Ph.D. Thesis
Doctor of Philosophy

DTU Compute
Department of Applied Mathematics and Computer Science

Characterization of absorption enhancers for orally administered therapeutic peptides in tablet formulations

Applying statistical learning

Soren Havelund Welling

Kongens Lyngby 2016

DTU Compute
Department of Applied Mathematics and Computer Science
Technical University of Denmark

Matematiktorvet
Building 303B
2800 Kongens Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

Summary

To develop a successful an oral formulation of insulin for treatment of type-2 diabetes patients would be a great mile stone in terms of convenience. Besides protecting insulin from enzymatic cleavage in the small intestine, the formulation must overcome the intestinal epithelia barrier. Absorption enhancers are needed to ensure even a few percent of insulin are taken up. In thesis article 1, various methods to measure the effect of absorption enhancement and enzyme stability of insulin were applied. The major class of absorption enhancers is surfactant-like enhancers and is thought to promote absorption by mildly perturbing the epithelial membranes of the small intestine. The Caco-2 (Carcinoma Colon) cells can grow an artificial epithelial layer, and are used to test the potency of new absorption enhancers. This project was aimed to identify new absorption enhancers, that are both potent and sufficiently soluble. Quantitative structural activity relationship (QSAR) modeling is an empiric approach to learn relationships between molecular formulas and the biochemical properties using statistical models. A public data set testing the potency of absorption enhancers in Caco-2 was used to build a QSAR model to screen for new potent permeation enhancers. Thesis article 2 contains likely the first QSAR model to predict absorption enhancement. The model was verified by predicting molecules not tested before in Caco-2. The Caco-2 model overestimates the clinical effect of lipophilic permeation enhancers. In the Caco-2 model all reagents are pre-dissolved, and therefore the assay cannot predict critical solubility issues and bile salt interactions in the final tablet formulation. A QSAR solubility model was built to foresee and avoid slow tablet dissolution. Due to enzyme kinetics, slow tablet dissolution will allow most insulin to be deactivated by intestinal enzymes. The combined predictions of potency and solubility, will likely provide a more useful *in-silico* screening of potential permeation enhancers.

Random forest was used to learn relationships between molecular descriptors and potency or solubility. However, unlike multiple linear regression, the explicitly stated random forest model is complex, and therefore difficult to interpret and communicate. Any supervised regression model can be understood as a high dimensional surface connecting any possible combination of molecular properties with a given prediction. This high dimensional surface is also difficult to comprehend, but for random forests, it was discovered that a method, feature contributions, was especially useful to decompose and visualize model structures. The visualization technique was named forest floor and could replace the otherwise widely use technique partial dependence plots, especially in terms of discovering interactions in the model structure. Thesis article

3 describes the forest floor method. An R package forestFloor was developed to compute feature contributions and visualize these according to the ideas of thesis article 3. Better interpretation of random forest models is an exciting interdisciplinary field, as it allows investigators of many backgrounds to find fairly complicated relationships in data sets without in advance specifying what parameters to estimate. Forest floor was used to explain how potency and solubility were predicted by random forest models.

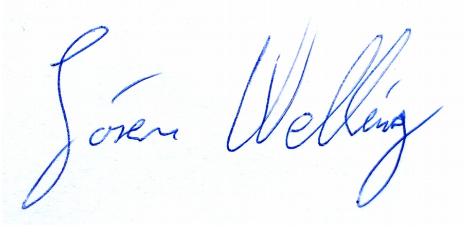
Preface

This Ph.D. thesis was prepared at the department of Applied Mathematics and Computer Science at the Technical University of Denmark in fulfillment of the requirements for acquiring a Industrial Ph.D. degree in Applied Mathematics and Computer Science.

A number of figures from third party sources have been copied or reproduced in thesis accordingly to the guide lines *Keep your thesis legal* [Joh+15]. For any figure stated as copied, I am not the copy right holder, and I have included the figure as less than a substantial part of someone others work to make a specific point. Other figures are either my creations, reproductions and/or copied from public available sources or mix of all three and can be copied and modified freely.

The results of Caco-2 and Ussing studies of thesis article 1, have been included before in my master dissertation. However the, enzymatic stability and calcium electrode studies and the preparation of the manuscript were not a part of the master thesis project.

Kongens Lyngby, October 7, 2016



Soren Havelund Welling

Acknowledgements

Thanks to Christian Vind for introduction to the Molecular Operating Environment and for consulting me in the early stage of the project. Thanks to Sara Øster Brebbia (Dirksen) and Sten B. Christensen for collecting permeability data.

Thanks to the many package authors of R. Thanks to the many people who care to answer questions in forums, write guides, blogs and tutorials. Thanks to *github.com*, *travis.cl*, *r-forge.r-project* and *CRAN* for hosting, building and testing the R package forestFloor.

This industrial PhD program have been sponsered by Innovation Fund Denmark and the STAR programme at Novo Nordisk.

A special thanks to my dear supervisors Hanne HF Refsgaard, Line K Clemmensen, Per B Brockhoff, Stephen T Buckley and Lars Hovgaard. Eventually I will pass on your advice to others, believing it were my own thoughts.

Contents

CHAPTER 1

Diabetes Type-2 and Oral Insulin

"Diabetes is one of the first diseases described in an Egyptian manuscript from c. 1500 BCE mentioning "too great emptying of the urine." The first described cases are believed to be type 1 diabetes. Indian physicians around the same time identified the disease and classified it as madhumeha or honey urine, noting that the urine would attract ants. The term "diabetes" or "to pass through" was first used in 230 BCE by the Greek Apollonius Of Memphis. The disease was rare during the time of the Roman empire with Galen commenting that he had only seen two cases during his career. Type 1 and type 2 diabetes were identified as separate conditions for the first time by the Indian physicians Sushruta and Charaka in 400–500 AD with type 1 associated with youth and type 2 with being overweight. The term "mellitus" or "from honey" was added by the Briton John Rolle in the late 1700s to separate the condition from diabetes insipidus which is also associated with frequent urination. Effective treatment was not developed until the early part of the 20th century when the Canadians Frederick Banting and Charles Best discovered insulin in 1921 and 1922. This was followed by the development of the long acting NPH insulin in the 1940s."

-https://en.wikipedia.org/wiki/History_of_diabetes [1]

[1] I dedicate this first quotation to Wikipedia a community curated encyclopedia. Open and free communities such as stackexchange.com (cross validated, stack-overflow, Tex, ...), R mailing list, youtube.com have taught me the most in the last three years. I hope scientific literature, soon also will become open source. That means open access to content and also a transparent community driven editing and reviewing process.

1.1 Diabetes

Diabetes is a major challenge to general health and quality of life in almost every country world wide. Type-2 accounts for the most incidents of diabetes and is strongly associated with lack of physical exercise, an unfavorable diet and obesity. Other risk factors are age and genetic predispositions. Type-2 diabetes is not only a problem in industrialized countries. World-wide 380 million people are estimated to be affected and 15% of deaths are attributable to diabetes. [Agu+13].

1.1.1 Blood sugar regulation in diabetic and healthy patients

Type-1 diabetes is defined as the inability to produce insulin due to the not fully understood auto-immune rejection of the insulin producing beta-cells. The typical onset of Type-1 diabetes is at child age or youth. Insulin is an important hormone for regulation of the human metabolism, as it promotes uptake of glucose in the liver, muscles and fat tissue. An oscillating level of insulin is required to regulate the energy metabolism during the day. Shortly after a meal, insulin is released to signal the start of glucose take up. In fasting state, insulin levels in healthy persons are low, as no new glucose is obtained from digestion. Insulin has an oppositely acting counterpart, glucagone, that promotes release of glucose primarily from the liver. Type-2 diabetes is defined by insulin resistance, where the peripheral tissues and the liver do not respond sufficiently to the endogenously produced insulin. When blood glucose levels exceeds 10 mmol/L (180 mg/dL), the kidney can no longer re-uptake all glucose from the excreted urine. The glucose in urine will sequentially increase osmolality and prevent the kidney from also reabsorbing water and hence the higher rate of sweet urine and the name diabetes mellitus. A healthy human can regulate the blood sugar within 3.9 mmol/L and 7.8 mmol/L throughout a normal day cycle. After the meals of the day, the food is metabolized and free glucose comes into blood circulation. Without regulation, a single meal would make the blood sugar far exceed normal levels[Sil10; Cow90].

To put these numbers in to perspective, ideally elevating the blood glucose from 3.9 to 7.8 mmol/L for adult of 80 kg and 8 liters of blood would only require $\frac{80\text{dL}(7.8-3.9)\text{mmol/L} \quad 180\text{mg/dL}}{10\text{mmol/L}} = 5.6\text{g}$ of glucose.

The pancreas, located adjacent to the first part of the intestine after the stomach, senses systemic blood sugar levels. The pancreas can also receive hormone signaling from the adjacent small intestines (e.g. GLP-1), indicating that food currently is being metabolised and systemic blood sugar soon will rise. Thus, whenever needed under and after a meal, the pancreas will release insulin. Insulin triggers a number of blood glucose lowering responses, rendering cellular uptake and storage of glucose. Glucose is stored short-term in the liver and muscles and in part converted to fat and stored long-term in fatty tissues. The liver is the major short term energy storage, taking up glucose and converting it to polymeric glucogen, that when needed can be reconverted into glucose and released to systemic circulation again. For type-2 pa-

tients, insulin secretion is constantly elevated to compensate for the insulin resistance, and therefore the pancreas cannot further regulate the glucose load from a meal, as it is already producing insulin almost as fast as possible [Sil10]. With a long-acting insulin analogue for a type-2 diabetic, the insulin requirement from the pancreas is no longer maxed out, such that the pancreas again can up and down regulate the insulin level during the day.

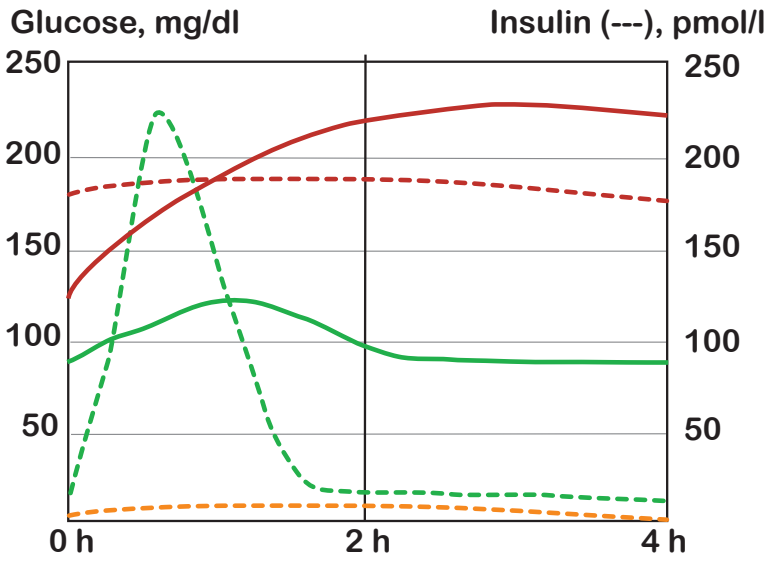


Figure 1.1: Glucose tolerance test: A typical healthy patient has a very low fasting insulin level (dashed green line) and a high transient response to regulate a glucose intake at 0 hours. A healthy patient is able to lower glucose level within 2 hours after glucose intake (green line). The insulin level of an untreated Type-2 patient (red dashed line) is already elevated and no further compensation is possible. The glucose level (red line) will not stabilize within 2 hours. Yellow dashed line represents a Type-1 patient with almost no endogenous insulin production. If untreated, glucose levels would exceed the ranges of this axis. Figure is reproduced by redrawing and combining illustrations from [Sil10; CL04].

A glucose tolerance test is used to diagnose diabetes. Figure 1.1 outlines a glucose tolerance test and the response from a healthy subject as well as for a type-1 and type-2 diabetic patient. A type-1 diabetic patient will have no endogenous regulation of blood sugar and both long acting and well timed short acting insulin therapy is needed. However, due to uncertainties of timing and variance of bio-availability, oral insulin therapy may at first only substitute long-acting insulin. As type-1 diabetics are

already treating themselves with injectable short-acting insulin in conjunction with meals, there is little rationale in replacing their long-acting basal insulin injectable formulation with an oral tablet formulation. In contrary for type-2 diabetics that only need a basal long acting insulin therapy to assist their pancreas in regulating the blood glucose, an oral insulin therapy would be a convenient option to have.

1.1.2 Treatment and compliance

Intensive anti-diabetic therapy is important to avoid or delay myocardial infarct, micro vascular diseases and kidney related complications [Hol+08; Bou+11; Gæd+08]. Having a chronic elevated high blood glucose level is simply very unhealthy long-term and will reduce the quality of life. As the type-2 diabetes progresses, glyceamic control is no longer achievable with a single oral agent such as Metformin or Sulfonylurea, which receptively lowers insulin resistance and increases endogenous insulin production. Injectable peptide based formulations of insulin-analogues or GLP-1-analogues will inevitably be considered at some point in the disease stage progression. Inconvenience and patient compliance are the main factors for not achieving the clinical recommendations for blood glucose control in insulin treatment of type II diabetes patients. To initiate injectable insulin therapy is a psychological barrier for Type-2 patients and a cause of worrying [Kor02]. Nevertheless insulin therapy will eventually become the outcome for most Type-2 patients. From early onset type-2 patients still have the ability to regulate blood sugar to some level and strict hourly control is not needed. In two groups of 24,000 and 10,000 patients of seniors aged 60-69, the former group was diagnosed with type-2 diabetes within last 9 years and the latter group has been diagnosed more than 9 years ago. When recently diagnosed, patients are prescribed oral hypoglycaemic agents (OHA). Therefore 50% of early diagnosed patients received metformin and/or sulfonylurea. Only 7.5% received insulin treatment. In contrast, in patients diagnosed more than 9 years ago, fully 65% are ordinated treatments based on injectable insulin [Hua+14]. Thus a typical type-2 diabetes disease progression will start with fairly convenient once a day tablets, mildly lowering glucose. Hereby the constant insulin production of the pancreas is lowered, giving the pancreas a insulin buffer capacity to regulate blood glucose throughout the day. With time, more insulin is needed to obtain sufficient blood glucose control. The disease will progress from a treatment only complimenting the anatomical blood sugar regulating mechanisms, to the insulin therapy becoming the main regulation of blood glucose.

Compliance is the extent of which the patient uses the medicine as prescribed by the physician. Especially for short acting injectable insulins, daily awareness and monitoring of blood glucose and storing injectable insulin refrigerated may be a large requirement for the patient. Type-1 patients who came to master these skills early in life, are likely to have a much higher compliance. Simply, the patient must learn to live by a fairly complex treatment regimen late in life. Compliance for only oral agents can already be as low is 50% after six months [Gar+13]. In a study

50% of insulin-naive patients perceived injectable insulin initiation as a failure. In a study of those patients who did not comply to an treatment regime with injectable insulin, the most common reasons given were planing to improve healthy behavior instead (25%), fear of injection (13%), negative impact on work (9%), concerns on long term medication (9%), inconvenience (6%), and not believing insulin was necessary (6%). Despite the various available anti diabetic agents for various stages of type-2 diabetes, it is indicated that less than 50% of patients achieve the aimed glucose control recommended and around two-thirds will die prematurely of cardiovascular disease [Gar+13]. In contrary it is argued that current injection pens actually have improved comfort for insulin injection so much, that needle phobia alone is not at strong argument for developing oral insulin formulations [Mah+14].

The possible introduction of oral insulin may provide a mid-way solution, especially for type-2 patients where other oral agents no longer are potent enough, yet with the same ease of administration as oral agents. Thus oral insulin may prolong the time the patient can regulate blood sugar without injectable insulin and perhaps improve compliance.

1.2 Drug Development Challenges of Oral Insulin

Oral formulations of insulin is not an obviously great idea. One broad intuitive explanation is: From natures side, an organism tend not take to up any foreign substances, and certainly not foreign proteins or peptides. External proteins and peptides are likely produced by foreign species, and therefore have been created to serve independent purposes, that not necessarily are in alignment with the survival of this organism.

Likewise, the human body has a series of barriers in the gastrointestinal tract before proteins such as insulin, reach systemic circulation. Figure 1.2 illustrates the upper gastrointestinal tract and the barriers for insulin.

1.2.1 Acid resistant coating

Normal protein digestion starts in the stomach with the enzyme pepsin cleaving protein amide-bonds next to lipophilic/aromatic amino acids. Insulin formulations are simply protected towards pepsin and acidic hydrolysis with an acid insoluble tablet coating. The coating is made of polymers with pH-dependent solubility. Acidic side groups will only deprotonate under neutral pH. Deprotonation, and hence ionization, increases the solubility of the otherwise non-ionic lipophilic polymers [CM99; Gab+10]. Most coatings are designed to dissolve at pH > 5.5 [Mah+14].

The sphincther, the opening muscle of the stomach, forwards some of the acidic stomach content to the upper duodenum and eventually the coated tablet or capsule. The insulin producing gland, the pancreas, is central to the gastro intestinal digestion. The pancreas will secrete to the pancreatic duct alkaline carbonate to neutralize the

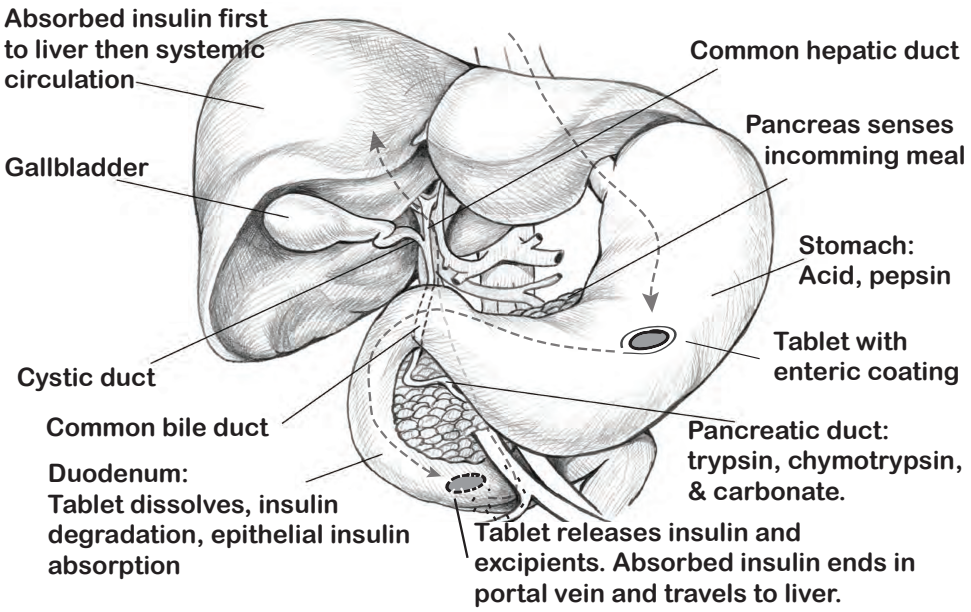


Figure 1.2: (1) Tablet coating protects against acidic hydrolysis and pepsin. (2) Release of basic carbonate, bile, trypsin and chymotrypsin. Pepsin is inactivated at neutral pH. Bile interferes with surfactant like absorption enhancers. (3) Tablet dissolves. Lipophilic absorption enhancers will slow down insulin release and allow trypsin and chymotrypsin to inactivate all insulin before absorption. (4) Insulin permeates duodenal and jenunal epithelia facilitated by absorption enhancers. Illustration kindly provided for public use by NIH: National Institute of Diabetes and Digestive and Kidney Diseases. `https://catalog.niddk.nih.gov/imagelibrary/detail.cfm?id=148` Original captions modified..

hydrochloric stomach acid. The stomach enzyme pepsin is deactivated by neutral pH, while neutral acting trypsin and chymotrypsin are released from the pancreas as well. Depending on the thickness and the acid groups of the coating polymer, the tablet/capsule will start dissolving immediately in the duodenum and jejunum or as late as in the colon.

The stomach empties approximately only every 50-120 minutes during fasting and even rarer for diabetic patients [Sil10; Cor+95; Gab+10], thus the accuracy of timing of the dose is not likely to match injectable insulin. Therefore, oral insulin is not likely to replace fast acting well timed doses of injectables used by type-1 diabetics in connection with meals. Oral insulin is most likely to replace longer acting insulin analogues, where the exact onset of action is of less concern.

1.2.2 The peptide itself: Size does matter, so does lasting long and stability

Presently several oral insulin formulations are in clinical phase two and three. These formulations can rely on a protein backbone modification to make insulin intrinsically more stable to enzymatic degradation. The formulations' approaches to increase the bio-availability are: absorption enhancers, enzyme inhibitors (soybean, citric acid) and micro/nano-encapsulated carriers [Agu+16]. Oral delivery of other peptides such as glukagon-like peptide-1 (GLP-1) analogues, octreotide (somatostain agonist), parathyroid hormone are also in clinical development in 2016 [Agu+16].

Peptide APIs in oral formulations targeting systemic circulation, that have been approved by FDA or EMA are Cyclosporin (MW 1200) Desmopressin (MW 1100), Taltirelin (MW 500) and glutathione (MW 300) [Agu+16]. The reason these four APIs have been successfully introduced to market before insulin is likely in part their significantly smaller size than monomeric insulin (MW 6000) and these peptides can therefore permeate the epithelial barrier more easily. With the current formulation technology it is only barely possible to administer insulin.

> "The selection of a suitable peptide for oral formulation is, therefore, a key commercial decision. For example, selecting a complex, high molecular weight (MW), narrow therapeutic index peptide, manufactured by a costly recombinant approach, requiring multiple daily oral administrations would be problematic." [Mah+14]

In this quote, Maher *et al* point out that oral peptide delivery for decades has been in its infancy and that we cannot suddenly deliver any type of peptide. In fact the current achievements are more attributable to the biotechnological advances allowing modification of the peptide and cheap production. If less than 5% insulin is absorbed, 20 times as much insulin must be produced.

Treatment with injectable peptides such as insulin, GLP-1 and growth hormone have become more convenient with fewer injections in the last decades, as extensive research has aimed to modify the intrinsic endogenous hormone to increase the apparent

and actual half-life. The apparent half-life is extended by formulation approaches only gradually releasing the insulin from the injection site, but also the systemic half-life can be extended [Arn+10]. Hereby, patients treated with a constant level of hormone, only have to inject themselves daily or even weekly. Oral peptide formulations at first will likely be attempted switches from their prenatally-marketed counterparts [Mah+14]. As only a couple of percentages of the peptides are likely absorbed, the relative variation of absorption is potentially very high [Gab+10]. A dosing interval significantly shorter than the half-life of the drug is a classic method for maintaining a relatively constant drug concentration within the therapeutic window [TR06].

Analogues with enzymatic stability are also required. The luminal enzyme activity by trypsin and chymotrypsin will likely inactivate released insulin in less than 5-15 minutes [Wel+14]. Co-formulation of soy-bean enzyme inhibitors [FUJ+85], covalent peptide protecters (SNAC) [BML13] and pH lowering [Wel+14] can only lower degradation 2 to 5-fold. Designing stable analogues is also needed to obtain sufficient stability. PEGylation, cyclization and modification of back-bone structure are known approaches to increase intrinsic peptide analogue stability [BML13].

1.2.3 Permeation enhancers to open the epithelial barrier

Thirdly peptides or proteins have to pass the epithelial barrier. Insulin (MW 6000 D) is likely near the upper limit for how big the proteins currently can be successful delivered. For Octertide, it was possible to select a small part of the peptide while still retaining activity [Agu+13].

Fatty acids with C8 to C16 chains have been used to open tight junction between epithelial cells and mildly perturb the phospholipid membrane of the epithelials cells. There has been an extensive research [BML13; Mah+09; AM90] uncovering how caprate (aka. C10) regulate calcium levels, and phosphyrolation cascades of the epithelial cells leading to opening of tight junctions. Nonetheless, in order to obtain a sufficient response caprate has to be released in intestinal lumen in concentrations close to the critical micelle contration which is where the perturbing effect on the phospholipid membrane sets in [BML13]. No specific technology increasing protein absorption significantly has emerged from the caprate-calcium-tight junction-theory. In practice fatty acids are surfactants, and most surfactants will destabilize the epithelial membrane and promote peptide absorption. The important part is how wide is the therapeutic window, potency versus toxicity and if these surfactant ate sufficiently soluble. I do not dismiss that, biological effects such as the C10-Calcium-tight junctions in theory may render one surfactant slightly more or less potent. However, such effects would be difficult to predict. The working hypothesis of this thesis is that central properties of surfactants are fairly possible to predict as they arise from non-complex physical phenomena, and new absorption enhancers could be discovered simply from the expected structure. In contrary it is not possible to learn the mechanism of receptor mediated permeation enhancers, as even the smallest change in the molecule could change the receptor-agonist affinity.

Where fatty acids have one deprotonated carboxylic acid as hydrophile head group, there several other possible head groups. Most enhancers have a mono acyl chain typically from 8 to 16 carbon atoms and some hydrophilic domaine. Other surfactant enhancer classes are e.g. the acyl-carnitines, acyl-cholates[LS00], phosphocholines[LLT99], acyl-maltosides [Pet+13] and acyl sulphates [AA93].

CHAPTER 2

Measure permeation enhancement

2.1 Studies to test peptide absorption

The goal for an oral formulation is to achieve a high bio-availability with a low dosage variance. That is to maximize the percentage of dose absorbed, and to minimize the day-to-day variance. However in order to learn how a formulation attempt may be failing, it is needed to break down the process into individual steps. The typical three steps to consider are dissolution of the tablet, enzymatic stability and permeation enhancement. The first article included in this thesis Section 2.3, *"The role of citric acid in oral peptide and protein formulations: Relationship between calcium and proteolysis inhibition."* [Wel+14], is a good example of the *in vitro* methods used to evaluate permeation enhancement and enzymatic stability. These three steps can be understood as a chain. If just one link fails, the overall result will be poor.

Studies to test permeation enhancement ranging from early proof-of-concept development to clinical trials are listed in Figure 2.1. Beyond this list focusing on permeation studies, also dedicated studies of dissolution, peptide stability and toxicology would e.g. be required.

In preclinical development, studies are formally categorized in latin. *Ex vivo* is outside the living and *in vivo*; isinside the living organism. Often *in vivo* refers to animal studies only and not human studies. Studies in human are termed clinical trials. It is obviously sensible to test many new insulin analogues and formulations in fairly inexpensive and fast studies. The term *in vitro*, in glass [vial], is used for lab bench studies. Lastly *in silico*, in silicium, is used for computer models.

Medical research progress by observing relationships and propose general causal mechanisms, that are later verified and utilized. Whereas clinical trials are the most representative for the intended population of patients. One may ask why to use *in vitro* studies at all, when these only are crude simulations of the clinical therapy. However, it is difficult to deduct why a clinical therapy gave no positive results, as the multiple potential steps of failure cannot be observed. Often only the final of the clinical trial can be observed. Moreover, negative confirmation is important to narrow in what part of a therapy was essential for a positive outcome. Ethical considerations limit designing clinical trials, that conflict with the best interest of patients. With *in vivo* studies it is possible to test such therapies, as long as the animal suffering is kept

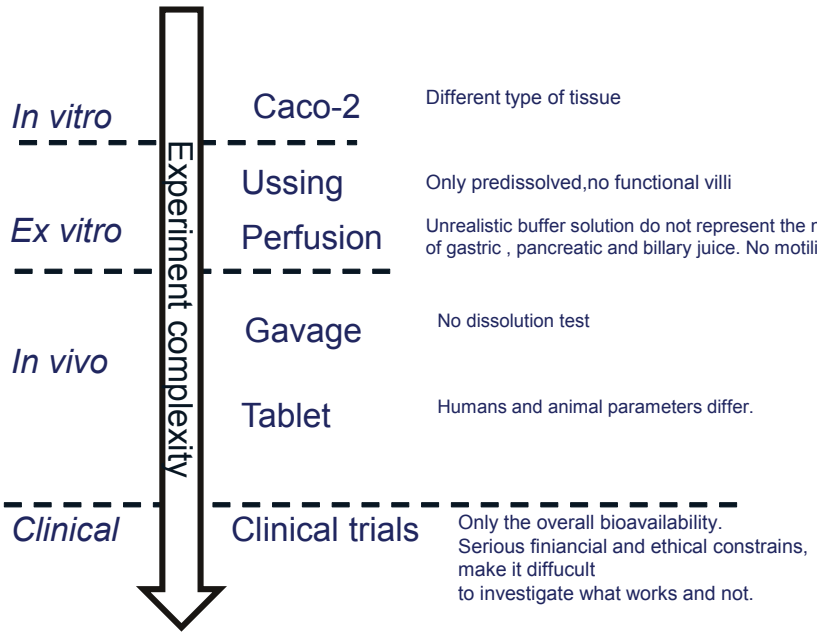


Figure 2.1: Schematic overview of types of experiments ranging from Caco-2 to clinical. Simple experiments to not accurately simulate the actual process, and there will be a number of biases which must not be over interpreted..

minimal. Still, the gastro-intestinal tract is a complicated piece of machinery and it is not possible to control or measure every aspect the peptide absorption process. Both in clinical studies and *in vivo* studies there can be a high subject-to-subject variance due to the variation physiology and genetic disposition. Therefore these studies may have a high unexplainable variance component. This variance component makes a step-wise incremental optimization of formulation challenging, as it becomes hard to evaluate what works, due to no significant change of bioavailability. In contrary, the lab bench *in vitro* experiments offer close control of the experiment conditions. These experiments are often fast and of relatively low cost, while the reproducibility and statistical power are high. Also the *in vitro* experiments can be split into individual steps as discussed in this chapter, such that every aspect increasing the bio-availability insulin can tested.

From a modeling perspective, a larger observation number is needed, especially for non-linear machine learning. Data from *in vitro* is less expensive to acquire in larger numbers. The *in vitro* experiments can often be crude and it is accepted as

a necessary evil, that the conditions do not exactly represent completely a clinical situation. Figure 2.1 lists the range of permeation experiments, and is intended to reflect this conflict between representative studies and elucidating studies.

When training machine learning models on data from *in vitro* experiments, we only expect the model to reflect the experiment. The prediction of such model will only be useful to that extend the experiment is a fair generalization of the insulin absorption. With a sufficient theoretical understanding of the overall insulin absorption process and the experiment biases, we can hopefully use the model appropriately and identify the causal relationships between formulation and outcome.

2.1.1 Caco-2 monolayers

Caco-2 monolayers have been a standard screening method for intestinal drug permeability. The Caco-2 permeation model, as other permeation models, consist of a donor chamber, an epithelial barrier and a receiver chamber. Caco-2 monolayers are colonic human cancer cells, which are much more prolific and easy to handle, than primary (human non-cancer) epithelial cells. When seeded on micro porous filter (Ø 2cm) on a 12-well plate, the Caco-2 cell line will grow and form epithelial monolayers in two weeks. A typical test will use 4 mono layers per tested treatment and the study is to be repeated on a different day. A concentration of the permeation drug is applied to the donor side, and the receiver side is sampled at 4 time points to measure the diffusion rate. During the test, the nutrient rich growth media is replaced with a minimal buffer solution. The buffer solution only contain a physiological level of electrolytes such as calcium, potassium, sodium, chloride and phosphate plus an acid/base buffer. These ions maintain isotonicity and a normal electrical membrane potential. Active membrane transporter receptors in the epithelial membrane rely on a given correct membrane potential.

Within standard drug development of small molecules, a low permeability result would likely lead to discarding the given analogue, as it is easier to find a new analogue with a favorable permeability. In insulin therapy, there are few alternatives to insulin-like analogues and therefore the low permeability must be alleviated by e.g. co-formulation of permeation enhancers. To use Caco-2 monolayers to test absorption enhancers likely account for a smaller part of the use of Caco-2 monolayers. Here the permeability of insulin is already low, and may be increased by a given absorption enhancer. For surfactant enhancers at high concentration, the monolayer can be completely disrupted, while at low concentration no useful effect is observed.

2.1.2 Measuring permeability

To estimate how potent a permeation enhancer is, permeation markers have been used. An obvious marker is human insulin itself, but insulin is difficult to handle and quantification by immunobinding-assay can be expensive. A number of of alternative permeation markers are typically used to compliment insulin. One of these are ^{14}C

or ^{3}H isotope labeled mannitol. Mannitol, is approximately as hydrophilic as glucose, and is thought to only diffuse passively via the paracellular pathway through the tight-junctions [ANA92; Art+94]. Unlike glucose, mannitol is not taken up by active transport, and is therefore a suitable passive transport marker. As mannitol is hydrophillic, it cannot permeate the lipophilic plasma membrane and can only permeate between the cells by the tight junctions. Mannitol is therefore a para-celullular passive permeation marker. Besides permeation of mannitol through the tight junctions, also permeation of electrolytes can be used as a marker of how open the junctions currently are.

Trans epithelial electrical resistance (TEER) can measure how open the paracellular tight junctions are, as the junction cross sectional area is proportional to the conductivity, and conductivity is inverse resistance. TEER is an non-invasive measurement easy to apply. A certain TEER response can be translated to change of insulin permeability. E.g. lowering the epithelial resistance 50% by the lauroyl carnitine chloride surfactant enhancer translated to a (40-fold increase) of permeability of insulin in Caco-2 model [Wel+14].

FITC-dextran 4000 dalton (FD4) is a florescent labeled sugar polymer of similar hydrophilicity and size as insulin. FD4 can mimic insulin as transport marker as it has similar molecular weight and hydrophobicity [VKB99]. FD4 on the other hand has no enzymatic instability and can be used to estimate what would have been the permeation of unchanged insulin, disregarding the enzymatic cleavage in the luminal space. FD4 can be specifically and accurately quantified, as it is a strong fluorophore.

2.1.3 Calculating permeability

Gradient driven diffusion of mannitol, FD-4 and insulin across the epithelial barrier are first order processes, where the transport per time (aka. flux) is proportional to the concentration gradient C_i (mol/L). Only a few percent of the total amount of transport marker will permeate the barrier and therefore will C_i be approximately unchanged throughout the experiment. Hereby becomes the apparent permeability $\boldsymbol{P}_{app}$ proportional to the transport rate or flux $\boldsymbol{J}$ (mol/s), which is usually estimated by sampling the receiver basolateral side 4 times. When plotting receiver concentration versus time, a fitted ordinary least squares slope is used as the overall flux through out the experiment. Permeability is the flux per concentration gradient and corrected for the cross sectional area of the chamber, $\boldsymbol{A}$ (cm^2). Therefore the apparent permeability is calculated as

$$\boldsymbol{P}_{app} = \boldsymbol{J}/\boldsymbol{A}/\boldsymbol{C}_i = \frac{d\boldsymbol{C}_r}{dt}/\boldsymbol{A}/\boldsymbol{C}_i \quad ,$$

where $\frac{d\boldsymbol{C}_r}{dt}$ is the slope of ordinary least squares regression of receiver chamber concentration versus time. The intercept with x-axis (time) is interpreted as the lag time, which is the time it takes to saturate the mucus layer, the thin unstirred water layer and the epithelial junctions. The lag time is typically a couple of minutes depending whether using Caco-2 or Ussing chamber. To observe an apparent constant flux e.g. throughout a one hour experiment is normal. The small decrease in concentration

gradient of some few percent throughout the study may ideally have caused deceleration of the flux. However, it is possible that the actual permeability of the epithelial barrier is increasing through the one hour flux study due to slow deterioration by the absorption enhancer, and therefore a constant flux is observed. In control epithelial barriers without absorption enhancer treatment, the barrier integrity is not expected to decline, but at the same time the flux is very low, so the concentration gradient do not change.

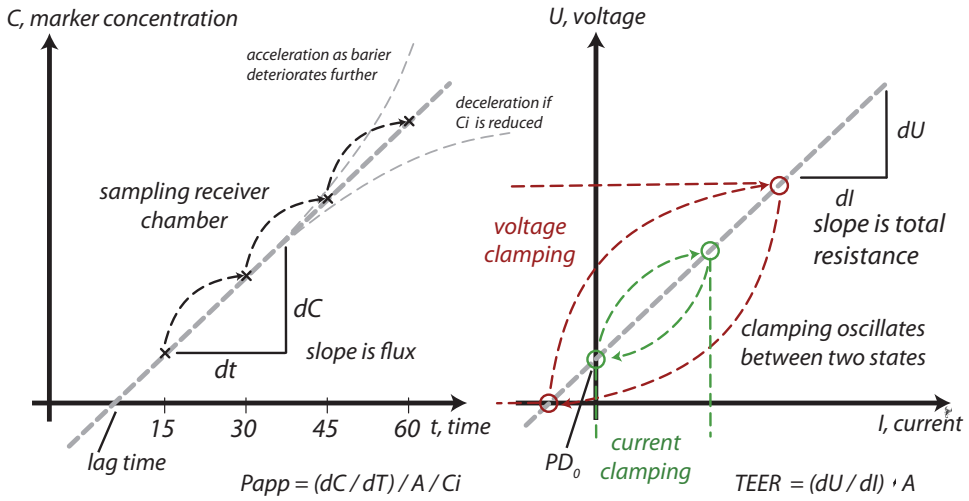


Figure 2.2: Illustration of how *(left)* $\boldsymbol{P}_{app}$ and *(right)* TEER is calculated..

The tight junctions are ideally water filled channels between the cells of the barrier. The water in the channels can be thought to have specific conductance depending on the concentration and diffusability of ions. When tight junction are widened, the cross-sectional area of tight-junction is changed. Similarly as marker flux is driven by a concentration difference, the electrical current $\boldsymbol{I}$ (Ampere μA) is driven by an electrical potential difference $\boldsymbol{U}$ (Voltage V). The electrical current $\boldsymbol{I}$ is analogue to the flux $\boldsymbol{J}$ and potential difference $\boldsymbol{U}$ to $\boldsymbol{C_i}$. Therefore the membrane conductivity corrected for membrane area $\boldsymbol{G_A}$ area can be expressed in parallel to $\boldsymbol{P_{app}}$, such that

$\boldsymbol{G_A = I/A/U}$.

The inverse of $\boldsymbol{G_A}$ is named TEER (trans epithelial electrical resistance)(Ohm cm^2) in drug transport studies. To summarize both $\boldsymbol{P_{app}}$ and inverse TEER describe how readily a flux of either molecules or ions are driven through the epithelial barrier by respectively a concentration gradient or a electrical potential gradient.

Figure 2.2 describes how to calculate the slopes for P_{app} and TEER. For P_{app} the receiver concentration $\boldsymbol{C}_r$ is plotted as a function of time, the slope represents flux

$\boldsymbol{J}$. TEER is either measured with voltage or current clamping. To measure TEER with voltage clamping is to control the electrical gradient (potential difference) with electrodes and measure the actual current. Whereas, current clamping is to drive a specified flux of ions (current) through the epithelial barrier and record the electrical gradient. In either case is the membrane potential difference U plotted as a function of current. The slope represents TEER not corrected for area. At zero applied current, the potential difference is not neccesarily zero, as the epithelia has an active energy dependent ion transport. The slopes obtained by current clamping and voltage clamping are exactly the same.

For direct measurement of the intrinsic permeability of a given drug molecule without permeation enhancers, TEER is mainly used to check barrier integrity. For measuring the potency of permeation enhancers to increase permeability, TEER can be used as indicator hereof. To directly measure the permeation of insulin would be preferable, but this is more slow, expensive and unlikely to find in already published results. As the current and potential difference can be manipulated with electrodes, the time resolution for TEER measurements is ideally in seconds. The time resolution for permeation markers is limited by the sampling rate and accuracy of the concentration determination. However, in Caco-2 studies the sampling rate for both TEER and $\boldsymbol{P}_{app}$, the sampling rate of Caco-2 studies are typically once per 15 minutes.

As the potency of permeation enhancers can vary greatly, it is not meaningful to test all enhancers by the same concentration. Like a under- and over-exposure in photography, a too low concentration will have no measurable effect and a too high concentration will kill the epithelial cells. The %TEER-decrease describes how much resistance the epithelium looses by a given permeation enhancer at one specific concentration. The measured %TEER decrease is likely a sigmoidal curve as a function of the logarithmic enhancer concentration. Whitehead *et al* produced the data set, that the permeation model of this thesis has been based on. Every enhancer was tested at 1%, 0.1% and 0.01% (w/w) and the resulting %TEER decrease was measured [WKM08]. If a given enhancer had no effect the total TEER would not change. In opposite, if the enhancer was overdosed, the epithelium would practically be dissolved and TEER would plummet to zero. In order to summarize how a permeation enhancer performed across the three concentrations, I decided to simply compute the average %TEER decrease for the three concentration levels. This potency average was named Tpot. If Tpot=0.5, the enhancer likely elicited ratio-TEER decreases (0.90, 0.50, 0.10) at the respective concentrations. Likewise for Tpot=.9 the ratio-TEER decreases may have been (1.00,0.95,0.75). To predict the %TEER-decrease at one specific concentration is both not that useful and may be difficult. To predict the potency of a permeation enhancer as defined by Tpot is more useful and realistic. If the potency of one enhancer is mediocre, it may be acceptable if the predicted solubility is outstanding. Then a high concentration of the permeation enhancer can be released at once.

2.2 Mechanisms of absorption enhancement in oral formulations

The formulation part of oral peptide therapy is mainly to ferry the highest possible total and relatively amounts of insulin unchanged through the upper gastro-intestinal tract. The two major barriers are enzymatic degradation and poor epithelial permeability. Other barriers are the mucus and thin unstirred water layer on top of the epithelia. These barriers are not regarded as prominently rate limiting bottle necks and are not given as much attention.

2.2.1 Preventing and measuring enzymatic degradation

Enzymatic stability of peptides vary greatly in the intestinal tract. The time span insulin can remain unchanged after tablet dissolution in the small intestinal lumen is considered to be from a single minute up to 15 minutes [Wel+14]. Enzymatic degradation rate (V_i) is assumed to follow Michalis-Menten kinetics, which describes the degradation rate per enzyme as a function of substrate/peptide concentration $[S]$, enzyme-substrate binding constant (K_m), and max enzyme degradation speed V_{max}. The kinetics is described in by this equation

$V_i = \frac{V_{max}[S]}{K_M+[S]}$.

The Michales-Menten kinetics describe, that at very low concentrations of substrate (peptide), the degradation rate will be proportional to the concentration of insulin in the luminal space, thus a first order decay process. The Michaelis-Menten kinetics opens for three ways to limit relative peptide degradation. First the apparent K_m binding constant can be increased by adding a competitive substrate. The enzyme will then bind to another substrate such as soy bean peptides or Bowman-Birk compounds. A very potent non competitive inhibitor would bind strongly to the enzyme, such that only a small amount of inhibitor would be needed. Such non-peptide inhibitors are in practice toxic and not suited for chronic treatment [Ber98; MKD80].

A second option, which is used in the article [Wel+14] of this thesis, is to lower the apparent V_{max}. Citric acid itself is not a substrate of trypsin or chymotrypsin. As citric acid is dissolved, it will partly release protons into the luminal space and thereby lower pH. The enzymes trypsin and chymotrypsin have a 10-fold lower V_{max} at pH=4. The last way to inhibit the relative degradation is to release as much substrate as fast as possible into the luminal space. As the substrate concentration increase $[S] >> K_m$ and $V_i \approx V_{max}$ there will be a theoretical upper limit for how much insulin can be degraded per time. Driving up the concentration of insulin, will surpass the linear range of degradation and approach the V_{max}. This approach could fairly be called hit and run, as enough insulin is released in such a short time and location, that the enzyme cannot degrade all at once. The latter approach is highly dependent on the dissolution time of the tablet. If the tablet dissolves over an hour, the insulin would be released very slowly in small concentrations, and the enzymes

would have plenty of opportunity to degrade all insulin. To simply drive up the insulin content of the tablet is not a likely successful strategy, as the insulin is very costly to produce. Also, there can be an upper boundary of how much insulin, that can be included in one tablet due to drug safety. Otherwise a case of unexpected high bioavailability could lead to a hazardous overdose and acute hypoglychemia.

2.3 Article 1: *The role of citric acid in oral peptide and protein formulations: Relationship between calcium chelation and proteolysis inhibition*

European Journal of Pharmaceutics and Biopharmaceutics 86 (2014) 544–551

Contents lists available at ScienceDirect

European Journal of Pharmaceutics and Biopharmaceutics

journal homepage: www.elsevier.com/locate/ejpb

Research paper

The role of citric acid in oral peptide and protein formulations: Relationship between calcium chelation and proteolysis inhibition

Søren H. Welling [a,b], František Hubálek [a], Jette Jacobsen [b], David J. Brayden [c], Ulrik L. Rahbek [a], Stephen T. Buckley [a,*]

[a] *Diabetes Research Unit, Novo Nordisk A/S, Måløv, Denmark*
[b] *Department of Pharmacy, Faculty of Health and Medical Sciences, University of Copenhagen, Copenhagen, Denmark*
[c] *UCD School of Veterinary Medicine and UCD Conway Institute, University College Dublin, Dublin, Ireland*

ARTICLE INFO

Article history:
Received 27 August 2013
Accepted in revised form 23 December 2013
Available online 31 December 2013

Keywords:
Oral peptide delivery
Citric acid
Proteolysis inhibition
Chelation
Intestinal drug permeability
Insulin

ABSTRACT

The excipient citric acid (CA) has been reported to improve oral absorption of peptides by different mechanisms. The balance between its related properties of calcium chelation and permeation enhancement compared to a proteolysis inhibition was examined. A predictive model of CA's calcium chelation activity was developed and verified experimentally using an ion-selective electrode. The effects of CA, its salt (citrate, Cit) and the established permeation enhancer, lauroyl carnitine chloride (LCC) were compared by measuring transepithelial electrical resistance (TEER) and permeability of insulin and FD4 across Caco-2 monolayers and rat small intestinal mucosae mounted in Ussing chambers. Proteolytic degradation of insulin was determined in rat luminal extracts across a range of pH values in the presence of CA. CA's capacity to chelate calcium decreased ~10-fold for each pH unit moving from pH 6 to pH 3. CA was an inferior weak permeation enhancer compared to LCC in both *in vitro* models using physiological buffers. At pH 4.5 however, degradation of insulin in rat luminal extracts was significantly inhibited in the presence of 10 mM CA. The capacity of CA to chelate luminal calcium does not occur significantly at the acidic pH values where it effectively inhibits proteolysis, which is its dominant action in oral peptide formulations. On account of insulin's low basal permeability, inclusion of alternative permeation enhancers is likely to be necessary to achieve sufficient oral bioavailability since this is a weak property of CA.

1. Introduction

Development of oral delivery systems for proteins and peptides offers the promise of improved patient compliance compared to conventional parenteral administration. Moreover, in the case of certain protein therapeutics (e.g., insulin), the physiological response elicited may exhibit a pharmacodynamic profile which more closely resembles the natural physiological response. However, delivery of protein therapeutics is severely hindered by poor absorption across the intestinal barrier and extensive degradation by proteolytic enzymes. Thus, to effectively overcome these impediments, a formulation strategy which can modulate both of these processes is necessary to achieve acceptable oral bioavailability with low intra-subject variation.

Although degradation of proteins by gastric enzymes and low pH may be overcome via inclusion of an enteric coating, the approach to minimise proteolytic activity in the small intestine, while simultaneously ensuring efficient release and permeation represents a more significant challenge. In this regard, one such concept extensively explored is that of acidic inhibition of proteolysis. Luminal proteases, such as trypsin and chymotrypsin, exhibit maximum activity at pH $\geqslant$ 6.5 [1,2] i.e., that typically observed in the pH microenvironment of the jejunum and ileum. Via adjustment of the local pH to values corresponding to pH < 6.5, proteolytic activity of enzymes such as chymotrypsin [1], the primary luminal degrading enzyme for insulin [2], can be significantly diminished.

Indeed, acidic inhibition of proteolysis as a strategy for the oral delivery of therapeutic peptides recently gained attention following Tarsa Therapeutics (Philadelphia, PA) successful completion of a phase III trial ('ORACAL') for orally delivered salmon calcitonin (sCT) [3]. Such technology typically comprises of an enteric coated capsule or tablet, which bypasses the stomach unchanged, along

Abbreviations: CA, citric acid; Cit, citrate; DOC, sodium deoxycholate; EDTA, ethylenediaminetetraacetic acid; ISE, ion selective electrode; KH, Krebs–Henseleit; LCC, lauroyl carnitine chloride; sCT, salmon calcitonin; TDC, taurodeoxycholate; TEER, transepithelial electrical resistance.

* Corresponding author. ADME Department, Novo Nordisk A/S, Novo Nordisk Park, 2760 Måløv, Denmark. Tel.: +45 3079 4609.
E-mail address: spby@novonordisk.com (S.T. Buckley).

0939-6411/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ejpb.2013.12.017

with a pH-lowering excipient contained in vesicles (e.g., an organic acid such as citric acid). Upon entry into the duodenum with its luminal pH range of between 5 and 6, pH-dependent disintegration of the polymer coating of the dosage form commences, followed by release from the vesicle of both co-localised API and citric acid (CA). Concomitant association of CA maintains a decrease in local pH, thus stabilising the co-released peptide. In this way, it facilitates a reduction in the luminal enzymatic activity, providing a higher concentration gradient of the API over time, which in turn promotes improved absorption and bioavailability [4,5].

Alongside pH-lowering agents, co-administration of an absorption enhancer(s) has generally been regarded as indispensable due to the inherently poor epithelial permeability properties of proteins and peptides [5,6]. Indeed, previous publications exploring this technology have employed LCC, an amphiphilic surfactant [5–7]. However, based upon the recent ORACAL sCT study, where an absorption enhancer was omitted, one may speculate that either the need for co-administration is diminished on account of the proposed permeation enhancing properties of citric acid or citrate (Cit) [3], or that enhancers might not be required for oral sCT where bioavailability of 1–3% is typical for marketed nasal versions of this particular potent molecule [8]. CA and Cit are GRAS excipients and have been widely employed in oral formulations of small molecules. Thus, despite this formulation strategy being comparatively new, a body of literature exists examining the multiple mechanisms by which CA, in its salt form (i.e., tri-sodium citrate) may promote oral absorption. Cit exhibits calcium chelating properties and evidence exists to suggest that it may increase paracellular absorption, by triggering disruption of tight junction complexes via depletion of intracellular calcium [9–11].

In this report, the potential mechanism of action of CA as both an acidic proteolysis inhibitor and calcium chelator/permeation enhancer was addressed and conclusions made as to which might be its dominant action at relevant pH values in the upper small intestine. *In silico* and *in vitro* determination of CA's calcium chelation activity and its capacity to prevent insulin degradation by peptidases across a broad range of pH values were obtained. From this data we assessed whether or not a common pH range existed over which *both* proteolysis inhibition and calcium chelation occurred. Finally, the capacity of CA/Cit to enhance permeability was investigated in Caco-2 monolayers and rat intestinal tissue and compared to that of lauroyl carnitine chloride (LCC), an established amphiphilic permeation enhancer [5–7] previously employed as an additional agent in pH-lowering oral peptide formulations.

2. Materials and methods

2.1. Materials

Caco-2 cells (ATCC-HTB-37) were obtained from American Type Culture Collection (ManassasVA). Cell culture media (Dulbecco's modified essential media (DMEM)) and penicillin/streptomycin were purchased from Lonza (Verviers, Belgium). All other supplements i.e., foetal bovine serum (FBS), HEPES buffer and non-essential amino acids (NEAA) as well as Hanks' balanced salt solution (HBSS) and trypsin were purchased from Gibco (Naerum, Denmark). Corning Transwell® filter inserts (1.12 cm^2 surface area, 0.4 μm pore diameter) were purchased from Fisher Scientific (Slangerup, Denmark). FITC-dextran 4 kDa (FD4) and D-glucose were purchased from Sigma Aldrich (Dublin, Ireland). Bovine serum albumin (BSA) was purchased from Sigma Aldrich (Copenhagen, Denmark). All other reagents were of the highest analytical grade.

The Iso-Insulin ELISA assay kit was purchased from Mercodia (Uppsala, Sweden). Lauroyl-DL-carnitine (LCC) was purchased from Chemos (Regenstauf, Germany). [^{3}H]-mannitol, [^{14}C]-mannitol and Ultima Gold® scintillation fluid were purchased from Perkin Elmer (Waltham, MA). Liquid scintillation counting was carried out using a TopCount C990201 or a TriCarb 2900TR liquid scintillation counter (both Perkin Elmer). Luminescent measurements were performed using a Spectramax® 250 or Gemini® microplate reader (both Molecular Devices, Sunnyvale, CA). Fluorescent measurements were performed on a Tecan® GENios fluorescent microplate reader (Tecan, Durham, NC).

2.2. Cell culture

Caco-2 cells (passage numbers 40–60) were seeded at a density of 2.5×10^5 cells/flask and grown to 70–90% confluence in DMEM (supplemented with 10% FBS, 100 U/ml penicillin and 100 μg/ml streptomycin and 1% (v/v) NEAA). For transport studies, Caco-2 monolayers were cultured on permeable Transwell® 12 mm diameter inserts with pore sizes of 0.4 μm at a density of 10^5 cells/cm^2 and used after 14–17 days in culture. Cells were cultured at 37 °C and 5% CO_2 atmosphere and the medium was changed every other day.

2.3. Modelling chelation activity of citric acid (CA)

A model to predict free calcium fraction was constructed as described in Supplementary materials. The conditional pKa and citrate (Cit) calcium chelation constant, K, corresponded to previously published values in which similar ionic strengths were applied [12–14]. The model was not corrected retrospectively to take account of the experimentally determined calcium electrode measurements.

2.4. Calcium electrode measurements

A pH-meter (744; Metrohm, Herisau, Switzerland) was fitted with a micro pH-electrode (6.0224.100; Metrohm), a calcium selective electrode (6.0508.110; Metrohm) and an AgCl reference electrode (Dri-Ref-L; World Precision Instruments, Sarasota, FL). All titrations were performed in calcium-free transport media at room temperature (RT). To 20 ml of calcium-free HBSS 300–500 μl CA or Cit (1.5–2 M) was added to yield a final solution of CA/Cit (30 mM) and pH values of 4, 5, 6, and 7.4. The solution was titrated with 40 mM $CaCl_2$ from 0.5 μl to 1310 μl [5×10^{-3}–2.5×10^0] mM $CaCl_2$. Electrical motive force (EMF, mV) and pH were concomitantly monitored during titration. Double standard curves of calcium added to transport media (without CA/Cit), assuming that free calcium concentration was equivalent to total calcium. Titrations were performed at room temperature to improve reproducibility. All solutions were maintained at room temperature for 4 h prior to titration, as the ISE was sensitive to temperature changes. Activity of the ISE was assessed within the pH range of 3–7.4.

2.5. In vitro inhibition of proteolysis

Cit and CA were added to zinc-free transport medium (see "Transepithelial transport studies in Caco-2″; zinc-free) to give Ca/Cit stock solutions a total concentration of 12.5 mM of CA species and a range of pH values (3.5–7.4). Subsequently, enzyme-rich washes were extracted from fasted rat duodenal lumens by rinsing 10 cm fresh duodenum with 10 ml water and instantly freezing the eluate at −80 °C until use. At time point zero, 100 mM recombinant human insulin (Novo Nordisk A/S, Copenhagen, Denmark) was mixed with duodenal extracts and CA stock solutions in a ratio of 1:1:8 respectively, yielding 10 mM insulin and 10 mM CA species. The kinetic study was performed using an autosampler robot

(Gilson 215 liquid handler; Middleton, WI) running 16 separate samples simultaneously. The reaction was sampled at six time points over a period of 120 min. Upon collection of each 20 μl sample, 50 μl trifluoroacetic acid (5% v/v) was immediately added and the samples refrigerated (4 °C) in order to stop the proteolytic degradation. Insulin content was quantified by Acquity UPLC consisting of an autosampler (Model Acq-SM), pump (Model Acq-BSM), column oven (Model Acq-SM) and detector (Model Acq-TUV; Waters, Milford, MA). RP-UPLC separation was achieved by Acquity BEH 1.7 μM C18 1 × 50 mm column (Waters), using a linear gradient of acetonitrile in 0.2 M sodium sulfate, 0.04 M sodium phosphate, pH = 7.2. Peaks were detected by UV absorption at 220 nm and quantified using a human insulin standard. Reaction rates were calculated as the slope of the linear least squares fit to the semi log-plot of concentration versus time, see Eq. (1). All reactions rates were derived by the natural logarithm, *e*.

$$C_t = C_0 \cdot e^{-kt} \Rightarrow \ln(C_t) = -k \cdot \mathbf{t} + \ln(C_0) \quad (1)$$

where t (s) is a given time point, C_t (μM) the remaining concentration of insulin at time point, t; k (min^{-1}) is the reaction constant and C_0 (μM) is the initial concentration of insulin.

2.6. Transepithelial transport studies across Caco-2 monolayers

Filter-grown monolayers were washed with warm HBSS buffer (138 mM NaCl, 5.3 mM KCl, 1.3 mM $CaCl_2$, 0.40 mM $MgSO_4$, 0.44 mM KH_2PO_4, 4.2 mM $NaHCO_3$ and 5.6 mM glucose; pH 7.4) supplemented with 0.1% BSA and 10 mM HEPES and allowed to equilibrate. Transepithelial electrical resistance (TEER) was measured with a chop-stick electrode (Millicell-ERS®, Millipore, Billerica, MA) prior to testing and monolayers with TEER values <600 Ω cm^2 were discarded. The buffer in the respective apical side was then replaced with a solution containing insulin (10 μM) and [^{3}H]-mannitol (0.8 μCi/ml) alone or in combination with CA (5 mM; pH 4.5) or LCC (1 mM), and the monolayers were incubated at 37 °C for 60 min. In some studies, Cit (20 mM; pH 7.4) was added to the basolateral side. Samples containing fluxed insulin from the donor apical side were collected from the basolateral compartments every 15 min for 1 h and human insulin content, was diluted to 100–10,000 ppm (~14–1400 mU/l), and was assayed using ELISA. Flux (J [mol/s]) was determined from steady-state appearance rates of insulin in the receiver fluid. The apparent permeability coefficient, P_{app} [cm/s], was calculated according to Eq. (2)

$$P_{app} = J/(A \cdot C_i) \quad (2)$$

where C_i (mol/cm^3) is the initial concentration of insulin in the donor fluid and A is the nominal surface area:1.12 cm^2 for Caco-2 monolayers and 0.63 cm^2 for intestinal mucosae.

2.7. Preparation of rat intestinal tissue for Ussing chamber studies

Studies were carried out in accordance with the UCD Animal Research Ethics Committee policy, on the use of post mortem animal tissue in research, as well as in adherence to the "Guide for the care and use of laboratory animals", (8th Edition, National Academy of Sciences, 2011. http://www.aaalac.org/resources/theguide.cfm). Male Wistar rats (250–300 g) (Charles River, Margate, UK) were euthanised by stunning and cervical dislocation. The lower jejunum and ileum (lower small intestine) was removed, opened along the mesenteric border and rinsed in warm oxygenated Krebs–Henseleit solution (KH; 118 mM NaCl, 4.7 mM KCl, 2.5 mM $CaCl_2$, 1.2 mM $MgSO_4$, 1.2 mM KH_2PO_4, 25 mM $NaHCO_3$ and 10 mM glucose) according to previous methods [15]. Tissue was pinned with the mucosal side down on a dissection board to expose the external muscularis, which was carefully removed with a size 5 fine forceps. The tissue was then mounted in Ussing chambers with a circular window area of 0.63 cm^2, bathed bilaterally with 5 ml KH and continuously gassed with 95% CO_2/5% O_2 at pH 7.4 and maintained at 37 °C. The transepithelial potential difference (PD, mV) and short circuit current (I_{sc}, μA) were measured across the lower small intestine. The tissue was voltage clamped to zero for 30 s and switched to open circuit configuration for 3 s by an automatic voltage clamp (EVC-4000 amplifier) and Pro-4 timer (both WPI, Hertfordshire, UK). Analogue data were digitised with a Mac Powerlab® data acquisition unit and analysed with Chart® software (AD Instruments, Oxford, UK). Following an equilibration period of 15 min, PD and I_{sc} were measured and TEER was calculated at regular time points from 0 to 120 min using Ohm's law.

2.8. Transepithelial transport studies in rat lower small intestinal tissue

Transport of [^{14}C]-mannitol and FITC-Dextran 4000 (FD4), non-degradable hydrophilic flux marker, was examined across lower small intestinal mucosae mounted in Ussing chambers. Briefly, [^{14}C]-mannitol (0.2 μCi/ml) and FD4 (1 mg/ml) were added to the apical chamber and flux was monitored periodically over 2 h by sampling the serosal chamber (200 μl) every 20 min for 2 h, and apically (200 μl) at time zero, while replenishing with fresh KH buffer at each sampling point. In some studies, CA (30 mM), Cit (30 mM) or LCC (3 mM) were simultaneously added to the apical chamber. Samples containing [^{14}C]-mannitol were mixed with scintillation fluid and read in a scintillation counter (TriCarb 2900TR, Perkin Elmer). Where samples contained FD4, fluorescence was measured in a fluorescence microplate reader (Spectramax Gemini, Molecular Devices) with $\lambda_{ex}/\lambda_{em}$ of 480/520 nm. The P_{app} was calculated according to Eq. (2).

2.9. Statistical data analysis

Statistical analysis was carried out using Prism-6® software (GraphPad, San Diego, USA) using two-tailed unpaired Student's *t*-tests unless otherwise stated. Results are presented as the mean ± standard deviation (SD, unbiased). The level of significance set was $P > 0.05$. Normal distribution of data was in general assumed except for apparent permeability data which elicited a relatively consistent standard deviation (%RSD) and was therefore log transformed prior to statistical analysis.

3. Results

3.1. Prediction of the chelation activity of citric acid (CA)

To estimate the chelation activity of CA, a mathematical model was employed (see Supplementary material). Chelation activity was defined as the average of the apparent formation constants of the individual species of CA, weighted according to their presence and expressed relative to the formation constant of Cit^{3-}, in theory the strongest chelator. The predicted relationship between each form of CA (H_2Cit^-, $HCit^{2-}$, Cit^{3-}) and their calcium chelation activity is depicted in Fig. 1. As illustrated, CA chelation capacity is especially high in the presence of high levels of Cit^{3-} and that is pH-dependent. Above pH 5, total chelation activity corresponds directly to the proportion of the Cit^{3-} species present. At values below pH 5 the fraction of Cit^{3-} is less than 0.1. In this range (i.e., pH < 5), $HCit^{2+}$ becomes relatively more dominant compared to Cit^{3-}. At pH > 5, while $HCit^{2+}$ exhibits some degree of chelation capacity, it is thought to be significantly less than that of Cit^{3-} at the higher pH values. Importantly, calcium chelation activity was

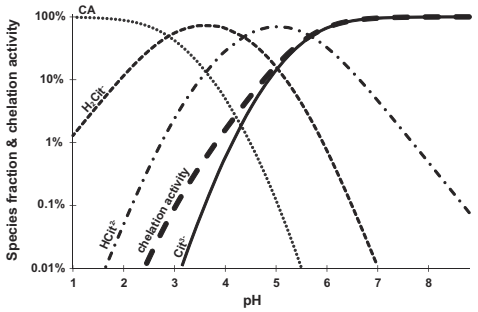


Fig. 1. Calculated CA chelation activity related to the theoretical concentration of the individual formats (ion and salt) at various pH values. The bold dashed line represents the summarised chelation activity of all formats. The model depicted applies to solutions of ionic strength between $I = 0.1–0.6$ M, but does not apply to diluted solutions.

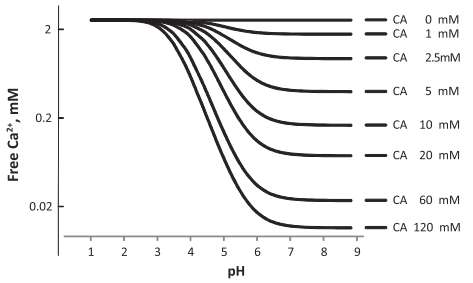


Fig. 2. Calculated free calcium concentration in media of 2.5 mM total calcium (KH buffer) at different concentrations of CA and pH values. The model depicted applies to solutions of ionic strength between $I = 0.1–0.6$ M, but does not apply to diluted solutions.

reduced ~10-fold for each unit of pH within the pH range of 3–6. As shown in Fig. 2, in order to chelate 99% of the total calcium in KH buffer at a pH of 7.4, very high concentrations of 120 mM CA (pH 5.5) or 60 mM CA (pH 7) would be required. For 90% chelation, 120 mM CA (pH 4.2) or 10 mM CA (pH 5.8) would be necessary. The prediction therefore is that at highly acidic local pH values, non-physiological concentrations of CA would be required to substantially chelate calcium, the most likely mechanism to open epithelial tight junctions for permeation enhancement.

3.2. Correlation of 'predicted' versus 'measured' free calcium (Ca^{2+})

CA, H_2Cit^-, $HCit^{2-}$ and Cit^{3-} form a buffer system in which the distribution of the species is a function of pH. Herein, all mentioned concentrations of CA or Cit are absolute and accompanied by a pH value from which the actual distribution of CA-species can be found (Fig. 1).

In order to validate the predictive model, total and free calcium were determined with a calcium-selective electrode following titration of 30 mM CA (at pH values of 4, 5, 6, and 7.4) with 10 μM–3 mM Ca^{2+}. Fig. 3 shows predicted free Ca^{2+} levels versus corresponding experimentally determined free Ca^{2+} levels. Within the range of 0.01–2 mM total Ca^{2+}, very accurate correlations were achieved ($R^2 > 0.98$). According to the conditions observed in Fig. 3, CA/Cit is at least 25-fold in excess compared to calcium. Consequently,

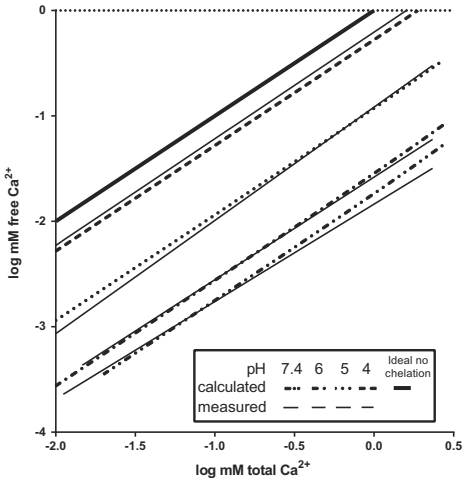


Fig. 3. Free calcium versus total calcium in solutions measured with an ion-selective electrode compared to the calculated free calcium. Solution: HBSS + HEPES (10 mM) + CA (30 mM). Dashed lines are calculated relationships of the total and free calcium. The thin black line — represents the experimentally determined calcium chelation. The thick black line ▬ represents the ideal i.e., no chelation, whereby total calcium = free calcium.

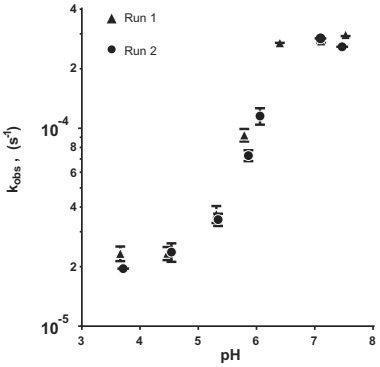


Fig. 4. Apparent reaction constants (k_{obs} natural log) of the proteolysis of insulin by rat duodenal extracts at various pH values in HBSS + HEPES (10 mM) with CA/Cit (10 mM). The proteolysis rate shows an exponential pattern of increase with pH. Insulin proteolysis was fitted as 1st order decay. Data points represent an average of two measurements ± SD from two separate runs. $n = 28$.

the titration slopes of free calcium versus total calcium achieved are highly linear and when plotted on a double-log scale display slopes close to 1. A minor departure from linearity was observed in the lower part of the standard slope (free calcium/mV) which fitted well with 2nd or 3rd order polynomial and was attributed to be a loss of sensitivity of the electrode in its lower detection range.

3.3. pH-dependent degradation of insulin

In the absence of proteolysis inhibition, insulin is readily degraded in the small intestine. However, the activity of most key

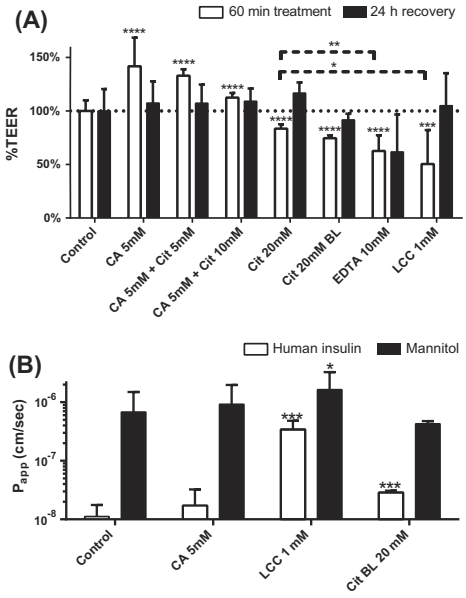


Fig. 5. (A) Effect of CA, Cit and LCC on TEER in Caco-2 monolayers. Normalised change of TEER (%) in monolayers after 60 min treatment with CA 5 mM (pH 4.5), CA 5 mM + Cit 5 mM (pH 5.2), CA 5 mM + Cit 10 mM (pH 5.5), Cit (basolateral) (20 mM, pH 7.4), EDTA (10 mM) or LCC (1 mM). Data are represented as means ± SD, $n = 7–50$. Separate Student's *t*-tests were carried out to evaluate if treatments significantly differed from the control. (B) P_{app} of insulin and [^{3}H]-mannitol across Caco-2 monolayers incubated with CA (5 mM, pH 4.5), Cit (20 mM, pH 4.5), LCC (1 mM, pH 7.4) for 60 min. Separate Student's *t*-tests were carried out to evaluate if treatments significantly differed from the control. Data are represented as means ± SD, $n = 3–12$.

proteolytic enzymes including trypsin, chymotrypsin and elastase is strongly influenced by pH. To investigate the precise relationship between pH and proteolysis rate, an *in vitro* proteolysis assay using native extracts of rat duodenum was performed in the presence of CA/Cit (10 mM). Throughout, the degradation of insulin was characterised by first order kinetics. A plot of insulin degradation versus time is provided in the Supplementary material. At pH 6.5–7.4, the native pH of the small intestine, the reaction rate proceeded much faster than at acidic pH values (Fig. 4). By decreasing the pH from 7 to 4.5 however, the reaction rate was reduced markedly (10-fold) ($p < 0.0001$). Hereafter from pH 4.5 to 4 the reaction rate remained unchanged. At a pH of 3.5, the reaction rate was an order of magnitude lower than that seen at pH 7.4. In order to exclude other potential confounding factors, the influence of both total Cit concentration and the quantity of NaCl added was evaluated. Although Cit (120 mM, pH 7.4) lowered the reaction rate, k_{obs}, two-fold ($p < 0.001$), NaCl (75–250 mM) elicited no significant effect and both the independent factor of NaCl and Cit appeared to have no physiological significance. These data emphasise the dramatic protection against pancreatic peptidases afforded by simply maintaining the pH at 3–4 with a 10 fold lower concentration of CA than that required to chelate calcium.

3.4. Effects of CA, Cit and LCC on Caco-2 monolayers: TEER and P_{app} values of [^{3}H]-mannitol and insulin

Treating Caco-2 monolayers with apical addition of LCC (1 mM) elicited a decrease in TEER of 50% in 60 min compared to that of untreated monolayers (Fig. 5A) ($p < 0.001$). Exposure to various concentrations of Cit (5–20 mM pH 7.4) apically or basolaterally however, resulted in smaller albeit concentration-dependent reductions in TEER and Cit was less potent and efficacious than the well-known chelator and permeation enhancer, EDTA and LCC (Fig. 5A). Surprisingly, CA (5 mM) at pH 4.5 elicited a significant increase in TEER of 50% relative to untreated monolayers ($p < 0.001$). Comparable responses were observed with 5 mM CA in combination with 5 mM and 10 mM Cit (Fig. 5A). Following incubations, transport medium was removed and fresh DMEM introduced; monolayers were then incubated for 24 h to assess TEER recovery. All treated cultures showed a full recovery in TEER, exhibiting comparable values to that of control monolayers after 24 h. Of note, addition of excessive amounts of CA, resulted in a lowering of pH < 4 where TEER recovery was not observed due to irreversible deterioration of the monolayer barrier, as earlier reported [16]. In summary, the data reveal that CA and/or Cit only have marginal effects on TEER, but nothing like the level of decrease which would be expected from an effective permeation enhancer (i.e., >50% TEER decrease).

The P_{app} values of insulin across Caco-2 monolayers showed a significant increase in the presence of LCC (1 mM), generating a 40-fold increase in transport (8×10^{-7} cm/s) compared to basal insulin P_{app} across untreated monolayers ($p < 0.01$) while the P_{app} of [^{3}H]-mannitol was 50% greater than in untreated monolayers (1×10^{-6} cm/s) ($p < 0.05$) (Fig. 5B). In contrast, CA (5 mM) had no effect on the P_{app} of either insulin or [^{3}H]-mannitol. While high concentrations of Cit (20 mM) produced a 2-fold increase of insulin permeability ($p < 0.001$), this was still significantly less than that of LCC and observed only when applied basolaterally. The effect apically was even smaller. Overall, the data confirm that, compared to the dramatic permeation enhancement induced by LCC, CA/Cit are not effective enhancers in Caco-2 monolayers and this is consistent with their nominal effect on TEER compared to positive controls.

3.5. Effect of CA, Cit and LCC on rat lower small intestinal mucosae: TEER and P_{app} of [^{14}C]-mannitol and FD4

The effects of CA, Cit and LCC on the electrophysiological responses of rat lower small intestinal tissue were examined in Ussing chambers. Addition of very high concentrations of Cit (30 mM, pH 7.4) elicited a sustained decrease in TEER over 120 min, which was significantly lower (~20%) than that observed in control tissue (Fig. 6A). In contrast, incubation with CA (30 mM, pH 3) gave rise to a significant, albeit transient increase in TEER (~20%) relative to control during the initial 20 min exposure. Thereafter, TEER values aligned with those of untreated controls. Although LCC did not provoke any significant decrease in TEER, a 2-fold increase in the permeability of FD4 was observed (Fig. 6B). In contrast, neither the TEER reduction elicited by Cit nor the TEER increase observed with CA was associated with changes in mannitol permeability. Permeability of [^{14}C]-mannitol remained statistically unaffected by the various treatments. Overall, these data do not indicate that CA/Cit is an effective permeation enhancer in rat lower small intestinal mucosae, whereas LCC was effective, at least for FD4.

4. Discussion

This work addressed the functional role of CA in formulation-based approaches for the delivery of peptides and proteins via the oral route. Specifically, we examined the interplay between its primary use as a pH-lowering agent and its additional function as a calcium chelator that might cause both tight junction openings and contribute to further inhibition of serine proteases. Modulation of intestinal pH via formulation is an attractive means to

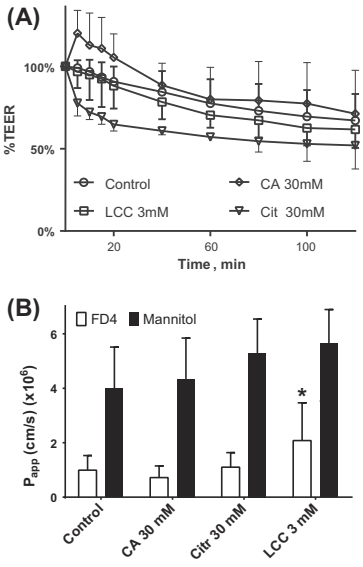


Fig. 6. (A) Effect of CA, Cit and LCC on TEER (A) and P_{app} (B) in rat lower small intestine in Ussing chambers upon mucosal-side incubation with CA (5 mM, pH 4.5), Cit (20 mM, pH 7.4) or LCC (3 mM) for 120 min. (B) P_{app} of FD4 and [^{14}C]-mannitol across rat lower small intestine incubated with CA (5 mM, pH 4.5), Cit (20 mM, pH 7.4) or LCC (3 mM) for 120 min. Data are represented as means ± SD, n = 3–4.

stabilise the protein against enzymatic degradation. Indeed, it has previously been used as a strategy for small intestinal delivery of sCT. Using CA (a prototype organic acid) and LCC (an amphiphilic surfactant), there was significant enhancement in oral bioavailability of sCT in dogs [4,5,17]. However, precise elucidation of the combined effects of CA on protein stability and permeability remains undetermined.

Chelation of calcium is an efficient means by which to modulate tight junction structures. Participation of Ca^{2+} in the establishment of epithelial cell junction networks has been widely demonstrated [18,19]. CA, a hydroxy tricarboxylic acid, is an efficient chelator in its salt form (i.e., citrate), capable of sequestering multivalent cations. Conditional chelation constants for Ca^{2+}–Cit^{3-} and Ca^{2+}–$HCit^{2-}$ are 1880 M^{-1} and 67 M^{-1}, respectively [14,12], while conditional CA $pKa_{1,2,3}$ values applied were [2.80; 4.08; 5.33] [13,14]. Such values are dependent on ionic strength and temperature. Thus, use of the applied model to predict chelation capacity potential is suggested to be restricted to conditions whereby I ranges from 0.1 M to 0.6 M and temperature ranges from 18 °C to 45 °C. KH and HBSS have ionic strengths of $I \sim 0.16$ M. In this context, our proposed model predicts that at pH values of 6–7, and above, optimal chelation activity is observed. Importantly, below pH 6 the apparent chelation constant is reduced ~10-fold for each pH unit. Therefore, at highly acidic pH values the concentration of CA/Cit necessary to chelate vast quantities of calcium dramatically increases. The pH microenvironment generated via release of CA from a pH-lowering formulation likely corresponds to a value of 4.5 or lower [17]. Under these circumstances, CA will primarily dissociate into $HCit^{2+}$ and not Cit^{3-} the predominant chelating species. In this chemical arrangement, its potency as a chelator of calcium ions is considerably reduced. Accordingly, this formulation approach is unlikely to engender significant chelation activity, but will be dominated by a capacity for pH-mediated peptidase inhibition.

Enzymatic degradation of proteins by luminal proteases represents a significant barrier to achieving a therapeutically relevant bioavailability. The ability of pH-lowering formulations to curtail this undesirable aspect of intestinal physiology is its primary attraction for oral peptide and protein delivery. Our *in vitro* investigations revealed that an EC_{50} of proteolysis is achieved at pH 6. However, given that extensive proteolytic degradation arises *in vivo*, in order to effectively protect sufficient quantities of intact protein, substantial serine protease inhibition is necessary. The kinetics studies pertaining to insulin degradation indicate that proteolysis activity exhibits an apparent linear (log scale) decrease down to a value of pH 4.5 where >90% inhibition was achieved. Given that proteins are extensively and rapidly degraded especially in the duodenal and jejunal regions of the GIT, formulations employing acidic inhibition of proteolysis should therefore aim to achieve a local pH of at least 4.5. In this regard, an *in vivo* study in dog found that capsules loaded with the maximal practicable quantity of CA, corresponding to 570 mg in a 680 mg tablet, yielded the highest bioavailability, by reducing the pH to as low as 3 [18]. It should be noted that degradation studies performed *in vitro* represent a system of optimal mixing conditions, which may not be present *in vivo*. Under such circumstances, lowering the pH beyond 4.5 likely ensures a greater acidic expanse within the small intestine, thus representing a local region in which protein degradation is limited for a period of time.

Chymotrypsin is the primary enzyme responsible for the degradation of insulin [2]. Corresponding studies examining chymotrypsin-mediated degradation of casein or denatured lysozyme exhibit a slope (k_{obs} versus pH) which closely resembles that generated in our investigations examining the degradation of insulin in rat duodenal enzyme extracts [1]. Studies suggest that the activity of chymotrypsin can be lowered 2-fold when free calcium is less than 0.05 mM [20]. Similarly, the activity of chymotrypsin has been shown to be lowered when Ca^{2+} is omitted [20] or when available Ca^{2+} is chelated by EDTA [21]. Our investigations however, indicate that the function of CA/Cit as a chelator is most pronounced at high pH values (i.e., >7). Indeed, we have observed that Cit 120 mM lowers the rate constant of chymotrypsin-mediated degradation of insulin by up to 35% at pH 7.4 due to its prevalent chelating action at that pH value (see Supplementary material). However, for pH-lowering formulations, where the local pH at the site of release is <5, its function as a chelator does not contribute to its peptidase inhibitory capacity, which is simply due to moving the pH optimum for serine proteases away from pH 7.4. Therefore, in practice the impact of CA/Cit on reduction of enzyme activity via calcium chelation is significantly less than that due to acidity *per se*. Trisodium citrate (120 mM) will provide a considerable increase to ionic strength possibly further reducing the activity of luminal proteolysis. However, on the contrary, addition of NaCl (75, 120 and 250 mM) did not significantly reduce the degradation rate of insulin by chymotrypsin (unpublished data).

On account of its large molecular weight size, transport of insulin (5800 MW, 8 Å) across the small intestinal epithelium is severely restricted. Indeed, basal permeability results in absorption of doses far below that required to achieve a therapeutic effect [22]. Amphiphilic permeability enhancers, a class of absorption enhancers including LCC and sodium taurodeoxycholate (TDC) promote trans- and paracellular absorption via mild recoverable membrane perturbation and disruption of tight junction complexes [23–25]. For example, permeation of sCT across rat ileal tissue in Ussing chambers can be increased 5-fold and 14-fold for LCC and TDC, respectively [6]. However, their successful incorporation in a pH-lowering formulation requires that they exhibit sufficient

solubility at low pH values. In this regard, LCC satisfies this requirement. As a consequence, it has been widely co-entrapped in formulations designed for acidic inhibition of proteolysis [5,7].

Nevertheless, in the ORACAL study for orally delivered sCT, no amphiphilic enhancer was required. One interpretation was that CA/Cit might be enhancing paracellular transport presumably via modulation of tight junctions [3], in addition to their effect on inhibiting peptidases by acidifying the pH. Supporting this, work by Okada and colleagues [26] revealed that addition of organic acids facilitated vaginal absorption enhancement of the luteinising hormone releasing hormone (LH–RH)-analogue, leuprolide. Furthermore, they observed a weak correlation between this effect and the chelating properties of the acids. However, the influence of pH on such effects was not addressed in this study. Collectively, our investigations indicate that even if intestinal luminal calcium chelation is associated with permeability enhancement, the pH conditions of acidic inhibition of proteolysis (i.e., pH $\leqslant$ 4.5) will extensively reduce chelation activity. Conceivably, the positive results yielded in the Phase III trial with sCT in the absence of a recognised permeability enhancer could be ascribed to the fact that sCT is half the molecular weight of insulin and exhibits a higher baseline permeability [27–31]. Moreover, since sCT is highly potent and marketed nasal versions are associated with no more than 1–3% bioavailability for required efficacy, the hurdles for an oral sCT are therefore much lower and would not require a recognised permeation enhancer once the excipient organic acid performs the pH-lowering role. This would not be the case for insulin where a much higher oral bioavailability would be required for a commercially-viable oral formulation. Consistent with this theory, ileal instillation of insulin with soy-bean proteolysis inhibitor alone did not lower blood glucose in rats, whereas glucose levels were significantly reduced when insulin was co-administered with the bile salt, sodium deoxycholate (DOC) [32], an efficient permeation enhancing agent.

There are numerous reports which demonstrate the effect of CA as a permeability enhancer. CA (1%, pH 7) enhanced nasal absorption of oil/water (O/W) emulsions containing indomethacin by 6.5-fold [33]. However, it should be noted that the nasal clearance/dilution is smaller than that observed in the GIT [34]. Moreover, the CA partitions in water yielding a concentration of 100 mM; thus, the concentration of CA reaching the nasal mucosa is likely much higher than in the intestine.

Insulin formulated with CA (10%, pH 1.72) for vaginal administration effectively lowered blood glucose to the same level as a 10 times lower intramuscularly injection. However, although such high concentrations of CA and low pH can increase permeability, they have also been shown to lead to extensive damage of mucosal tissue [26]. In the context of chronic administration, such adverse effects are undesirable.

Our *in vitro* (Caco-2 monolayers) and *ex vivo* (rat small intestinal tissue) transport studies indicate that the ability of CA/Cit (5–30 mM) to increase permeability of insulin or FD4 is negligible, exhibiting effects which are lower than that of LCC (1 or 3 mM), regardless of pH. Examination of absorption processes *ex vivo* can be confounded by enzymatic degradation. Thus, FD4 being a non-degradable hydrophilic macromolecule-sized compound represents a useful surrogate marker compound for insulin in Ussing chamber-based transport studies, facilitating exclusive examination of the absorptive process alone. The concentrations of LCC employed correspond to those which lie below that which can adversely impact tissue viability over the course of permeation study [25]. In the case of CA, the concentrations used represent those which are anticipated to be found locally in the lumen at the site of release of such oral formulation(s). Compared to excised intestinal tissue, Caco-2 monolayers represent a more fragile model system. Thus, pH should not be reduced beyond 4 [16] and LCC and CA concentrations can likewise be reduced to appropriate functional concentrations which ensure the cell monolayers recover following treatment.

In line with earlier findings for EDTA [19], basolateral application of Cit elicited a more marked decrease in TEER. This differential susceptibility could be attributed to compositional differences in tight junction structures at the apical and basolateral interfaces as previously shown for EDTA [18]. Application to the basolateral side ensures direct exposure of the highly calcium-dependent zonula adherens proteins – structures which are implicated in preservation of monolayer integrity. In this regard, apical addition of Cit will have little impact. However, translation of these basolateral specific effects *in vivo* is not particularly relevant; given the fact that exposure is restricted to the apical membrane of the intestinal epithelia upon release from the dosage form. Collectively, these results strongly suggest that apically applied CA, by means of significant calcium chelation at pH 7.4, is not sufficient to elicit significant augmentation of insulin permeability, notwithstanding its ability to trigger acidic inhibition of proteolysis. Nevertheless, this observation does not preclude the possibility that CA/Cit could potentially chelate calcium in a pH neutral micro-environment below the mucus layer covering the intestinal epithelium.

5. Conclusions

It is evident that the pH range over which CA effectively inhibits proteolysis and that whereby Cit exerts calcium chelating properties does not coincide. At pH 3–4, the capacity of CA to inhibit small intestinal serine proteases is high, and this is due to sub-optimal pH values for those enzymes rather than to calcium chelation. Moreover, *in vitro* and *ex vivo* investigations indicate that the capacity of Cit/CA to exert significant permeation enhancement on human intestinal monolayers and isolated rat small intestinal mucosae is extremely low. While oral delivery of a few potent small peptides including sCT may be successfully achieved in the presence of formulated peptidase inhibitors (e.g., CA) in the absence of permeation enhancers, larger and more impermeable peptides and proteins will require an absorption enhancing agent in the formulation.

Acknowledgments

Signe Beck Petersen (Novo Nordisk A/S) is thanked for her helpful advice regarding Ussing chamber experiments. A calcium ion electrode was kindly provided by Jesper Østergaard (University of Copenhagen). The authors are grateful for the technical assistance of Lisette Gammelgaard Nielsen, Anette Heerwagen, Gitte Hedelund, Trine Moghaddam (Novo Nordisk A/S) and Patrick Kearns (UCD). SH Welling, F Hubálek, UL Rahbek and ST Buckley are either affiliated with or employees of Novo Nordisk A/S. This study was funded in part by Novo Nordisk A/S and Science Foundation Ireland Grant SRC/07/B1154.

Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.ejpb.2013.12.017.

References

[1] A.D. McLaren, E.F. Estermann, Influence of pH on the activity of chymotrypsin at a solid-liquid interface, Arch. Biochem. Biophys. 68 (1957) 157–160.
[2] R. Schilling, A. Mitra, Degradation of insulin by trypsin and alpha-chymotrypsin, Pharm. Res. 8 (1991) 721–727.
[3] N. Binkley, M. Bolognese, A. Sidorowicz-Bialynicka, T. Vally, R. Trout, C. Miller, C.E. Buben, J.P. Gilligan, D.S. Krause, A phase 3 trial of the efficacy and safety of

oral recombinant calcitonin: the oral calcitonin in postmenopausal osteoporosis (ORACAL) trial, J. Bone Miner. Res. 27 (2012) 1821–1829.
[4] J.P.F. Bai, L.L. Chang, J.H. Guo, Effects of polyacrylic polymers on the lumenal proteolysis of peptide drugs in the colon, J. Pharm. Sci. 84 (1995) 1291–1294.
[5] Y.H. Lee, P.J. Sinko, Oral delivery of salmon calcitonin, Adv. Drug Deliv. Rev. 42 (2000) 225–238.
[6] P. Sinko, Y.H. Lee, V. Makhey, G. Leesman, J. Sutyak, H. Yu, B. Perry, C. Smith, P. Hu, E. Wagner, L. Falzone, L. McWhorter, J. Gilligan, W. Stern, Biopharmaceutical approaches for developing and assessing oral peptide delivery strategies and systems: in vitro permeability and in vivo oral absorption of salmon calcitonin, Pharm. Res. 16 (1999) 527–533.
[7] N.M. Mehta, Oral Delivery and recombinant production of peptide hormones Part I: making oral delivery possible, BioPharm Int. 17 (2004).
[8] M.J. Cho, J.F. Scieszka, P.S. Burton, Citric acid as an adjuvant for transepithelial transport, Int. J. Pharm. 52 (1989) 79–81.
[9] M. Grant, M.A. Leone-Bay, Peptide therapeutics: it's all in the delivery, Ther. Deliv. 3 (2012) 981–996.
[10] D.P. Froment, B.A. Molitoris, B. Buddington, N. Miller, A.C. Alfrey, Site and mechanism of enhanced gastrointestinal absorption of aluminum by citrate, Kidney Int. 36 (1989) 978–984.
[11] C.R. Nolan, J.R. Califano, C.A. Butzin, Influence of calcium acetate or calcium citrate on intestinal aluminum absorption, Kidney Int. 38 (1990) 937–941.
[12] R.P. Singh, Y.D. Yeboah, E.R. Pambid, P. Debayle, Stability constant of the calcium-citrate(3-) ion pair complex, J. Chem. Eng. Data 36 (1991) 52–54.
[13] A.J. Meyer, M. Popp, Free Ca2+ in tissue 22+ -selective electrodes, J. Exp. Bot. 48 (1997) 337–344.
[14] J.L. Meyer, Formation constants for interaction of citrate with calcium and magnesium ions, Anal. Biochem. 62 (1974) 295–300.
[15] A.W. Cuthbert, H.S. Margolius, Kinins stimulate net chloride secretion by the rat colon, Br. J. Pharmacol. 75 (1982) 587–598.
[16] G. Borchard, H.L. Lueβen, A.G. de Boer, J.C. Verhoef, C.M. Lehr, H.E. Junginger, The potential of mucoadhesive polymers in enhancing intestinal peptide drug absorption. III: Effects of chitosan-glutamate and carbomer on epithelial tight junctions in vitro, J. Control. Release. 39 (1996) 131–138.
[17] Y.H. Lee, B. Perry, S. Labruno, H. Lee, W. Stern, L. Falzone, P. Sinko, Impact of regional intestinal pH modulation on absorption of peptide drugs: oral absorption studies of salmon calcitonin in beagle dogs, Pharm. Res. 16 (1999) 1233–1239.
[18] M. Tomita, M. Hayashi, S. Awazu, Absorption-enhancing mechanism of EDTA, caprate, and decanoylcarnitine in Caco-2 cells, J. Pharm. Sci. 85 (1996) 608–611.
[19] A.B.J. Noach, Y. Kurosaki, M.C.M. Blom-Roosemalen, A.G. de Boer, D.D. Breimer, Cell-polarity dependent effect of chelation on the paracellular permeability of confluent caco-2 cell monolayers, Int. J. Pharm. 90 (1993) 229–237.
[20] T. Sasaki, H. Kise, Increase of catalytic activity of α-chymotrypsin by metal salts for transesterification of an amino acid ester in ethanol, Biosci. Biotechnol. Biochem. 61 (1997) 1196–1197.
[21] J.P. Bai, L.L. Chang, Transepithelial transport of insulin: I. Insulin degradation by insulin-degrading enzyme in small intestinal epithelium, Pharm. Res. 12 (1995) 1171–1175.
[22] G.P. Carino, E. Mathiowitz, Oral insulin delivery, Adv. Drug Deliv. Rev. 35 (1999) 249–257.
[23] A.C. Chao, M.T. Taylor, P.E. Daddona, M. Broughall, J.A. Fix, Molecular weight-dependent paracellular transport of fluorescent model compounds induced by palmitoylcarnitine chloride across the human intestinal epithelial cell line Caco-2, J. Drug Target. 6 (1998) 37–43.
[24] J.A. Fix, K. Engle, P.A. Porter, P.S. Leppert, S.J. Selk, C.R. Gardner, J. Alexander, Acylcarnitines: drug absorption-enhancing agents in the gastrointestinal tract, Am. J. Physiol. 251 (1986). G332-40.
[25] E.L. LeCluyse, S.C. Sutton, J.A. Fix, In vitro effects of long-chain acylcarnitines on the permeability, transepithelial electrical resistance and morphology of rat colonic mucosa, J. Pharmacol. Exp. Ther. 265 (1993) 955–962.
[26] H. Okada, I. Yamazaki, Y. Ogawa, S. Hirai, T. Yashiki, H. Mima, Vaginal absorption of a potent luteinizing hormone-releasing hormone analog (leuprolide) in rats I: Absorption by various routes and absorption enhancement, J. Pharm. Sci. 71 (1982) 1367–1371.
[27] H. Ichikawa, N.A. Peppas, Novel complexation hydrogels for oral peptide delivery: in vitro evaluation of their cytocompatibility and insulin-transport enhancing effects using Caco-2 cell monolayers, J. Biomed. Mater. Res. 67A (2003) 609–617.
[28] A.C. Foss, N.A. Peppas, Investigation of the cytotoxicity and insulin transport of acrylic-based copolymer protein delivery systems in contact with caco-2 cultures, Eur. J. Pharm. Biopharm. 57 (2004) 447–455.
[29] K.H. Song, S.J. Chung, C.K. Shim, Preparation and evaluation of proliposomes containing salmon calcitonin, J. Control. Release. 84 (2002) 27–37.
[30] P.S. Hiremath, K.S. Soppimath, G.V. Betageri, Proliposomes of exemestane for improved oral delivery: formulation and in vitro evaluation using PAMPA, Caco-2 and rat intestine, Int. J. Pharm. 380 (2009) 96–104.
[31] R. Shah, M. Khan, Regional permeability of salmon calcitonin in isolated rat gastrointestinal tracts: transport mechanism using Caco-2 cell monolayer, AAPS J. 6 (2004) 36–40.
[32] M. Kidron, H. Bar-On, E.M. Berry, E. Ziv, The absorption of insulin from various regions of the rat intestine, Life Sci. 31 (1982) 2837–2841.
[33] E. Karasulu, A. Yavasglu, Z. Evrensanal, Y. Uyanikgil, H.Y. Karasulu, Permeation studies and histological examination of sheep nasal mucosa following administration of different nasal formulations with or without absorption enhancers, Drug. Deliv. 15 (2008) 219–225.
[34] A. Allen, G. Flemstrom, Gastroduodenal mucus bicarbonate barrier: protection against acid and pepsin, Am. J. Physiol. Cell Physiol. 288 (2005).

CHAPTER 3
Introduction to tools of supervised machine learning

3.1 Supervised machine learning to predict and to learn

Supervised machine learning covers regression, classification and probability estimation models built on labeled data. Regression is to predict scalars (numbers), classification to predict class membership or lastly to rather predict a probability distribution across classes.

The direct motive of supervised modeling is to predict a certain target information, that is otherwise expensive/tedious to measure, or only reveal it self in the future, or the measuring is invasive and will destroy the object of interest. The target is predicted by learning a simple or perhaps complex relationship between easy accessible features and the target from a labeled training data set. When a useful relationship has been established with a model, target predictions can be made for a unlabeled data set without the target.

An indirect motive of supervised machine learning is to elucidate a general relationship between features and targets. One example of an indirect motive is when modeling the contraceptive method choice reference data set [Lic13], see also results part of forest floor article in Section 5.3. Here, +1000 Indonesian married women had answered a questioneer on contraception and socio-economic status. To build a model to accurately predict contraception method choice based on 10 questions on socio-economic status was never the actual motive. It has little practical use to ask 10 questions to only estimate one answer. Why not just ask the right question at first? However, the structure of an accurately predicting model can be an useful empiric proposal for a general relationship. Scientifically, the next step is to form testable causal link theories inspired by the captured empiric relationships.

A typical labeled data set, is organized as a data table with one column with desirable target information and a series of columns with feature information. Every row is an independent observation of one target and some features. A practical example is the public abalone data set. Here, a marine biologist may be interested in estimating the age of abalones (shellfish). However, determining age is tedious and requires to sacrifice each specimen to study the broken shell under a microscope after

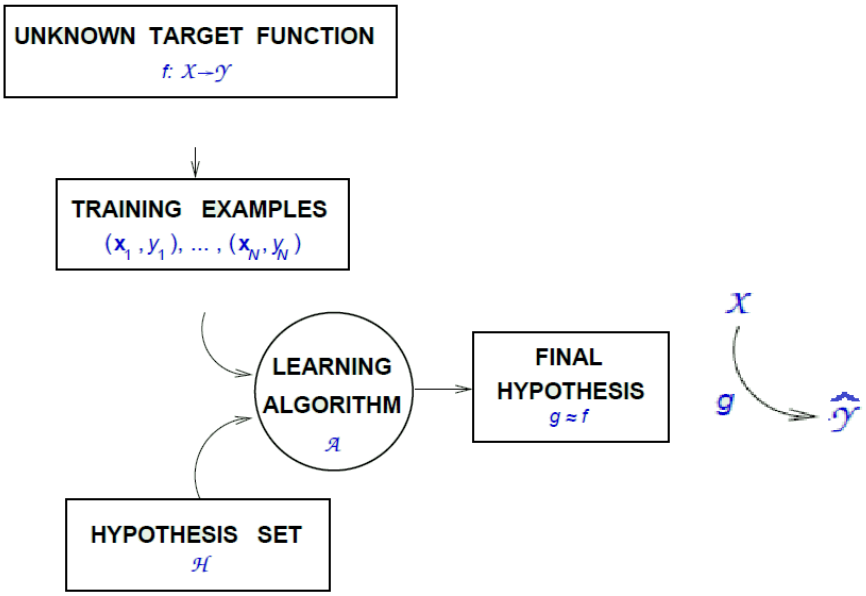


Figure 3.1: We can imagine an unknown target function f in nature, that generates the N observed training examples X_i y_i for $i \in 1, 2, ..., N$. With a hypothesis set H, that is the set of hyper parameters we will try for A. Each hypothesis will yield a model. With model selection one final hypothesis and the resulting model g is chosen. The model g is a function that mimics f. With g we can start to predict how f behaves (direct motive), or secondary evaluate the structure of g to both challenge the validity of g and our current beliefs of f. The figure is copied from a set of Caltech lecture slides [Abu12]. .

chemical staining. To measure the size and weight and to observe the gender is in contrary easy [Lic13]. Therefore the marine biologist can use a supervised regression model to learn a relationship between morphology and age, and use this relationship to predict effortless the age of new specimens. Each row of the data set will be one abalone randomly sampled from the ocean, a target column (y) will describe the age of each abalonem, and feature columns will describe other features measured.

3.1.1 Univariate regression

Perhaps a single feature such as weight ($x_{\cdot 1}$) would be almost perfectly linear related to age ($y_{\cdot}$). In such case uni-variate linear regression (ordinary least squares) would be a sufficient model. Where $\hat{y} = b_1 x_{\cdot 1} + b_0$, and where b_1 and b_0 are chosen to minimize a loss function evaluating the training error. Let $x_{\cdot 1}$ be a vector of weight measurements and let $y_{\cdot}$ be a vector age. Both $x_{\cdot}1$ and $y_{\cdot}$ are of length N, the sample size of the training set, and the elements are enumerated from 1 to N^{th} observation by i, such that y_i is the age of the i^{th} abalone, and x_{i1} is the weight.

Perhaps the absolute abalone growth rate increases with age, and therefore the age of larger abalones are in general overestimated. By plotting the relationship and inspecting the residuals it may be evident, that linear fit is not optimal. To overcome this, the model may manually be expanded with a quadratic term such that $\hat{y} = b_2 x_{\cdot 1}^2 + b_1 x_{\cdot 1} + b_0$. Thus in this manual approach, first a linear fit was made, and by inspecting the residuals it was obvious that transforming the weight measurements by non-linear quadratic transfer function would improve the linear relationship.

3.1.2 Multiple linear regression and interaction terms

The user may now start to include several transposed features and interaction terms. There is no direct limitation in linear regression to not model non-linear relationships and interactions terms. All these terms just have to be stated manually. For a small set of features, and where the goodness-of-fit of a linear model is already fair, it is easy to inspect the residuals to observe, how a linear fit may be inadequate. Hereafter one can specify some useful transfer functions and interaction terms to obtain an even better model. However, when the number of features become large, it will become tedious to manually state terms in the model. If the feature relationship is fairly complex (many interaction terms and non-linearities), then no two-dimensional residual plot can reveal what transfer functions to use. A fast fix is to state a model with several transfer functions for each feature and interactions for each pair of features.

One limitation is the degrees of freedom. In a linear ordinary least squares model, if the number of parameters to fit in the model outnumbers the number of observations, there is no longer one unique fit with a minimal loss function score, but rather a subset of solutions all with a constant error. It takes only an offset and a slope to connect two points with a line, or an offset and two slope coefficients to describe

a plane connecting three points. Likewise, 18 points can always be connected by an 17-dimensional hyper plane plus an offset. The accuracy of a model should not be evalauted by training error, especially when then number of parameters are close to as many observations and/or if the observations are noisy.

> "With four parameters I can fit an elephant, and with five I can make him wiggle his trunk" - J Neumann [Wik16b]

Psychologist and economist Daniel Kahneman noted when working with notoriously noisy data from psychology tests, that multiple linear regression models explained the training data well, however the established model predicted poorly future results and could not be reproduced. Each subject (person) would be exposed to a number of sub tests outputting score values. Each sub test was designed to be related to e.g. leadership performance by some measurable definition. The test scores could be combined to predict future leadership for interviewed candidates with a multiple linear regression model, however Kahneman preferred to simply use the summed total score as a predictor, thus giving each sub test the same weight. He found this approach more accurate and reproducible than multiple linear regression [Kah11].

The individual psychological tests aimed to reflect the same target and these were likely linearly related (collinearity). Moreover, the psychological observations were inherently noisy and the training set size modest. This was a recipe for a poor overfitted multiple linear regression model. The best ordinary least sqaures fit may be a spurious ratio between the sub tests, where some tests even are accredited by a negative coefficient, although having a positive linear relationship to the target.

Kahnemans practice of forcing all tests to have the same coefficient size is in practice the same as regularization, although a fairly crude version. A more elegant regularization method is the elastic net (EN). EN regression allow to do anything between classic multiple regression and a very strong regularization where all coefficient tend to have low values of equal size. Regularization tend to even the dependency on all features, unlike using only one feature greedily. Intuitively, to rely more evenly on different information sources under noisy conditions will lead to more robust models. Secondly increased regularization tend to prevent finding some very specific ratio between correlated features to explain the target, but rather use the weighted average of correlated features to predict.

The elastic net coefficient estimation Elastic-net do not only minimize squared target residuals but also the squared coefficents (L2/lambda penalization) and absolute coeeficients (L1/alpha penalization). An elastic net algorithm will likely use a convex optimization search to find the coefficients that minimizes this loss function,

$$\hat{\beta}_{elastic_{net}} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{N} (y_i - \hat{y})^2 + \sum_{j=1}^{p} \lambda(\alpha\beta_j^2 + (\alpha - 1)|\beta_j|) \quad , \tag{3.1}$$

where N is the number of observations/rows, and $\hat{y}_i$ is the i^{th} prediction $\hat{y}_i = \beta_0 + \beta_{1:p} x_i$. and p is the number of features/columns [FHT01]. The λ and α parameter are specified before the convex optimization search. λ controls the overall

penalization of coefficient sizes. α is a value between 0 and 1. When 1, only the sum squared coefficents is penalized ($L2$), whereas for $alpha = 0$ only the sum of absolute coefficients is penalized ($L1$). Any value in between is a linear combination. Strict $L2$ penalization can never drive a coefficient entirely down to zero, as squaring a small number below 1 makes it even smaller. With the $L1$ penalization it is possible to drive coefficients down to zero, thus effectively omitting them from the model. $L1$ penalization can be used for feature selection [FHT01].

The elestic net estimator allow to introduce two types of gradual regularization driving down the effective number of parameters in the model. Elastic net can be seen as a learning algorithm A that takes a training set T $((x_1, y_1), ..., (x_N, y_N))$ and a set of hyper parameters H (α and λ) as input. The leaning algorithm will then output a model function $g = A(T, w)$. The model function can make predictions $g(x_i) = \hat{y}_i$. The hyper parameters α and λ should be chosen to obtain the lowest future prediction error and not necessarily to obtain the lowest training error, as the ordinary least squares fit do. To estimate what model will work best with future predictions cross-validation is used.

3.2 Cross-validation

There are infinite models that can explain a given training set perfectly. Especially a sufficient complex model, can always connect the dots in some way. In hind sight virtually anything can be explained. Cross-validation is a simulation of how well a given model will predict unseen data.

There are alternatives to cross-validation such as the 'Akaike information criterion' which will weight training set prediction accuracy against a penalty for model complexity. However the Akaike information criterion do not translate well to non-linear algorithmic models where the effective degrees of freedom is difficult to estimate.

Cross-validation is performed for two purposes. First, model selection, that is to compare our set of hypotheses to choose the most promising approach. In practice that is to select the most useful learning algorithm and optimize the hyper parameters and perhaps perform some feature selection. Secondly, to estimate the prediction error for future data generated by the same underlying function f for a given built model g [FHT01].

3.2.1 Segregation

The simplest cross-validation scheme is segregation. The data set of data examples is randomly divided into e.g. 50% training observations, 25% validation (aka. calibration) and 25% test observations. The learning algorithm uses the training set to estimate model parameters, the validation set is used to choose the hyper parameters of learning algorithm and the test set is used to estimate the future prediction error [FHT01].

3.2.2 Unbiased estimation of prediction error

To perform model selection and to estimate prediction error is often the same procedure. However, especially if the set of model hypotheses is large, there may be some hypotheses, that only by chance produced a good model fit of the validation set. Selecting the hypothesis with lowest validation error may be the best strategy, however the low validation error of this hypothesis is an inseparable combination of skill plus luck. Where the skill component remains, the luck component is inherently non reproducible and will regress towards the mean [Kah11]. Thus, we don't know exactly why a given hypothesis turned out to be the best to fit the validation set. The best hypothesis will subsequently most likely perform worse on the test set than on the validation set, as luck is difficult to reproduce. For a large hypothesis set, where all models have a low explained variance, we should expect the luck component to be substantial. The *Inverse gamblers fallacy* would be the extreme cases, where there is no skill and only luck. A gambler may erronously believe that she should keep playing, as she has been in luck the last couple of rounds. An extended version of the *Infinite monkeys theorem* explains the problem well. Theoreme states, that in a room sit infinite monkeys and type randomly on type writers, one monkey will eventually have written the complete works of Shakespeare. Would you hire this monkey expecting it to write more brilliant works in the future? [**wiki:gamblers**; **monkeys**]

This problem is very related to family-wise error of multiple hypotheses testing in classical statistics and e.g. the Tukey and Bonferoni corrections. However no such corrections exists for general machine learning.

In conclusion when ever making a model selection or a parameter estimation based on a set of data, the same data set cannot provide an unbiased estimation of the model performance. A new cross-validation must be wrapped around the current procedure. Lastly also within machine learning literature, publication bias inevitably exists. An article including a model procedure with a good final test score, is more likely to be submitted and accepted. But as a component of luck may have had an influence, the published test set performances, will in average regress towards the mean, if the study were to be reproduced. The test score is valuable as an unbiased estimation of the future model prediction error. However paradoxically, if we act on this test score, it is no longer unbiased. How can a test score be valuable, if we cannot use the information? There is an obvious conflict of interest. Should a researcher publish the model with the better test score, or ideally the model with the most unbiased test score. Machine laerning competions offers a good solution to resolve the conflict of interest. Here an extra and final test set will only be revealed after the competition is over. Then each competitor can only focus on building the best model to predict future data. Cross-validation will be used, but only to guide the competitor to make optimal choices. If the competitor produces over optimistic internal cross-validation, then she is only cheating her self [Ser15].

3.2.3 Independent and identical sampling

Any prediction error estimation from a cross-validation is based on the assumption of independent and identical sampling. Returning to Figure 3.1, it is assumed that that the distribution of training examples, test examples and future examples to predict are identical distributed. That is that X_{test} and X_{future} are drawn from the same larger X_{all} distribution. Also, the sampling procedure recording a sequence of observations must be independent [Le 10]. E.g. if there is a strong auto-correlation with an oscillation interval larger than the sample size, a sample average may not be very close to the population average, as all observations happen to be recorded in e.g. a sampling period with only observations above average. Furthermore the low sample variance will lead to an over confident estimation of model accuracy.

3.2.4 Transient or constant underlying systems

Typically, cross-validation assumes that the underlying true function f, see Figure 3.1 remains unchanged. When the underlying true function is a physical rule of nature, no drift of f is expected. However if the underlying function is drifting, the expected prediction error are probably over optimistic. Within economics, this problem is known as the Lucas Critique [Wik16a]. It is naive to believe economical models successfully can predict future events from historical data. In a developing society, events will be unprecedented due to new technology and transient social constructs. The rules of the game may simply be changing too fast. To some extend modeling strategies in transient systems, can be validated with 'back-testing', that is a cross-validation over time testing an online learning system.

It can be difficult to decide if new observation diverge from the predicted, due to noise or a drift in the underlying system f. If one expects the underlying system to both be drifting and to be noisy, it is a tough situation. Any small training sample will have a substantial sampling error, and any large sample will be sampled across a period where the underlying function f already have drifted.

3.2.5 Defining training and prediction error

The estimated training or prediction error is based on a loss function to score how much different a prediction is to the true target. Also when all predictions have been scored, an aggregation method is needed to summarize the expected performance. Typically the average is used as aggregation method, and for regression, squared residuals is used as the loss function. Therefore, the training and prediction error is

$$err = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}(x_i)) \quad , \tag{3.2}$$

where i iterates a set of 1 to N test or training observations and where the squared difference $L(a,b) = (a-b)^2$ is the loss function, L. Both for regression and classification

it is worth to consider what aggregation rule and loss function to use, such that the trained model is the most useful. E.g. if a forecaster is financially penalized proportionally to the forecast error, then the loss function should certainly also be absolute error $L(a, b) = |a - b|$. If the forecaster will be fired if any prediction is worse than a certain margin, one may choose only to summarize the top or bottom outliers to focus on reducing these. Also the prediction error may be far from constant across the feature space X, and therefore would an unconditional aggregate, such as averaging be a crude estimate. Instead the prediction error for each prediction of X_i should be estimated by a error function f_e, such that $X_i \; err_{X_i} = E[L(y_i, \hat{y}_i)|X_i] = f_e(X_i)$. For classification the loss function should reflect the actual cost of false positives / false negatives or the loss function should maximize a reward. Classification problems can be redefined as probability estimation problems, applying a proper score rule as loss function. Probability estimation is a more elegant way to maximize a defined reward, than classification. Classification can only apply one fixed threshold of certainty, when assigning predicted classes to test observations [Har+84]. At best the model selection and test could be based on a simulation reflecting, what would happen if the predictions were utilized. This is known from back-testing e.g. stock trading robots. The robot will e.g. provide predictions of the coming days price fluctuations over a year, and a trading simulation will measure how well a given strategy can utilize these predictions to increase portfolio net worth and to minimize the risk (variance). Then the loss function and aggregation method is coupled to exactly, what the user intend to achieve. However, it may be difficult to estimate the outcome of good and poor predictions, when in opposite to stock trading the rules of the game is unknown. How the accuracy of the predictions of permeation enhancers provided in this thesis will affect the company Novo Nordisk is really hard to say, and therefore it is difficult to devise a test metric that optimizes wealth or happiness. A good guess would be, that most often the permeation enhancer predictions will mean nothing at all and for rare wild cards, a whole lot. This is based on the notion that most candidates in drug development will fail, and only few candidates will drive the revenue. It is also possible to scale the target information in a way such that a standard loss function would be useful. In Section 2.1.3 the target Tpot could have been transformed before passing it to a machine learning algorithm.

3.2.6 Other cross-validation regimes

Cross-validation by segregating the data set into one training, one validation and one test set is easy to implement. However, the random segregation itself contributes to the uncertainty of the estimated prediction error. This can be avoided by 10 times segregating the data set, building the model and estimating the prediction error. The average of the repeated estimated prediction errors will have a lower variability. n-fold cross-validation is a stratified version, where each observation is selected once for the test set and $n - 1$ times for the training set. n-fold cross-validation is more efficient than random segregation to lower the variance of segregation. When the user both

need to perform a grid search in the hypothesis space, select the best model based on prediction of the validation set and compute an unbiased prediction error the cross-validation regime may become complicated. Krstajic *et al* provide a useful review for various cross-validation regimes [Krs+14]. After model selection and prediction error estimation, the learning algorithm is executed a last time with the favored hyper parameters and the full data set as training set. The resulting model will be used in practice. To include all observations will increase the model performance.

3.3 Algorithmic models

So far in this introduction multiple linear regression have worked well even for non-linear problems, if the user can come up with transfer functions that make the feature relate linearly to the target. However, for multivariate data set, where an initial multiple linear regression model has a low goodness-of-fit, it can be difficult to decide from residual plots by each feature, what series of transfer function and interaction terms would improve the model fit. Algorithmic models are loosly termed as model structures rather defined by the algorithm, than the output mathematical expressions. However, models can ultimately be expressed in a mathematical expression, it may just be a very long and incomprehensible one. Popular algorithm models presently are radial basis support vector machine (RB-SVM), random forest (RF), gradient boosting trees (GBT) and neural nets (NN). An introduction to each type of model is beyond the scope of this thesis and especially neural nets has today a large field special tailored for the purpose sub-models, such as convolutional nets, recurrent nets, deep layered nets or any combination of these.

From a users view point, all these algorithmic learners have in common, that features can be inputted without specifying transfer functions, and the resulting model will to some extend automatically: Fit non-linear relationships, fit interactions and provide regularization plus robustness to outliers. A given algorithmic models should be considered instead, whenever this seem to out perform linear regression in cross-validated prediction error.

In ordinary least squares regression, there are no hyper parameters, only one fit with the lowest training error. In the elastic net fit, λ and α could be adjusted and for each set of these hyper parameters named w, there would be a solution of parameters / coefficients. In contrary to linaer regression, the model parameters to fit are not necessarily stated in advance. New ones can be created in the fitting process. Each algorithmic learner will have a set hyper parameters, and if these are calibrated poorly, the resulting model fit will be inferior. Unlike manually stated linear regression models, the parameters of algorithmic tend not only to additively influence the prediction. The parameters also influence each other such that the overall learning algorithm is very flexible. The downside, it is much harder to interpret one parameter, when it must be understood in the context of a perhaps complicated network of parameters.

The random forest algorithm was extensively used throughout this thesis. Probably either of above mentioned methods RBF-SVM, GBT and some versions of NN, could have replaced the use of random forest in this thesis. The prediction error difference, may even have been so small, that it is only in competitions, it would matter which learner to pick. In a competition situation, one may also choose to use all the learners and combine several trained models in a joint ensemble.

The random forest algorithm often perform quite well with the one default set of hyper parameters, whereas GBT have no recommended default set of hyper parameters, but likely there is a smaller subset of hyper parameters that do perform better than random forest. From a user perspective, GBT can provide a slightly improved performance on the expense of application time and run time. In order to calibrate the GBT learner well, for a given data set, it is likely necessary to perform a substantial grid search and thus in fact run the learner algorithm several times. With formalized grid search and cross-validation tools such as caret package, few extra lines of code are required [Kuh15].

Although the trained models are stated differently, a good rule of thumb is, that when the various models achieve very comparable cross-validation results their effective model structures must match. In appendix Section A.2.5 there is included a simulation to illustrate how similar a RF and RBF-SVM fit are for interpolation. However, when then RF and RBF-SVM model is used for extrapolation outside the proximity of training observations, the model structures heavily disagree. Figure 3.2 depicts how similar the effective model structures are in proximity to data points, and how different extrapolated predictions are.

3.4 Random Forest

3.4.1 bagging

Random forest is a bootstrap aggregated ensemble model meaning it is a collection of independently trained sub models, where each model is trained on a bootstrapped sample of the training set. The prediction error of bagged ensemble models can be lower than the average prediction error of the individual models.

Bootstrapping the training set means to sample a random set of observations. This bootstrap set has the same size as training set and could include every observation of the training set. This would be very unlikely. By default (uniform distribution, sampling with replacement) the chance, that an observation is not included, is approximately 37%. The chance an observation is sampled at least once is %63, see Appendix A.2.9 for an exact sampling description.

For random forest the same learner, decision tree, is bagged perhaps 500 times. The ensemble prediction is typically the average vote of the ensemble. Besides random forest, bagging is an easy way to combine different model approaches to achieve a perhaps lower prediction than any model approach alone. As bagging independently train each model, the computation can easily be distributed across multiple CPUs or

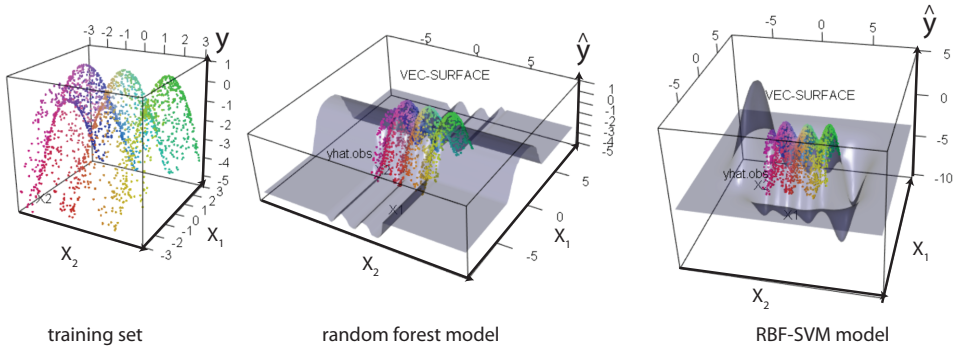


Figure 3.2: Left plot, a training set has been simulated from $y = sin(x_1\pi)\frac{1}{2}x_2^2$ where x_1 and x_2 are drawn from an uniform distribution within $[-3; 3]$. Middle and left plot, RF and RBF-SVM have respectively been trained on the data set, providing two completely similar model structures within the proximity of training data. The the shape of the RF and RBF-SVM model structures heavily disagree outside training set area. See the simulation code in A.2.5..

nodes in a server cluster. Appendix A.2.8 provides an elaborate parallel implemented example, where logistic regression is combined with random forest to form a new and more accurate ensemble.

When only bagging multiple linear regression models, the acquired ensemble can actually be reduced to one linear regression model where each coefficient is the average of the corresponding coefficients in each sub model. Hereby bagging provides a regularized model similar to increasing the *lambda*-parameter of elastic net and where $alpha = 1$. For ensembles of non-linear models, there are no obvious average model. However the bagging still provide regularization, offering a more robust model estimation, than a single learner model alone.

3.4.2 Introduction to decision trees

The random forest algorithm is an improvement of the classification and regression tree algorithm (CART) [Bre+84; Bre01]. The individual decision trees of random forest are built by applying a collection of steps recursively.

1. A collection of observations is called a node and a node has a prediction, that is the average target. A node is terminal, either as there are no valid features left to split by or if a stop criterion has been triggered. Stop criteria are: Exceeding max allowed nodes in tree (infinite by default), reaching minimal allowed node size (5 for regression, 1 for classification), or if all targets are the same.

2. For any intermediary node, a split rule is defined by a break point. For a break point for a given numeric feature, any observation having a feature value below or equal is forwarded to a left daughter node and otherwise to a right daughter node. Every possible split will be tried. For categorical features, any unique separation of categories into two daughter nodes will be tried.

   A loss function evaluates and select the best split. By default the loss function is the sum of squared residuals for regression and a variation of gini-index for classification. The regression loss function choose the split, that maximizes the variance across daughter nodes and minimizes intra-daughter node variance. The classification loss function choose the split that produces two nodes most unlike a uniform class distribution. The loss function implementation is discussed in detail in Appendix A.2.11.

3. For each daughter node restart the procedure from step 1.

In Figure 3.3 a simple regression tree is built. Left side shows a geometrical representation of the model structure. Right side shows a graph representation of the same model. The nodes are enumerated as these are created. For a CART tree, the training set and root node is not the same, as all observations is used to build the tree. In a random forest ensemble, all trees share the same training set. The training observations passed to each root node of each tree is a bootstrap of the training set. The root node is split into node 2 and node 3. The recursive algorithm proceed with daughter node 2 first (left). If left node is terminal, the algorithm will return to parent node and proceed to right daughter node. If right daughter node is terminal, the algorithm will return to parent-parent node. Therefore the algorithm will visit the nodes in this sequence: root - 2 - 4 - 2 - 5 - 2 - root - 3 - 6 - 3 - 7 - 3 - root.

Although decision trees are a collection simple univariate binary step functions, the stacked trees can ideally estimate almost any structure including interactions. The tree example of Figure 3.3, do in fact contain an interaction where high X_1 give a higher terminal prediction, when X_2 is also high, and the opposite if X_2 is low. In appendixA.2.4 there is a simple simulation verifying that random forest can fit interactions.

Whereas a two-way interaction term in a multiple linear regression is simply defined as $\hat{y} = \beta_0 + \beta_{12}x_1x_2 + ...$, the interactions of decision trees are not easy to comprehend when represented as an equation of stacked step functions. It is neither easy to comprehend the interactions by a graph representation, especially when the trees have more than 50 nodes. This has led to that interactions are only vaguely defined and discussed for random forests [Bou+14]. In Section 5.2, I argue product interaction terms as known from linear regression, cannot represent any type of interactions in random forest models. Also, I provide my geometrical definition of interactions in random forest models.

Although random forest is a powerful learning algorithm to estimate interactions for a series of data problems, there is a limit for how complex an interaction the random forest learning algorithm can fit. Although a given tree with a very high

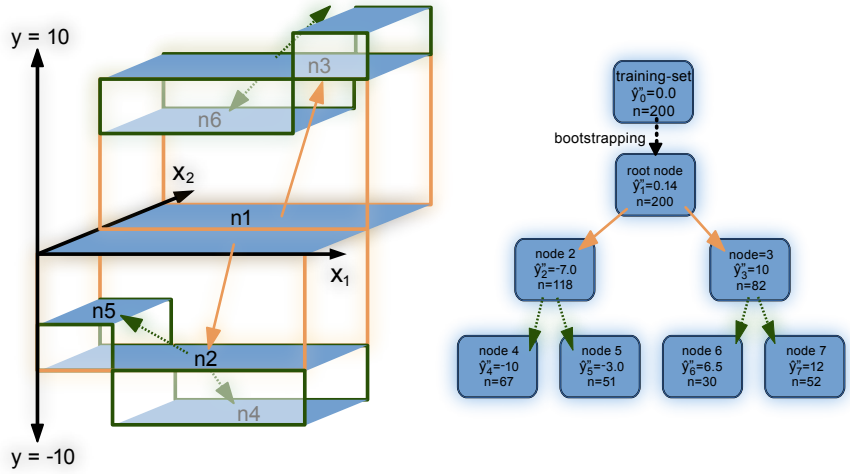


Figure 3.3: Two representations of the same model tree. Left, a geometrical representation of the model surface. Right, a graph representation of the same tree model. Nodes are enumerated as created by the learning algorithm. $\hat{y}_j''$ are the node predictions. n is the node size. X_1 and X_2 axes are two numerical variables and y axis the prediction axis. This figure is copied and modified from [Wel+16]..

number of nodes could potentially approximate any model structure, the random forest algorithm cannot necessarily grow that tree, because the loss function consider only the immediate best univariate split. In Section 5.3.1, *Supplementary materials for "Forest Floor Visualizations of Random Forest - 1.3 Shallowness of Random forest"* there are provided examples of high-order interactions, that random forest cannot fit. This is the reason why neural networks will supersede random forest when a very complex model structure is needed.

3.4.3 Regularization in random forest

Single fully grown decision trees have a low bias, meaning the learning algorithm is flexible enough to build a model to approximate almost any function, if there is enough training observations and the noise level is low. However as mentioned, decision trees cannot fit high-order interactions, see Section 5.3.1. If the function to approximate is not a step function itself, but some rounded structure, it will take high number terminal nodes to approximate. Similarly it takes a high number of square pixels to draw a seemingly round circle on a screen. To obtain the lowest bias, the decision tree

must be fully grown. Unfortunately, when every terminal node prediction is made up by only one observation, the model will be noise labile. For the rare occasion of noise free data sets, a single fully grown decision tree can work well. In Figure 3.4, to the left a noise free data set simulated by the function $y = (X_1^2 + X_2^3)^2 - (X_1^2 + X_2^3)$, where X_1 and X_2 are drawn from uniform distributions. To the right, a decision tree with one observation per terminal node. Overall the decision tree approximate the underlying function well, the cross-validated explained variance is 96%.

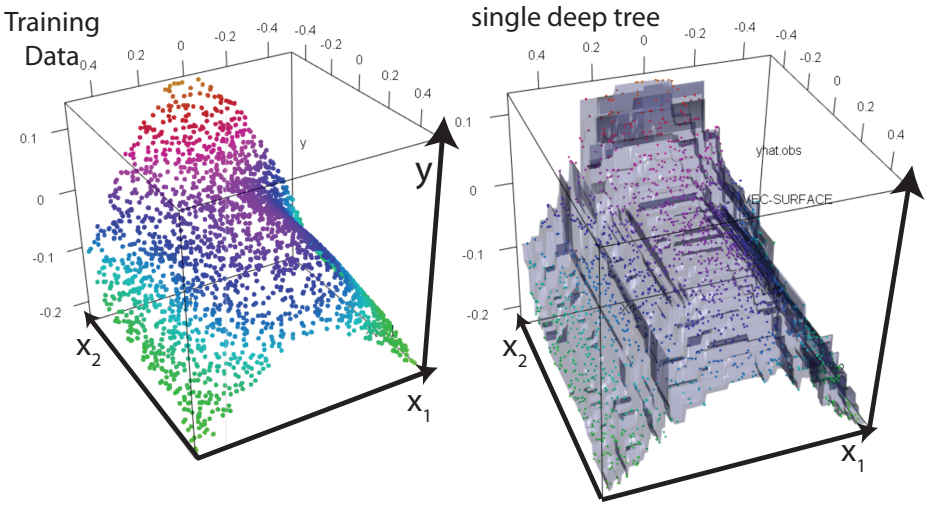


Figure 3.4: Single regression tree fit to 2500 data points without noise. $y = (X_1^2 + X_2^3)^2 - (X_1^2 + X_2^3)$, where X_1 and X_2 are drawn from uniform distributions..

If the same data set is added a normal distributed noise on the target values, the model variance in its predictions will rise up. In Figure 3.5, the same simulated data set has been added noise to the target, such that 33% of the variance is non explainable noise. An optimal model would therefore be able to explain approximately 67%. Model (a) is a single fully grown decision tree. The cross-validated explained variance is however only 30%. In model (b), the decision tree is regularized by limiting the depth of the decision tree, such that no node will be split if having less than 150 observations. The cross-validated explained variance is 50%. The model is no longer unstable, however as seen in the Figure 3.5, the model is now very crude and square. Thus by a bias-variance trade-off, the model have become stable by constraining the model surface complexity. Ensemble regularization with bagging can lower the decision tree variance without increasing the bias as much. Model (c) is a default random forest model, minimal nodesize = 5, number of trees is 500. The cross-

validated explained variance of the model is 60%. The default RF model structure surface still contain some random ripples that attributes to the model variance. To regularize the model beyond a default random forest model, there are four approaches: random variable sub space(see next section), bootstrap size, terminal node size and max nodes. As with model (b) the tree depth can be limited by restricting minimal node size or max nodes. In model (d) a trees of the random forest are limited to a minimal nodes size of 50. No node with less than 50 observations will be split. To increase the robustness of RF models directly by limiting the depth of each tree is recommended e.g. in Elements of Statistical Learning [FHT01] and in this simulation yields a cross-validated explained variance of 63.5%. However I find, when limiting tree depth directly, the tree correlation increases. Visually model (d) seems smoothed compared to (c). However model (d) also seems to retain some bias similar to the structure of model (b). Limiting bootstrap sample size to e.g. 10% of N training observations instead will also lower the tree depth as fewer splits can be made before each tree is fully grown. The tree correlation will be very low, as each bootstrap are much more different. Model (e) is a random forest where bootstrap sample size has been lowered to 250 instead of 2500, the model structure appear even more rounded and the cross-validation explained variance is 64.5%, slightly better than standard way to regularize RF. The slightly lower prediction error of model (e) over model (d) comes with a price. In Appendix A.2.12 a simulation show sample size regularization requires more trees to converge to a minimal prediction error, than with node size regularization as recommended by Friedman *et al.* In terms of training speed, fewer observations per tree approximately cancels the more trees required. This finding is interesting as sample size regularization, seem not to be mentioned much in random forest literature.

3.4.4 Random subspace regularization

Node size and sample size regularization as seen in Figure 3.5 tend to smooth the model structure. A third type of regularization is central for the random forest algorithm, called random variable subspace regularization. At any split only a random sample of features (aka. variables) are available to split by. This hyper parameter is called *mtry*, relating to the feature training matrix of n rows and m columns. If *mtry* is 3, only three random columns can be accessed by the algorithm for any split. Setting $mtry = 1$, will force the algorithm to use all features evenly. Setting *mtry* equal to the total number of features will allow the algorithm to greedily split by one dominant feature first and then only deeper down the tree make minor corrections by other features. Therefore random variable subspace regularization have a similar effect as $L2$-regularization of elastic net. A low *mtry* tend to distribute the dependence on features. Moreover, a low *mtry* decrease the tree correlation as different features will be used to make the first split. However low *mtry* tend to increase bias, especially interactions are fitted less well. In Appendix A.1 model structures are visualized for different values of *mtry*.

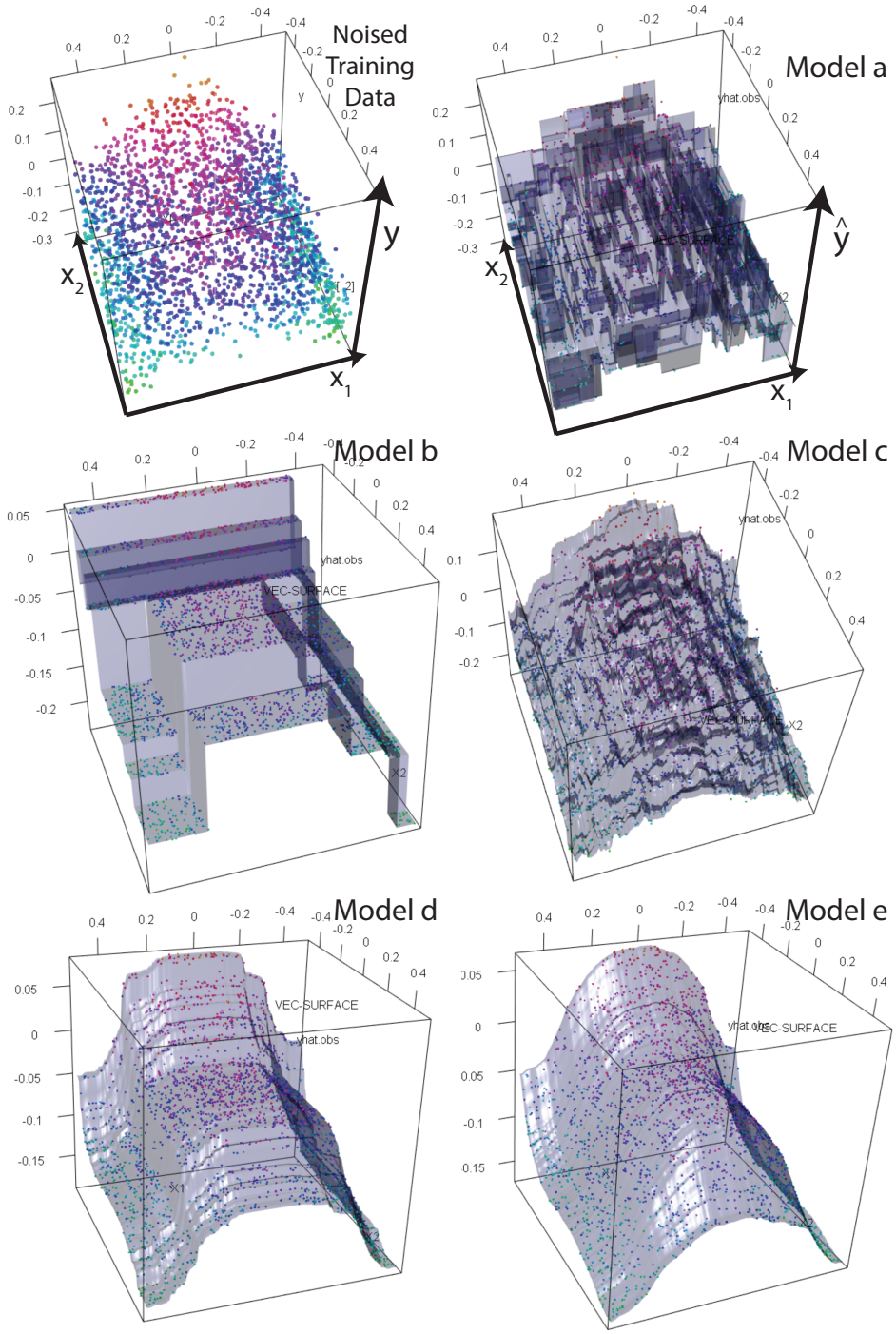


Figure 3.5: Single regression tree fit to 2500 data points with a 33% unexplainable noise/variance component, $y = (X_1^2 + X_3^3)^2 - (X_1^2 + X_2^3)$, where X_1 and X_2 are drawn from uniform distributions. (a) single tree, min node size is 5 , (b) single tree, min node size is 150, (c) ensemble of 500 trees, min node size is 5, (d) ensemble 1500 trees, min node size is 50, (e) ensemble of 1500 trees, min node size is 5, bootstrap sample size is 250.

3.4.5 Final choice of regularization

The standard approach is to find the set of hyper parameters that give the lowest cross-validated prediction error. However, already by visually inspecting the model surface, the user should consider if the model surface is plausible. Would it be realistic to reproduce e.g. the complex surface of model (a) and (c), if a new data set was sampled. If the user in fact expect the underlying function to be complex and the unexplainable noise component to be low, then the user should lean towards a model with less regularization and accept model (c). If the user expect the noise component to be substantial, then the user should lean towards more regularization, as a complex model structure would be unrealistic to estimate. Also as discussed in Section 3.2.2, model selection may be misleading. The model selected by the validation set, could be mostly selected due to luck and not skill. Luck happens to regress towards the average, it cannot be reproduced [Kah11]. Therefore, there is no standard cross-validation procedure that can calibrate any model. The user must inevitably include her own expectation to the underlying function that generates the data.

Moreover in terms of communication, the user can regularize the model beyond what seems optimal to lower the prediction error, as the model structure especially when high-dimensional may be easier to explore and communicate. An over regularized random forest model structure may be easier to visualize and yet far more accurate, than a linear regression model.

CHAPTER 4

Predict permeation enhancement

A main goal of this thesis has been to use the preclinical data more efficiently. Instead of testing randomly new absorption enhancers or what intuitively seems promising, a model built on previous experiences, can swiftly evaluate larger libraries of molecules.

4.0.1 Work flow and early challenges

Very early in the project three areas to predict were identified. These were solubility, critical micelle concentration (CMC) and permeation enhancement. In Novo Nordisk for purposes securing intellectual properties, the default rule is that no in-house data generated by the drug development projects can be published. Therefore all data used in this PhD thesis were already public available from third-party sources or conducted in experiments for this thesis only.

Figure 4.1 outlines the work flow of establishing a model to predict permeation enhancement.

The work flow outlines a bottom up approach, where very simple *in-vitro* experiments are conducted for series of candidate molecules. Each molecular formula is first characterized by external models simulating the molecule and calculating molecular descriptors, see the methods part 3.2 of article in Section 4.1. A relationship is established with the learning algorithm between the molecular formula and the *in-vitro* response.

Data on *in-vitro* Caco-2 permeation enhancement or similar preclinical studies were not expected alone to be useful to select new enhancers. *In-vitro* studies tend to overestimate the importance and impact of lipophilicity and the overall effect of absorption enhancer. Enhancers with C16 carbon chains are found 10-50 fold more potent than their C10 counter parts. [Mah+09; TT08]. Nevertheless C16 carbon chain based permeation enhancers, have quite disappointing not delivered the same potency in *in-vivo* studies. An obvious testimony is that there is no public announced clincal trials using C16 surfactant peptide absorption enhancers [Agu+16].

A starting project hypothesis was, that critical micelle concentration (CMC) was correlated with high absorption potency. Therefore initially, to collect data sets on CMC values was a central goal. Being able to predict CMC could assist the selection of permeation enhancer candidates. However, I found no literature that in fact shows,

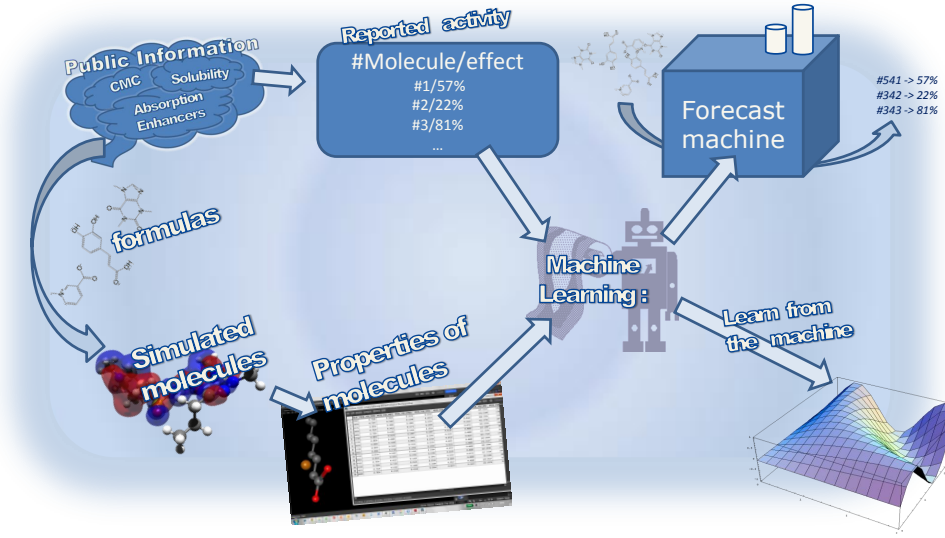


Figure 4.1: Public information are tables of measured activity (target) linked to unique molecule formulas. Formulas of molecules can be simulated with e.g. force field models to extract calculated properties (features / descriptors). A machine learning algorithm uses target and features to output a model. The model is used as a forecasting machine, and the model can be inspected to learn from the machine..

that low CMC values directly should cause permeation enhancement. Rather CMC is simply being one of more collective attributes associated with lipophilicity [RK12]. Surfactants can be effective below their CMC [XO00]. Therefore, it would make more sense to build predictive model using a target that resembled the permeation enhancement the most. Therefore Caco-2 measurements was finally favored over CMC measurements as model target.

Molecular solubility in watery buffers is not surprisingly negatively associated with lipophilicity / hydrophibicity. In a worst case scenario potency predictions and solubility predictions would be highly negatively correlated. Then to the models, permeation potency and in-solubility would be two indistinguishable phenomena. Such a failure of the model destinguish insolubility with permeation enhancement would contract the observed population of permeation enhancers in litterature, as elongating the carbon chain of any molecule making it more lipophile, does not necessarily make a potent permeation enhancer. In the article of Section 4.1, I used an early version of the developed random forest diagnostic package forestFloor [Wel+16], to discover that the trained permeation enhancement model in fact had captured this distinction

in an interaction term. See Section 5.2 of how I define and classify interactions for non-linear models such as random forest.

4.0.2 Challenges of a top down modeling of permeation enhancement

In opposite to the bottom up approac of modeling simplified *in-vitro* experiments, a top down approach was also being followed early in the project. Hypothesis associated with top down approach was, that molecules claimed in literature such as articles and patent applications to be permeation enhancers could be used as training examples.

One data set was obtained in an effort to use text mining to collect examples of compounds being mentioned for being permeation enhancers and/or absorption enhancers. The data set was acquired as a part of Sarah Brebbia Dirksen (Øster) master thesis project in 2012. A list of 167 compounds originating from patent applications and research articles was compiled [Bre12].

Such a set of examples would not only reflect *in-vitro* studies, but also a series of unknown limitations such as toxicology, formulation in-capabilities etc. Thus a model build on target data in the end stage of process, may contain a number of favorable biases, such that the model e.g. tend not to suggest toxic molecules as permeation enhancers.

It was assumed that permeation enhancement is a rare property of molecules, such that any random molecule would likely not be an enhancer. Random molecules of similar weight was acquired with the software ACDlab. Hereafter 49 chemical descriptors were computed for both sets of molecules using the software packages ACDlabs and Molecular operating environment (MOE). Linear discriminant analysis has been used to learn a class separation rule to identify typical enhancer like molecules [Bre12].

Although a valid cross-validation of the classification model predicted a nearly perfect classification accuracy, the model failed in a proof of concept to test predicted molecules in-house in the Caco-2 model. There are a number reasons why this approach can have failed in spite of a promising cross-validation. To be fair some of these reasons were discussed in the fine master thesis of [Bre12]. However, in the clear light of hind sight, let us in the following section evaluate the implicit assumptions of this top-down approach.

4.0.3 Assumptions of the top-down model

A requirement was that training and future test observations had to be drawn from same distributions, see Section 3.2.3. The training set originated with positive examples from the pharmaceutical literature and with negative examples from what appeared to be a random synthesis chemical library. It was assumed that all molecules from random synthesis library were not effective as enhancers, and I agree on this as a fair assumption. However visually inspecting the random synthesis molecules,

revealed these molecules were far from typical excipients of pharmaceutical formulations. Exotic heavy atoms as uranium! or what appear to be a truly random patterns of side group substitutions was observed. If this library in fact was generated as an random or an exhaustive search of possible molecule graphs below a given molar weight, then the average molecules will look far from the typical excipients as these are certainly not random molecules for a number of reasons. The typical excipient, especially surfactant enhancers, are not very substituted, not very branched. If generating a molecule starting from one carbon atom, and any choice have equal probability, should this carbon chain branch out here or not? Should there be substituted something here or not? The average molecule will be very branched and highly substituted, simply because there the combinatorial space is bigger, than for non-branched non-substituted carbon chains. Thus the classification model has been trained to recognize the difference between two groups of molecule that are obviously different. A part of the reason the two classes of molecules are different may well be properties related to being permeation enhancers or not. However any other differences between the groups will be confounded permeation enhancement.

Most likely such classification model would be used to scan through a library of approved pharmaceutical excipients, thus only drawing observations to predict from the pharmaceutical molecule distribution. Suddenly the model may recognize many of the molecules as permeation enhancers, simply because these are not branched enough or highly substituted. The model will suggest too many molecules as permeation enhancers and not all can be tested. One could use the linear discriminant score to rank the predicted molecules to select those, who where most likely permeation enhancers. However classic linear discriminant analysis, do not employ a proper score function such as the Breier score and is not calibrated to rank how likely predicted molecules are permeation enhancers.

Supervised machine learning rely on, that proximate observations have similar targets on the smallest scale in the feature space, such that some interpolation is a meaningful prediction. However, if including receptor mediated permeation enhancers or calcium chelators. Receptor-agonist interactions are very delicate processes that in two ways are analogous to a lock and key. First a permeation enhancer agonist will like a key fit in a membrane receptor lock and trigger a response. Secondly, just the slightest change of the key or lock could disable the activation. To map any possible key combination would take a very high number of training examples. Likewise for receptor mediated permeation enhancers or calcium chelators, just a single reconfiguration of an atom, could completely change the biological response. A few hundred molecules mentioned in litterature is likely very far from mapping the entire complex mechanism of a living cell. In contrary surfactant-like permeation enhancers will likely to some extend elicit their effect through surface tension depression. A single atom reconfiguration will likely not completely revert the physiochemical properties and an interpolation on the small scale will be meaning full.

4.1 Article 2: *In silico modelling of permeation enhancement potency in Caco-2 monolayers based on molecular descriptors and random forest*

European Journal of Pharmaceutics and Biopharmaceutics 94 (2015) 152–159

Contents lists available at ScienceDirect

European Journal of Pharmaceutics and Biopharmaceutics

journal homepage: www.elsevier.com/locate/ejpb

Research paper

In silico modelling of permeation enhancement potency in Caco-2 monolayers based on molecular descriptors and random forest

Søren H. Welling [a,b], Line K.H. Clemmensen [b], Stephen T. Buckley [a], Lars Hovgaard [a], Per B. Brockhoff [b], Hanne H.F. Refsgaard [a,*]

[a] Global Research, Novo Nordisk A/S, Novo Nordisk Park, 2760 Måløv, Denmark
[b] Technical University of Denmark, DTU Compute, 2800 Kgs. Lyngby, Denmark

ARTICLE INFO

Article history:
Received 30 January 2015
Revised 14 May 2015
Accepted in revised form 17 May 2015
Available online 21 May 2015

Keywords:
Permeation enhancers
Caco-2
Random forest
QSAR
Surfactants

ABSTRACT

Structural traits of permeation enhancers are important determinants of their capacity to promote enhanced drug absorption. Therefore, in order to obtain a better understanding of structure–activity relationships for permeation enhancers, a Quantitative Structural Activity Relationship (QSAR) model has been developed.

The random forest-QSAR model was based upon Caco-2 data for 41 surfactant-like permeation enhancers from Whitehead et al. (2008) and molecular descriptors calculated from their structure.

The QSAR model was validated by two test-sets: (i) an eleven compound experimental set with Caco-2 data and (ii) nine compounds with Caco-2 data from literature. Feature contributions, a recent developed diagnostic tool, was applied to elucidate the contribution of individual molecular descriptors to the predicted potency. Feature contributions provided easy interpretable suggestions of important structural properties for potent permeation enhancers such as segregation of hydrophilic and lipophilic domains. Focusing on surfactant-like properties, it is possible to model the potency of the complex pharmaceutical excipients, permeation enhancers. For the first time, a QSAR model has been developed for permeation enhancement. The model is a valuable *in silico* approach for both screening of new permeation enhancers and physicochemical optimisation of surfactant enhancer systems.

1. Introduction

Development of oral delivery systems for proteins and peptides offers the promise of improved patient compliance compared to conventional parenteral administration. However, bioavailability is, in part, limited due to poor absorption of proteins across the intestinal epithelial barrier. To effectively deliver a protein systemically this barrier can be modulated by the presence of permeation enhancers [1].

Quantitative Structural Activity Relationship, QSAR methods have been applied extensively for exploration of structural properties of importance for oral absorption of new chemical entities, e.g., QSAR models have been developed for permeability [2] and solubility [3–5]. To our knowledge, no QSAR model for permeation enhancement has previously been published.

Some permeation enhancers have specific mechanisms of action, e.g., modulating the function of tight junctions in the plasma membrane such as zona-occludens-toxin [6], EDTA [7] or melittin [8]. However, the majority of permeation enhancers are primarily surfactants and will non-specifically disrupt the lipid bilayer packing of phospholipids in the epithelial membrane [1]. Surfactants are molecules having segregated lipophilic and hydrophilic domains. Water soluble surfactants tend to pool in the surfaces of water/air and water/lipid, lowering the surface tension. Lowering of surface tensions of water/air surfaces and the ability to enhance the permeability across lipid bilayers correlated

Abbreviations: C6, sodium hexanoate; C8, sodium octanoate; c8G, octylglucoside; C10, sodium decanoate/caprate; c12PC, dodecylphosphocholine; c12GPC, dodecanoylglycerophosphocholine; c14GP, myristoylglycerophosphate; CART, classification and regression tree; CDC, chenodeoxycholate; DDM, dodecylmaltoside; EDTA, ethylenediaminetetraacetic acid; GCC, glycochenocholate; GH, glycyrrhizinate; LCC, lauroylcarnitinechloride; LOO-CV, leave-one-out cross validation; MOE, molecular operating environment; PCC, palmitoyl carnitine chloride; QSAR, quantitative structural activity relationship; SM, simomenine; RMSE, root mean square error; r_p, Pearsons correlation coefficient; r_s, Spearman rank correlation coefficient; SD, standard deviation; TEER, transepithelial electrical resistance; TDM, tetradecylmaltoside; TDS, sodium tetradecyl sulphate; TDM, tetradodecyl maltoside; TC, taurocholate; T_{pot}, TEER potency; UC, Ursocholate.

* Corresponding author at: Insulin Pharmacology Research, Novo Nordisk A/S, Novo Nordisk Park, 2760 Måløv, Denmark.
E-mail address: hare@novonordisk.com (H.H.F. Refsgaard).

http://dx.doi.org/10.1016/j.ejpb.2015.05.012
0939-6411/

well for a selection of surfactant-like permeation enhancers [9]. General relations between molecular structures and physicochemical properties of surfactants are thoroughly described by Rosen [10]. Several properties of surfactants, including surface pressure, have previously been modelled with a QSAR approach applying both linear regression and non-linear machine learning models as artificial neural networks, support vector machine or random forest [5,11,12]. Combining the above mentioned concepts, it seems plausible that a QSAR-model of surfactant-like permeation enhancement could be constructed.

Our modelling is based on a Caco-2 data set for 41 surfactant permeation enhancers from Whitehead [13,14] tested in cell monolayers across three concentrations. Hereby, trade-offs between potency, pathway and safety amongst a selection of mainly surfactant-like permeation enhancers were investigated. For this article only the potency data was used. *In vitro* Caco-2 monolayers are cultures of functional, differentiated enterocytes and are widely employed to evaluate permeability rates of drug candidates or pre-formulations [15]. The Caco-2 data for permeation enhancers from Whitehead [13,14] together with molecular descriptors calculated from structure of these surfactants were the basis for the QSAR model.

Non-linear machine learning models can have superior predictive capabilities compared to classical statistical explanatory modelling. However, such machine learning models are often complex "black boxes" – difficult to interpret and discuss [16]. This article presents a promising method to elucidate the interplay of features comprising good permeation enhancers within the complex non-linear model of random forest. Therefore, based on the developed model, we here can recommend ranges of the selected molecular descriptors to obtain high permeation enhancement potency.

2. Materials and methods

2.1. Materials

Caco-2 cells (ATTC-HTB-37) were obtained from American Type Culture Collection (Manassas, VA). Cell culture media (Dulbecco's modified essential media (DMEM)) and penicillin/streptomycin were purchased from Lonza (Verviers, Belgium). All other supplements (i.e., foetal bovine serum, HEPES buffer and non-essential amino acids (NEAA)) as well as Hanks' balanced salt solution (HBSS) and trypsin were purchased from Gibco, Life Technologies (Carlsbad, CA). Corning Transwell® filter inserts (1.12 cm^2 surface area, 0.4 μm pore diameter) were purchased from Fisher Scientific (Waltham, MA). Bovine serum albumin (BSA) was purchased from Sigma Aldrich (St. Louis, MO). All other reagents were of the highest analytical grade.

2.2. Cell culture and TEER measurements

Caco-2 cells (passage numbers 41–49) were seeded at a density of 2.5×10^5 cells/flask and grown to 70–90% confluence in DMEM (supplemented with 10% FBS, 100 U/ml penicillin and 100 μg/ml streptomycin and 1% (v/v) NEAA). For transport studies, Caco-2 monolayers were cultured on permeable Transwell® 12 mm diameter inserts at a density of 10^5 cells/cm^2 and used after 14–17 days in culture. Cells were cultured at 37 °C and 5% CO_2 atmosphere and the medium was changed every other day. Monolayers were equilibrated in HBSS-based transport buffer 1 h prior to testing. Transepithelial electrical resistance (TEER) was measured with a chop-stick electrode (Millicell-ERS®, Millipore, Billerica, MA) prior to testing, and monolayers with TEER values <600 Ω cm^2 were discarded. TEER was measured after 1 h exposure to permeation enhancers.

3. Data processing

3.1. Training set

Whitehead et al., tested the ability of 51 permeation enhancers to lower the barrier integrity marker %TEER in Caco-2 cells at 1%, 0.1% and 0.01% (w/v) and published the data set as supplementary materials in two papers [13,14]. Of the 51 permeation enhancers reported, forty-two had computable molecular structures (non-mixtures) and were a wide selection of enhancers which were ascribed to 10 different categories of surfactants: Anionic surfactants, cationic surfactants, zwitterionic surfactants, non-ionic surfactants, bile salts, fatty acids, fatty esters, fatty amines, sodium salts of fatty acids, nitrogen-containing rings and others [13]. EDTA (a calcium chelator) was excluded from the training set because of a non-surfactant-like mechanism together with high potency. The remaining 41 permeation enhancers had surfactant-like structures or low potency e.g., urea could be described as an ineffective surfactant without permeation enhancement effect.

TEER-potency (T_{pot}) was defined to concatenate measurements of TEER%-decrease (EP) at the three different concentrations (0.01%, 0.1% and 1% w/v) into one target variable. T_{pot} was simply defined as the mean TEER%-decrease across the three concentrations as given in Eq. (1). $T_{pot} = 1$ corresponds to a permeation enhancer lowering TEER% completely at 0.01% (w/v) and $T_{pot} = 0$ translates to no effect of a permeation enhancer on TEER% even at 1% (w/v). The TEER%-decrease EP is defined as in Eq. (2) and depends of the TEER% before and after treatment with enhancer plus $TEER\%_+$ the background filter resistance.

$$T_{pot} = \frac{EP[0.01\%] + EP[0.1\%] + EP[1\%]}{3} \quad (1)$$

$$EP = 1 - \frac{TEER\%_{AE} - TEER\%_+}{TEER\%_{noAE} - TEER\%_+} \quad (2)$$

From a statistical point of view the loss of information is minimal, as the TEER%-values of the three concentrations were highly correlated. The loadings of the first principal component of a principal component analysis resembled the definition of T_{pot} and this principal component explained 71% of the variance. From a practical viewpoint T_{pot} could be seen as a linear approximation of pEC50 (−log effective concentration (w/v) of where 50% TEER-decrease is observed), see Eq. (3). pEC50 itself is dimensionless.

Thus, for a given permeation enhancer having a potency of pEC50 = 1 the corresponding value of $T_{pot} = 0.5$.

$$T_{pot} = \frac{pEC50 + 0.5}{3}, \quad \text{for } pEC50 \in [-0.5; 2.5] \quad (3)$$

3.2. Software packages, descriptors and model design

The open source R statistical software (v 3.02) was acquired freely from http://www.r-project.org and Rstudio integrated development environment (v 0.98.501) also acquired freely from http://www.rstudio.com. The R-package 'randomForest' (v.4.6) [17,18] was used in the random forest-QSAR model. CAS identification numbers of compounds in the training set were converted to mol-files through SciFinder [19]. Mol-files bundled in sdf-files were imported to the software application MOE [20] and sequentially pre-processed with the following functions: '*wash*' (simulating an ideal solubilised molecular form), '*partial charges MMFFA96x*' calculating the electron densities necessary for a number of descriptor algorithms, and finally '*energy minimize*' relaxing the molecule in the minimum state. All 2D molecular descriptors provided by MOE were computed. The subgroup of 3D descriptors '*vsurf*' [21] plus the single 3D descriptor '*dipole*' were calculated as they were relatively fast to compute and therefore suitable for

screening purposes. One new descriptor carbon chain length (CCL) was implemented through R. CCL is the length of the longest saturated non-substituted aliphatic carbon chain of a given permeation enhancer. Table 1 explains the simple implementation of CCL. After 266 molecular descriptors were acquired, a variable filtering was performed to increase prediction performance. First, fifteen descriptors were excluded for having the same value for more than 95% of the actual training-set. A descriptor having the same value for all permeation enhancers does not provide any information and is problematic for some algorithms which e.g., divide by the variance, which will be zero. Subsequently, 143 redundant descriptors were filtered off, one at a time, until no remaining descriptor pair-wise correlations exceeded $r_p = 0.9$ (Pearson correlation). This correlation-filtering was a simplified implementation of the CORCHOP routine [22]. Lastly, the remaining descriptors were filtered by their Spearman rank correlation coefficient (r_s) to the target variable, T_{pot}. As Spearman rank correlation utilises the target parameter (T_{pot}), it was computed separately on training data for each fold of the cross-validations, so as to avoid latently overfitting. Nevertheless, the random forest algorithm was a robust model and the root mean square error estimated by leave one out cross-validation ($RMSE_{LOO\text{-}CV}$) exhibited a variation of less than 20% for any reasonable subsets of pruning parameters. The 30 best r_s-correlating (or inverse-correlating) descriptors were included in the model. See Table 2 gives an overview of the descriptors selected for the model. Fig. 1 depicts the data flow from molecular formulas, computation of molecular descriptors, variable filtering, model training and cross-validation.

The default parameters of the random forest model were used as provided in the R-CRAN package 'randomForest', though the number of decision trees grown was set to 10,000 or 50,000 to ensure a conveniently high reproducibility between model-runs. Variable importance was computed for any descriptor and described the deterioration in prediction accuracy of the model, when permuting the particular descriptor. Variable importance was used to rank the importance of the descriptors and did not influence the model predictions. However, variable importance was a valuable tool to identify the molecular descriptors/features most important for predicting surfactant-like permeation enhancement.

To assess the mechanics of the random forest-QSAR, the package rfFC [23,24], which is a diagnostic extension for random forest, was acquired from https://r-forge.r-project.org/projects/rffc/. rfFC provides forest contributions which is the mean contribution of a given variable to the T_{pot} prediction of a given permeation enhancer.

3.3. Experimental test set

A set of 11 compounds and an additional 3 compounds from the training set were tested in Caco-2 monolayers to generate an experimental test set for validation of the developed random forest-QSAR model. Contrary to the experimental setup of the training data, the experimental test conducted for this paper differs in terms of media (HBSS versus DMEM, respectively) and incubation times (60 min versus 15 min, respectively). Likewise, morphology of Caco-2 monolayers is expected to have some inter-lab variation [25]. The most lipophilic permeation enhancers were barely soluble at 1% (w/v) at 37 °C and needed to be maintained at this temperature during the experiment at all times to avoid precipitation. Model predictions were compared to experimentally measured values of T_{pot}. The Squared Pearson correlation coefficient (r_p^2) and the root mean square error of ordinary least square fit ($RMSE_{OLS}$) were used as the validation criteria for the linear relationship between model predictions and experimental values. It is acceptable that the slope and offset deviates from 1 and 0 respectively as the absolute measured T_{pot} is method specific.

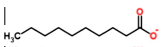
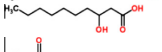
Table 2
Overview of the 30 descriptors applied in the random forest-QSAR model predicting the potency of surfactant-like permeation enhancers in Caco-2 monolayers. Descriptors were computed through MOE (18) except CCL "carbon chain length" implemented for this article.

Group of descriptors:	Amount used	Names of descriptors as available in MOE
Atom counts and bond counts	3	a_nN a_nS b_double
Kier–Hall & Kappa shape:	1	chi1_C
Adjacency and distance matrix:	8	BCUT_SLOGP_0, BCUT_SLOGP_3 BCUT_SMR_3 GCUT_PEOE_0 GCUT_PEOE_3 GCUT_SMR_0 wienerPath
Pharmacophore feature:	1	a_base
Partial charge:	10	Q_PC+ PEOE_RPC+ PEOE_PC+ Q_VSA_PPOS Q_VSA_POL Q_VSA_FPNEG PEOE_VSA_POL PEOE_VSA_FPPOS PEOE_VSA+5 PEOE_VSA+1 PEOE_VSA-1
Surface area, volume and shape:	4	vsurf_IW3 vsurf_ID8 vsurf_CP vsurf_Wp 2
Conformation dependent charge:	1	dipole
Physical properties:	1	log*P*(o/w)
New descriptor in this article:	1	CCL "carbon chain length"

3.4. Literature test set

Based on a literature search, nine permeation enhancers were included as a literature test set (Table 3). For all included

Table 1
Examples of the new descriptor carbon chain length (CCL). CCL estimates the longest sequence of saturated carbon atoms by counting the longest sequence of capital C's in a corresponding SMILES representation of the structure.

Name	Structure	Smiles underscoring	CCL count
Decanoate		O=C(O)CCCCCCCCC	9
3-Hydroxydecanoic acid		CCCCCCCC(CC(=O)O)O	8
Benzoic acid		O=C(O)c1ccccc1	1

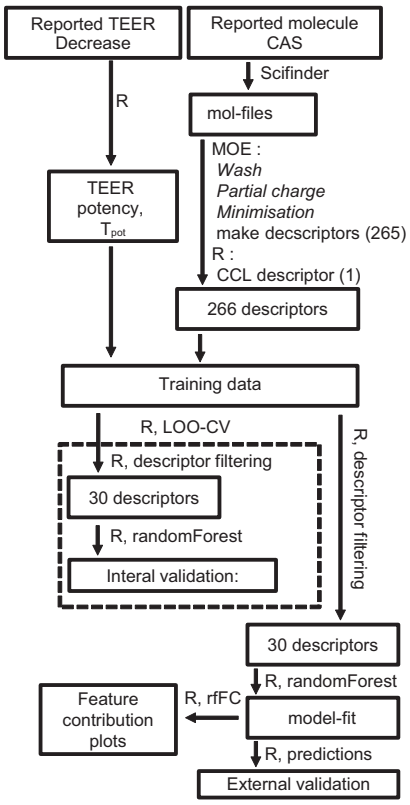


Fig. 1. Scheme of the modelling process. From top, TEER-data and provided chemical identification (CAS) were processed into TEER potency (T_{pot}) and molecular descriptors. Descriptor filtering is embedded in the leave-on-out cross-validation (LOO-CV). The final random forest model was both used for prediction of external test sets and for diagnostic interpretation through the rfFC-package. R (R) and molecular operating environment (MOE) are software applications. CCL, carbon chain length, molecular descriptor, see Table 1.

Table 3
Permeation enhancers from literature tested in Caco-2 monolayers. EC50% is the estimated concentration (w/V)% where the permeation enhancer will lower TEER 50%. pEC50 is the negative logarithm to EC50%. RF predicted potency (w/V)% is the average ability of the permeation enhancer to lower TEER at 1%. 0.1% and 0.01% (w/V)%.

CAS	Compound name	EC50, (w/V)%	pEC50	Refs.
6080-33-7	Simomenine (SM)	2.0	−.30	[26]
81-24-3	Tauro cholate (TC)	.80	−.096	[27,28]
474-25-9	Cheno deoxy cholate (CDC)	.50	.30	[28]
128-13-2	Ursocholate (UC)	.50	.30	[28]
29836-26-8	Glyco octyl (c8G)	.40	.40	[29]
68797-35-3	Glycyrrhizinate (GC)	50	.70	[30]
325465-45-0	Myristoyl glycerol phosphate (c14GP)	.10	1.0	[31]
20559-18-6	Lauryl glycerol phospho choline (c12GPC)	.025	1.6	[31]
29557-51-5	Dodecyl phosphate choline (c12PC)	.021	1.7	[31]

permeation enhancers from the literature, pEC50 was estimated by interpolation to compare across various experimentally applied concentrations. pEC50 is the negative logarithm of EC50 and has an approximate linear relation to T_{pot}, as described in (Eq. (3)). The model performance was validated by the accuracy of the pEC50 prediction for the permeation enhancer in the literature test set. Again the main criteria for comparison were r^2_p and $RMSE_{OLS}$ between interpolated pEC50 values and predicted T_{pot} values.

4. Results

A random forest-QSAR model was developed based on a 41 compound training set from literature [13,14] and permeation enhancement potency (TEER% Caco-2) values were matched with molecular descriptors. The predictability of the model was tested through validation. Three types of validation were applied: Internal leave-one-out cross validation, (LOO-CV), experimental validation and literature validation. Lastly, the mechanics from the autonomous random forest-QSAR model was extracted to provide a complimentary insight into which molecular properties there are important for permeation enhancement potency.

4.1. Model validation

Internal cross-validation was used throughout the process of designing a predictive generalisable model of permeation enhancement. Table 4 summarises the validation outcome. Both the internal and experimental validation showed $RMSE_{OLS}$ = 0.16–0.17. This error was a sixth of the entire 0–1 range of the T_{pot} scale. As T_{pot} summarises three concentration levels 1% to 0.1% to 0.01% (w/v) with a 10-fold span between each step, the accuracy was interpreted as to confirm that the model could predict within which 10-fold concentration a given permeation enhancer was effective. Likewise, for the literature validation the RMSE was 0.39, which corresponds to less than half of one unit on the pEC50 scale. One unit of pEC50 is equal to a 10-fold change in 50% effective concentration.

Fig. 2 shows plots of the three types of validations. In part A and B the predicted T_{pot} values are plotted against the measured values for the training set data from Whitehead et al. [13,14] and for the experimental data set. Fig. 2C depict the correlation between predicted T_{pot} potencies and the actual pEC50 values for the literature test set. The internal LOO validation correlation coefficient was lower, r^2_p = 0.57 (Fig. 2A), than for the external test-sets, r^2_p = 0.65–0.66 (Figs. 2B and 1C).

Eleven permeation enhancers were evaluated in Caco-2 monolayers as an experimental test set (Fig. 2B). Biotin and benzoate are widely used food additives and were intended as negative

Table 4
Summary of the validation of the random forest QSAR model predicting potency (%TEER) of permeation enhancers in Caco-2 monolayer. T_{pot}, a measure of enhancer potency defined as mean decrease of %TEER when applying 1%, 0.1% and 0.01%(w/v) in Caco-2 monolayers. pEC50 is the estimated concentration of which %TEER is decreased 50%. CV-LOO, internal cross validation – LOO. $RMSE_{ols}$, root mean square error of ordinary least square prediction fit. (a) RMSE, root-mean-square-error adjusted to compare across T_{pot} and pEC50 (Eq. (3) in Section 2).

	Training-set	Test-set, experimental	Test-set, literature
Number of enhancers	41	11	9
Data origin	1 article	Experimental	7 articles
Target value	T_{pot}	T_{pot}	pEC50%(T^a_{pot})
Model correlation, r^2_p	57% (LOO-CV)	66%	65%
Model error, $RMSE_{OLS}$	0.17	0.16	0.39(0.16[a])

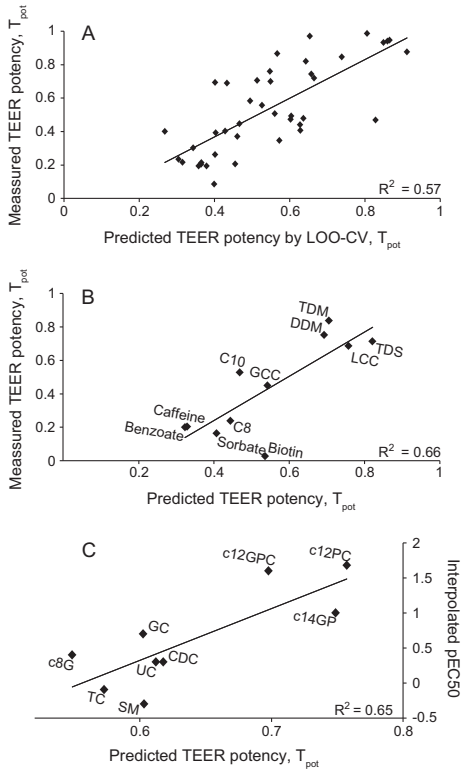


Fig. 2. Validation plots of random forest-QSAR model predicting the potency (%TEER) of permeation enhancers in Caco-2 monolayer. *X*-axis is the predicted target response of permeation enhancers' ability to lower %TEER in the Caco-2 monolayers. *Y*-axis is the true target response. (A) Internal cross-validation plot validating the predictability within the training-set, (B) validation of experimental test set, (C) validation of literature test set. T_{pot}, mean decrease of %TEER in Caco-2 monolayers of a given permeation enhancer applied at concentrations of 1%, 0.1% and 0.01%. pEC50, negated 10 base logarithm of concentration (%w/v) of 50% TEER.

controls. None of these compounds were measured to be potent permeation enhancers. All permeation enhancers in the experimental test set, with the exception of biotin, were well predicted. Biotin was predicted to elicit a moderate potency, but was devoid of any significant effects when tested in Caco-2 cells. Three compounds SDS, C6 and PCC from the training set were retested to verify, that the experimental setup applied here could reproduce findings from Whitehead et al. (data not shown).

Fig. 2C show the predicted T_{pot} potencies and the actual pEC50 values for the literature test set. The nine enhancers were surfactants with a single well defined molecular structure and sufficient data points published to estimate a pEC50 value. The range of interpolated pEC50 values from the literature data ranged from −0.3 to 1.7 corresponding to that the most potent permeation enhancer had ∼100 times higher potency than the weakest. Three of the nine compounds Myristoyl glycerol phosphate (c14GP), Lauryl glycerol phospho choline (c12PC) and Dodecyl phosphate choline (c12GPC) were markedly more potent than predicted by the model. The exact predicted rankings of the second most and third most potent compounds of the experimental test-set were not correct, but within the expected uncertainty of the model. The same was seen for the group of low potent permeation enhancers.

4.2. Reviewing descriptors useful for prediction of permeation enhancement

Of the 266 descriptors assessed, 30 descriptors were applied after filtering in the model. Names and grouping of the used descriptors can be seen in Table 2 in the method section. The 16 most important descriptors were included in a Spearman rank correlation matrix depicting their internal rank correlation within the training set (Fig. 3) and their rank correlation with the target parameter T_{pot}. The strongest absolute correlation coefficient, 0.89, was between variables PEOE_VSA-1 and log*P*.oW. No correlation could exceed the correlation filter limit of 0.9, as described in Section 2. All 16 descriptors were found to be rank correlated with the target value T_{pot}. The descriptors absolute rank correlations to T_{pot} ranged from $r_s = 0.34$ to $r_s = 0.63$.

To interpret the precise contribution of each descriptor within the model, a diagnostic method termed 'Feature Contributions' [23,24] was used. A novel diagnostic plot of the feature contributions is presented in Fig. 4 The feature contributions of the 16 most important descriptors describing permeation enhancer-potency (T_{pot}) in the training-set were plotted against their respective descriptor values. This provided an intuitively graphical interpretation of how features within the random forest-QSAR model context affected the T_{pot} prediction. It represents an innovative way to graphically present the computed feature contributions. This expansion of a regular random forest model summarises the total partial descriptor contribution for any permeation enhancer in the training set. The predicted T_{pot} values for a given enhancer are equal to the sum of all partial descriptor contributions, which again is dependent of the actual feature values, as outlined in Fig. 4.

Fig. 4 shows that descriptor [2, BCUT_SLOGP_0], and [3, vsurf_ID8] had sharp thresholds separating the positive (i.e., beneficial) and negative contributions to the T_{pot} value of each permeation enhancer. For [1, dipole] there was also a separation between positive and negative contribution to the T_{pot} value.

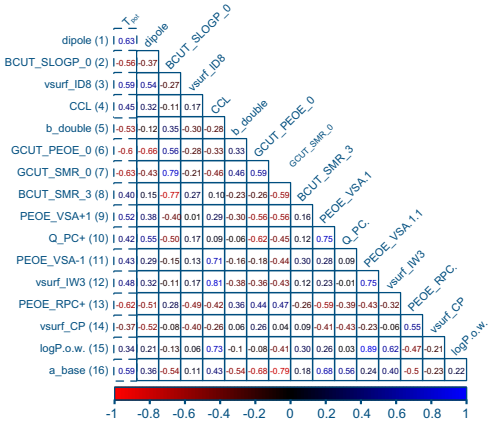


Fig. 3. Spearman-correlation matrix of the 16 most important molecular descriptors listed by decreasing variable importance (most important first) within the random forest-model. The target variable T_{pot}, the ability of permeation enhancers to lower electrical resistance in Caco-2 monolayers across concentrations 0.1–1%(w/v), has also been included.

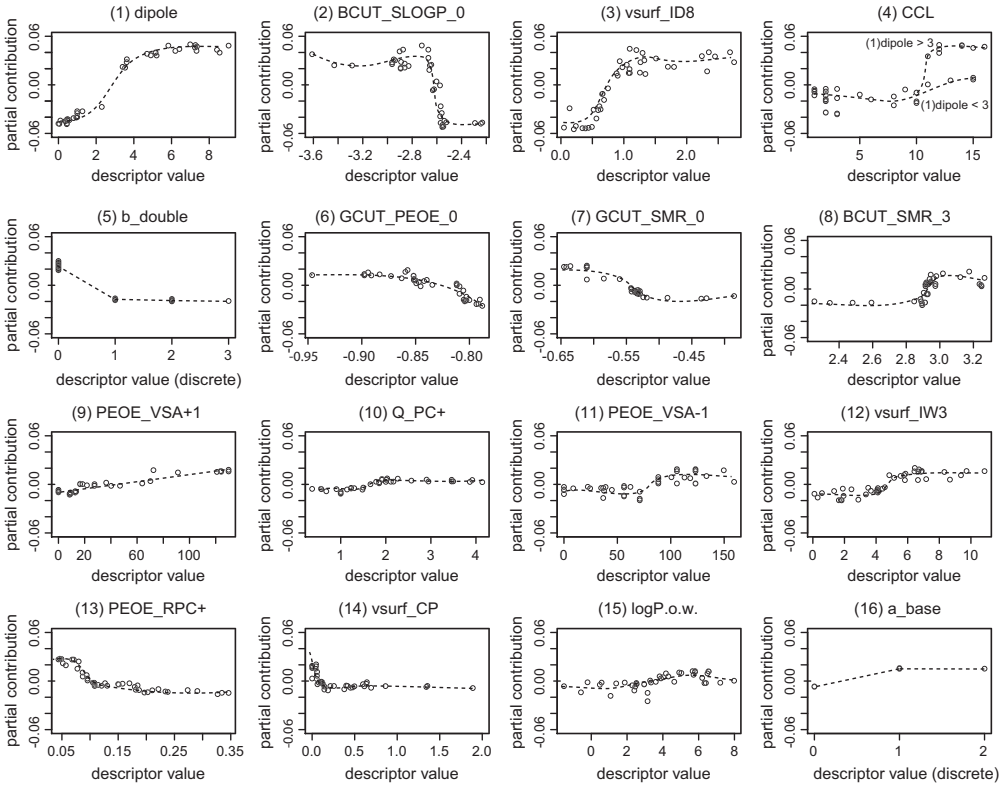


Fig. 4. Random forest feature contributions diagnostics. Scatter plots of partial descriptor contributions versus the individual descriptor values for each observation of the training set for the 16 descriptors with highest 'importance' within the random forest model. Scatter plots enumerated in descending order of 'importance'. Dashed trend lines were added to the plots. Each plot outlines a partial function of a variable as its role in the model.

Good permeation enhancers, compounds with high T_{pot} values, had high dipole > 3, [2, BCUT_SLOGP_0] below −2.7 and [3, vsurf_ID8] > 1.

[1, dipole] reflected the overall dipole moment calculated from the partial charges of the molecule. The descriptor contribution of [1, dipole] within the random forest model was well described as a function of the descriptor value itself. Thus, there was no interaction with other descriptors. On the contrary, the descriptor contributions of descriptor [4, CCL], the maximum aliphatic carbon chain length, varied for many permeation enhancers having the exact same chain length. This pointed to an interaction phenomenon between descriptors. When only emphasising permeation enhancer above the threshold value for dipole > 3, these permeation enhancers were all accredited positively for having a high CCL value. Oppositely, enhancers with dipole < 3 were accredited neutral for any CCL value. The interpretation drawn was that molecules with a high dipole moment are likely to have a hydrophilic domain and if combined with an aliphatic carbon chain of length > 10, the molecules are likely to have surfactant properties. Conversely, compounds with no significant hydrophilic groups such as oils, would not function as enhancers alone despite long carbon chains.

Descriptor [3, vsurf_ID8] reflected the hydrophilic domains separation from the lipophilic domains which was expected to be a central surfactant-like property. More precisely, the [3, vsurf_ID8] reflected the distribution of hydrophobic or hydrated domains and their distance from the mass centre. It was observed that the model evaluated low [14, Vsurf_CP] values as being beneficial. [14, Vsurf_CP] is a micelle critical packing parameter. Cone shaped surfactants would in generally have a low [14, Vsurf_CP] value and thereby a low critical packing number. A low critical packing number favours micellar aggregation, not liposomal. Throughout Fig. 4, the descriptors were generally declining in magnitude of feature contributions, and thus less influential.

5. Discussion

The random forest-QSAR model of permeation enhancement in Caco-2 cells was shown to provide reasonable estimates of permeation enhancer potency. Such a model has to the knowledge of the authors not been developed previously. Within the paradigm, that surfactant-like properties are key features of most enhancers, it was confirmed that a model could be constructed inspired by the *in silico*, *in vitro* and *in vivo* models of surfactant surface tension depression [9,11,12].

In the case of a new modelling area, a large amount of descriptors could possibly become useful. However, the relatively small

size of training examples makes such a selection process challenging. The training set of 41 permeation enhancers and 266 descriptors is an example of sparse (small n – large p) modelling which can lead to overfitted non-generalisable models. Filtering/pruning constants, redundant and non-correlated descriptors improve model performance.

The ensemble method, random forest, is an extension of the classification and regression tree, CART. Decision tree models branch out/split data into increasingly smaller sub groups of samples having the most similar target response. Such CART decision trees are highly adaptable of many types of data, but also easily overfitted to the training data. Thus, the model becomes adaptable but highly noise sensitive. The random forest model is an extra layer to the CART, reducing noise without losing its adeptness. In short, random forest is an ensemble of many of such uncorrelated decision trees (e.g., 500). Though each tree is susceptible to random inference, the average prediction of many decision trees have been shown to be much less prone to overfit thus its conclusions/predictions are more generalisable across data sets [17]. That said, the conclusions from any model approach will always be limited by the diversity of the training set. In this example, scientific literature, the basis of this model training, tend to have a bias towards not reporting any compounds which lack an enhancing effect. In the case of this training set, all compounds elicited at least a low enhancement effect. A feature of decision tree-based models is that, they cannot extrapolate beyond the target range (T_{pot}) of the training set. Likewise, caffeine and benzoate were predicted in absolute terms to be more potent than experimentally measured, as no learning examples could suggest such a weak potency. Nonetheless, this is not of much practical concern in terms of predicting new permeation enhancer candidates. A useful model does not have to distinguish very weak enhancers from non-enhancers.

C10, sodium decanoate, is one of the most described enhancers in literature [7,9,32,33]. Amongst the reported mechanisms of C10 are phosphorylation cascades and intracellular calcium signalling leading to tight junction opening [9,33]. Such mechanisms are far too complex to be captured from a training sample of this size. Conceivably, this may be why C10 was predicted to be a mediocre permeation enhancer, yet elicited a relatively stronger potency (Fig. 2B), caused by components not captured by the model. It is expected that doubling or tripling the size of the training set would improve prediction accuracy significantly. This would require testing another 40–80 permeation enhancers in three concentrations in Caco-2 monolayer.

Fig. 4, the feature contributions versus descriptor values of each training permeation enhancer, provides a novel and very useful way to learn from the random forest-model. For example, a molecule having a dipole > 3, a Vsurf_ID8 > 1 and a BCUT_SLOGP_0 < −2.7 and CCL > 10 would appear to bear promising starting point. Furthermore, the data in Fig. 4 suggested interaction for e.g. CCL > 10, only contributing positively conditioned when dipole > 3. That carbon chain length is only conditionally advantageous matches the general understanding of surfactant-like properties. Such simple rules can help to understand what modifications of an enhancer can be made without incurring a loss of potency. The abundance of partial charge related descriptors (see Table 2) was interpreted as a consequence of, that most surfactants have one or more polar domains neighbouring carbon hydride domains and an induced dipole moment across the border [12].

The feature contributions technique represents a novel approach to data analysis and has the potential to be employed as a powerful explorative tool within many scientific areas. QSAR models based on algorithm models such as random forest are designed to map associations (not necessarily causal) between features and the target parameters to optimise predictions. It should be noted that this is also the case for classical statistical approaches [16]. Nevertheless, as discussed above, the suggestions from the feature contributions are plausible causal from a physicochemical point of view.

Other core aspects relating to oral protein formulation such as solubility, stability and metabolism are not encompassed in the existing approach. Thus, their inclusion is necessary in order to yield a fully predictive model of protein permeation. When designing/screening for new enhancers as excipients in protein-based drug formulations, various other requirements, such as solubility, should be considered.

Thus, by applying the described *in silico* model an *a priori* prediction of the permeation enhancer potency of a surfactant can be determined based upon its structure and hence obviate the need for extensive permeability screening of novel compounds.

6. Conclusions

Random forest-QSAR modelling utilising molecular descriptors calculated from the molecular structure was shown useful for predicting permeation enhancer potency. Although absorption of proteins is a complex biologic phenomena, the surfactant-like properties of permeation enhancers comprise a relatively manageable component.

Sparse data combined with the biological noise (unexplained) component is a challenge to build a robust predictive model. To reduce the estimation error, the prediction challenge was alleviated in two ways:

(1) TEER readings of three concentration levels were joined into a single value target (T_{pot}) to create an approachable modelling question: Is the potency of a new surfactant-like enhancer high, medium or low?
(2) Filtering of correlated descriptors to reduce redundant information and to remove descriptors with no univariate correlation to target parameter was performed to avoid too many descriptors being progressed to the random forest model with few training examples.

From the validations employed i.e., internal cross-validation, experimental validation and literature validation, the model was found to predict potency of permeation enhancers. Furthermore, it was possible to extract common structural features for high potency enhancers. Such knowledge is useful to assess the credibility of the built model and/or inspire our understanding of what makes a surfactant-like permeation enhancer potent.

Hereby, we have outlined how to robustly perform *in silico* screening for permeation enhancers with non-linear random forest, with the possibility to assess and learn from the model. The provided QSAR model forms a good basis for a systematically approach for the development of oral therapeutics formulated with potent permeation enhancers.

Acknowledgments

Christian Vind assisted with molecular descriptors and Sten B. Christensen assisted with literature search.

This work, a part of an industrial Ph.D project for Søren Welling, was granted by The Danish Agency for Science, Technology and Innovation and the company Novo Nordisk A/S.

Søren Welling, Hanne Refsgaard, Stephen Buckley and Lars Hovgaard are employees and/or shareholders of Novo Nordisk A/S.

Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.ejpb.2015.05.012.

References

[1] B.J. Aungst, Absorption enhancers: applications and advances, AAPS J. 14 (2012) 10–18.
[2] H.H.F. Refsgaard, B.F. Jensen, P.B. Brockhoff, S.B. Padkjaer, M. Guldbrandt, M.S. Christensen, In silico prediction of membrane permeability from calculated molecular parameters, J. Med. Chem. (2005) 805–811.
[3] B. Fredsted, P.B. Brockhoff, C. Vind, S.B. Padkjær, H.H.F. Refsgaard, In silico classification of solubility using binary k-nearest neighbor and physicochemical descriptors, Mol. Inform. 26 (2007) 452–459.
[4] D.S. Palmer, N.M. O'Boyle, R.C. Glen, J.B.O. Mitchell, Random forest models to predict aqueous solubility, J. Chem. Inf. Model. 47 (2007) 150–158.
[5] L.D. Hughes, D.S. Palmer, F. Nigsch, J.B.O. Mitchell, Why are some properties more difficult to predict than others? A study of QSPR models of solubility, melting point, and log P, J. Chem. Inf. Model. 48 (2008) 220–232.
[6] A. Fassano, S. Uzzau, Modulation of intestinal tight Junctions by zonula occludens toxin permits enteral administration of insulin and other macromolecules in an animal model, J. Clin. Invest. 99 (1997) 1158–1164.
[7] M. Tomita, M. Hayashi, S. Awazu, Absorption-enhancing mechanism of EDTA, caprate, and decanoylcarnitine, J. Pharm. Sci. 85 (1996) 608–611.
[8] S. Maher, L. Feighery, D.J. Brayden, S. McClean, Melittin as an epithelial permeability enhancer I: investigation of its mechanism of action in Caco-2 monolayers, Pharm. Res. 24 (2007) 1336–1345.
[9] W.J. Xia, H. Onyuksel, Mechanistic studies on surfactant-induced membrane permeability enhancement, Pharm. Res. 17 (2000) 612–618.
[10] M.J. Rosen, Surfactants and Interfacial Phenomena, third ed., Hoboken, New Jersey, 2000.
[11] Z.W. Wang, D.Y. Huanga, G.Z. Li, X.Y. Zhang, L.L. Liao, Effectiveness of surface tension reduction by anionic surfactants-quantitative structure-property relationships, J. Dispers. Sci. Technol. 24 (2003) 653–658.
[12] J. Hu, X. Xhang, Z. Wang, A review on progress in QSPR studies for surfactants, Int. J. Mol. Sci. 11 (2010) 1020–1047.
[13] K. Whitehead, S. Mitragotri, Mechanistic analysis of chemical permeation enhancers for oral drug delivery, Pharm. Res. 25 (2008) 1412–1419.
[14] K. Whitehead, N. Karr, S. Mitragotri, Safe and effective permeation enhancers for oral drug delivery, Pharm. Res. 25 (2008) 1782–1788.
[15] B. Sarmento, F. Andrade, S.B. da Silva, F. Rodrigues, J. das Neves, D. Ferreira, Cell-based in vitro models for predicting drug permeability, Expert Opin. Drug Metab. Toxicol. 8 (2012) 607–621.
[16] G. Shmueli, To explain or to predict?, Stat Sci. 25 (2010) 289–310.
[17] L. Breiman, Random forests, Mach. Learn. 45 (2001) 5–32.
[18] A. Liaw, M. Wiener, Classification and regression by randomForest, R News 2 (2002) 18–22.
[19] SciFinder Scholar, version 2014, Chemical Abstracts Service. Columbus, OH, 2014; RN (multiple look-ups, +50).
[20] Molecular operating environment (MOE), 2012-2013.8, Chemical Computing Group Inc., 1010 Sherbrooke St. West, Suite #910, Montreal, QC, Canada, H3A 2R7, 2013.
[21] G. Cruciania, P. Crivorib, P.A. Carruptb, B. Testab, Molecular fields in quantitative structure–permeation relationships: the VolSurf approach, Theochem 503 (2000) 17–30.
[22] D.J. Linvingstone, E. Rahr, CORCHOP – an interactive routine for the dimension reduction of large QSAR data sets, Quant. Struct. – Act. Relat. 8 (1989) 103–108.
[23] A. Palczewska, J. Palczewskiy, R.M. Robinson, D. Neagux, Interpreting random forest classification models using a feature contribution method, in: T. Bouabana-Tebibel, S.H. Rubin (Eds.), Advances in Intelligent Systems and Computing: Integration of Reusable Systems, Springer, Heidelberg, 2014, pp. 193–218.
[24] V.E. Kuz'min, P.G. Polishchuk, A.G. Artemenko, S.A. Andronati, Interpretation of QSAR models based on random forest methods, Mol. Inform. 30 (2011) 593–603.
[25] R. Hayeshi, C. Hilgendorf, P. Artursson, P. Augustijns, B. Brodin, P. Dehertogh, et al., Comparison of drug transporter gene expression and functionality in Caco-2 cells from 10 different laboratories, Eur. J. Pharm. Sci. 35 (2008) 383–396.
[26] Z. Lu, W. Chen, A. Viljoen, J.H. Hamman, Effect of sinomenine on the in vitro intestinal epithelial transport of selected compounds, Phytother. Res. 24 (2010) 211–218.
[27] U. Werner, T. Kissel, M. Reers, Effects of permeation enhancers on the transport of a peptidomimetic thrombin inhibitor (CRC 220) in a human intestinal cell line (Caco-2), Pharm. Res. 13 (1996) 1219–1227.
[28] S. Michael, M. Thöle, R. Dillmann, A. Fahr, J. Drewe, G. Fricker, Improvement of intestinal peptide absorption by a synthetic bile acid derivative, cholylsarcosine, Eur. J. Pharm. Sci. 10 (2000) 133–140.
[29] P.P. Tirumalasetty, J.G. Eley, Permeability enhancing effects of the alkylglycoside, octylglucoside, on insulin permeation across epithelial membrane in vitro, J. Pharm. Pharm. Sci. 9 (2006) 32–39.
[30] M. Sakai, T. Imai, H. Ohtake, H. Azuma, M. Otagiri, Effects of absorption enhancers on the transport of model compounds in Caco-2 cell monolayers: assessment by confocal laser scanning microscopy, J. Pharm. Sci. 86 (1997) 779–785.
[31] D.Z. Liu, E.L. Lecluyse, D.R. Thakker, Dodecylphosphocholine-mediated enhancement of paracellular permeability and cytotoxicity in Caco-2 cell monolayers, J. Pharm. Sci. 88 (1999) 1161–1168.
[32] S. Maher, T.W. Leonard, J. Jacobsen, D.J. Brayden, Safety and efficacy of sodium caprate in promoting oral drug absorption: from in vitro to the clinic, Adv. Drug Deliv. Rev. 61 (2009) 1427–1449.
[33] T. Lindmark, Y. Kimura, P. Artursson, Absorption enhancement through intracellular regulation of tight junction permeability by medium chain fatty acids in Caco-2, J. Pharmacol. Exp. Ther. 284 (1998) 362–369.

CHAPTER 5

Interpretation of random forest models

5.1 Modeling with random forest

The random forest model started to play a bigger role for this thesis as it showed to outperform linear regularized regression models as partial least squares and elastic net. Also RF seem to outperform the non-linear k-nearest neighbor. The random forest algorithm is certainly not always the best choice of a model, the non free lunch theorem states that there is no optimal silver bullet. That said Random forest together with radial support vector machine and gradient boosting was in comparison on 110 data sets found to be the in generel top performing algorithm in terms of cross-validated accuracy [**wainer2016comparison**]. If the underlying model generating the data is truly linear, e.g. in octane concentration determination in near-infrared light spectroscopy [**kalivas1997two**], then linear regression well perform better than random forest.

5.2 One interpretation of interactions

Throughout the work of this thesis, there have been an emphasis on identifying and visualizing interactions. In this process, I have developed my own definitions of interactions, that especially relate to geometrical interpretations of model structures. In our paper *Forest Floor Visualizations of Random Forests* we introduce an interaction as:

> *"Interactions in the model structure mean that the model predictions in part rely on the interplay on two or more features. Thus, the interaction parts of a model structure cannot be reduced to additive scoring rules, one for each feature. Likewise, to plot single feature-to-prediction relationships is not a sufficient context for visualizing any interactions."* [Wel+16]

I would like to elaborate on this definition of interactions. First definitions of the model structures, feature spaces and prediction spaces are needed. A given model

structure [1] f maps from feature space X to a prediction space $\hat{y}$, such that $\hat{y} = f(X)$. X is a real valued euclidean vector space, constituted by d axes x_1,...,x_d. Any point is a unique combination of feature values. For regression[2] the prediction space $\hat{y}$ is simply an one-dimensional real axis. We understand f as a function which connect any point in the feature space to some point in prediction space. To discuss how to describe interactions, I will introduce a simple structure. We can imagine the surface of a regression function f in a Eucledian 3D space, for two features x_1 and x_2 spanning the horizontal plane and one vertical prediction axis. We call this joint feature space and prediction space for the mapping space. We can imagine the structure of f is a geographical landscape model of altitude as function of coordinates. This f mapping has potentially valleys and mountains. Essentially, if f can be reduced into separate additive score functions, we can for any (x_1,x_2) coordinate position in this landscape not only use f to calculate the altitude, but also f decomposed into f_1 and f_2, such that $\hat{y} = f(X) = f_1(x_1) + f_2(x_2)$. When the mapping space is only 3D, it is not obvious, why we want to decompose f. However, for higher dimensional models, we cannot directly visualize nor comprehend the geometrical structure. By decomposing the model structure we allow ourselves to split the full structure in to simpler pieces we can visualize and understand. Unfortunately not all landscapes can be reduced to such additive scoring rules. Returning to the altitude model f, imagine a landscape model of a single pyramid surrounded by a flat dessert, see Figure 5.1. A function describing the local height of this landscape would reflect, that only when x_1 AND x_2 match the position of the pyramid, then the predicted altitude should rise above the sands. To decompose this topological map into f_1 + f_2 would analogously correspond to two observers on either side of the pyramid, where neither can see the entire pyramidn but only their respective front sides. Either observer would notice that in the middle of their respective fields of view, the altitude is elevated due to the pyramid. Each observer could generalize what they see to a partial function parallel to their field of view. However, as they cannot see their back side of the pyramid, neither of the observers will know if this is generally true for any position on the other axis. If we simply combined by averaging the altitude predictions of the two naive observers, the model structure would more look like the roof of a tower, not a pyramid.

In the mini review (letter) of boulesteix *et al* points out that the definition of interations can be vague and ambiguous in recent litterature [Bou+14]. They refer to the simple classical statistical logit-model, that contains a well defined two-feature saddle point interaction. Here f is decomposed such that $\hat{y} = f(X) = P(\beta_1 x_1 + \beta_2 x_2 + \beta_{12} x_1 x_2)$ where P is the logit transfer function $P(z) = log(\frac{z}{1-z})$. The logit function is used to let $\hat{y}$ describe the probability of some binary outcome. We could generalize and say P could be any polynomial function. An interaction term stated as the product of

[1] Throughout Chapter 3, f has referred to the underlying function to match the notation of Figure 3.1[Abu12]. However, in this chapter f refers to the model structure to match the notation of the forestFloor article [Wel+16].

[2] See the forestFloor article [Wel+16] or this how to handle categorical features, an idea originally from Friedmans gradient boosting and partial dependence article [Fri01].

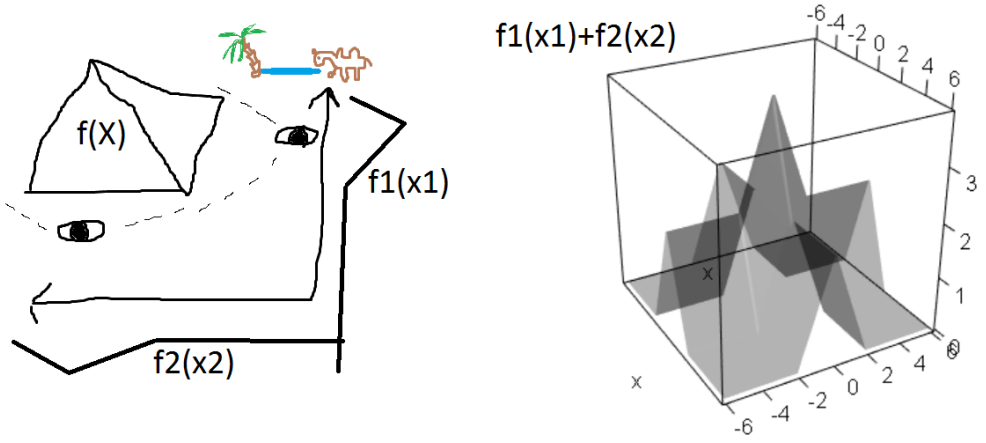


Figure 5.1: Left, pyramid in flat sands, an example of a model structure with non-global non-saddle interaction. XY-plane describes the coordinates. Z-axis is the altitude. Two naive observers f1 and f2 cannot approximate the shape of the pyramid alone. Right, approximated shape of the pyramid if f1 and f2 are combined additively..

two features is called a saddle point structure, as the shape resembles a saddle. Even when the product of two features is wrapped in a transfer function P, this interaction is a saddle point. The transfer function P can be seen as a compression/rearrangement of the $\hat{y}$ prediction axis. The limitation of the saddle point structure is that it can neither fit the shape of the pyramid, because the pyramid is a local interaction in an otherwise flat dessert sand landscape. My postulate is that there is no combination of compressing, stretching or looping of the $\hat{y}$ axis, which can morph a saddle point structure into the pyramid structure. Therefore, I postulate that logistic models comprised by the coefficient weighted sum of main effects and a number of saddle point interactions and any transfer function P is not a suitable basis for decomposing any model structure. There are structures, which do not fit this scheme. For any f_{12} there is not necessarily a P such that $f_{12}(x_1, x_2) = P(\beta x_1 + \beta x_2 + \beta_{12} x_1 x_2)$.

To decompose any possible landscape, including the pyramid, let $f(X) = f_1(x_1) + f_2(x_2) + f_1 2(x_1, x_2)$. If $f_{12} = f$ then not much have been accomplished. In fact their are a lot of really unuseful decompositions of f. In general it would be helpful to describe as much of the structure as possible as additive main effects and only what absolutely necessary as a higher order interaction. Let us imagine the pyramid is placed in a landscape steadily descending towards a ocean. Then the f_1 and/or f_2 can describe this general descending trend, while f_{12} describe the pyramid.

A non-linear regression model of d features can be decomposed into bias/offset, d main effects, $d-1$ second order effects, and a number of higher order effects. The type

of decomposition we aim for is one, that explains as much of the structural features in lower order effects. In the following, I derive that the number of possible components approximately doubles per feature.

For a model of d features there will be effects of d different orders. The smallest of first order (main effect) and the highest of d^{th} order. The number of combinations to draw i features out of d-features can be described as binomials. Summing all binomial counts from $i = 1$ to d give the number possible components/effects of the system. Therefore, n_d the number of possible effects are calculated as, $n_d = \sum_{i=1}^{d} \binom{i}{d}$. Furthermore the number n_d can be defined by n_{d-1} as, $n_d = 2n_{d-1} + d$. For $d = 10$ there are staggering 2036 possible components effects. Suddenly one may get sweaty hands, when to learn there are so many different effects of a multivariate model with 10 features. The promise that decomposition of the model structure would bring simplicity and clarity seems far from fulfilled. Fortunately, we can ignore a high number of these components when emphasizing model structures trained by random forest or any other reasonable machine learning algorithm. As shown in supplementary materials for the forest floor article, random forest can only poorly fit a saddle point interaction of 3 orders and 4 orders are nearly impossible. We live in a reality where fifth order interactions always will be near impossible to robustly estimate with statistical models and finitely sized training sets. Adhering to the Occam's Razor guideline, that we should always pick the simplest model explaining the observed data, all effects of more than 2 or 3 orders can in practice be disregarded. Furthermore, a subset of features will often be used more frequently than others in splits in the trees of the random forest. Interaction terms can only be learned in a tree model, when one split by one feature is conditioned by another feature split upstream. As the most dominant features tend be the ones upstream, interactions in the model structure are most likely found in the model structure as second order interactions between one relatively dominant feature (high variable importance) and some other feature. Therefore, interactions between weak features can be disregarded at first. Therefore, the a random forest model can be decomposed into d main effects and a lower number of second order effects, where at least one feature likely have been the favorite splitting feature. To summarize, an interaction effect of i^{th} order is one that cannot be fully be described by any combination of lower order effects.

This broad definition of interaction effects can also cover classification and multi classification, where f map to a $K-1$ dimensional simplex probability space, where K is the number of classes. Second order interaction effects for a probabilitic classification model mapping to 3 classes($K = 3$) are visualized in the forest floor article, see figure 12 [Wel+16]. However, the surface of this second order interaction effect is not 3 dimensional as the pyramid example. For a second order interactions($i = 2$), the mapping space has $i+K-1 = 2+3-1 = 4$ dimensions. In the forest floor article a such 4D problem is visualized in Figure 12. Here a 2D triangular (K-1)-simplex diagram is used to describe the probability distribution of predictions, and different

color gradients in turn represent the different feature axes. [3]

Returning to pyramid example. Let's say f_1 described the gradual descending of the landscape towards the ocean. As f_1 is non-linear, it could describe a single sand dune, a local maximum, parallel to the beach. This local sand dune now have to stretch infinitely to both sides, such that for any x_2 the effect of f_1 is static. We can use this observation to make a bottom up definition of interactions also. Let S be a subset of all features X of the model. Let T be the complimentary subset of features. An interaction effect by subset S, f_S, has the order equal to size of $|S|$. We would know that f_S is an adequately detailed decomposition of f, as for every parallel plane spanned by the features in S, the curvature of f_S align with f. In contrary when the interaction curvature is far from the same by any combination of feature vales of T, we know we are missing an important interaction with some feature(s) from T. Furthermore, that feature of T, by which the S-plane curvature changes the most, is the one that need to be included in S, in order to obtain a better generalized effect. This bottom up definition of interaction effects lend its notion of complimentary feature subsets S and T from Friedmans paper on gradient boosted machines and partial dependence plots [Fri01]. What I effectively state here is, that if a partial dependence plot of subset S is a poor generalization of f, then an interaction with one or more features from T are missing.

5.3 Article 3: *Forest Floor Visualizations of Random Forests*

First version submitted to arXiv.org the 30^{th} of May 2016. Latest version (3^{rd}) submitted to arXiv.org the 4^{th} of July 2016.

[3] If number of classes $K >= 5$ is higher than 5, the (K-1)-simplex diagram would require a 4D or higher visualization. Instead Figure 10 of article depicts how to plot probability predictions by each class by a single axis.

Forest Floor Visualizations of Random Forests

Soeren H. Welling[1,2], Hanne H.F. Refsgaard[2], Per B. Brockhoff[1] and Line H. Clemmensen*[1]

[1]Department of Applied Mathmatics and Computer Science, Technical University of Denmark, Matematiktorvet, Building 324, 2800 Kgs. Lyngby, Denmark
[2]Novo Nordisk Global Research, Novo Nordisk Park 1, 2760 Maaloev, Denmark

July 5, 2016

Abstract

We propose a novel methodology, forest floor, to visualize and interpret random forest (RF) models. RF is a popular and useful tool for non-linear multi-variate classification and regression, which yields a good trade-off between robustness (low variance) and adaptiveness (low bias). Direct interpretation of a RF model is difficult, as the explicit ensemble model of hundreds of deep trees is complex. Nonetheless, it is possible to visualize a RF model fit by its mapping from feature space to prediction space. Hereby the user is first presented with the overall geometrical shape of the model structure, and when needed one can zoom in on local details. Dimensional reduction by projection is used to visualize high dimensional shapes. The traditional method to visualize RF model structure, partial dependence plots, achieve this by averaging multiple parallel projections. We suggest to first use feature contributions, a method to decompose trees by splitting features, and then subsequently perform projections. The advantages of forest floor over partial dependence plots is that interactions are not masked by averaging. As a consequence, it is possible to locate interactions, which are not visualized in a given projection. Furthermore, we introduce: a goodness-of-visualization measure, use of colour gradients to identify interactions and an out-of-bag cross validated variant of feature contributions.

1 Introduction

We propose a new methodology, forest floor, to visualize regression and classification problems through feature contributions of decision tree ensembles such as random forest (RF). Hereby, it is possible to visualize an underlying system of interest even when the system is of higher dimensions, non-linear, and noisy. 2D or 3D visualizations of a higher-dimensional structure may lead to details, especially interactions, not being identifiable. Interactions in the model structure mean that the model predictions in part rely on the interplay on two or more features. Thus, the interaction parts of a model structure cannot be reduced to additive scoring rules, one for each feature. Likewise, to plot single feature-to-prediction relationships is not a sufficient context for visualizing any interactions. Often a series of complimentary visualizations are needed to produce an adequate representation. It can be quite time consuming to look through any possible low dimensional projection of the model structure to check for interactions. Forest floor guides the user in order to locate prominent interactions in the RF model structure and to estimate how influential these are.

For RF modeling, hyper parameter tuning is not critical and default parameters will yield acceptable model fits and visualizations in most situations [10, 23]. Therefore, it is relatively effortless to train a RF model. In general, for any system where a model has a superior prediction performance, it should be of great interest to learn its model structure. Even within statistical fields, where decision tree ensembles are far from standard practice, such insight from a data driven analysis can inspire how to improve goodness-of-fit of a given model driven analysis.

Although the RF algorithm by Breimann [3] has achieved the most journal citations, other later decision tree ensemble models/algorithms such as ExtraTrees [14], conditional inference forest [8], Aborist [21], Ranger [26] and sklearn.random.forest [17] will often outperform the original RF on either prediction performance and/or speed. These models/algorithms differ only in their software im-

*lkhc@dtu.dk

plementation, split criterion, agreggation or in how deep the trees are grown. Therefore all variations are compatible with the forest floor methodology. Another interesting variant, rotation forest [19], does not make univariate splits and is therefore unfortunately not directly compatible with forest floor visualizations. To expand the use of feature contributions and forest floor, we also experimented with computing feature contributions for gradient boosted trees [6]. This is possible, as splits still are univariate and trees contribute additively to the ensemble prediction. A proof-of-concept of computing feature contributions on gradient boosted regression trees and visualizations are provided in supplementary materials.

Decision trees, as well as other machine learning algorithms, such as support vector machines and artificial neural networks can fit regression and classification problems of complex and noisy data, often with a high prediction performance evaluated by prediction of test sets, n-fold cross validation, or out-of-bag (OOB) cross validation. The algorithms yield data driven models, where only little prior belief and understanding is required. Instead, a high number of observation are needed to calibrate the adaptive models. The models themselves are complex black-boxes and can be difficult to interpret. If a data driven model can reflect the system with an impressive prediction performance, the visualization of the model may deduce knowledge on how to interpret the system of interest. In particular, a good trade-off between generalization power and low bias is of great help, as this trade-off in essence sets the boundary for what is signal and what is noise. The found signal is the model fit, which can be represented as the mapping from feature space to prediction space (output, target, response variable, dependent variable, y). The noise is the residual variance of the model. The estimated noise component will both be due to random/external effects but also lack of fit.

1.1 Overview of the article

In this article we introduce the forest floor methodology. The central part is to define a new mapping space visualization, forest floor. Forest floor rely on the feature contributions method [9][16], rather than averaging many projections (partial dependence) [6] or projecting the average (sensitivity analysis) [5]. In Section 1.2 these previous mapping space visualizations are introduced and the challenges to overcome are discussed. In the theory section, 2.1, we discuss the feature space, prediction space and the joined mapping space for any regression or classification model and define local increments as vectors in the prediction space. Properties of the RF algorithm by Breimann [3] and the feature contributions method by Kuz'min *et al* [9] and Palczewska *et al* [16] are highlighted and illustrated in section 2.2. In section 2.3 we argue that the prediction of any node in any tree is a point in the prediction space and the local increments are the vectors that connect the nodes of the trees. Any prediction for any observation is basically a summed sequence of local increments plus the grand mean or base rate. Since local increments are vectors and not a tree graph, the sum of vectors is not dependent on the order of the sequence. In Section 2.4 we show how that feature contributions, a particular reordering of local increments by splitting feature, can be used to decompose the model structure 2.4. We also introduce a new cross-validated variant of feature contributions and provide an elaborated definition of feature contribution to also account exactly for the bootstrapping process and/or stratification.

The materials and methods sections, 3.1 and 3.2, provide instructions on how to reproduce all visualization in this paper. The result section 4 is dedicated to three practical examples of visualizing models with forest floor. The three examples are a simulated toy data set, a regression problem (white whine quality) and a classification problem (contraception method choice). A low-dimensional visualization is not likely to convey all aspects of a given RF mapping surface. For all practical examples, we describe how to find an adequate series of visualizations that do.

1.2 Representations of random forest models

A RF model fit, like other decision tree based models, can be represented by the graphs of the multiple trees. Few small tree graphs can be visualized and comprehended. However, multiple fully grown trees are typically needed to obtain an optimal prediction performance. Such a representation cannot easily be comprehended and is thus inappropriate for interpretation of model fits. A random forest fit can be seen as a large set of split rules which can be reduced to a smaller set of simpler rules, when accepting a given increase in bias. This approach has been used to reduce the model complexity [13]. But if the minimal set of rules still contains a large number, e.g. hundreds or thousands, then this simplified model fit is still incomprehensible. It is neither certain which rules have influence on predictions nor which rules tend to cancel each other out. We believe that the rule-set or tree-structure representations are mainly appropriate to understand how a RF algorithm possibly can model data. On the other hand, these representations are indeed inappropriate for interpreting RF model fits and conveying the overall model structure. For that pur-

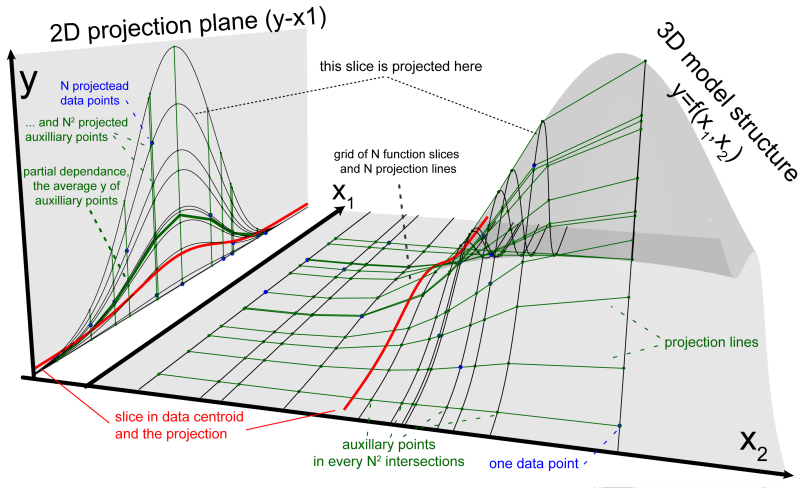


Figure 1: Illustration of sensitivity analysis and partial dependence plots. The grey response surface depicts a given learned model structure of two input features (X_1 and X_2) and one prediction axis ($\hat{y}$). 11 data points vs. predictions are depicted as blue dots. 1D-sensitivity analysis (fat red lines): one partial function slice intersects the centroid where $X_2 = \overline{X}_2$ an is projected to the X_1-y plane. ICE plot: Multiple function slices (black lines) all parallel to $X - 1$ intersects each one data point and all slices are projected to the X_1-y plane. Partial dependence plots: Each data point intersected by one black line is projected to any black lines (green points). The green point outline a grid. All green and blue points are projected into the X_1-y plane, and the fat green line connects the average prediction values as a function of $X - 1$. This illustration can be generalized to any dimensional reduction.

pose, a mapping space visualization is superior in terms of visualization and communication.

If we join the feature space and prediction space, this function will be represented as a geometrical shape of points. Each point represents one prediction for a given feature combination. This geometrical shape is the model structure and is an exact representation of the model itself. Nevertheless, for a given d-dimensional problem where $d > 3$, this is still difficult to visualize or even comprehend. Instead, one may project/slice or decompose the high-dimensional mapping into a number of marginal visualization where small subsets of features can be investigated in turns. This allows us to comprehend the isolated interplay of one or a few features in the model structure.

Following, we will introduce previous examples of mapping space visualizations to specify what forest floor aims to improve. Different types of sensitivity analysis (SA) were used by Cortez and Embrechts to make such investigations [5], we will here discuss sensitivity analysis and data based sensitivity analysis. First a supervised machine learning model is trained. Next the model is probed. That means to input a set of simulated feature observations (points in feature space) into the model fit and record the output (target predictions). Instead of probing the entire high-dimensional mapping space, only one confined slice of fewer dimensions is probed in order to make feasible visualizations.

The simplest visualization in SA is one dimensional (1D-SA), where a single feature is varied in a range of combinations, and this range will span the X-axis of the visualization. When two features are varied (2D-SA), the resulting grid of combinations will span the XY-plane. All other features must be fixed at e.g. the mean value, the feature centroid of the training set. The model fit is probed with these observations and the resulting predictions will be plotted by the Z-axis. The obtained line/surface will now visualize one particular 2D or 3D slice of the full mapping structure.

In figure 1, a non-linear regression model structure ($y = sin(X_1)^8 sin(X_2)^8 + \epsilon$) is represented by the grey transparent surface. The model has two feature axes in the horizontal XY-plan and the prediction axis by the vertical Z-axis. Thus, the mapping space has 3 dimensions and the model structure is some curved 2D-surface which connect any given feature combination with one prediction. The red line/slice in the model structure is the example of an 1D-SA visualization. This single slice is projected into the X_1-Z plane. This 1D-SA projection portrays the partial effect of feature X_1 in the special case, where other features are set to mean observed value. Notice that the red line almost completely misses the local hill in the model structure. A single low dimensional slice of the mapping structure can easily miss prominent local interactions, when number of model dimensions is high.

A 2D-SA slice can explain a main effect and/or the possible interaction within two selected features. Figure 1 only illustrates a 1D-SA slice projection, but represents the idea of any projection. The depicted model structure itself could infact be a 2D-SA projection of a higher dimensional model structure. Whether a given slice is a good generalization of the full mapping structure is unknown. A good generalization means that any parallel slices, where the fixed features are set to another combination, yield the same XYZ-visualization, with only perhaps a fixed offset in the prediction axis (Z) [7]. We will for now term that such visualization has a high goodness-of-visualization. In section 2.4 we will propose a metric for goodness-of-visualization. For a data structure with only additive effects and no interactions, the obtained model mapping structure is likely to have no interactions as well as any slice will be identical to its many parallel counterparts. In Figure 1, all the black parallel slices to the red slices give different projection lines in the mirror plane which could not be corrected by a simple offset. Therefore the model structure must have an interaction which cannot be seen in this projection alone. The iceBOX package displays multiple projection lines to search for masked interactions and is a good alternative to the forest floor approach [7].

A second concern is whether a given slice or slices extrapolate the training data. For a RF model with a satisfactory cross validated prediction performance, the mapping structure will represent the underlying data structure, but only within the proximity of the training data. Extrapolated areas of the mapping structure are far from guaranteed to represent an underlying data structure. Several different non-linear learners (RF, SVM, ANN, etc.) may easily have comparable model structures in the proximity to training data points, whereas far from the training set the models will heavily disagree. For RF models containing dominant interaction effects, the mapping structure on the borders of the training data becomes noise sensitive, as decision trees only can extrapolate parallel to feature axes, as the splits only are univariate. RF models only containing additive main effects have stable and smooth mapping structure at the borders of the training data. Model extrapolation of random forests with dominant interaction effects have been illustrated in supplementary materials.

SA plots remain a useful tool. When forest floor yield plots of similar structure, these plots generally represents the model mapping well. Visualization of multiple parallel projections, the so called ICE plots (individual conditional expectation) with the

ICEbox package, can also reveal interactions. However multiple projection lines cannot directly filter out main effects by other features. These will tend to offset the projection lines on the prediction axis. Centered ICE (c-ICE) visualizations do adjust this offset by centering the prediction axis for all projections in one specific location [7].

A frequently used visualization method proposed by Friedman is the partial dependence plot (PD) which is the same as what Cortez and Embrechts later have termed data-based sensitivity analysis (DSA)[5, 6]. In Figure 1, the green fat line in the mirror plane represents a partial dependence projection. Whereas 1D-SA and 2D-SA only project the slice intersecting e.g. the training data centroid, the partial dependence plot projects multiple slices. Each projected slice intersects one data point. The partial dependence line is the average prediction values of all slices. Thus, the obtained PD visualization summarizes all parallel slices of the mapping structure by averaging. To summerize, SA averages and then projects, whereas PD projects and then averages. ICE-plot projects many slices and do not aggregate. The PD approach may improve generalization across slices as it up-weighs the parts of mapping structure, that are well represented by data points. Still, interactions between varying and fixed features will be lost by averaging. Furthermore, the PD projections form a regular data grid spanned by the data observations. See the grid of black and green lines on the model structure surface in Figure 1. However, for data sets with high feature collinearity, data points will mainly be positioned in one diagonal of the grid, whereas the remaining part of the grid will span extrapolated parts of the model structure. This extrapolation occur for both SA, PD and ICE-plots.

Feature contributions was introduced by Kuz'min [9] for RF regression and elaborated by Palczewska *et al* [16] to also cover RF multi-classification. Feature contributions are RF predictions split into components by each feature. Feature contributions are essentially computed utilizing information from the tree networks of a RF model. Feature contributions have not before been used or understood in conjunction with the idea of function mapping structures. The contribution of this paper, is to show that feature contributions can be understood as a different way of slicing the mapping structure. From this insight the methodology, forest floor, was developed.

We have developed a number of tools to increase the usefulness of the forest floor methodology. These are: Out-of-bag cross validated feature contributions to increase robustness without increasing computation time, goodness-of-visualization tests to evaluate how well slices generalize the mapping structures and color gradients traversing mapping space to visually identify latent sources of interactions. Furthermore, the methods have been implemented as a freely available R-package, from which all mapping visualizations of this paper originate. The R-package forestFloor [25] aims to assist the user visualizing a given RF model fit through a serious of appropriately chosen visualizations.

2 Theory and calculation

Here is provided a new notation for RF regression and classification to combine a mapping space representation with the feature contributions method developed by Kuz'min [9] and Palczewska *et al.* [16]. Moreover to obtain an exact decomposition of the model structure, we expand the previous notion of feature contribution to also cover the initial bootstrap and/or stratification step for each decision tree. For RF multi-classification we describe a probabilistic (K-1)-simplex prediction space, to improve the interpretation of feature contributions. Lastly we introduce how to calculate out-of-bag cross-validated feature contributions.

2.1 Defining regression and classification mappings

Any regression model f_r can be seen as a mapping between a d-dimensional feature space $X \in \mathbb{R}^d$ and and a prediction scale $\hat{y} \in \mathbb{R}^1$

$$\hat{y} = f_r(X) \quad , \tag{1}$$

where X represents the infinite set of points in the feature space. A subset of points in X can be notated as e.g. X_t where t is a defined set. Single value entries of a countable subset of X is notated as x_{ij} where $i \in \{1, ..., N\}$ (N points) and $j \in \{1, ..., d\}$ (d features). $\hat{y}$ represents the entire prediction scale, where $\hat{y}_t$ could be a subset, if countable with point entries $\hat{y}_i$.

The entire mapping can be represented as a d-dimensional (hyper)surface S in a $d+1$-dimensional mapping space V. S can be understood as a learned model structure trained on a set of training observations, t. Obviously, if $d \in \{1, 2\}$, then S can conveniently be plotted by Cartesian axes as a 2D function plot or a 3D response surface (prediction as function of two features). Each label of a categorical feature can be assigned an integer value from 1 to K' categories and thus also be plotted.

A classification model can be seen as a mapping from $X \in \mathbb{R}^d$ to $\hat{y} \in \{1, 2, ..., K\}$. Some models, as RF, provides a probabilistic prediction (pluralistic voting) of class membership $\hat{p}_k$ for any class $k \in \{1, 2, ..., K\}$ and assign the class membership hereafter. Thus, the probabilistic classification

model f_c is a mapping from X to the probability space P,

$$f_c(X) = P \quad . \tag{2}$$

Any point in P is a possible prediction $\hat{p}$ with a unique probability distribution over K mutually exclusive classes, such that $\hat{p} = \{\hat{p}_1, \hat{p}_2, ..., \hat{p}_K\}$. As class memberships are mutually exclusive, the sum of the class probabilities is always one, $|\hat{p}|^1 = 1$. Therefore the probability space is a *K-1* dimensional simplex [15], which contains any possible combination of assigned probabilities to K mutually exclusive classes, see Figure 2 . The K axes, which assign probability of 0 to 1, are not orthogonal, meaning it is not possible to modify the assigned probability of one class without affecting at least one other.

The classification mapping can be represented by simply joining the simplex-space with the feature space, but this would only allow a 2D or 3D visualization when $(d + K - 1) \in \{2, 3\}$, thus either maximally a 2 feature problem for 2 classes, or a 1 feature separation for 3 classes. Instead, this mapping can also be represented as K separate d-dimensional surfaces S_k in a $d + 1$-dimensional space V with d axes representing features and one axis $\hat{p}_k$ representing the probability of either of the K classes. Thus, we align the directions of all K probability axes to reduce the dimensionality of the mapping space with $K - 2$ dimensions. Then, any line parallel to the probability axis $\hat{p}_k$, will intersect every S_k surface, describing the predicted probability of the k^{th} class at this point of input features. The sum of predicted probabilities of all intersections for any such line will be equal to one. To summarize, multi classification model structures are more difficult to visualize, as each class adds another dimension to the mapping space. It is possible to plot the individual predicted probability of each class and overlay these plots. Figure 2 summarizes the mapping topology for regression, for binary classification, and for multi classification.

RF mapping for both regression and classification can jointly be defined as

$$\hat{y} = f(X) \quad . \tag{3}$$

Here $\hat{y}$ is the c-dimensional prediction space. For regression, $c = 1$, f maps to a 1-dimensional prediction scale. For classification, $c = K$ classes, and f maps to a prediction vector space, where the k^{th} dimension predicts the probability of class k. For classification the predictions $\hat{y}$ can be any point within the $(K - 1)$-simplex. On the other hand, the training examples y can only be of one class each, which are the K vertices (corners) of the $(K - 1)$-simplex.

We define a local increment vector, L, pointing from $\hat{y}_i$ to $\hat{y}_j$ in a prediction space of c dimensions, such that

$$L_{ij} = \hat{y}_j - \hat{y}_i = \{\hat{y}_{j1} - \hat{y}_{i1}, ..., \hat{y}_{jc} - \hat{y}_{ic}\} \quad . \tag{4}$$

For regression, where $(c = 1)$, the local increment is a scalar with either a positive or negative direction. For classification, $(c > 1)$, the local increment is a vector with c elements, one for each class. Each node of a RF model fit is a prediction, which is a specific point in the prediction space. Local increments are the connections between nodes, describing the change of prediction. Computing the thousands or millions of local increments for trees and nodes, and sum these individually for each observation and feature is essentially the feature contributions method.

2.2 Properties of random forest related to feature contributions

RF is an ensemble of bootstrapped decision trees for either regression or classification. Figure 3 illustrates how the RF algorithm operates for regression. For each of the trees (1 to n_{tree}) the training set is bootstrapped (random sampling with replacement). In average $(\frac{N-1}{N})^N \approx 0.37$ of N observations will not be included in each bootstrap. These observations are called out-of-bag (OOB). Thus for any tree, a selection of observations will be 'in-bag' and used to train/grow the tree starting from the root node. Any node will have a node prediction which is defined by in-bag observations in that node.

$$\hat{y}''_j = \frac{1}{n_j} \sum_{i=1}^{n_j} y_{ij} \tag{5}$$

For a regression tree, the node prediction of the j^{th} node $\hat{y}''_j$ is equal to the mean of in-bag target values in the j^{th} node. Where y_{ji} is the target value of the i^{th} observation in the j^{th} node. n_j is the number of observations in the j^{th} node. Thus we are only computing a node prediction from in-bag elements.

For classification, the probabilistic node prediction p_{jk} of the class k of the node j is equal to the number of in-bag observations of class k divided with total number of in-bag observations in the node.

$$\hat{p}_{jk} = \frac{n_{jk}}{n_j} \quad . \tag{6}$$

A node prediction $\hat{y}''_j$ can also describe all class probabilities at once as a vector corresponding to a point in the $(K - 1)$-simplex space.

$$\hat{y}''_j = \{\hat{p}_{(j,1)}, ..., \hat{p}_{(j,K)}\} \tag{7}$$

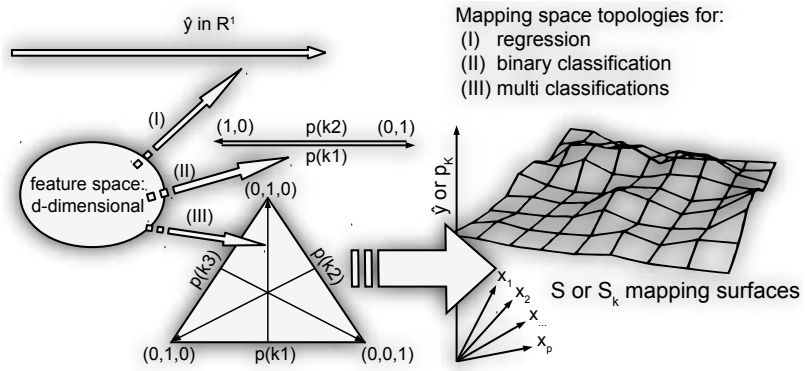


Figure 2: Topologies of random forest model represented as a function mapping from d-dimensional feature space to one of the following prediction spaces: (a) regression, 1-dimensional scale; (b) binary classification, $K = 2 - 1$ probability simplex reducible to a 1-dimensional probability scale; (c) multi-classification, probabilistic $(K - 1)$-simplex. The mapping can be represented as a high-dimensional surface S, in a joined feature and prediction space linking any combination of features to a given prediction. For multi-classification, S can be split into multiple S_k surfaces describing predicted probability for each of K individual classes.

For classification $c > 1$, the class probabilities of any node will always sum to 1 for any node:

$$|\hat{y}_j''|^1 = \sum_{k=1}^{K} p_{jk} = 1 \quad . \tag{8}$$

Therefore, the elements of any local increment vector for classification, see Equation 4 will always sum to zero. This is not true for the local increment scalars of regression, $c = 1$.

For an original RF implementation [10], predictions of terminal nodes of classification trees are reduced to a single majority vote. Other implementations such as sklearn.randomForestClassifier [17] would rather pass on the probabilistic vote from terminals nodes and only on the ensemble level perform reduction by majority vote or just keep the full probabilistic average. In practice, implementations of feature contributions usually have to re-estimate node predictions. A feature contributions implementation such as forest floor should match the specific rule of terminal node predictions of the specific model algorithm.

A node is by default terminal if there are 5 or less in-bag observations left for regression or a single in-bag observation for classification. Any non-terminal node will be split into two daughter nodes to satisfy a loss-function. For regression the loss function is typically the sum of squared residuals.

For classification, a Gini criterion is used as the loss function. That is to select the split yielding the lowest node size weighted Gini impurity. Gini impurity (g) is 1 minus the sum of squared class prevalence ratios in nodes, $g = 1 - \sum_{k=1}^{K} \hat{p}_{jk}^2$. Gini impurity is in fact the equation of a K-dimensional hypersphere, where $\sqrt{1-g}$ is the radius and all $\hat{p}_{jk}$ are the coordinates. The $(K-1)$-simplex space intersects this hypersphere where all prevalences sum to one, $1 = \sum_{k=1}^{K} \hat{p}_{jk}$. Therefore for a $K = 3$ classification, a Gini loss function isobar appear as a 2D-circle, when visualized in the $(K - 1)$-simplex space. One circular isobar is drawn in Figure 4. The Gini loss function chooses the split placing two daughter nodes the furthest from the center of the $(K - 1)$-simplex.

Splitting numerical features of ratio-, ordinal- or integer-scale is all the same for RF. A break point will direct observations lower or equal to the left node. Splitting by categorical features is to find the best binomial combination of categories designated for either daughter node. A feature with 8 categories will have $2^{8-1} - 1 = 63$ possible binary splits. Any available break point are evaluated by the loss-function, but the RF algorithm is constrained to only access a random selection of the features in each node. The amount of features available, *mtry*, can e.g. be a third of the total amount of features. This *random variables subspace* and bootstrapping will ensure decorrelation of trees and feature regularization without overly increasing the bias of each fit. Each fully grown tree is most likely highly overfitted, as the individual predictions of each terminal node are dictated by 5 or less observations. Combining the votes of many overfitted but decorrelated trees form an ensemble with lowered variance and without increased bias. Out-of-bag(OOB) predic-

tions are calculated for each terminal nodes. As OOB observations are not used actively in growing the trees of the forest, they can serve as an internal cross validation which yields similar results as a 5 fold cross validation [23]. The prediction of individual trees are written as $\hat{y}'_{ij}$ for $i \in \{1, ..., N\}$ observations predicted by $j \in \{1, ..., n_{tree}\}$. The ensemble predictions are computed as

$$\hat{y}_i = \frac{1}{n_{tree}} \sum_{j=1}^{n_{tree}} \hat{y}'_{ij} \quad , \tag{9}$$

and the OOB cross validated ensemble predictions $\tilde{y}_i$ are computed as

$$\tilde{y}_i = \frac{1}{|\tilde{J}_i|} \sum_{j \subseteq \tilde{J}_i} \hat{y}'_{ij} \quad , \tag{10}$$

where $\tilde{J}_i$ is the subset of $\{1, ..., n_{tree}\}$ trees, where i^{th} observation is OOB. $|\tilde{J}_i|$ is the size of the subset $\tilde{J}_i$. Thus let any training observation i iterate through the $\tilde{J}_i$ subset of trees, defined as those trees where i was not in-bag, and find the mean of terminal node predictions.

To obtain value/class predictions of new observations, the observations will be forwarded through all trees according to the established split rules. A tree prediction is dictated by the terminal node a given observation ends up in. The ensemble prediction of a RF model fit will by default be the average for regression and the majority vote for classification. Figure 3 explains graphically the structure of a single regression tree by feature x_1 and x_2. First all bootstrapped observations exist within the node n1. The mean prediction value of n1 is in this example 0.14 a slight offset compared to the training set prediction mean of 0. The first split is over a break point in x_2, dividing n1 into n2 with low prediction value and n3 with a high prediction value. Both n2 and n3 are further split by x_1. Interestingly, n2 and n3 have almost opposite splits by x_1. In n2, high x_1 leads to a lower prediction, while reversely in n3. This illustrated tree have only grown 7 nodes. Nonetheless, the tree contains an interaction term, where high $x1$ only contribute positively to the prediction $\hat{y}$ when conditioned by high x_2.

2.3 Local increments and feature contributions

This section explains how feature contributions are computed. This paper expands the feature contributions defined by Palczewska *et al* [16] to also account for bootstrapping and/or stratification and to allow OOB cross validation. Feature contributions summarize the pathways any observation (a given combination of input features) will take through the many decision trees in a RF model. Each sub node of the trees holds a prediction, which is average observed target of observations populating it, see Equations 5 & 6. The sum of the many steps from node to node (local increments) is for regression exactly the resulting large step from the grand mean of the training set to the given numeric target prediction. Likewise for classification, the large step is from base rate to a probabilistic target prediction. A proof hereof is provided in supplementary materials. As these many small steps towards the final prediction is an additive process, it is possible to reorder the sequence of steps and end up by the same prediction. The important implication hereof is that the RF model structure can be decomposed into additive sub models, each with the same dimensionality. As each sub model structure is the sum local increments of decision splits by one specific feature, each sub model structure tend to only describe the main effect of this one specific feature plus perhaps interactions with other features.

In order to efficiently describe how variations of feature contributions are computed, a notation of how to access any local increment in a given RF model fit is formulated. We define L as a list of lists of lists containing all local increments. L is defined in the following three levels (observations, trees, increments):

1. L_i is a list with $i \in \{1, ..., N\}$, and N is the number of observations predicted by the forest. i is the i^{th} observation.

2. Each element of L_i, called L_j is a list with $j \in \{1, ..., n_{tree}\}$, and n_{tree} is the number of trees in the ensemble.

3. Each element of L_j, called L_k is a list with $k \in \{1, ..., n_{increment,i,j}\}$, and $n_{increment,i,j}$ is the number of increments encountered by the i_{th} observation in the j^{th} tree.

Note that L can be ordered as a 2-dimensional array (i observation, j tree) where each element is a sequence of local increments specific for the i^{th} observation in the j^{th} tree. Overall, we can access any local increment in L with L_{ijk}. Depending on the model type, L will contain local increments as scalars for regression or as vectors for classification. The first local increment $k = 1$ for any tree and observation in L_{ijk} is the step from node 0 (training set) to node 1 (root node of tree). Thus the k^{th} local increment steps from the parent node $k-1$ to a daughter node k. The local increment L_{ijk} is the change of node prediction $\hat{y}''_{ijk} - \hat{y}''_{ij(k-1)}$

Equation 11 describes how any prediction can be computed from L_{ijk} as the sum of all local increments plus grand mean or base rate. A proof hereof can be found in the supplementary materials.

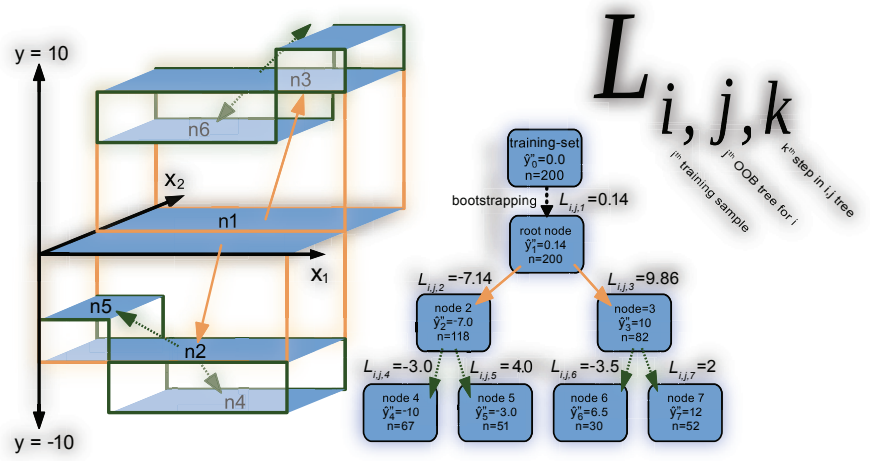


Figure 3: Random forest and local increments explained. Left, an 3D illustration of a small regression tree of 7 nodes. Right, the same tree described by node means($\overline{u}$), node size(n) and local increments L_{ijk}. L is subsetted by observation, tree, node and feature. A observation falling in e.g. node 4, will have a prediction as the sum of the local increments in its path plus the grand mean of the training set.

The target prediction $\hat{y}_i$ is computed as

$$\hat{y}_i = \frac{\sum_{j=1}^{n_{tree}} \sum_{k=1}^{n_{increment,i,j}} L_{ijk}}{n_{trees}} + \overline{y} \quad , \qquad (11)$$

where L_{ijk} is a local increment and where $\overline{y}$ is the grand mean or base-rate. The numerator is a scalar for regression and a vector for classification. The denominator, n_{tree}, is always a scalar.

So far the prediction of the i^{th} observation is the grand mean (regression) or the base-rate (classification) plus the sum of all local increments L_{ijk} encountered by this i^{th} observation divided by n_{trees}.

Figure 4 is a new geometrical representation of local increments for a 3-class classification. Figure 4 is not intended as a model structure visualization, but rather as a representation of how decision trees branch out in the prediction space. Each node in the classification tree can be seen as a probabilistic prediction defining a point in a probabilistic $(K-1)$-simplex. Figure 4 depicts node predictions and local increments for a small tree with four terminal nodes. To this tree graph is appended a node (T) for training set to the root node of the tree. This train node represents the class distribution of the training set. The bootstrap increment leads to the root node. This step is often small and a result of random uniform sampling w/o replacement. If applying class stratification, the length and direction of this step can be controlled. Stratification corresponds to defining a prior expected class distribution, which will be the position of the root nodes in the prediction space. From here all trees will branch out from this point. The following local increments and nodes comprise the entire tree. Any split produces two nodes and two local increments of opposite direction. If not of equal node size, there will be one shorter local increment defined of many in-bag observations and one longer local increment defined of fewer in-bag observations. This is a consequence of that class distributions of daughter nodes multiplied by the node sizes and added together is exactly equal to class distribution of parent node multiplied by its node size. This symmetry effect can be found in Figure 11 in section 4.3. For the unbalanced binary features *wives' religion*, *wives working* and *media exposure* the prediction is offset a lot for a few observations, while the prediction of remaining many observations will only change a little in the exact opposite direction. For regression and binary classification such a direction is essentially one-dimensional and can be positive or negative. For multi classification the direction is a vector of K elements with the restriction that the sum of elements is zero. In Figure 4, the circle represents a Gini loss function isobar. The further away (euclidean distance) nodes are placed from uniform class distribution the better a split according to RF Gini loss function. The best kind of split is one placing both daughter nodes onto two of the K vertices of the (K-1)-simplex.

For the training set, a cross validated OOB-

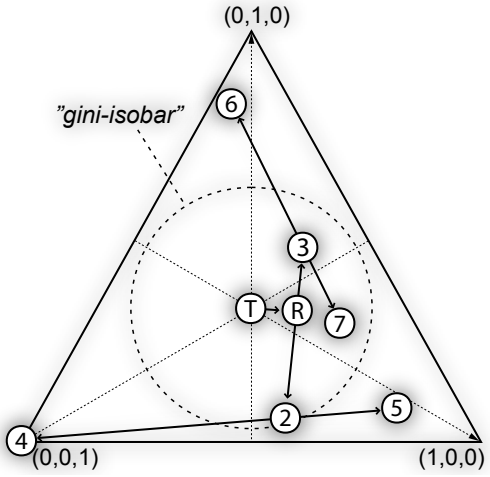


Figure 4: A representation of how node predictions and local increments for a small classification tree with four terminal nodes. The first node in center represents the class distribution of a balanced training set (T). The bootstrap increment leads to the root node of the tree (R). The following local increments and nodes comprises the entire tree. Any split produces two local increments of opposite direction. The circle represents Gini loss function isobar. The further the two nodes (weighted by size) are from uniform class distribution the better a split according to the Gini loss function.

prediction $\tilde{y}$ can be formulated as

$$\tilde{y}_i = \frac{\sum_{j \subseteq \tilde{J}_i} \sum_{k=1}^{n_{increments,i,j}} L_{ijk}}{|\tilde{J}_i|} + \overline{y} \quad , \qquad (12)$$

where $\tilde{J}_i$ is the subset of trees where i^{th} sample is OOB. One can reason, that if Equation 11 is true for any set of trees, then Equation 12 must also be true for a given subset of any trees, such as the OOB subset $\tilde{J}_i$, see supplementary materials.

When predicting the training set with an RF model, any training observation $i \in \{1, ..., N\}$ will have a high proximity to itself, that is, it will in any in-bag tree both define the in-bag node predictions of the terminal node and be predicted by the very same terminal node. For data sets with a high noise level this becomes a problem and the points S_i of model structure S will overfit the sampled training set observations T_i, and visualizations hereof will look more noisy. If the RF training parameter minimum terminal node size is increased and/or bootstrap sample size is lowered then training observation i will have a lower influence on its own prediction and visualizations will not look noisy.

To compute feature contributions, the summed local increments over each observation and feature, it is necessary to keep a record of splitting features in each parent node. In Equation 11, the i^{th} observation in the j^{th} tree encountered the local increments for $k \in \{1, ..., n_{increments,i,j}\}$. For this i^{th} observation in j^{th} tree, let H_{ijl} be the subset of local increments where the parent node was split by the l^{th} feature. The local increments of bootstrapping are assigned to *feature 0*. The letter H is used, as K already is used to describe number of classes.

This distinction between OOB-predictions $\tilde{y}$ and regular test predictions $\hat{y}$ of training set now becomes important as how to feature contributions are defined. Previously [16, 9] feature contributions have been defined for regression and classification analogous to this:

$$F_{il} = \frac{\sum_{j=1}^{n_{tree}} \sum_{k \subseteq H_{ijl}} L_{ijk}}{n_{tree}} \quad , \qquad (13)$$

Here F_{il}, the feature contribution of the i^{th} observation for the l^{th} feature, is a subtotal of local increments L_{ijk}, where k only iterates over H_{ijl}, which is those times the parent nodes were split by feature l.

This definition of feature contributions is fine if: (a) the noise level is low or (b) if feature contributions F only is computed for some test set different from training set or (c) if the user is confident, that the model structure is not over fitted. It would be possible to cross validate by segregating the data set in a training set and test set to avoid over fitted visualizations. To discard data points is not desirable for a data set with limited observations. It would be possible to perform an n-fold cross valida-

tion, but n-fold random forests would be necessary to train.

We propose to compute feature contributions for the OOB cross validated predictions. OOB cross validated predictions are only the sum of local increments over trees where i^{th} observation was OOB, see Equation 12. Analogously, we OOB feature contributions $\tilde{F}_{il}$ as

$$\tilde{F}_{il} = \frac{\sum_{j \subseteq \tilde{J}_i} \sum_{k \subseteq H_{ijl}} L_{ijk}}{|\tilde{J}_i|} \quad , \tag{14}$$

where j only iterates the subset of trees $\tilde{J}_i$, and where i^{th} observation was OOB. $|\tilde{J}_i|$ is the total number of times the i^{th} observation was OOB and the size of the subset $\tilde{J}_i$. Equation 14 is used in forest floor visualizations to compute cross validated feature contributions of the training set predictions.

2.4 Decomposing the mapping surface with feature contributions

We can compute the OOB cross validated set of points $\tilde{S}_i = \{X_i, \tilde{y}_i\}$ for $i \in T$ the training set. That is the combination by training features X_i and the cross validated predictions $\tilde{y}_i$, where $c = 1$ for regression and $c > 1$ for classification. To decompose $\tilde{S}_i$, then $\tilde{y}_i\}$ is expanded with $\tilde{F}_{il}$, such that:

$$\tilde{y}_i = \sum_{l=0}^{d} \tilde{F}_{il} + \overline{y} \quad . \tag{15}$$

Likewise non cross-validated $\hat{y}_i$ is a sum of non cross-validated F,

$$\hat{y}_i = \sum_{l=0}^{d} F_{il} + \overline{y} \quad . \tag{16}$$

The ensemble prediction $\hat{y}$ or $\tilde{y}$ is equal to sum of local increments + grand mean / base rate, see Equation 11,12. As sequences of additive vectors can be rearranged, it is possible to compute sub totals of local increments of the full prediction. Feature contributions is just the subtotal of encountered local increments for the for the i^{th} observation where the parent node was split by the l^{th} feature.

Notice feature 0 ($l = 0$) is included to accurately account for the normally small and negligible feature contribution of random bootstrapping. For an increasing number of trees, this bootstrapping feature contribution will approach zero. However, if the bootstrapping is stratified F_{i0} and $\tilde{F}_{i0}$ is equal to local increment from training set base rate $\overline{y}$ to the chosen stratification rate in every root node.

Figure 5 illustrates OOB cross validated feature contributions and regular feature contributions. A so called "one-way feature contribution plot" is a single feature contribution column plotted against the values of the corresponding feature. In Figure 5 the "one-way feature contribution plot" can be seen as projections of $\tilde{F}$. Conveniently, the main effects of either feature x_1 and x_2 have been separated with feature contributions before the projection into the 2D plane. In Figure 5, the goodness-of-visualization fit to the projected feature contributions can be seen for both $\tilde{F}_{i1}$ and $\tilde{F}_{i2}$. If it is possible to re-estimate the set feature contributions e.g. $\tilde{F}_{i1}$ with some estimator f only by the feature context of the visualization, it is certain, that no interactions have been missed. Thus the model structure do not contain any interaction effect with feature x_1. To quantify this we use a leave-one-out cross validation,

$$GOV(\hat{f}_\lambda) = cor(\hat{g}_{.l}, \tilde{F}_{.l})^2 \quad , \tag{17}$$

here the goodness-of-visualization (GOV), is the pearson correlation between LOO predicted feature contributions. Where $\hat{g}_{il} = \hat{f}_i^{-i}(X_{i\lambda})$ is the leave-one-out prediction of the $\tilde{F}_{il}$ feature contribution of the i^{th} observation for the l^{th} feature. λ is the features which are used to fit the estimator. When $\lambda = l$, GOV quantifies how well feature contribution of the l^{th} feature $\tilde{F}_{.l}$ is explained as a main effect. In Figure 5 $\tilde{F}_{.1}$ is predicted by $X_{.1}$ and $\tilde{F}_{.2}$ is predicted by $X_{.2}$. GOV can also quantify other visualization contexts than main effect plots. E.g. in Figure 7 of result section the goodness of a visualization context of two features x_3 and x_4 is quantified, where $\lambda = \{3, 4\}$.

3 Materials and methods

3.1 Data and software

The real datasets *contraceptive method choice* (cmc) and *white wine quality* (wwq) were acquired from the UCI machine learning repository [4, 11]. All algorithms were implemented in R (3.2.4) [18] and developed in Rstudio (0.99.892) [20]. The main functionality is available as the R-package, *forestFloor* (1.9.5) [25], published on the repository CRAN. If not stated otherwise all RF models was trained with the CRAN package *randomForest* [10] by default parameters except keep.inbag=TRUE in order to reconstruct the individual pathways of observations through the trees. To reproduce result section, R scripts for each data example have been included in the package.

3.2 Simulating toy data

To demonstrate that the visualizations in the result section 4 provide correct representations of the data structure, it is beneficial to use simulated (toy) data from a given hidden function. Such functions as Friedman#1 and 'Mexican hat' are known examples [1]. To illustrate the principal functionality

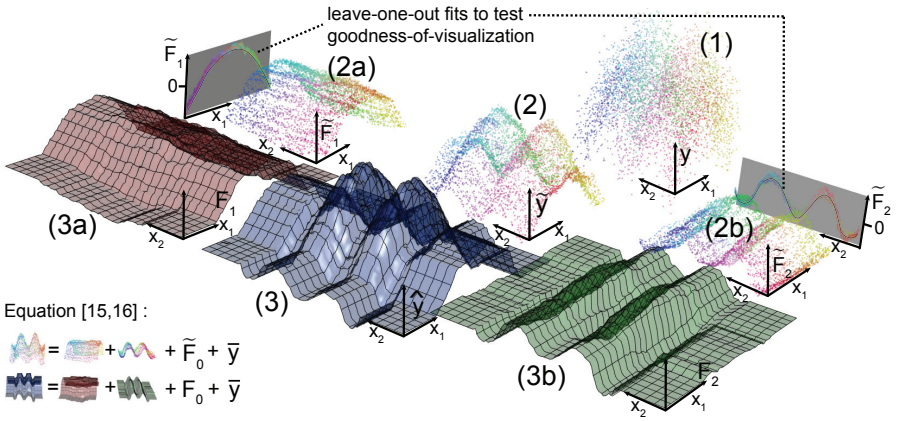


Figure 5: (1) Simulated data set of 5000 observations, $y_i = f(X_i) = -(X_{i1})^2 - cos(X_{i2})) + \epsilon_i$ where X_{i1} and X_{i2} are drawn from a uniform distribution such that $X_1 \in [-\frac{\pi}{2}; \frac{\pi}{2}], X_2 \in [0; 8\pi]$. For all plotted points, a colour gradient (hue color wheel) is used to mark different combinations of X_1 and X_2. (2) Out-Of-Bag cross-validated predictions $\tilde{y}$ are plotted. (2a/2b) $\tilde{y}$ is decomposed into feature contributions $\tilde{F}_1$ and $\tilde{F}_2$ and projected into a 2D plane, see Equation 14 and 15. Either contain almost only variance from the two main effects $-(X_1)^2$ or $cos(X_2)$. (3) Blue surface depict the full model structure, $\hat{y} = f(X)$. To either side (3a/3b) $\hat{y}$ is decomposed into F_1 and F_2, see Equation 13. The sum of cross-validated feature contributions by each observation plus the grand mean $\overline{y}$ is equal to the cross-validated predictions, and vice versa for non-cross validated. F_0 is the corrections for random or stratified bootstrapping. If no stratification, F_0 will be negligibly small. This illustration also generalizes more input features/dimensions and probabilistic classification.

of forestFloor a new hidden function, G is defined. G is the ideal hidden structure, which cannot be observed directly. The toy function was defined as $G(X)+\epsilon = G^*(X) = y = x_1^2+\frac{1}{2}sin(2\pi x_2)+x_3x_4+\epsilon k$ and was sampled 5000 times. x_i were sampled from a uniform distribution $U(-1,1)$. The noise variable ϵ was sampled from a normal distribution $N(0,1)$ and k was set such that the Pearson correlation $cor(G(X),G^*(X)) = 0.75$. Thus the true unexplainable variances component is roundly 25% of the total variance. The level of detail, RF can capture from hidden structure G, declines as the noise increases.

4 Results

Three data sets were modeled with RF regression or RF classification and subsequently explored with forest floor. The examples demonstrate how feature contributions can be used to visualize the data structure and how to identify unaccounted interactions in a visualization.

4.1 Random forest regression of *toy data*

A default RF regression model was trained on the toy data set with a hidden structure, $y = x_1^2 + \frac{1}{2}sin(2\pi x_2) + x_3x_4$. Figure 6 plots feature contribution of all six features against the training set feature values of the toy data. This type of plotting illustrates the main-effects, as feature contributions by each feature were plotted against their respective feature values. Hereby, the mapping surface S was visualized as the sum of d partial functions(blacklines), one for each feature. As the feature contributions retained any variance (main effects + interactions) associated with the node splits by each feature, it was possible to visually verify and test the goodness-of-visualization. Notice that main effect plots of x_1 and x_2 form nonlinear patterns representing the underlying additive x_1^2 and $\frac{1}{2}sin(2\pi x_2)$ contributions to the target y. Therefore, the leave-one-out R^2 goodness-of-visualization was > 0.95 for both these plots. As the explained variance of feature contributions of x_1 and x_2 was more than 95% when fitted as main effects, there was no considerable unaccounted interactions. On the other hand, feature contributions of x_3 and x_4 were poorly explained in main the effect plots. The GOV was poor, less than $R^2 < 0.1$. It was hence concluded that plotting the one-way feature contributions of x_3 and x_4 did not assist to explain the structure of S. Feature contributions of x_5 and x_6 were also poorly explained but contained no large variance and were therefore not interesting to explore further. The features x_5 and x_6 could also be identified as unrelated to the target y for having a very low variable importance (not shown). To include such uncorrelated/unrelated features illustrated the base line of random fluctuations in the mapping structure. This helped to assess whether a given local structure only was a random ripple.

As the feature contributions of x_3 and x_4 were inadequately accounted for, a broader context was needed to understand the hidden structure. To identify interactions relevant for the feature contribution of x_3 a color gradient (red-green-blue) was applied in mapping space V along the x_3 axis. The color of any other observation in any other plot was decided by its projected position on the x_3 axis. Low values were assigned red and high values blue. Figure 6 depicts the main effects feature contribution plot of x_1,...,x_6 with the applied color gradient to x_3. Any main effect feature contribution plot of features who neither correlate and neither interact with x_3 will show a random color pattern. Such features were $x1$, $x2$, $x5$ and $x6$, which neither correlated nor interacted with x_3. Plots of only correlated features would reproduce the same horizontal color pattern. In the extreme case, a feature identical to $x3$ would reproduce the exact same horizontal color pattern. Plots of only interacting features would reproduce the color gradient vertically along the feature contribution axis. A combination of correlation and interaction would make the color gradient reappear diagonally. In Figure 6 the color gradient suggests, that x_3 interacted with x_4 due to the vertical color gradient in the plot of x_4. In Figure 7 their combined feature contributions were plotted in the context of both feature x_3 and x_4. In this 3D plot it was observed, that the 2D rule of color gradients of interacting features was a basic consequence of perspective. Both color patterns of x_3 and x_4 could be reproduced by rotating the 3D plot. In this 3D plot, there was no large deviation of feature contributions from the fitted grey. Thus, it was evident that any structure of S related to x_3 and x_4 were well explained in the joined context of both features x_3 and x_4. The GOV of this fit was $R^2 > .9$. Therefore, this second order effect plot was an appropriate representation of how x_3 and x_4 contribute to the target y. The depicted saddle-point structure of Figure 7 was expected, as the product of x_3 and x_4 contributed additively to the target y. Overall, the model surface S, could be represented by two one-way plot of x_1 and x_2 and one two-way plot of x_3 and x_4. Hereby the hidden structure of the toy data was fully recovered.

4.2 Random forest regression of *white wine quality* (wwq)

The previous example of forest floor visualization was an idealized example with uncorrelated features and either representing clear main effect or

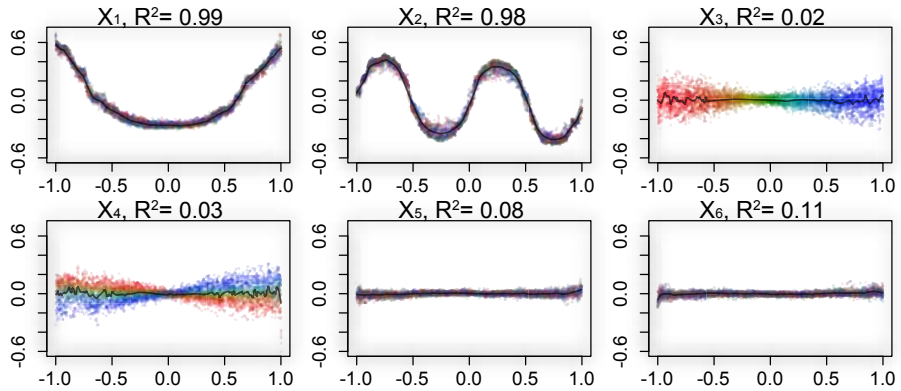


Figure 6: Forest floor main effect plot of a RF mapping structure trained on hidden function $y = {x_1}^2 + \frac{1}{2}sin(\pi x_2) + x_3x_4 + k\epsilon$. x_5 and x_6 have no relation to y and were included only to illustrate a base line signal. A color gradient parallel to $x3$ is applied to identify latent interaction with $x4$. Leave-one-out k-nearest neighbor gaussian kernel estimation provides goodness-of-visualization(black line & R^2 correlation) to evaluate how well each feature contribution can be explained as a main effect.

clear interaction effects. The white wine quality data set (wwq) is an example of mixed main effects and interactions by most features. The target, consumer panel ratings(1-10) of wines, was predicted on basis of 11 chemical features. A default RF model was trained and explained 56% of variance and the mean absolute error was 0.42 rating levels matching the previous best found model performance [5]. To explore the model structure of S, first all main effect plots were inspected. Figure 8 depicts all plots by all 11 features. Features were sorted in reading direction by variable importance to present most influential feature first. A color gradient along the most influential feature, *alcohol*, was applied to search for interactions. Hereby it was observed that *density* was negatively correlated with *alcohol*, that *volatile acidity* interacted with *alcohol* and that *residual sugar* both correlated and interacted with *alcohol*. The observed correlation between *residual sugar*, *density* and *alcohol* is trivial, where low-density *alcohol* linearly lowers *density* while high-density *residual sugar* increases *density*. Close to 98% of the scaled variance of these three features can be described by two principal components. This information redundancy was expected to affect variable importance of the three implicated features and to lower the general variance of the respective feature contributions. Although the overall structure suggested that alcohol content in general was associated with higher preference scores, there was a local cluster identified as low *alcohol*, high *residual sugar* and low *pH* which was associated with high preference scores also. Figure 8 suggested that wines could achieve a high preference score when *residual sugar*≈17, *pH*≈2.9, *citric acid*≈.35 and *fixed acidity*<7 despite a low alcohol content. Such white wines was perhaps by the consumer panel attributed fruity and fresh. Any found interaction could be investigated with several color gradients and two-way forest floor plots. It was chosen to investigate the interactions of *volatile acidity*, as this feature was the third most important feature, whereas the goodness-of-visualization of the one-way forest floor plot was only $R^2 = 0.69$. Two-way forest floor plot was therefore a more suitable representations of this effect. The color gradient along alcohol content already suggested a notable interaction between *volatile acidity* and *alcohol*. Figure 9 depicts the two-way forest floor plot of feature contributions of *volatile acidity* in the context of itself and the feature *alcohol*. The goodness-of-visualization was then $R^2 = 0.94$. Therefore, the residual variance of feature contributions not explained by this plot was low. For wines with alcohol content more than 10% (blue area) *volatile acidity* appeared slightly positively to preference score. For wines with lower than 10% *alcohol* (red area) *volatile acidity* appeared to contribute negatively to preference score.

4.3 Random forest multi-classification: *Contraceptive method choice* (cmc)

To illustrate the capabilities of forest floor for multi-classification the data set cmc was chosen. The data set originates from a survey of 1473 non-pregnant wives in Indonesia in 1987 comparing cur-

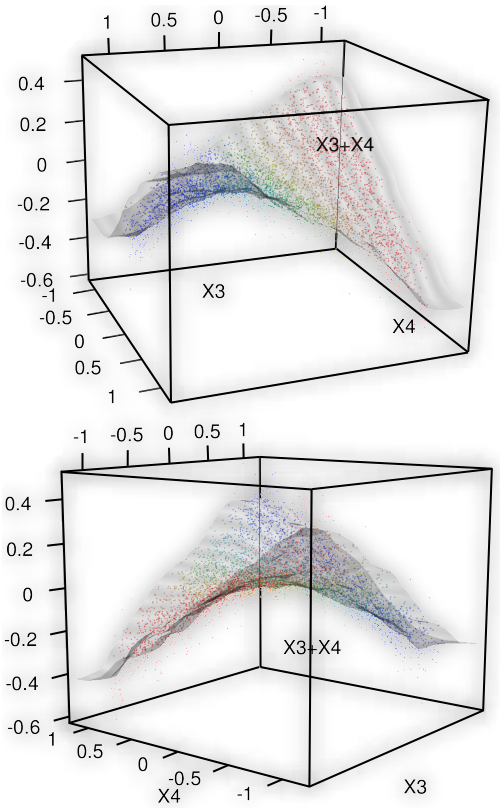


Figure 7: One forest floor interaction plot. XY-plan represent feature values x_3 and x_4 and Z-axis is the summed feature contributions of $\tilde{F}_{i3} + \tilde{F}_{i4}$. goodness-of-visualization is evaluated with leave-one-out k-nearest neighbor gaussian kernel estimation (grey surface, $R^2 = .90$). This indicates no remaining latent interactions related to features x_3 and x_4.

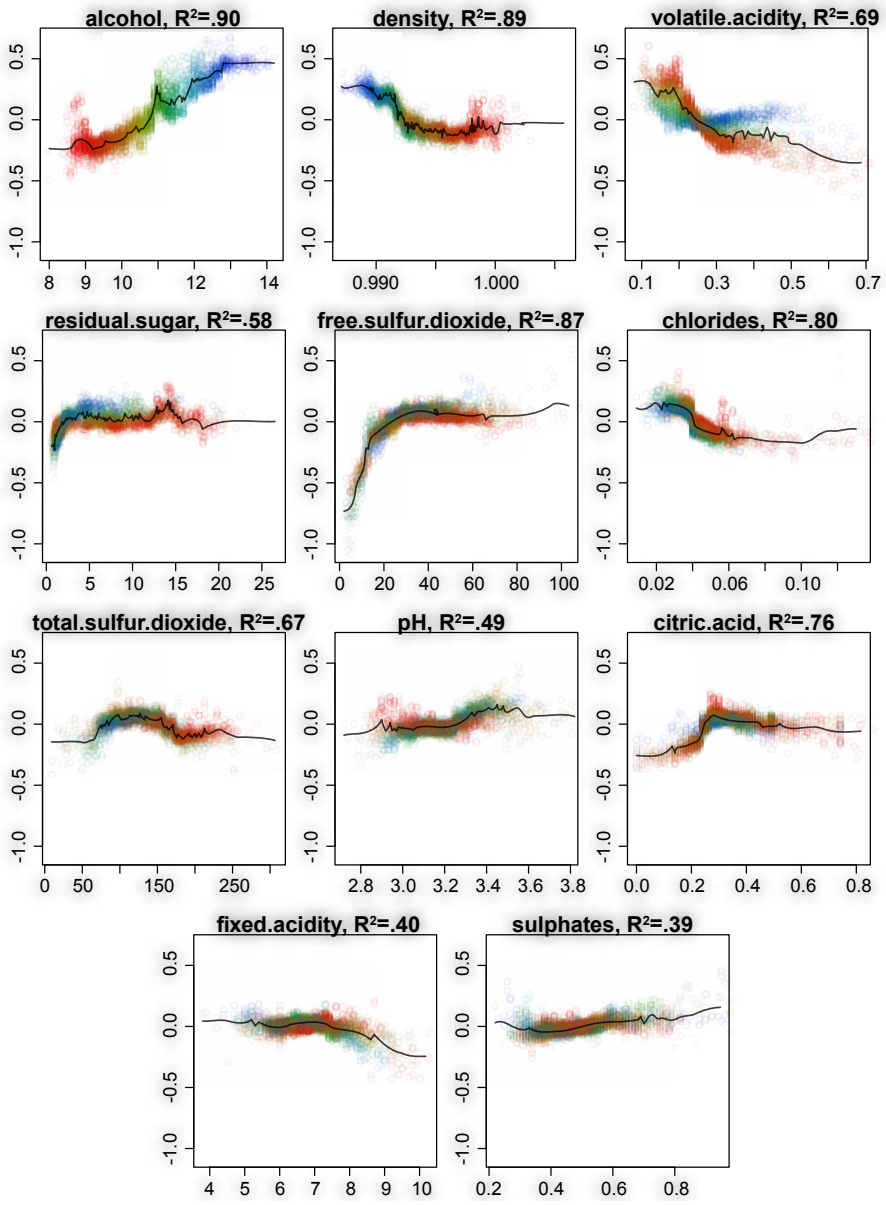


Figure 8: Forest floor main effect plots of random forest mapping structure of model predicting panel ratings of 4900 white wines on basis of chemical properties. The plots are arranged according to variable importance. X-axis are variable values and Y-axis the corresponding cross validated feature contributions. Color gradient in all plots are parallel to the feature *alcohol* (content w/w). goodness-of-visualization is evaluated with leave-one-out k-nearest neighbor estimation (black line , $R^2 values$)

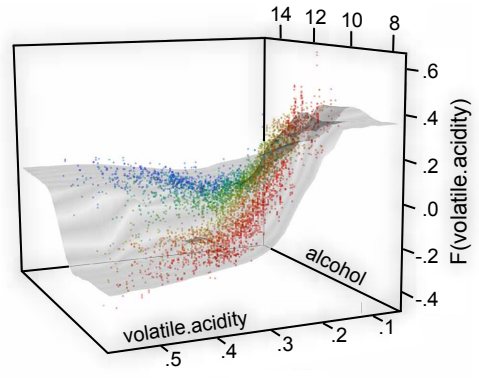


Figure 9: Forest floor interaction plot: Feature contribution of *volatile acidity* versus feature values of *volatile acidity* and *alcohol*. Color gradient is parallel to *alcohol* axis. goodness-of-visualization is evaluated with leave-one-out k-nearest neighbor estimation (grey surface and $R^2 = 0.93$)

rent choice of contraception with socioeconomic features. These features were, *wives' age* (16-49), *wives' education level* (1-4), *husbands' education* (1-4) , *n_children* (0-16), *wives' religion* (0 (not islam), 1 (islam)), *wives working* (0 (yes), 1(no)), *husbands' occupation* (I,II,III,VI), *standard-of-living index* (1-4), *media exposure* (0=Good, 1=not good) and the target *contraceptive method choice* (1=no-use (629), 2=long term(333), 3=short term (511)).

In the *cmc* data set the choice of contraception was far from fully described by the available features [12]. The OOB cross validated RF model error-rate was .44. Assuming wives did not use contraception (the most prevalent case) yielded a $\frac{629}{1473} = .57$ error rate. Anyhow, if the RF model performance would be regarded as good by domain specialists, the model structure could possibly provide insights to the socioeconomic mechanisms in play. Hyper parameters *Sample size* and *mtry* were tuned to yield the best OOB cross validated performance. Optimal parameters was found to be bootstrap *sample size*= 100 and *mtry* = 2. A lower *sample size* can increase robustness by tree decorrelation but also introduce more bias. To lower *sample size* of trees can be advantageous, when explained variance component is less than 50%. Thus a RF model different from default settings, was chosen to slightly improve predictions and to simplify/smooth the mapping structure to explore. Hereby the mapping structure may better represent the underlying social/economic mechanisms, that the specific data structure of survey reflects.

Three types of plots were constructed to investigate the mapping structure. As the number of features was $d = 9$ and number of classes was $c = 3$, a full dimensional mapping space visualization would require 12 dimensions. As shown in Figure 2, probability axes can be aligned along the y-axis, to reduce the number of dimensions to represent prediction space to only one. Also, when the cross validated predictions were decomposed into cross validated feature contributions, only 2 dimensions were needed to plot any main-effect. These plots resembled one-way forest floor regression plots although coloring was reserved to identify class of predicted probability. Otherwise each class by each feature would need to be plotted separately. Black assigns no usage. Red assigns long-term usage and green assigns short-term usage. Figure 10 illustrated the main effects of each feature of a RF-fit, the y-axis describes the additive change of predicted probability for each observation for each each class. The actual feature value for each observation was depicted by the x-axis. Thus any observation were placed three times in each plot by the same feature value in three colors once for each three classes. The sum of changed probability over classes for any observation must be zero, see Equation 8. Overall, Figure 10 showed that main effects were dominant, as most variance was explained by the respective features. *n_children* was the most important feature strongly predicting (probability change up to +/- .30) that wives with 0 or 1 child tended not to use contraception. On the other hand, more than 4 children predicted a slight increase in either type of contraception. Except for a preference separation for long-term contraception over short-term for wives with more 7 children, the *n_children* feature was not found useful to predict the choosing betwen the two types of contraception. *Wives's education* especially separated between no-use of contraception

and long-term use, where lowest level predicted up to +/-10% probability change. With more education the wives tended to use long-term contraception over no usage. The use of short-term contraception was comparably unchanged as a function of *wives' education*. *Wives' age*, the third most important feature, favored short-term contraception for wives younger than 30, while long-term and no contraception for wives elder than 30. After 40 years, either use of contraception declined. *Husbands' education* elicited same pattern as *wives' education* though size of effect was half. A small subgroup of 7% was reported to have a not good *media exposure* and this predicted a probability increase in no contraception of 8%. Type of *Husband' occupation* favored for category *I* long-term by 5% over short-term, whereas category *III* predicted an opposite 3% effect. Standard of living predicted a pattern much similar to *husband's eduction*. A small subgroup (15%) of wives were not muslim, and this predicted a 5% increase in short-term contraception over long-term usage and no usage. Lastly for a subgroup of 25% working wives was predicted a very slight increase (2%) of no-usage over short-term.

The main effects for this 3-class problem could also be depicted as a series of $(3-1)$-dimensional simplexes, where the position in the triangle depicts the predicted probability distribution for any observation. Colors can either depict true class (black: no-usage, red: long-term and green: short term) or colors can depict a feature (low value (red), middle (green), high(blue)). Figure 11 depicts all main effects in bi-simplex plots, with left simplex colored by cross-validated true class separation, and right simplex colored by feature value distribution across the simplex space. Figure 11 depicts 10 pairs of simplexes. Lines were added to the simplexes to illustrate majority vote. Only 17% of wives were predicted to use long-term contraception even though 22% of the sample population did so. Because RF models effectively used the sampled base rate as prior (marked as a blue cross) and the effective separation was weak, predictions tended to be skewed towards largest class away from smallest class. A different prior than the sampled base rate could be set by stratified bootstrapping of each tree in a random forest model. E.g. to stratify sampling by target class would move the blue cross to the middle of the simplex, and roughly a third of predictions would fall into either class. Stratified bootstrapping would e.g. be reasonable if the preferred contraception is expected to be different in the full population than in the training population.

In the second total separation simplex, to present an overview of any differences in socioeconomic status, principal component analysis was used to reduce the full feature space to two principal color components. Here a purple cluster indicated no-usage, a green cluster was shifted towards long-term usage, light blue cluster predicted short-term usage, and a dark-blue cluster predicted short-term or no usage. The color separation was not perfect, partly because the separation problem was difficult and partly because PCA cannot fully characterizes a potential nonlinear mapping surface of random-forest. To colour be several features at the same time, seemed to be most useful for data sets with high linear feature collinearity.

The left of following bi-plots of simplexes depicted the effective separation of true class separation by any feature contribution. The right simplex depicted the separation as a function of the corresponding feature (by color). This second simplex could be used both to illustrate the main effect of each feature and to assess whether higher order effects were present. For features with small set of levels such as womans education, a separation in four clusters (red(1), brown(2), pale blue(3), deep blue(4)) could be seen. Education level 1 and 2 were partly joined. The local centroids of these cluster was interpreted as the main effect, and the deviation from the centroids as higher order effects + unfiltered noise. For all simplexes the global centroid and prior is the (blue cross).

The series of bi-plot simplexes of Figure 11 could illustrate with finer detail the predicted probability distribution for any observation, whereas the precise feature value was depicted with less fidelity than in Figure 10.

The three features *media exposure*, *wives' religion* and *wives working* were binary and showed the largest change of predicted probability in the smallest subgroups. This observation was regarded trivial, as the group size weighted probability change across a binary feature split must have equal size. Thus few observations can change prediction a lot, if many observations only change prediction a little in a opposite direction. This was regarded a property for all binary decision tree models and Figure 4 in Section 2.3 depicted a similar pattern of how local increments would propagate in a probability simplex.

To search for higher order effects, similar to forest floor regression, simplex plots can in turn be colored by other features. In Figure 12 the simplex plots of *wives' age* and *wives' education* was printed 3 times each. From left to right, color gradients illustrated respectively *wives' age*, *wives' education*, and lastly *n_children*. The simplexes in the diagonal reproduced the main effect coloring from Figure 11, whereas other depicted simplexes possibly would detail 2^{nd} order interactions. E.g. *wives' education* of Figure 12 showed the four clusters, one for each education level. The distance from any point to its local cluster as a mix of higher order

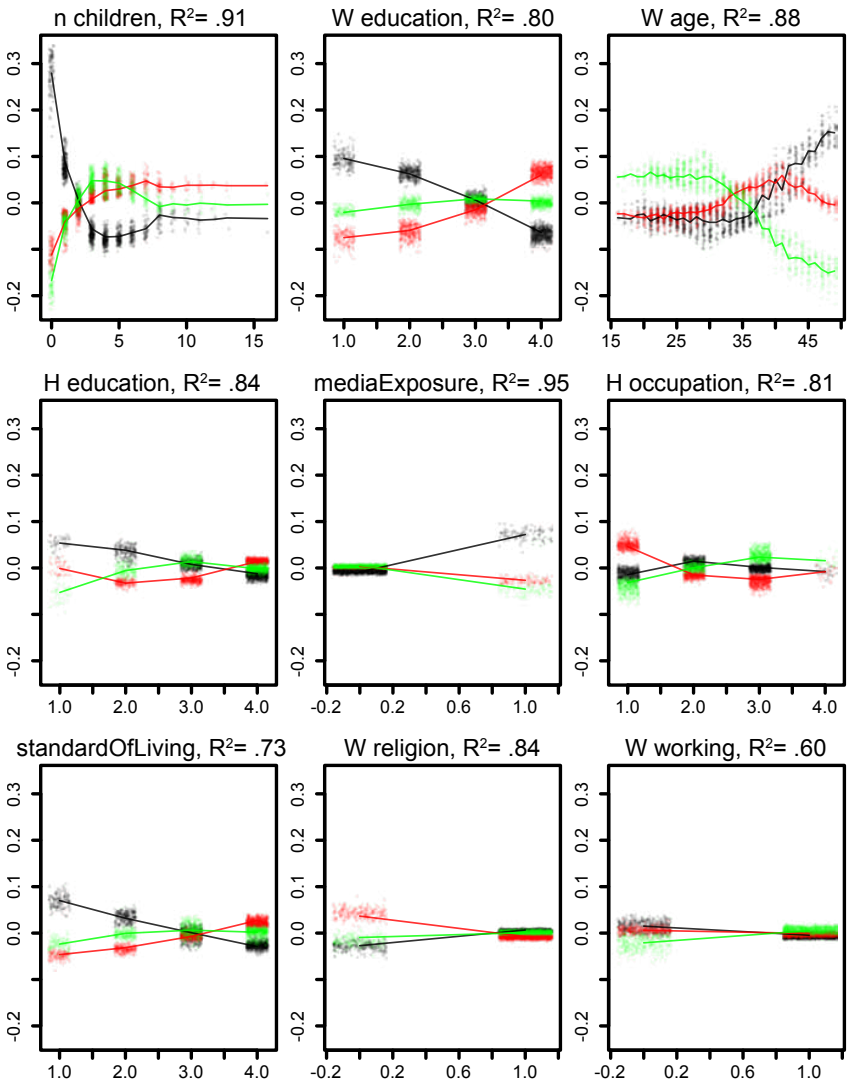


Figure 10: Cross validated feature contributions for each feature for each class(black, red, green) and for all training observations plotted against the corresponding feature values. Categorical features are coded with integers. Feature contributions can be understood as change of predicted class probability attributed to a given feature.

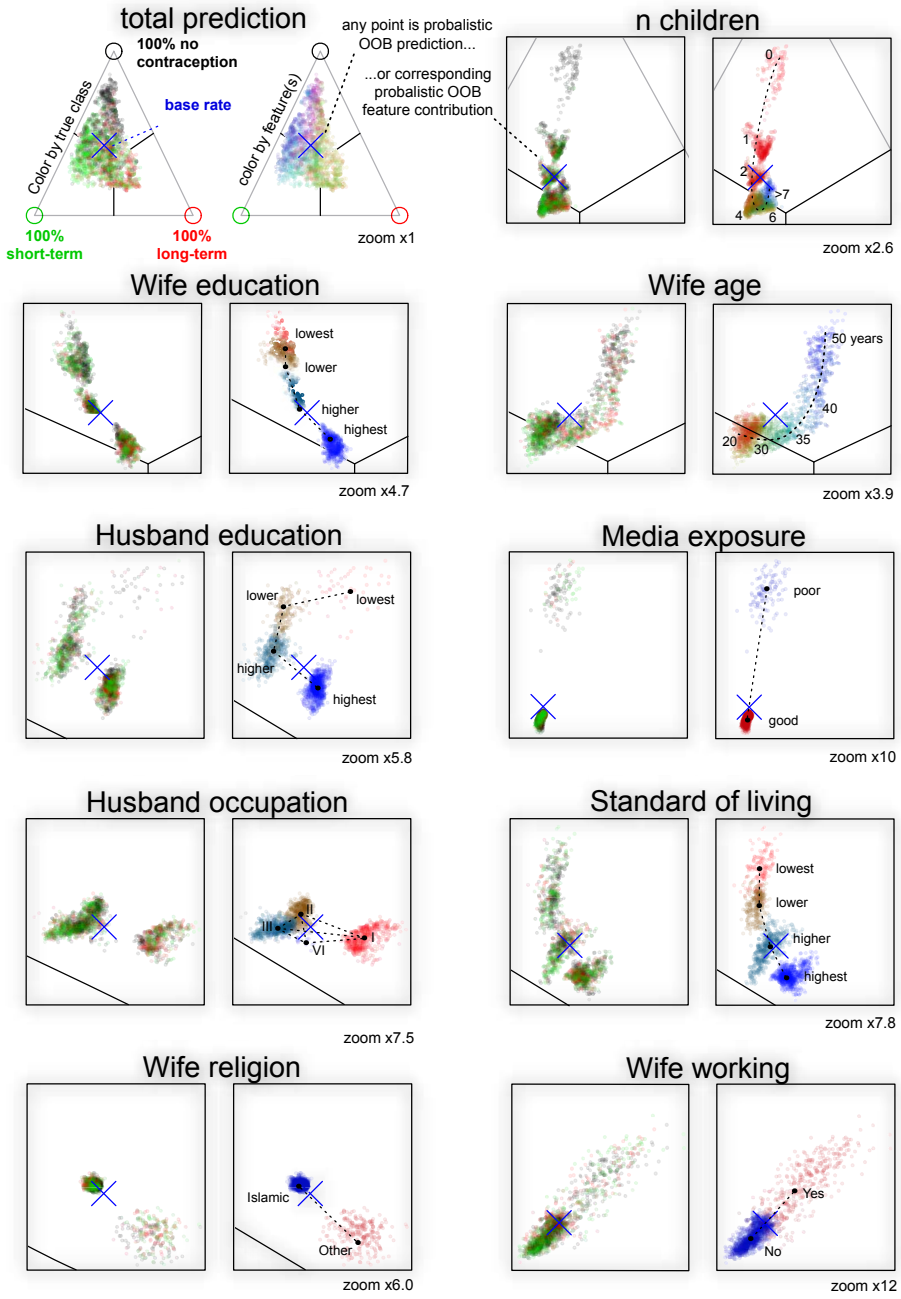


Figure 11: From top left: Cross validated predicted class probability colored by true class and a PCA color gradient describing observation diversity. Following pairs of plots, were the predicted probability decomposed into feature contributions. Left colored by true class, right colored by corresponding feature value. Red is minimal value, blue is maximal value. Blue cross is class base rate of training set. Dashed lines are drawn manually to assist interpretation of main effects.

effects and a small noise component. It was found that wives with highest education aged 20 were predicted more likely to use contraception than when aged 25. Wives' with highest education and few children (red) preferred short term contraception over long term. As the features *n_children* and *wives' age* are correlated, these will both interact with *wives' education*, not only one.

5 Discussion

Forest floor is a methodology to visualize the mapping structure of a RF model using feature contributions. RF can be termed a predictive algorithmic model, designed to have a high predictive accuracy on the expense of model transparency [22, 3]. RF could also be termed as data driven, as the model can adapt itself to the data with little guidance. The opposite is a theory driven model where the user manually choose an explicitly and clearly stated model to capture the data structure. A practical advantage of using RF, is when the user have little prior knowledge or theory on the subject. The majority of nonlinear machine learning algorithms models have in common, that the resulting model stated as an equation is fairly complex in the eyes of a human user. The complexity may be difficult to avoid if the model should be able to capture an unknown structure. But exactly when little prior theory is given, that is when the model should inspire the interpretation of the data structure. A dualistic approach is to choose both a perhaps linear explanatory model to interpret the system and a machine learning algorithm to get the most accurate predictions [22]. Such an approach may leave a gap between users comprehension and the actual structure of the nonlinear model. If the user is far from understanding a certain data-structure any optimization cannot hardly evolve from brute trial-and-error searches such as grid search or ant-colony-optimization methods.

For nonlinear high-dimensional multivariate models, it is not straight forward to visualize the trained mapping function. The provided visualizations can be understood as slices or projections of the mapping structure. It appears that a given series of 2D and/or 3D projections can jointly explain the structure of a RF mapping surfacesa. The quantifiable goodness-of-visualization measure describes how well the variance of the full structure can be explained in the context of the provided feature axis. If a large component of feature contribution variance remains unexplained, there is likely an unaccounted interaction pattern associated with this feature. Thus an advantage of forest floor is, that it aids the user to learn what local interaction effects are not yet visualized. With feature contributions it is possible to make an interpretation of what variance is attributed main effects, second order effects or higher order effects. Feature contributions can be computed from the training set itself and thus do not extrapolate the training set. The training set is used to set boundaries for model structure, such that extrapolated and unrelated model structures are not visualized. Feature contributions can be combined with the out-of-bag concept allowing cross validation to avoid presenting an overfitted mapping structure. Visualizations of cross validated feature contributions appear less noisy.

Color gradients allowed to include one or two extra dimensions in an illustration thus otherwise limited of three dimension. Color gradients traversing entire mapping space was used to highlight selected latent dimensions in a series of main effect plots to pinpoint missing interactions. We perceive colors as a combination of three channels red, green and blue. Thus, it may seem possible to visualize three additional dimensions in colors. Nonetheless, the ranges of color saturation and brightness should be constrained to avoid indistinguishable grey color tones and to ensure a minimal contrast to the background. Such considerations, limited color gradients to provide only two additional dimensions at maximum. It was possible to summarize a high-dimensional structure with e.g. principal component analysis and apply color gradients along the first 2 loading vectors, such as in Figure 11. In practice, we found a sequence of 1-dimensional color gradients best suited to uncover latent interaction structures in a RF model fit.

Feature contributions were first described in the context of RF regression, where a given feature can contribute either positively or negatively to a given prediction [9]. Next, the concept of feature contributions has previously been extended to classification, where the categorical majority vote labeling were replaced with numeric probability predictions [16]. We have argued that these probabilistic predictions are confined in a prediction space defined the $(K-1)$-simplex, for model with K classes. Any node in any tree will itself be a prediction and have a position in this space. We argue local increments are in fact vectors connecting nodes in the $(K-1)$-simplex space. The first local increment (the bootstrap increment) of any tree will be the vector connecting the class distribution of the training set to the class distribution of the root node. As the bootstrap increments will point randomly in any direction, the sum of a large number of such will approach the zero vector if no stratification is chosen. For stratification by true class, the bootstrap increments will connect the training set class distribution point in the $(K-1)$-simplex to the point in the $(K-1)$-simplex chosen by stratification.

The Gini loss function can be understood as

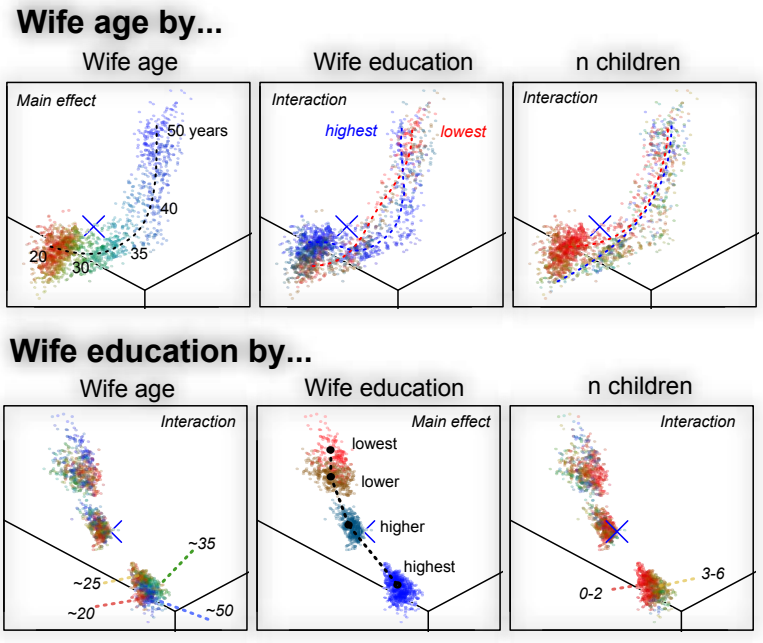


Figure 12: Feature contributions for the three most important features plotted row-wise. Each plot is colored column-wise by corresponding feature values. Dash lines are drawn manually to assist interpretation of interactions.

maximizing the squared distance of node positions to the center of $(K-1)$-simplex (equal class probability). Therefore any split by Gini will place the daughter nodes the furthest from the center, weighted by node size. As the classification trees are fully grown, the terminal nodes of one pure class can only be positioned on the vertices of the simplex. In Figure 11 was shown that the distribution of classes in the training set will function effectively as the prior of the RF model. If the user do not expect to find the same class distribution in future predictions as in training set, this prior can be moved in the simplex by stratification during the bootstrap process. In Figure 11 the center blue cross marked that the average root node center was skewed towards class 1 (no contraception) as 42% of the wives did not use any contraception. As class separation by the RF model was not strong the majority of predictions fall close to this prior base rate. In supplementary materials a RF model was trained with bootstrap stratification by true class such that the average root node is positioned in the center of the $(K-1)$-simplex and following predicted class probabilities were also centred around this point. Figure 4 depicted how any node-split will produce two new nodes with local increments in perfectly opposite direction. Thus, training set predictions will always be centred around this point.

Direct plotting of K class probabilities requires $K-1$ dimensions. This is possible for 3 or 4 classes with 2D plot or 3D plot respectively. The context of feature values can only be included as one extra axis or as color gradients. We have shown that the axis of the $(K-1)$-simplex can be aligned such that only one axis is needed to visualize the feature contributions as seen in Figure 10. This frees 1 or 2 axis to provide an adequate feature value context. In such visualization each observation will be plotted one time for each predicted class probability. Colors can be used to distinguish the classes.

In a previous article we trained a molecular descriptor model with RF to predict protein permeation enhancement in an epithelial cell model (Caco-2) [24]. A diagnostic tool was missed to address why such a model would be credible and to communicate intuitively the found pattern to fellow chemists/biologist with little knowledge of machine learning. We first stumbled upon feature contributions in the two articles [16, 9] and experimented to plot these feature contributions against the feature values. The R package rfFC [2] provided the first computations of feature contributions and was an inspiration to the design of the forestFloor package [25]. Hereafter we discovered partial dependence plots and sensitivity analysis [5, 6]. Now in hindsight we can report the set of advantages to forest floor, especially the tracking of unaccounted interactions such that no strong interaction will be overlooked when visualizing the mapping structure.

The following citation by Friedman [6] originates from an article from 2001 discussing the usefulness of partial dependence plots on nonlinear functions: *"Given the general complexity of these generated targets as a function of their arguments, it is unlikely that one would ever be able to uncover their complete detailed functional form through a series of such partial dependence plots. The goal is to obtain an understandable description of some of the important aspects of the functional relationship."* [6]

Indeed the structure of RF models can be highly complex and visualizations are unlikely to present every detail at once. Therefore a visualization toolset should assist the user to navigate the mapping structure. This has been done by isolating the part of the model structure related to the data structure, by evaluating the goodness-of-visualization of a given plot, and by pointing to where locally in the model structure a sizable latent interaction is not yet visualized. Our goal is to present complex models as adequately detailed visualizations. In a RF model there will likely always be a baseline of random ripples in the mapping structure, that we do not expect to be able to reproduce. These ripples are partly filtered of by using the out-of-bag cross validated feature contributions. Other ripples occur due to biases of the RF algorithm. Especially does the RF model structure surface contain wave like curvature parallel to the feature axes due to the univariate step functions of RF, see RF surfaces in Supplementary Materials.

We predict that 4D projections of a third order interaction rarely would be needed for the RF algorithm. In supplementary materials we have provided a simulation suggesting that RF only poorly can fit interactions higher than second order even when trained on 10.000 observations without any noise. This can be explained as the RF algorithm is limited in its potential complexity as the algorithm only can perform univariate splits decided by an immediate loss function. Another algorithm such as rotation forest [19] is not limited to perform univariate splits and therefore better on such simulated tasks with higher order interactions. What initially was an interaction effect can be rearranged into a main effect by new combined features. Multivariate split methods are not compatible with forest floor, but they are compatible with the generic methods partial dependence plots and sensitivity analysis [6, 5].

6 Conclusion

Forest floor has extended the tool-box to visualize the mapping structure of RF models. The geometrical relationship between random forest models

and feature contributions has been described. For RF multi-classification it was useful to understand the prediction space as a $(K-1)$-simplex probability space. Hereby the feature contributions can be interpreted as changes of predicted probability due to a given feature. A $(K-1)$-simplex prediction space can also visualize how the training set stratification affect RF predictions. Target class stratification is effective to modify the prior for the RF model.

We have emphasized that parts of a mapping structure which extrapolates the training set are irrelevant. To extract only the relevant mapping structure, feature contributions are computed only from the training set itself. Two new variants of feature contributions have been introduced to avoid inherent overfitting when using training set predictions. These variants of feature contributions are out-of-bag cross validated feature contributions, and n-fold cross validated feature contributions.

Feature contributions from a single feature can contain variance from main effects and/or interaction effects. A measure of goodness-of-visualization has been introduced to evaluate if the feature contributions of a given feature alone can be explained in the context of itself. If not, color gradients traversing the mapping space can be used to pinpoint overlooked interactions within feature contributions and features. Sizable interactions can be visualized in two-way interaction plots in the context of two features and perhaps even a third feature as color gradient. Again a goodness-of-visualization can be computed and evaluated for such a visualization.

Ultimately, it is difficult to communicate a context of more than 2 or 3 dimensions + target dimension(s). Thus fourth order interactions would be difficult to visualize and communicate. Anyhow, such visualizations are likely not missed, as the random forest algorithm could not fit fourth order interactions well and had a poor efficiency already with third order interactions.

As forest floor can break down a RF model fit into effects attributed to each feature and assist to find adequate context to understand these effects. It is intended that RF no longer should be seen as a non interpretable model. Learned associations between features and targets should inspire new ideas of the underlying possible causality structure.

References

[1] Monther Alhamdoosh and Dianhui Wang. Fast decorrelated neural network ensembles with random weights. *Information Sciences*, 264(0):104 – 117, 2014. Serious Games.

[2] Richard Marchese Robinson Anna Palczewska. *rfFC: Random Forest Feature Contributions*, 2015. R package version 1.0/r6.

[3] Leo Breiman. Statistical modeling: The two cultures. *Statistical Science*, 16(3):pp. 199–215, 2001.

[4] Paulo Cortez. UCI machine learning repository, 2009.

[5] Paulo Cortez and Mark J. Embrechts. Using sensitivity analysis and visualization techniques to open black box data mining models. *Information Sciences*, 225(0):1 – 17, 2013.

[6] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[7] Alex Goldstein, Adam Kapelner, Justin Bleich, and Emil Pitkin. Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *Journal of Computational and Graphical Statistics*, 24(1):44–65, 2015.

[8] Torsten Hothorn, Kurt Hornik, and Achim Zeileis. Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical statistics*, 15(3):651–674, 2006.

[9] Victor E. Kuz'min, Pavel G. Polishchuk, Anatoly G. Artemenko, and Sergey A. Andronati. Interpretation of qsar models based on random forest methods. *Molecular Informatics*, 30(6-7):593–603, 2011.

[10] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002.

[11] Tjen-Sien Lim. UCI machine learning repository, 1987.

[12] Tjen-Sien Lim, Wei-Yin Loh, and Yu-Shan Shih. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning*, 40(3):203–228, 2000.

[13] Sheng Liu, Shamitha Dissanayake, Sanjay Patel, Xin Dang, Todd Mlsna, Yixin Chen, and Dawn Wilkins. Learning accurate and interpretable models based on regularized random forests regression. *BMC Systems Biology*, 8(Suppl 3):S5, 2014.

[14] Raphael Maree, Pierre Geurts, Justus Piater, and Louis Wehenkel. Random subwindows for robust image classification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 34–40. IEEE, 2005.

[15] Deirdre B. O'Brien, Maya R. Gupta, and Robert M. Gray. Cost-sensitive multi-class classification from probability estimates. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 712–719, New York, NY, USA, 2008. ACM.

[16] Anna Palczewska, Jan Palczewski, Richard Marchese Robinson, and Daniel Neagu. Interpreting random forest classification models using a feature contribution method. In Thouraya Bouabana-Tebibel and Stuart H. Rubin, editors, *Integration of Reusable Systems*, volume 263 of *Advances in Intelligent Systems and Computing*, pages 193–218. Springer International Publishing, 2014.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[18] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2015.

[19] Juan José Rodriguez, Ludmila I Kuncheva, and Carlos J Alonso. Rotation forest: A new classifier ensemble method. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(10):1619–1630, 2006.

[20] RStudio Team. *RStudio: Integrated Development Environment for R*. RStudio, Inc., Boston, MA, 2015.

[21] Mark Seligman. *Rborist: Extensible, Parallelizable Implementation of the Random Forest Algorithm*, 2015. R package version 0.1-0.

[22] Galit Shmueli. To explain or to predict? *Statistical science*, pages 289–310, 2010.

[23] † Vladimir Svetnik, *, † Andy Liaw, † Christopher Tong, ‡ J. Christopher Culberson, § Robert P. Sheridan, , and Bradley P. Feuston‡. Random forest: A classification and regression tool for compound classification and qsar modeling. *Journal of Chemical Information and Computer Sciences*, 43(6):1947–1958, 2003. PMID: 14632445.

[24] Soeren H. Welling, Line K.H. Clemmensen, Stephen T. Buckley, Lars Hovgaard, Per B. Brockhoff, and Hanne H.F. Refsgaard. In silico modelling of permeation enhancement potency in caco-2 monolayers based on molecular descriptors and random forest. *European Journal of Pharmaceutics and Biopharmaceutics*, 94(0):152 – 159, 2015.

[25] Soeren Havelund Welling. *forestFloor: Visualizes Random Forests with Feature Contributions*, 2015. R package version 1.8.6.

[26] M. N. Wright and A. Ziegler. ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R. *ArXiv e-prints*, August 2015.

5.3.1 Supplementary materials for: Forest Floor Visualizations of Random Forests

Suplementary materials for: *"Forest Floor Visualizations of Random Forests".*

Soeren H. Welling, Line K.H. Clemmensen, Hanne H.F. Refsgaard, & Per B. Brockhoff

May 31, 2016

1 Proof for Equation [11] and [12] of article.

Part 1 - Any sequence of d-dimensional vectors: Denote a sequence of $n+1$ real vectors (or scalars) describing points in a $\mathbb{R}^d$ d-dimensional space as $\hat{y}''_k$ for $k \in \{0, ..., n\}$. The difference between any two adjacent vectors is defined as $L_k = \hat{y}''_k - \hat{y}''_{k-1}$ for $k \in \{1, ..., n\}$.

Lemma 1:

$$\hat{y}''_n = \sum_{k=1}^{n} L_k + \hat{y}''_0 \tag{1}$$

Proof 1:

For $k \in \{1, ..., n\}$, $\hat{y}''_k$ is the additive part of L_k and $\hat{y}''_{k-1}$ the substractive part.

When summing every L_k, all intermediary vectors of the sequence cancel out.

$\sum_{k=1}^{n} L_j = (\hat{y}''_1 - \hat{y}''_0) + (\hat{y}''_2 - \hat{y}''_1) + (\hat{y}''_n - \hat{y}''_{n-1}) = \hat{y}''_n - \hat{y}''_0$

Replacing $\sum_{k=1}^{n} L_k$ with $\hat{y}''_n - \hat{y}''_0$ in stated Lemma 1, one obtain

$\hat{y}''_n = \hat{y}''_n - \hat{y}''_0 + \hat{y}''_0$

Part 2 - a single tree: A tree is a hirachial graph. The first node, node 0, is connected to node 1. Every node from node 1 is either terminal and only connected to one parent node or

an intermediary node and has two daughter nodes. Every node of a tree has a prediction $\hat{y}''_k$ which is a real vector/scalar with exactly d dimensions.

Notes for part 2

For regression, a node prediction is a real scalar and computed as the target mean of inbag samples passing through the node. For classification a vector of d dimensions, where d is the number of classes in the training set, and each element from 1 *to d describe the prevalence ratio of inbag samples by a given class. Notice some random forest implementation use majority voting in terminal nodes. Here majority class element will be 1 and the remaining 0. Virtually any other prediction rule for nodes in classification trees outputting real valued vectors of length* $|\hat{y}''_k|^1 = 1$ *would be acceptable. Virtually any other prediction rule for nodes in regression trees outputting real values would be acceptable.*

An observation is an entity which will take one direct path of steps through the tree, starting from node 0 and ending in a terminal node. Observations are enumerated for $i \in \{1, ..., N\}$. Each observation will attain a sequence of predictions, one for each node it passes through. Each prediction is a real vector/scalar and written $\hat{y}''_{ik}$, where k sequentially enumerates the n nodes of the path for observation i. As n may differ for each observation i, it is thus written n_i. In one tree, any observation step sequence share the same first node 0 and node 1 also called the root node of the tree. A local increment (L_{ik}) is defined as a vector describing the prediction difference from $(k-1)^{th}$ to the k^{th} node for observation i.

Therefore we write $L_{ik} = \hat{y}''_{i,k} - \hat{y}''_{i,k-1}$ for $k \in \{1, ..., n\}$ for $n_i >= 1$.

The first node y_0 of one tree contain all observations and the prediction is the training set base rate / grand mean. y_0 can also be written as $\overline{y}$. The tree prediction of the i^{th} observation $\hat{y}'_i$, is defined as defined as the terminal node $\hat{y}'_i = \hat{y}''_{ik}$ where $k = n_i$.

Lemma 2

$$\hat{y}'_i = \sum_{k=1}^{n_i} L_{ik} + \overline{y} \text{ for any } i \in \{1, ..., N\} \tag{2}$$

Proof 2: As a given sequence of node predictions $\hat{y}''_{ik}$ for a given observation i are real

vectors/scalars, then the local increments of this sequence must be a part of any sequence postulated in lemma 1. Replacing $\overline{y}$ with y_0 and $\hat{y}'_i$ with $\hat{y}''_n$ we obtain lemma 1. Thus lemma 2 must be true also.

Part 3 - the test set prediction of any ensemble of trees The tree prediction of the i^{th} observation of the j^{th} tree is written $\hat{y}'_i$. An ensemble prediction $\hat{y}_i$ of $ntree$ decision trees is equal to the mean of the tree predictions $\hat{y}'_{ij}$ for each i observation. $\hat{y}_i = \frac{1}{ntree}\sum_j^{ntree}\hat{y}'_{ij}$ for A local increment of the j^{th} tree L_{ik} can be written L_{ijk} the number of local increments/steps for the i^{th} sample in the j^{th} tree can be written n_{ij}.

Lemma 3

$$\hat{y}_i = \frac{\sum_{j=1}^{n_{tree}}\sum_{k=1}^{n_{ij}} L_{ijk}}{n_{tree}} + \overline{y}, \tag{3}$$

Proof 3:

$\hat{y}_i = \frac{\sum_{j=1}^{n_{tree}}\sum_{k=1}^{n_{ij}} L_{ijk}}{n_{tree}} + \overline{y}$

$\hat{y}_i = \frac{\sum_{j=1}^{n_{tree}}\sum_{k=1}^{n_{ij}} (L_{ijk}+\overline{y})}{n_{tree}}$, use Lemma 2 to replace L_{ijk} with prediction of j^{th} tree $\hat{y}'_{ij}$

$\hat{y}_i = \frac{\sum_{j=1}^{n_{tree}}\hat{y}'_{ij}}{n_{tree}}$, this is the definition of the ensemble prediction

Part 4 For any j^{th} tree, any training observation i is either be designated as inbag or out-of-bag (OOB). The OOB prediction $\tilde{y}_i$ computed from a subset of all trees $\{1, ..., ntree\}$ where i is OOB, we call this subset for $\tilde{J}_i$ and this set will have $nOOB_{tree,i}$ members. The OOB ensemble prediction is defined as the mean prediction of OOB tree for the th observation. $\tilde{y}_i = \frac{1}{nOOB_{tree,i}}\sum_{j\in\tilde{J}_i} y_{ij}$ where subset $\tilde{J}_i \subseteq \{1, ..., ntree\}$.

Lemma 4

$$\tilde{y}_i = \frac{\sum_{j\in\tilde{J}_i}\sum_{k=1}^{n_{ij}} L_{ijk}}{n_{tree}} + \overline{y} \tag{4}$$

Proof 4: As lemma 3 was shown for any set of trees in an ensemble, and as lemma 4 is just the special case for particular subsets of trees, then lemma 4 must be true also.

1.1 How to highlight the mapping structure of a local cluster

In the white wines quality (wwq) data set. A local interaction was identified among wines with the lowest alcohol content ($< 9.3\%$). In spite of low alcohol content in general lead to lower preference predictions, a subgroup of low alcohol wines deviated from this main effect. It was possible to further characterize this local interaction in the mapping structure of the trained RF model. Any wine of alcohol content more than than 9.3 was colored transparent grey. Remaining low alcohol wines were colored by the feature contribution of alcohol, such that wines with a relatively positive impact of low alcohol content were marked blue and wines with a relatively negative impact were marked red. Intermediate wines were green. Main effect plots by all features were colored by these scheme as depicted. Hereby it was possible to visualize the local interaction. It was possible to observe that the most clear differences between wines marked blue and wines marked red was the content of chlorides, citric˙acid and residual˙sugar. This observation characterized a certain cluster of fruity wines (acidic and sweet) of high preference despite low alcohol content.

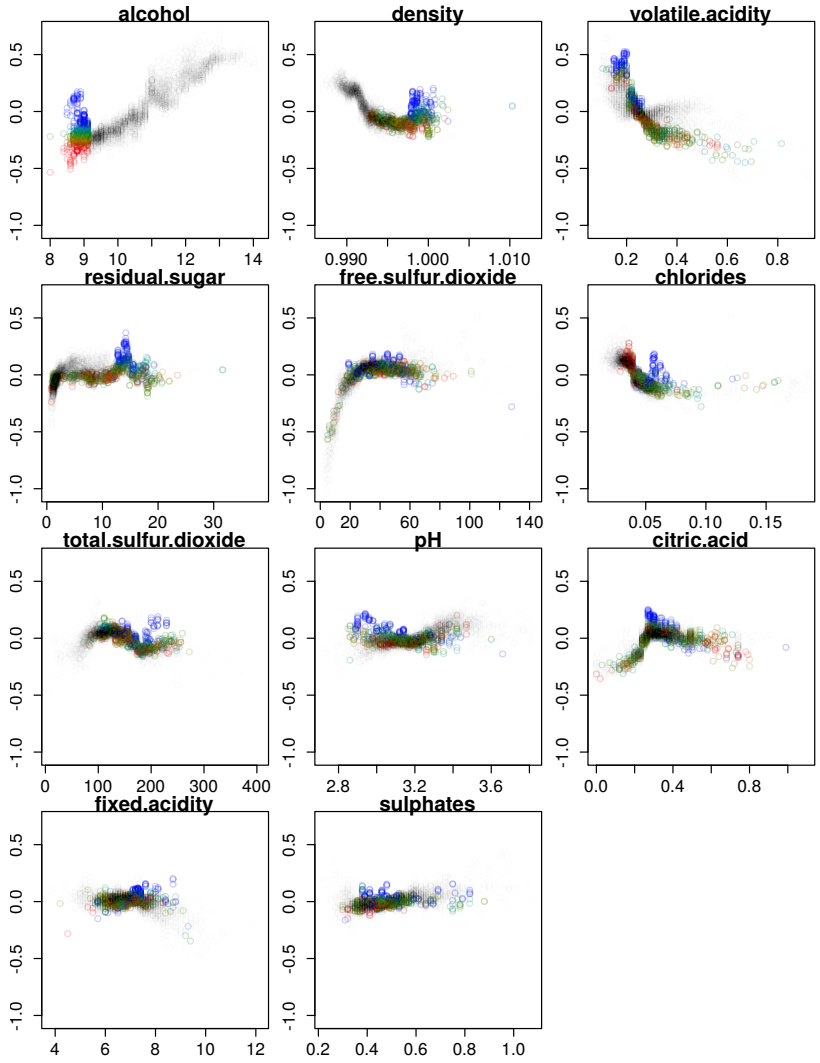


Figure 1: Cross-validated main effect feature contributions of predicted preferences of 4900 white whines. The color gradient along feature contributions of alcohol characterizes the specific interaction pattern between low alcohol content and remaining features.

1.2 RF mapping unrelated to data structure

Two normal distributed(N(0,1)) variables x_1 and x_2 is related to a target y by either $G_1(X) = y_1 = (x_1)^2 + 2sin(2x_2)$ or $G_2(X) = y_2 = x_1x_2$. 3000 samples were drawn and a default RF-model was trained. A grid of 300 grid lines and 300^2 grid points was formed. Each grid point represented a combination of x_1 and x_2 from -7 to 7 such that the entire grid extended the range of sampled values 3 times. Any grid point of x_1 and x_2 was predicted by the RF model. The predicted $\hat{y}$ was plotted as a function of x_1 and x_2 in a 3D plot. The mapping structure was represented as a surface outlined by the grid points and colored by high $\hat{y}$ (red/high, green/low). The mapping of the training set is represented by the set blue points on the mapping surface. For the data structure G_1 there is no unstable boundary effect as the partial quadratic function of x_1 and the partial sine function of x_2 do no interact and simply intersect additively in the region of the training set (blue points). The saddle-point structure of G_2 is not the sum of two additive partial functions. In a rectangular boundary of were training set was observed a series of ripples in the mapping structure was observed. Here predictions alternated between high and low values. This boundary mapping structure do not reflect the data structure of G_2. Likely as RF only performs univariate splits, it can only capture interaction effects by splitting data into sub groups. As these sub group becomes less populated at the boundaries of the data set the fit becomes markedly unstable.

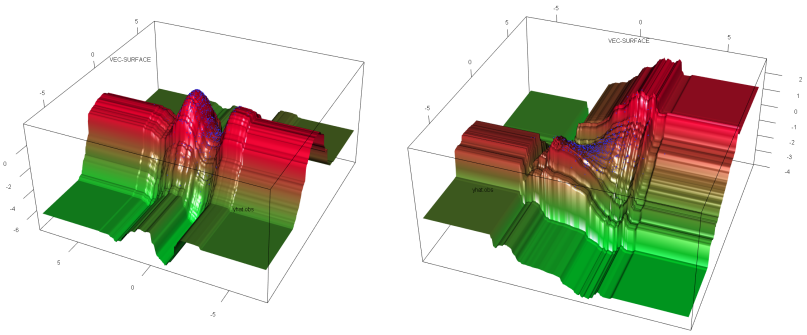


Figure 2: RF regression model structure of two hidden functions $y_1 = (x_1)^2 + 2sin(2x_2)$ (left) or $y_2 = x_1 x_2$ (right). Red-green color gradient is parallel to the vertical target axis, $\hat{y}$. Positions marked blue are the training examples used to train the mapping structure. The visualized surface extrapolates the trainnig set 100% in each direction. Left plot(y_1) depicts a stable main effect only structure. Right plot(y_2) depicts a unstable interaction effect only structure.

1.3 Shallowness of Random forest

Although splits of nodes in RF is performed univariately, RF can still capture interactions due to the many local rules applied. Presumably as the sequential decisions performed by RF satisfy only an immediate loss function of each split and splits are only univariate, RF cannot grow decision trees to capture 4^{th} order interactions or higher. To test the ability of RF to captivate data structures of various complexity, three hidden structures were designed. A series of i variables x_i were drawn from a distribution and multiplied. The structure have no error component. Figure 3 depicts from $d = 1$(light green) to $d = 6$(red) the ability of random forest models to fit a training set of N train samples. A single main effect is modelled with almost no error already from 100 observations. A second order interaction needs 100-200 samples to explain 75% of the variance when cross validated. A third order interaction in a feature space of continous variables ("saddle" & "sineprod") requires 10,000 samples to explain 75% variance cross validated.

"saddle"

$$y_d = \prod_{i=1}^{d} x_i, x_i \in N(0,1) \tag{5}$$

"sineProd"

$$y_d = \prod_{i=1}^{d} sin(x_i), x_i \in U(-\pi/2; \pi/2) \tag{6}$$

"binaryProd" (notice only -1 or 1 is sampled)

$$y_d = \prod_{i=1}^{d} x_i, x_i \in U\{-1, 1\} \tag{7}$$

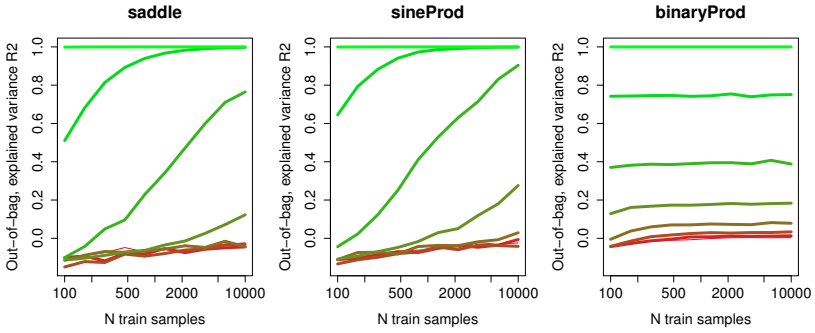


Figure 3: How many orders of interactions can RF capture? Three structures saddle, sineProd and binaryProd, ranging from main effect(light green) 6th order of interaction(red line). RF already becomes an poor estimator at 3rd order interactions.

1.4 The effect of stratification

Stratified bootstrapping by target variable moves weighted centroid of cross validated training predictions to the center of the simplex. Hereby, highly prevalent classes are down-sampled, but every sample will likely participate at least in a small number of trees. Appendix Figure 5 depicts such a stratified RF model, where root node is balanced in respect of target classes. Besides the centroid of prediction were moved to the center of K-1 probability simplex, the general structure of the model structure seemed similar to the non-stratified version in manuscript.

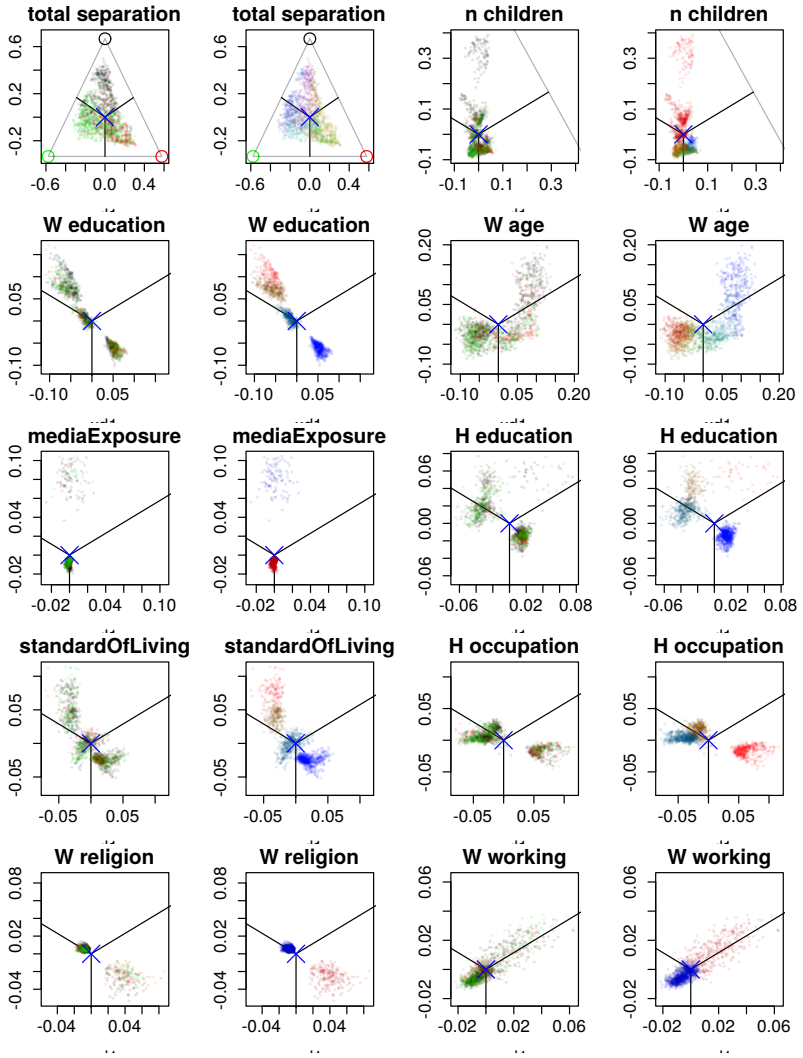


Figure 4: Feature contributions for contraceptive method choice (cmc) data set when RF was trained with target class stratification. Blue cross marks average root node which is also the center of the average cross validated prediction.

1.5 forest floor visualizations of gradient boosted tree

Gradient boosted trees suggested by Friedman is a boosted ensemble, where each new tree is fitted to the residuals of the current ensemble of trees [2]. Nonetheless, all grown trees in the ensemble are regular decision trees similar to trees of random forest ensembles. The greadient boosted ensemble prediction is the sum of votes, whereas for a random forest ensemble it is the average vote. In either case, both boosted trees and bagged trees contribute additively to the ensemble prediction. Therefore can every prediction be split into local increments and the feature-wise subtotals, named feature contributions can be computed. Presently, the perhaps most popular gradient boosting algorithm is XGBoost [1]. To make a fast proof-of-concept we preferred not to write an entirely new adaptor for XGBoost, but rather to write a wrapper around the randomForest implementation [3], making it behave as a gradient boosted ensemble and retain compatibility with forestFloor. This short wrapper is printed below and included in the forestFloor package as an example script **ffGradientBoost.R**.

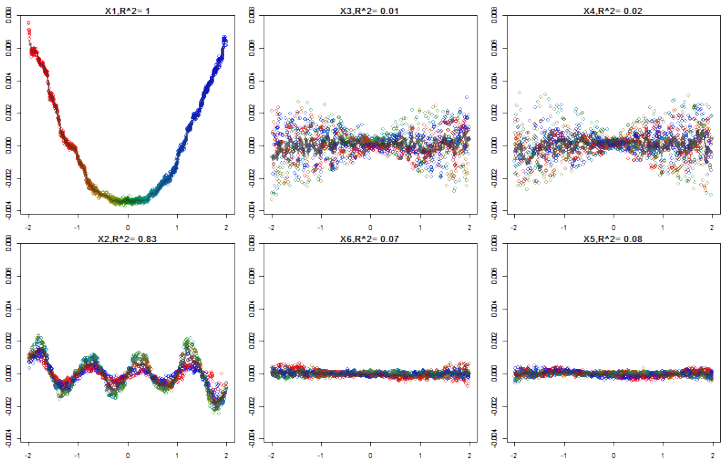


Figure 5: forestFloor visualization of a simpleBoost model. simpleBoost is a gradient boosted tree ensemble, implemented as a simple wrapper of the CRAN randomForest algorithm.

```
1
library(randomForest);library(forestFloor)
3 #simulate data
X     = data.frame(replicate(6,4*(runif(3000)-.5)))
5 Xtest = data.frame(replicate(6,4*(runif(1500)-.5)))
y     = with(X,X1^2+sin(X2*2*pi)+X3*X4) + rnorm(3000)/3
7 ytest = with(Xtest,X1^2+sin(X2*6*pi)+X3*X4) + rnorm(3000)/3

9 #define boosted tree wrapper
simpleBoost = function(
11   X,y,    #training data
  M=100,  #boosting iterations and ntrees
13   v=.1,   #learning rate
  ...) {  #other parameters passed to randomForest
15   y_hat = y * 0  #latest ensemble prediction
  res_hat = 0    #residuals hereof...
17   Fx = list()    #list for trees
  for(m in 1:M) {
19     y_hat = y_hat + res_hat * v  #update prediction, by learning rate
    res = y - y_hat              #compute residuals
21     hx = randomForest(X,res,ntree=1,keep.inbag=T,...) #grow tree on residuals
    res_hat = predict(hx,X)                           #predict residuals
23     cat("SD=",sd(res),  "\n")    #print
    hx$forest$nodepred = hx$forest$nodepred * v #multiply nodepredictions by learning rate
25     Fx[[m]] = hx  #append tree to forest
  }
27   Fx = do.call(combine,Fx) #combine trees with randomForest::combine()
  Fx$y = y #append y
29   Fx$oob.times = apply(Fx$inbag,1,function(x) sum(!x)) #update oob.times
  class(Fx) = c("simpleBoost","randomForest") #make simpleBoost a subclass of randomForest
31   return(Fx)
}
33

predict.simpleBoost = function(Fx,X) {
35   class(Fx) = "randomForest"
  predMatrix = predict(Fx,X,predict.all = T)$individual
37   ntrees = dim(predMatrix)[2]
  return(apply(predMatrix,1,sum))
39 }

41 plot.simpleBoost = function(Fx,X,ytest,add=F,...) { #plots learning curve
  class(Fx) = "randomForest"
43   predMatrix = predict(Fx,X,predict.all = T)$individual
  ntrees = dim(predMatrix)[2]
45   allPreds = apply(predMatrix,1,cumsum)
  preds = apply(allPreds,1,function(pred) sd(ytest-pred))
47   if(add) plot=points
  plot(1:ntrees,preds,...)
49   return()
```

```
}

#build gradient boosted forest
rb = simpleBoost(X,y,M=300,replace=F,mtry=6,sampsize=500,v=0.005)

#make forestFloor plots
ffb = forestFloor(rb,X,Xtest)
#correct for that tree votes of gradient boosts are summed, not averaged.
#forestFloor will as default divide by the same number as here multiplied with
ffb$FCmatrix = ffb$FCmatrix * c(rb$oob.times,rep(rb$ntree,sum(!ffb$isTrain)))

#plot forestFloor for OOB-CV feature contributions and regular feature contributions
plot(ffb,plotTest=T,col=fcol(ffb,3,plotTest = TRUE))
plot(ffb,plotTest=F,col=fcol(ffb,1,plotTest = FALSE))

#validate model structure
pred = predict(rb,X)
predtest = predict(rb,Xtest)
plot(y,pred,col="#00000034")
plot(rb,Xtest,ytest,log="x")
vec.plot(rb,X,i.var=1:2)

#export plot
png(file = "ffGradientBoost.png", bg = "transparent",width=800,height = 500)
plot(ffb,plotTest=T,col=fcol(ffb,1))
rect(1, 5, 3, 7, col = "white")
dev.off()
```

References

[1] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *arXiv preprint arXiv:1603.02754*, 2016.

[2] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[3] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002.

CHAPTER 6

Discussion and conclusion

In agreement with Novo Nordisk, the specific methods of combining permeability predictions and solubility predictions and the actual suggested candidates have not been included in this thesis. However the distribution of solubility predictions and permeation predictions is a good basis for a discussion of whether it is possible to search for permeation enhancers that are both soluble and sufficiently potent.

6.0.1 Is predicted potency the same as insolubility

In Figure 6.1 the predicted solubility and permeability are plotted for 10,000 molecules. Permeability predictions are based on a model similar to the one described in article of Section 4.1. As the predictive model rely on molecular descriptor computations with the MOE software suite, and this take in average 1 second per molecule to compute. The full available million molecule search set would have taken 11 days to run. To narrow down the search set, molecules were selected by criteria of rotatable bonds and molar weight in anticipation, that molecules with less than a certain fraction of rotatable bonds would likely not have fatty carbon chains.

As discussed in Section 4.0.1, it was a major consideration, that the permeability model and solubility model both perhaps only would recognize lipophilicity-like properties as predictors, and thus perform exactly opposite predictions. However this seems not to be the case for the 10,000 predicted molecules, as there are molecules with various predicted ratios between solubility and permeability enhancement. However, in the figure no molecules ocour in the top right 'unrealistic' corner, where molecules would be predicted as both very soluble and very potent.

In Figure 6.1 the majority of molecules are not potent enough. Sodium caprate has been set as the lower bar of required potency = 0.67. A potency of 0.67 predicts that if a compound was used as permation enhancer and added to three Caco-2 monolayers in solution at 1%, 0.1% and 0.01% *(w/w)*, the electrical resistance across the monolayers would in average be lowered 67%. As discussed in article from Section 4.1, the permeation model overestimates the potency of non permeation enhancers, as the training set contained no non-permation enhancers. Thus the permeation model likely has not learned to separate non potent from weekly potent. However, a permeation enhancement potency below sodium caprate (C10) is probably too weak, since no C8 or C6 permation enhancers have been successfully included in a formulation. Therefore the most important task of the model is to distinguish the highly potent enhancers from the medium potent.

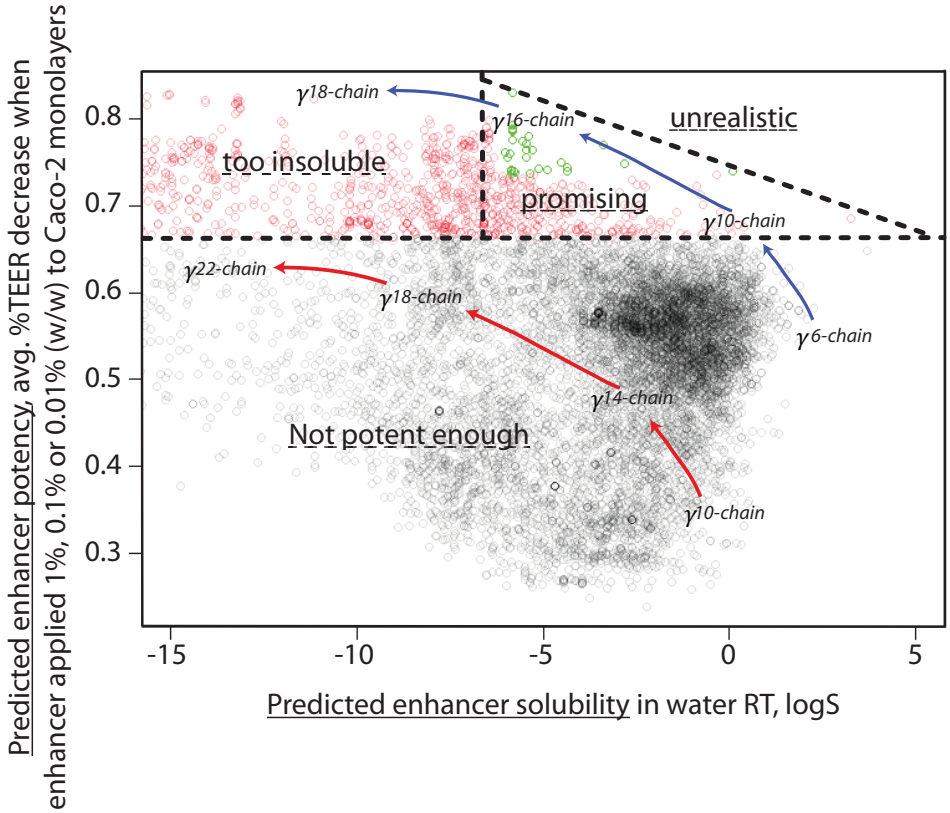


Figure 6.1: Predictions of logS water solubility and permeation enhancement in Caco-2 model for 10,000 compounds. Few molecules are both soluble and potent permeation enhancers. Figure has been segmented into groups of 'too insoluble', 'not potent enough', 'promising' and 'unrealistic'. For a given group of surfactant permeation enhancers, the acylic carbon chain can be reduced or elongated (γ). The blue arrows from γ6 to γ18 exemplify a given enhancer group, where the γ18 is too insoluble and γ6 not potent enough. For some other enhancer family with a less favorable head group, the chain length path (red arrows) may never cross into the 'promising' segment..

6.0.2 Uncertainty of predictions

In Section 4.1 the root mean square prediction error of the permeation enhancer model was found to be 0.16, on the defined potency scale from 0 to 1. The prediction error was estimated by cross-validation and on a external data set. Therefore it is reasonable to expect the model can recognize molecules as highly potent, medium potent or non-potent permeation enhancers. The prediction error (standard deviation) of the solubility model as discussed in the results part of the draft "Learning the structure of random forest models QSAR modelling: Predicting molecular Solubilty" in Section A.1 estimated by cross-validation is 0.64 on the logS scale. A standard deviation on logS scale of 0.64 translates to a 4-fold deviation in absolute solubility. As the training molecules in the *Huskonnen et al* data set [Pal+07] span the logS scale from -10 to +2, a prediction error of 0.64(RMSE) seems as a quite accurate model. Likewise, the cross-validated estimated explained variance of the model is 90%. However to function as an permeation enhancer, if logS is below -6, the dissolution rate would likely be too slow.

Most surfactant based permeation enhancers have an ability to form micelles, such that the apparent solubility of very lipophilic permeation enhancers could be close -1. However the lipophilic permeation enhancers will have a low dissolution rate. The predictions of a theoretical logS replaces the intrinsic dissolution rate, that likely would have been the best property to predict. No large public data set, was found for intrinsic dissolution

Solubility models [Del04; Pal+07] are mainly derived from solubility measurements at room temperature 20-25 degrees Celcius. As the solubilization occurs at 37 degrees Celcius, some molecules may improve solubility more than others. Noyes-Whitney dissolution rate equation predict dissolution as proportional to the concentration gradient. The more soluble an enhancer is, the proportionally faster it can ideally pass from solid form to mono-meric form and diffuse away from the tablet. The temperature sensitivity of molecules has not been accounted for. All logS prediction match 20 to 25 degrees celcius. Most likely the majority of molecules will have a higher logS value at 37 degrees, however some molecules may be more sensitive to temperature than others. This will contribute with extra uncertainty, when extrapolating to 37 degrees Celcius. One future approach could be to model the temperature sensitivity for a smaller data set and use these temperature sensitivity predictions to correct predictions at room temperature [Kli+16].

6.0.3 Interplay of dissolution rate and insulin permeability

Permeation enhancers can increase the permeability of insulin across the epithelial barrier. Insulin degrade with time due to enzymes in the luminal space of the small intestine. To reduce the time in luminal space, the permeation enhancer must contribute to a fast efficient dissolution of the tablet. A very slow release will give apparent zero-order enzyme reactions an advantage to break down all insulin. Also the permeation enhancer may be cleared from the site of action faster than the release

from the tablet, such that the overall concentration of permeation enhancer in the epithelial membrane will be too low to have an effect.

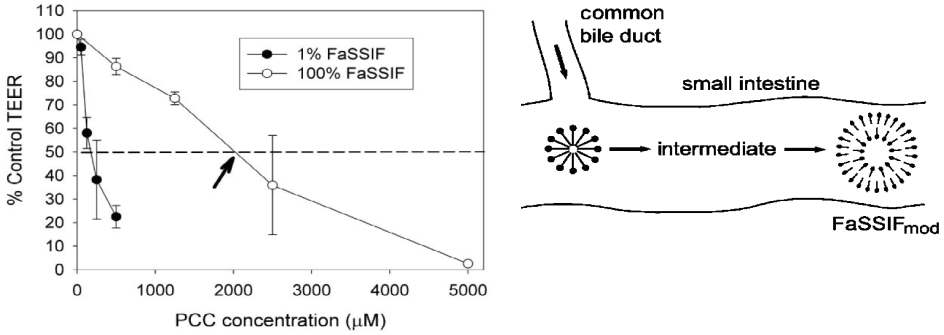


Figure 6.2: Two figures copied from [TT08; Naw+11] to illustrate why the potency of lipophilic permeation enhancers are likely overestimated in *in-vitro* studies using surfactant free buffers. Left, palmitoyl carnitine a C16 surfactant enhancer was found by Tippin *et al* to be 12 times less potent in a FaSSIF buffer with taurocholate and lechitine an EC_{50}. Right, in the intestinal lumen taurocholate and lechitine will be released from the bile duct and form micelles and liposomes. Small concentrations of long chain lipophilic enhancer will absorbed by the liposomes instead of the epithelial membrane.S.

Identifying new potent enhancers with the Caco-2 cell model do not emphasize the dissolution speed. As a substitute for intrinsic dissolution speed measurements, logS-solubility was used instead, as intrinsic dissolution measurements were too sparse and not available in the public domain. Predicting both solubility and permeation enhancement allowed to correct for the over-optimistic scoring of lipophilic permeation enhancers in the Caco-2 model. As the Caco-2 model uses watery HBSS buffer, the lipophilic permeation enhancers will have no other place to bind than the epithelial membrane. However under *in-vivo* conditions, there will be plenty of competing sites to bind such as billary liposomes and micelles. Tippin *et al* has showed that in-vitro models using HBSS buffers favor lipophilic permeation enhancers [TT08]. In a buffer such as FaSSIF, that contain small amounts of inert surfactants, the lipophilic enhancers will be absorbed into the inert micelles instead of the site of action. On the other hand, a medium potent permeation enhancer such as sodium caprate can dissolve from the tablet fast enough, that the concentration of caprate is much higher than the inert surfactants. The change of potency for a lipophilic enhancer palmitoyl carnitine is described in Figure 6.2.

6.0.4 Interpretation of random forest

The regression random forest learner was used to build predictive models. The explicit structure of the trained random forest models is too complicated to comprehend. Feature contributions was used, in order to evaluate what properties of molecules that constitutes soluble and potent permeation enhancers. With forest floor plots a distinct interaction in the trained random forest model structure was discovered. Specifically it was recognized that long carbon chains only increased the predicted potency, if the molecule also had a dipole moment above a certain threshold. This seems to be a reasonable rule to identify surfactants, as these must have both a hydrophile and lipophile domain and there will likely be a dipole moment between the two domains [RK12].

As the forest floor method of visualizing feature contributions, had not been described before in literature and it seemed to have some advantages over other diagnostic methods for similar purposes, it was very interesting to develop a generic visualization tool which could assist other random forest users in various fields to understand the overall structure of the trained model. The outcome has been a statistical package computing feature contributions and diagnostic tools to direct how the high dimensional model can be reduced to low dimensional visualizations. Today the package has been downloaded 5000 times, I hope to see the amount of users grow substantially the next years.

APPENDIX A

An Appendix

A.1 Draft: *Learning the structure of random forest models in QSAR modelling: Predicting molecular Solubility*

This draft has been prepared, however not finished. The draft describes how the solubility prediction model was trained and how to visualize the model structure with forestFloor.

DOI: 10.1002/minf.200((full DOI will be filled in by the editorial staff))

Learning the structure of random forest models in QSAR modelling: Predicting molecular Solubility

Søren H. Welling(1,2), Line KH Clemmensen(1), Per B. Brockhoff(1,2) and Hanne HF Refsgaard(1,2).

Dedication((optional))

Abstract: Random forest (RF) models are used in QSPR models to predict solubility by the molecular structure. Non-linear models, such as RF, have been difficult to interpret as the model of many trees each of many nodes is far too complex to comprehend. Instead a model can be understood as a high-dimensional mapping structure which can be decomposed into a series of main effects and interactions. With feature contributions, and a newly developed tool it is possible to produce 2D and 3D visualizations to browse the model structure. We have built a model of 12 standard molecular descriptors on a very cited data set of 1200 molecules and illustrated how a RF model fit weigh the information to produce predictions of solubility. It appears that interactions between used descriptors have a minor contribution on solubility prediction accuracy. The exemplified particular RF model fit can be boiled down to a series non-linear transfer functions, one for each descriptor, and some minor interactions. Moreover, the error of making such a specific generalization can be quantified. The proposed tool will likely be useful to interpret many other RF based QSAR models.

Keywords: keyword 1, keyword 2, keyword 3, keyword 4, keyword 5

1 Introduction

Quantitative structural activity/property relationship (QSAR/QSPR) models have been used to perform solubility predictions, and have e.g. been used in the pharmaceutical industry to select drug candidates for oral delivery. Insufficient solubility is likely lead to lower bioavailability [10]. Related, QSAR models have been used to estimate impact of pesticides on aquatic environment as a function estimated molecular octane/water partition coefficients (logP) [11].

QSAR models represent an empirical approach to establish a relationship between measured properties such as molecular solubility and a numerical description of molecules. Molecular formulas, SMILES or connection tables are graph representations of connected atoms by different types of bonds[cite]. These representations can be encoded to produce numerical descriptions, such as *molecular weight* or *ratio of rotatable bonds.* Molecular descriptors can make use of physicochemical theoretical calculations to estimate internal partial charges between atoms to predict e.g. polarity of the molecule [8,PEOE]. Prediction by other empirical derived models of logP(*SlogP*) or molar refractivity(*SMR*), can be reused to predict solubility [4(SlogP/SMR)]. Molecular descriptors, based on atomic contributions or functional group contributions, will naively view the molecule as a simple sum of its atoms or functional groups. Scores for each type of atom or functional group are fitted to explain a data set of measured logP or molar refractivity. Other descriptors such as Kierhall can quantify how branched the molecular graph is[cite]. Finally encodings can perform a 2D or 3D force field simulations predicting an energy favourable conformation of the molecule (MFA,dipole[cite]).

[section on the previous models and descriptors from ESOL, huuskonnen, Delaney,]

Multiple linear regression (MLR) has been used to find a linear relationship between molecular descriptors and the predicted property. Often within a narrow selection of related molecules [cite sulphonate prediction] or when the molecular descriptors are well designed, linear models will perform well[zheng]. The last couple of decades, non-linear models such as support vector machines, neural nets and random forest have improved the prediction performance[cite some review]. These algorithmic models do not rule out unspecified non-linear relationships and neither interactions.

[1] *Department of Applied Mathmatics and Computer Science, Technical University of Denmark, Matematiktorvet, Building 324, 2800 Kgs. Lyngby, Denmark*

[2] *Novo Nordisk Global Research, Novo Nordisk Park 1, 2760 Maaloev, Denmark*
**e-mail:HARE@NOVONORDISK , +45 3075 0367*

Supporting Information for this article is available on the WWW under www.molinf.com

[Continue with new models, new palmer, laura, bergstrom]

Ideally by improving molecular descriptors a complex non-linear regression model would not be needed. In practice it is difficult to adapt to an unknown non-linearity, especially when high. A successful RF model structure should not only be considered as a black-box structure, but it should inspire to new and better feature engineering. A deeper insight of the trained model structure of RF could improve our understanding of predicted molecular solubility and point to further improvements of molecular descriptors.

1.2 Article, aim, goal, approach

We will demonstrate how a RF model structure can be systematically deconstructed and visualized to describe the learned QSAR between molecular descriptors and solubility. In a previous article[cite], we used the concept of feature contributions[kuzmin, anna] to illustrate how a random forest model was able to predict a specific biological activity of molecules in a cell-culture model. Hereafter we investigated the topology of feature contributions and made a new tool, forest floor, to visualize and understand the structure of random forest models[cite]. In this article we build a conventional QSPR solubility model based on a highly cited and reused data set[((huskonnen)), Delaney, palmer, bergs-trøm, laura] to discuss how RF utilize this information of molecular descriptors to produce predictions of solubility. [specify athours contribution, hertil gør palmer normalt så langt går vi videere]

[Beskriv composition af artikel]

2 Method

2.1 Introduction to random forest regression

A random forest model[leo] is a bootstrap aggregate ensemble model (bagging). It consists of hundreds or thousands of individual decision trees, whom are aggregated to form a joint robust, yet adaptive, ensemble model. Growing each decision tree starts with drawing *N* samples from the training set with replacement. Hereby, in average a .631 fraction out of N training set molecules are sampled to the root node of a tree at least once. These samples for a given tree are the inbag samples. The root node has a node prediction defined as the average measured solubility of the molecules in the node. The mean square error (mse) of the root node prediction can be reduced by splitting into two daughter nodes. Molecular descriptor are used to search for a splitting rule. A splitting rule (larger or equal to a value by one molecular descriptor) will split the node into two daughter nodes. One daughter node will have a higher molecular solubility and one with a lower than the parent node. The best split will lower mse of predicted solubility in the daughter nodes the most. By default, only a random third of descriptors are evaluated in any node to ensure not only one dominant molecular descriptor greedily is used first. Every node is recursively split until node size reaches 5 or less. Then the node is designated as terminal node. To perform predictions, new samples are passed down the tree according to the split rules. The terminal nodes will make up the possible solubility predictions of the tree. These almost fully grown trees are likely low biased, as the potential model structure is very flexible. Though individually, a tree has a high variance as each prediction is based on only 5 or less samples. To counter the instability of each tree, the bootstrapping and only evaluating a random third of descriptors in every node ensure low correlation between trees. When trees are less correlated and the variance is random and symmetrical, the learned structure will be amplified and the variance will be averaged out [breiman].

For every tree, a set of molecules will be out-of-bag (OOB) in contrary to inbag, when used to grow the tree. To estimate the accuracy of the model fit, it must be cross validated. Any sample will be OOB in roundly one third of the trees in the model and thee tree can independently predict this sample. The cross validated prediction error is the expected performance of the model if new molecules were predicted, assuming these molecules were drawn independently from the same population as the training set. OOB cross validation is faster and yields comparable estimates as 5-fold cross validation [wessivik]. Variable importance (VI) can be used to order molecular descriptors by usefulness to the model. VI is the decrease of OOB cross validation performance (mse) if a given molecular descriptor, after growing trees, but before predicting OOB samples, was permuted (random shuffled) [caroline strobl]. VI can be used for variable selection or as in the visualizations of this paper, to bring the attention to the most useful variables first. Random forest and feature contributions can also be used for probabilistic classification[mig, anna]. In a QSAR context mainly regression is used.

2.2 Decomposing a RF model with feature contributions

The applied methodology, forest floor, does not visualize directly the decision trees of the random forest. With hundreds or thousands of trees, it is intractable for a user to comprehend the overall structure of a trained RF model by inspecting the trees. Instead the model can be understood as the learned mapping function (*f*), that maps from a feature space of molecular descriptors (*X*) to a physicochemical target ($\hat{\boldsymbol{y}}$). *X* has as many dimensions as features in the model. The geometrical shape of the model mapping can neither be visualized nor comprehended directly, as the mapping is likely non-linear and high dimensional. Instead, projections or decompositions are needed to visualize the structure with only 2-3 dimensions. Feature contributions [kuzmin, anna] serve as a particular useful decomposition of the prediction for each descriptor, which assist to choose the optimal visualization of the model structure.

A random forest algorithm ($\boldsymbol{g}$) when trained on a data set of *N* solubility measurements $\boldsymbol{y_i}, \boldsymbol{i} \in \{\boldsymbol{1}, \ldots, \boldsymbol{N}\}$ and encoded molecular features ($\boldsymbol{X_i}$) adjusted with a set of parameters ($\boldsymbol{\omega}$) will yield a model fit ($\boldsymbol{f}$). This model fit maps from any point in feature space ($\boldsymbol{X}$) of molecular descriptors to a predicted solubility scale ($\hat{\boldsymbol{y}}$). This mapping can be understood as a high dimensional geometrical structure. A decomposition is used to visualize and navigate what model structure connects $\boldsymbol{X}$ and $\hat{\boldsymbol{y}}$ in 2D or 3D visualizations. The simplest and perhaps adequately correct decomposition splits the solubility prediction into separate effects with one unique function to explain each molecular descriptor.

$$\underbrace{\boxed{\begin{array}{c}\textit{Training}\\ \boldsymbol{g(X_i, y_i, \omega)}\end{array}} \rightarrow \boxed{\begin{array}{c}\textit{Model fit}\\ \boldsymbol{\hat{y}_i = f(X_i)}\end{array}} \approx \boxed{\begin{array}{c}\textit{Main effects only}\\ \boldsymbol{f_1(x_{1i}) + f_2(x_{i2}) + \cdots + f_j(x_{ij}) + \bar{y}}\end{array}}}_{\underline{\textit{2D structures}}} \approx \underbrace{\boxed{\begin{array}{c}\textit{2}^{nd}\textit{ order interactions}\\ \boldsymbol{f_{1,2}(x_{1i}, x_{2i}) + \cdots + \bar{y}}\end{array}}}_{\underline{\textit{3D structures}}}$$

Figure x: The Random forest algorithm is a function that when given a data set and training parameters will output a model fit. This model structure can as a start be interpreted as concisting of main effects only and visualized in 2D. Any deviation from a main effect only can be visualized as a 2nd order interaction.

[up to 2nd order interactions, include. Make dash line boxes to indicateconcept formula]

Hereby the model fit $\boldsymbol{f}$ can be simplified to series of additive functions $\boldsymbol{f_1}$+$\boldsymbol{f_2}$+…, which separately can be plotted in 2D. Feature contributions are used to estimate such additive functions and allows an isolated interpretation of each molecular descriptor.

2.3 Computation of feature contributions?

Every root-, intermediary- or terminal node of a decision tree is an individual prediction. When a parent node is split by a given variable, the daughter nodes will each receive some of the inbag samples and hereby construct two new predictions. A local increment is the change of node predictions from a parent node to a daughter node. For any sample, the RF prediction is simply the sum of all its encountered local increments divided by number of trees plus the grand mean of the training set. Feature contributions are constructed by the same local increments divided by number of trees, but feature contributions are summed separately for each sample by each variable. Thus a feature contribution can be understood as the average change of prediction for one sample molecule due to the information of one specific molecular descriptor - given all other molecular descriptors. *Given all*, means in practice that any interaction structure is preserved in the feature contributions.

When computing feature contributions for a training set, the yielded feature contributions can be arranged as a matrix with same dimensions as the molecular descriptor training matrix X_{ij}. Feature contributions can be denoted F_{ij}. Any prediction $\hat{\boldsymbol{y}}_{\boldsymbol{i}}$ can be split into separate contributions attributed each of the molecular descriptors plus the grand mean of all solubility measurements ($\bar{\boldsymbol{y}}$).

$$\hat{\boldsymbol{y}}_{\boldsymbol{i}} = \sum_{\boldsymbol{j=1}}^{\boldsymbol{p}} \boldsymbol{F_{ij}} + \bar{\boldsymbol{y}}$$

To estimate the most accurate RF model structure it is most efficient to use any available training sample. To visualize the model structure it is also preferable to use all training predictions to compute feature contributions. Just as training predictions of a RF model can be out-of-bag cross-validated, so can feature contributions. Cross validated feature contributions yields fewer random ripples in the visualized model structure. These random ripples arise from the inherent overfitting of individual decision trees. [forestFloor]

2.4 Plotting, quantifying goodness-of-visualization and identifying latent interactions.

The first way to plot feature contributions for a given molecular descriptor is as a function of the corresponding descriptor values, and this function can be fitted with an estimator. For this purpose, we suggest an estimator based on leave-one-out k-nearest neighbour Gaussian distance weighting, as it can fit most RF model structures and produces a fast cross-validation.

$$\boldsymbol{E(F_{,j}, X_{,j}) \rightarrow f_j(X_{ij}) = \hat{F}_{ij}}$$

Hereby is obtained, a 2-axes plot of feature contributions (y-axis) versus the corresponding molecular descriptor values (x-axis) plus a fitted line describing the trending main effect not considering any interactions. See Figure 1 of the result section as an example. In Figure 1 the y-axis is feature contributions for any molecule in training set by a specific molecular descriptor (x-axis).

The fitted line may be an inadequate description, as a random forest model possibly may also have captured one or more interaction effects related to this molecular descriptor. The cross-validated explained variance of the feature contributions (R^2) by the fitted estimator quantified how well the 2D visualization describes the descriptor effect as a main effect only.

$$\boldsymbol{R^2_{f_j} = 1 - \frac{\sum_{i=1}^{N}(F_{ij} - \hat{F}_{ij})^2}{\sum_{i=1}^{N}(F_{ij})^2}}$$

If the explained variance is e.g. only 50%, one may choose to find a better context to understand the feature contribution. A broader context can be plotted as a 3D plot where the feature contribution e.g. can be plotted as a function of the two descriptors, e.g. by the first and second descriptor

j={1,2}. Again the feature contributions can be fitted with an estimator and the goodness-of-fit can be quantified.

$$\boldsymbol{E}\left(\boldsymbol{F}_{i,(1,2)}, \boldsymbol{X}_{i,(1,2)}\right) \rightarrow \boldsymbol{f}_{\boldsymbol{j}}\left(\boldsymbol{X}_{\boldsymbol{i},(1,2)}\right) = \widehat{\boldsymbol{F}}_{\boldsymbol{ij}}$$

In the 3D plot the estimated fit will no longer be a line but a surface, see the fitted surfaces in Figure 2. Unexplained variance of the estimated surface may remain; perhaps a 4D visualization is needed to explain an interaction between 3 molecular descriptors. Fortunately, we observe for random forest models in several data sets, that main effects tend to dominate over second order effects, which tend to dominate over higher order effects[cite me]. Thus, visualizing a model structure in 2D and 3D is likely adequate for most practical purposes.

Colour gradients can be used to provide one extra dimension. The molecule samples in a visualisation can be assigned to a colour gradient reflecting a latent variable to visually identify possible local or global interactions. A local interaction is understood as an interaction effect only learned in a smaller confined part of the model structure. A local interaction for a group of molecules can be highlighted with a colour pattern. In Figure 4 in result section such a highlighting is used to visualize the local model structure for 57 polychlorinated bi-phenyl molecules .

2.5 Software implementations

All visualizations in this article were produced with the R package forestFloor (1.8.9) [cite forestFloor cran]. The supplementary file of this article contains scripts to reproduce the model and visualizations of this paper. The forestFloor package depends on the rgl[Duncan, version] package to produce 3D visualizations, the kknn[cite] package for function estimators and the Rcpp [eddelbuettel, versoion] package to integrate functions implemented in C++ with the R environment. The RF models were trained with randomForest packae [liaw, version]. All packages are available from the CRAN repository [cite cran].

2.6 Data set and molecular descriptors

A public data set by Huuskonnen *et al*[3] was chosen because it is well cited and as it has been reused in many other datasets [palmer,Delaney,bergstrøm,wiisinger,Laura]. Training set and test set were merged in to on single data set of 1256 molecules. SMILES were imported to the software with the application MOE [cite] and sequentially pre-processed with the following functions: 'wash' (simulating an ideal solubilised molecular form), 'partial charges MMFFA96x' calculating the electron densities necessary for a number of descriptor algorithms, and finally 'energy minimize' relaxing the molecule in the minimum 3D state as suggested by [palmer]. To limit the scope of this article, only a small selection of 12 common and useful descriptors identified by Palmer *et al*[palmer] were used. The full data set with descriptors is provided in supplementary materials.

3 Results

3.1 Visualising main effects

A default random forest model of 2000 trees and mtry=4 was trained on the data set. Mean test error of 20 repeated 10-fold was 0.636(+/-0.004) sd? and r^2_s = .903(+/-0.001). Which was a similar performance as [palmer, huskonnen, laura?, hou?]. With the default RF model, out-of-bag feature contributions for every molecule were plotted as a function of the respective features/descriptors (main effect plots). *SlogP* was the most important descriptor by variable permutation importance and plotted first in upper left corner, followed by other descriptors in a decreasing order. A negative linear relation with solubility contribution was observed. High *SlogP* yielded negative contribution to solubility. A flattening of the main effect curve was observed in both ends. Fitted lines and calculated explained variance hereof described how well each molecular descriptor could be regarded as a main effect. The explained feature contribution variance by fitted main effect lines ranged from 90%-87% for the first 7 most important descriptors. Hereafter declined the explained main effect to range .71 to .48 explained variance. And the least important descriptor was only explained 15% as a main effect. Hence, the latter 5 variables were poorly described as main effects, where at the same time less influential for the model prediction deemed on the variable importance and as seen in Figure 1 the absence of feature contribution variance. Thus, overall to visualize the entire random forest model fit as strictly additive explained by the sum 12 main effect estimators explained 89% of the cross validated predictions. Thus to view these descriptors as contributing individually additively to the prediction of solubility would be a fair generalization of this particular instance of a random forest model fit.

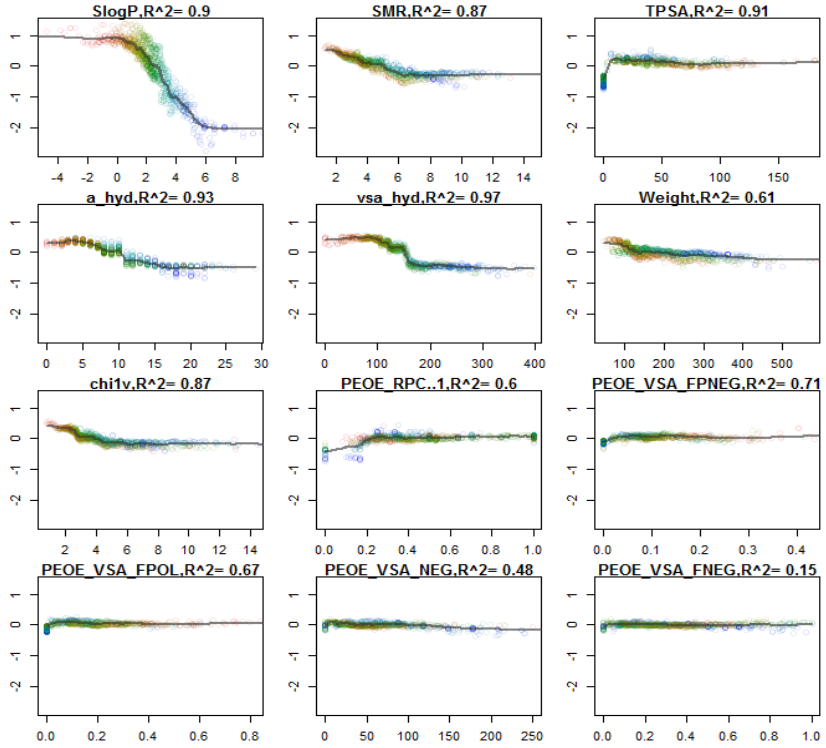


Figure 1. *Main-effect illustration of the 12 descriptors ordered by variable importance. Each molecule is represented once in each plot as a point of a specific colour. Point colour is by defined SlogP descriptor value of each molecule, corresponding to horizontal colour gradient in SlogP plots. A horizontal/diagonal gradient indicates local interactions with* SlogP. *Black lines + R^2 values are estimated fits, a strictly non-interaction interpretation of molecular descriptor effect, as described in equation 1.*

3.2 Identifying and visualizing interactions

To step beyond a strictly main effect interpretation, interaction effects must be identified. A colour gradient (red-yellow-green-teal-blue) horizontally aligned with the *SlogP* axis was used to characterize *SlogP* value of molecules in all other plots. Each molecule will have the exact same colour in all plots. Correlations and interactions with *SlogP* were visually highlighted with this colour gradient. Molecular descriptors correlating with *SlogP*, reproduced the colour gradient horizontally as observed for *SMR*, *PEOE_VSA_NEG*, *vsa_hyd*, *a_hyd*, *Weight*, *chi1v*). Other descriptors *TPSA* and *PEOE_VSA_FPOL* showed a reversed horizontal colour gradient as these descriptors negatively correlated with *SlogP* within the data set R_p~0.5. For all descriptors, deviations from fitted main effect lines were observed. Thus, the variance of each individual feature contribution could not entirely be explained by the descriptor alone. Molecules with specific *SlogP* values indicated by colour gradient were observed to deviate from the fitted lines in specific patterns. Hence, such deviations from a pure main effect could be explained by the many upstream decision splits by the *SlogP* or other correlated descriptors. In Figure 1 a low *Weight*(<120 Dalton) was attributed to a positive contribution to solubility, only when *SlogP*<1.5 (red/yellow). Molecules with high (*SlogP*>4, blue) had a feature contribution near zero for any molecular weight. Only 61% of the feature contribution variance of *Weight* was explained by the fitted main effect line. The remaining variance was thus attributed to interactions, such as the interaction with *SlogP* identified with the colour gradient. *Weight* was a descriptor with medium importance, yet poorly explained as a main effect. Hence, it was found as needed to elucidate the model contribution of *Weight* further. Figure 2 depicts in 3D the feature contributions of *Weight* for every molecule plotted by *Weight* and *SlogP*.

Again the interaction effect between *SlogP* and *Weight* could be observed. The fitted surface, explains the contribution of *Weight* (z-axis) as a main effect by *Weight* (x-axis) itself and as an interaction by *SlogP* (y-axis). This fit increased the explained feature contribution variance to 90%. In figure 2, it

was observed that there were no examples of molecules with low *Weight* and low *SlogP*. Thus this part the RF model structure is extrapolated and the model structure is less likely to be predictive for any such molecule. That the boiling point of small apolar molecules (e.g. propane, halothane etc.) is far below room temperature likely explains no such learning examples exist.

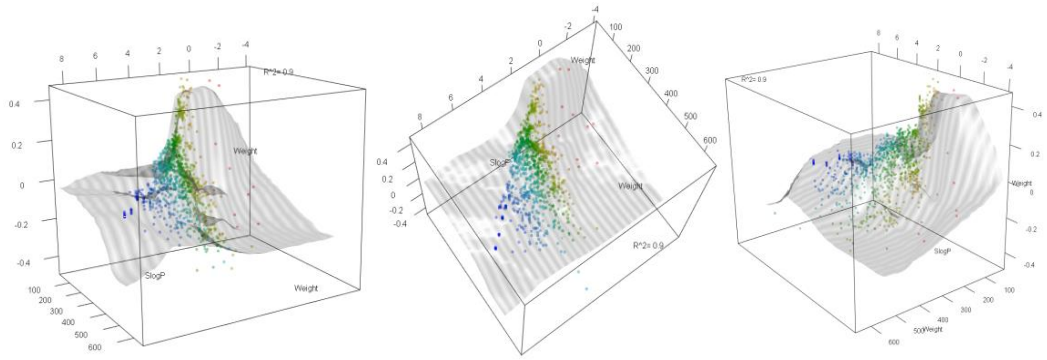


Figure 2. Feature contributions of *Weight* (z-axis) versus feature values *Weight* (X-axis) and feature values *SlogP* (Y-axis). Surface visualizes the fitted estimator, which describes 90% of the variance. Colour gradient parallel to *SlogP* axis as in figure 1. Image visualizes an interaction where *Weight* contributes most to solubility prediction when *SlogP* is negative.

The *SMR* feature was the second most important feature. The main effect of *SMR* feature contribution (molecular refraction by atom contributions) was explained 87%. When viewed as an interaction with *logS*, 95% of the feature contribution variance was explained. *SMR* is intended to approximate the polarizability of molecules, such that these e.g. can form induced dipoles in polar solvents and obtain an energy favourable charged interaction with water[cite]. Such an effect may have been anticipated to contribute in general positively to solubility, but in fact as main effect SMR contribute negatively to solubility. As molar refractivity is the 'polarizability per molecule', this measure was highly correlated with Weight (r_p =.93). If the SMR feature was divided by Weight and the RF model was refitted. The *SMR* feature dropped to the 11th most important feature and the main effect was flat.

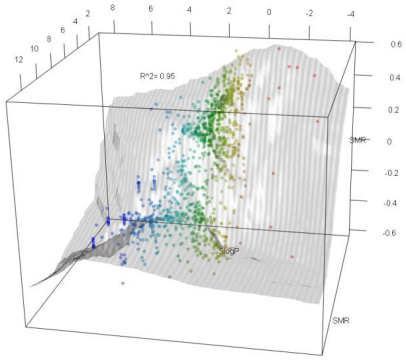


Figure 3. Interaction plot of *SMR* feature contribution as function *SMR* and *SlogP*. This fitted estimator describes 95% of variance of the feature contributions of *SMR*.

3.2 Identifying local effects

In main effect plot figure 1, a distinct group of 57 molecules with low *logS* showed distinct interactions in *SlogP, SMR, TPSA and PEOE_RPC..1*. The group of molecules can be identified in figure 2 middle plot, as having a perfect linear relationship between logS and SMR (r_p=1). In figure 4, the position of these molecules in the model structure was highlighted by colouring any other molecule black. The observed interactions was for *SlogP* a flattening of the negative contribution to solubility of molecules with *SlogP* above 5 whereas ~15 non PCP molecule with SlogP >5 were predicted decreasing soluble as a function of *SlogP*. For SMR a linear reduction in solubility as contrary to the general main effect.

Furthermore these molecules were not only on a line but only placed on 10 different steps with equal distance between them. The molecules were isolated and showed in table 3.

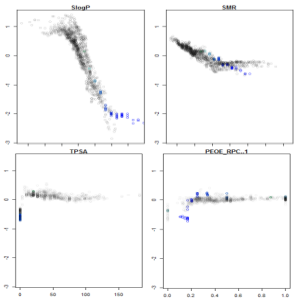


Figure 4. Highlighted feature contributions of PCB molecules.

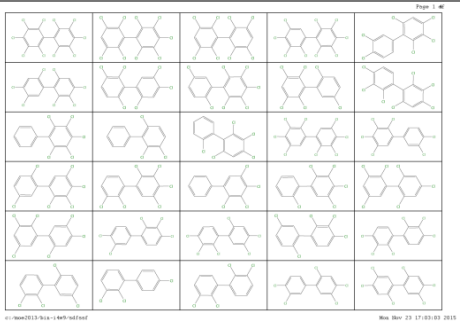


Table 1: Depiction of 36 PCB molecules. What kind of table would be fine here?

It showed that all molecules were polychlorinated biphenyl compounds (PCB). As both *SlogP* and *SMR* are defined by atomic contributions, all PCB with the same amount of substituted chloride atoms will have same *SlogP* and *SMR* values. In fact only two features *chi1v* and *PEOE_RPC..1* produced unique feature values for PCB's with same amount of chloride atoms. But these differences in values were minute, and they were more likely to arise from a non-deterministic convergence algorithm estimating the partial charges[cite method]. Also there appeared to be no obvious relationship between these two features and solubility beyond the number of chloride atoms. Moreover the random forest model fit did not seem to capture any relationship related to substitution pattern, as the OOB cross-validated predictions for these PCB with equal amounts of chloride atoms did not correlate with the actual solubility. Predictions ranged only 0.12 logS units for PCB with same amount of chloride atomes, where the predictions ranged 1.3 logS units. Thus the random forest model was unable with the 12 selected features to predict the relationship between PCB substitution pattern and solubility.

3.2 Model structure is affected by training parameters

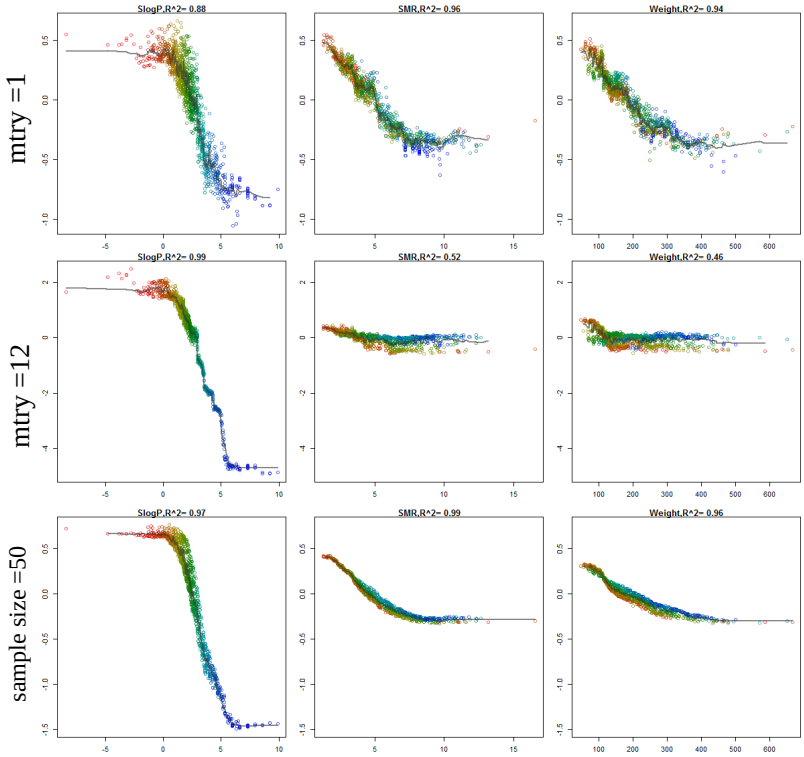


Figure 5: Model structure varies with the parameters. Low mtry(1), uniform use of features. All variables have main effect and interaction effects. High mtry(12,all), algorithm will greedily use best feature first, other features are mainly used for interaction effects. Low sample size(50), smoothens model structure, interactions reduced, model approaches strictly additive model. Sample size is by default 1250 and mtry is by default 4.

Discussion [unfinished]

Choosing a correct set of low dimensional visualizations to account for a complex model structure is not necessarily fully attainable [friedman]. Forest floor can identify and quantify the residuals of any visualization, such that the depiction of the RF model structure can iteratively be elaborated until a sufficiently correct depiction has been attained. Any high dimensional structure cannot be visualized in two or three dimension. In a regression context, a main effect requires 2 dimensions, a 2nd order (interaction) effect requires 3 dimensions and a 3rd order requires 4 dimensions. That said it is possible to understand the 3D structure of a DNA helix from a 2D drawing, and likewise the 4D Kleinn-bottle structure in form a 3D representation. RF is a relatively shallow model and 3rd or higher order interactions or seems almost absent.

This presented methodology of decomposing effects by descriptors, estimating main effects and interactions effects is one representation of the model structure. Another representation such as the actual trained ensemble of decision trees is concise but is too complex to lend itself to a clear interpretation. Another representation, such partial dependence plots can e.g. in 3D describe an interaction effect between two variables. But classic PD plots are not guaranteed to well generalize the overall high dimensional structure, nor do they point to the location of potential sizeable latent interactions. Thus, the forest floor is a methodology that provides the investigator means to browse the model structure of a random forest model and quantify how well a given low dimensional representation, as a series of visualisations, describe the overall structure.

With

3.1 discussion of other methods

With another method to visualize a mapping such partial dependence plots, to uncover hidden interactions and avoid to extrapolation is more difficult.

Today, mainly variable importance [palmer, laura, others] is used in conjunction with random forest models to interpret the model. Variable importance describes the loss of cross-validated predictive performance when each variable in turns were permuted. VI only approximates the usefulness of each molecular descriptor. VI does not outline how each descriptor is used by the model.

[insert in result section] A group of PCB molecules were identified as to elicit a distinctive interaction pattern. With the 12 selected molecular descriptors, was the chloride substitution pattern of this PCP molecules not learned. *SlogP* and *SMR* the most important descriptors are e.g. themselves based predictions on *logP* and molar refractivity for 10.000 measured molecules. Predictions are based summing empirical derived scores for each atom in the molecule. Atoms are categorized by atom number and type bonding to neighbouring atoms. Thus for PCB molecules having the same number of substituted chloride atoms all scores will be exactly alike. [maybe two extra sentrences of why neither other descriptors has any clue of this effect.] Ghavami et al. [6] produced a regression model only to predict solubility of PCB molecules and found that 90 percent of the variance of PCB log solubility can be attributed the number chloride atoms in a linear regression model. Introducing counts of ortho-, meta- and para configuration contributed to explain up to 97% cross-validated variance of the log solubility of PCB molecules. As the PCB molecules collapse to only extending a string of connecting points in the feature space, where each point concist of PCB molecules with same amount chloride atoms, the sampling density around these PCB molecules is high. Thus, is the random forest model able to fit a very specific structure accounting for the solubility variance related to chloride atoms in PCB molecules. If predicting the solubility of a random molecule, it would be unlikely to fall within the small sub feature space of PCB molecules. If it did fall within this subspace, the learned relationship from PCB's would dominate the prediction of the RF model.

First Main Text Paragraph----without indentation.
Main Text Paragraph----with indentation.

((Insert schemes above the captions. **Note:** Please do **not** combine scheme and caption in a textbox or frame))

Scheme 1. Scheme Caption.

Main Text Paragraph----with indentation.

((Insert figures above the captions. **Note:** Please do **not** combine figure and caption in a textbox or frame))

Figure 1. Figure Caption.

Main Text Paragraph----with indentation.

Table 1. Table Caption. ((**Note:** Please do **not** include the table in a textbox or frame))

Head 1[a]	Head 2	Head 3[b]	Head 4[c]	Head 5
Column 1	Column 2	Column 3	Column 4	Column 5
Column 1	Column 2	Column 3	Column 4	Column 5

[a] Table Footnote.
[b] …

3 Conclusions

First Main Text Paragraph----without indentation.
Main Text Paragraph----with indentation.

Acknowledgements

Acknowledgements Text.

References

[1] ((Reference 1, Example for **Journals**))a) A. Author, B. Coauthor, *Mol. Inf.* **2009**, *1*, 1-10; b) A. Author, B. Coauthor, *Angew. Chem.* **2006**, *118*, 1-5; *Angew. Chem. Int. Ed.* **2006**, *45*, 1-5.
[2] ((Reference 2, Example for **Books**))J. W. Grate, G. C. Frye, in *Sensors Update*, Vol. 2 (Eds: H. Baltes, W. Göpel, J. Hesse), Wiley-VCH, Weinheim **1996**, pp. 10-20.))
[3]

[1]
ESOL: Estimating Aqueous Solubility Directly from Molecular Structure
John S. Delaney*
Syngenta, Jealott's Hill International Research Centre, Bracknell, Berkshire, RG42 6EY, United Kingdom
Received October 29, 2003

[2]
Global and Local Computational Models for Aqueous Solubility Prediction of Drug-Like Molecules
Christel A. S. Bergström ,† Carola M. Wassvik ,† Ulf Norinder ,†‡ Kristina Luthman ,§* and Per Artursson †
Center for Pharmaceutical Informatics, Department of Pharmacy, Uppsala University, Uppsala Biomedical Center, P.O. Box 580, SE-751 23 Uppsala, Sweden, Department of Medicinal Chemistry, AstraZeneca R&D, SE−151 85 Södertälje, Sweden, and Department of Chemistry, Medicinal Chemistry, Göteborg University, SE−412 96 Göteborg, Sweden
J. Chem. Inf. Comput. Sci., 2004, 44 (4), pp 1477–1488
DOI: 10.1021/ci049909h
Publication Date (Web): June 23, 2004
Copyright © 2004 American Chemical Society

[3]
Random Forest Models To Predict Aqueous Solubility
David S. Palmer , Noel M. O'Boyle ,† Robert C. Glen , and John B. O. Mitchell *
Unilever Centre for Molecular Science Informatics, Department of Chemistry, University of Cambridge, Lensfield Road, Cambridge CB2 1EW, United Kingdom
J. Chem. Inf. Model., 2007, 47 (1), pp 150–158
DOI: 10.1021/ci060164k
Publication Date (Web): December 2, 2006
Copyright © 2007 American Chemical Society

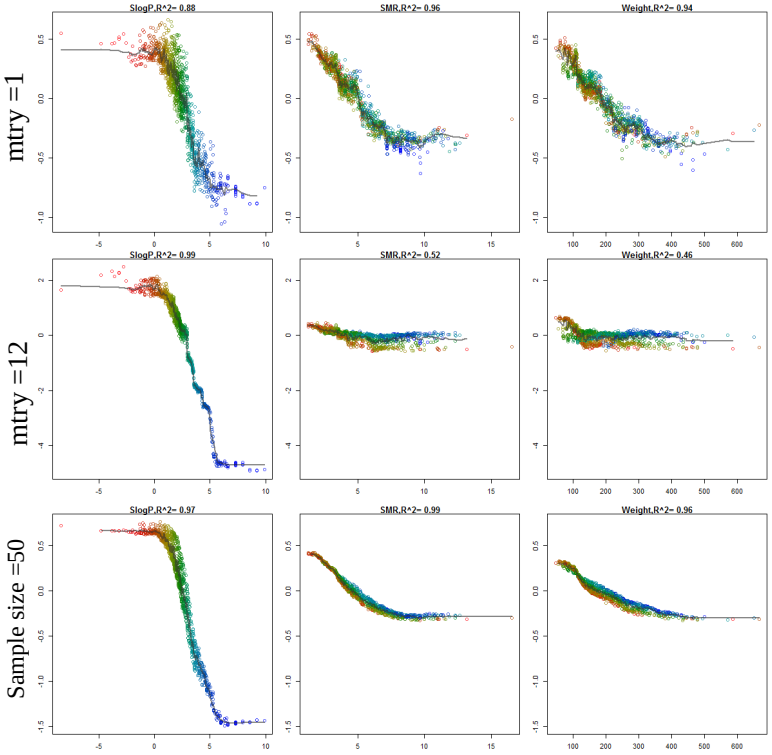
[4] Wildman, S.A., Crippen, G.M.; Prediction of Physiochemical Parameters by Atomic Contributions; J. Chem. Inf. Comput. Sci. 39 No. 5 (1999) 868–873.

[6] Ertl, P., Rohde, B., Selzer, P.; Fast Calculation of Molecular Polar Surface Area as a Sum of Fragment-Based Contributions and Its Application to the Prediction of Drug Transport Properties; J. Med. Chem. 43 (2000) 3714–3717.
[7] [Cruciani 2000]Cruciani, G., Crivori, P., Carrupt, P.-A., Testa, B.; Molecular Fields in Quantitative Structure-Permeation Relationships: the VolSurf Approach; J. Mol. Struct. (Theochem) 503 (2000) 17–30.
[8] Gasteiger, J., Marsili, M.; Iterative Partial Equalization of Orbital Electronegativity - A Rapid Access to Atomic Charges; Tetrahedron 36 (1980) 3219.,

[9] Prediction of drug solubility from structure. William L. Jorgensena, , , Erin M. Duffyb . doi:10.1016/S0169-409X(02)00008-X
[10] Yu, L. X.; Amidon, G. L.; Polli, J. E.; Zhao, H.; Mehta, M. U.; Conner, D. P.; Shah, V. P.; Lesko, L. J.; Chen, M.-L.; Lee, V. H. Biopharmaceutics classification system: the scientific basis for biowaiver extensions. Pharm. Res. 2002, 19 (7), 921−925
[11] OVERVIEW OF DATA AND CONCEPTUAL APPROACHES FOR DERIVATION OF QUANTITATIVE STRUCTURE–ACTIVITY RELATIONSHIPS FOR ECOTOXICOLOGICAL EFFECTS OF ORGANIC CHEMICALS STEVEN P. BRADBURY,† CHRISTINE L. RUSSOM,*† GERALD T. ANKLEY,† T. WAYNE SCHULTZ,‡ and JOHN D. WALKER§ †U.S. Environmental Protection Agency, National Health and Environmental Effect

Received: ((will be filled in by the editorial staff))
Accepted: ((will be filled in by the editorial staff))
Published online: ((will be filled in by the editorial staff))

t showed that all molecules were polychlorinated biphenyl compounds. As both SlogP and SMR are computed by atomic contributions, all PCB with the same amount of substituted chloride atoms will have same SlogP and SMR values. In fact only two features chi1v and PEOE_RPC..1 produced different feature values for PCB's with same amount of chloride atoms. But these differences in values were minute, and they were more likely to arise from a non-deterministic convergence algorithm defining the partial charges. Also there appeared to no relationship between these features, chloirde substitution configuration and actual logS. More over the OOB cross-validated predictions for these PCB varied only 0.12 units where the average variation with PCB with equal many chlorides was 1.3. And this OOB cross validated variation within PCB with equal amounts of chloride did not correlate with actual solubility. Thus the random forest model was unable with the 12 selected features to predict the relationship between PCB substitution configuration and solubility. 90 Percent of the variance of PCB solubility can be attributed the number chloride atoms or any other Ghavami et al. [6] showed how counting ortho meta and para configuration contribute to explain upto 97% cross-validated variance.


SlogP,R^2= 0.88
SMR,R^2= 0.96
Weight,R^2= 0.94
SlogP,R^2= 0.99
SMR,R^2= 0.52
Weight,R^2= 0.46
SlogP,R^2= 0.97
SMR,R^2= 0.99
Weight,R^2= 0.96
mtry =1
mtry =12
Sample size =50

A.2 Random forest Q and A answers with illustrations and code examples

During the last three years, two forums *Stack Overflow* `http://stackoverflow.com/` for learning programming and *Cross Validated* `stats.stackexchange.com` for learning statistics have been invaluable daily sources of fast answers to questions on all levels. Whenever you get stuck and that probably happens 10 times a day, you only have to complain to google about the problem in a sentence. As we are six billions on this planet, odds are favorable for, that some person on the same level already has had this problem before. This person posed the same question in the same silly wording as you do now, and some helpful person has already answered with a practical example, and perhaps also provided answers to other question, you really ought to be asking instead. The cross validated site is relatively small with only 83,000 questions (July 2016), where *Stack Overflow* has 12,000,000 questions to cover most programming languages. For the entertainment value and to test my knowledge on random forest, I started to answer questions my self. By putting my beliefs on public display, I sometimes receive swift criticism, when I am wrong. I have answered 92 questions. Each answer required in average a page and often comprises visulizations, coding, references to other material and took a half to 3 hours to write. I have included some of my favorite answers here, and have referred to these through this thesis. The cross-validated site has almost 100.000 users, and reputation voting wise, I rank top 200. My answers have presently in total 75,000 page views (50.000 views/yr), which is probably some 4 orders of magnitude more than this chapter ever will be read.

A.2.1 Handling unbalanced data with with random forest

2,871 5 17 review help

R package for Weighted Random Forest? classwt option?

I'm trying to use Random Forest to predict the outcome of an extremely imbalanced data set (the minority class rate is about only 1% or even less). Because the traditional Random Forest algorithm minimizes the overall error rate, rather than paying special attention to the minority classes, it is not directly applicable on imbalanced data. So I want to assign a high cost to misclassification of the minority class (cost sensitive learning).

I read several sources that we can use the option `classwt` of `randomForest` in R, but I don't know how to use this. And do we have any other alternatives to the `randomForest` funtion?

r | random-forest

edited Aug 17 '15 at 13:11
Antoine
1,454 7 25

asked Jun 19 '15 at 11:50
Matemattica
529 4 20

1 Answer

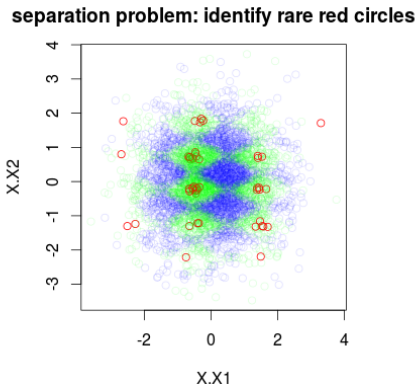
This thread refers to two other threads and a fine article on this matter. It seems classweighting and downsampling are equally good. I use downsampling as described below.

Remember the training set must be large as only 1% will characterize the rare class. Less than 25~50 samples of this class probably will be problematic. Few samples characterizing the class will inevitably make the learned pattern crude and less reproducible.

RF uses majority voting as default. The class prevalences of the training set will operate as some kind of effective prior. Thus unless the rare class is perfectly separable, it is unlikely this rare class will win a majority voting when predicting. Instead of aggregating by majority vote, you can aggregate vote fractions.

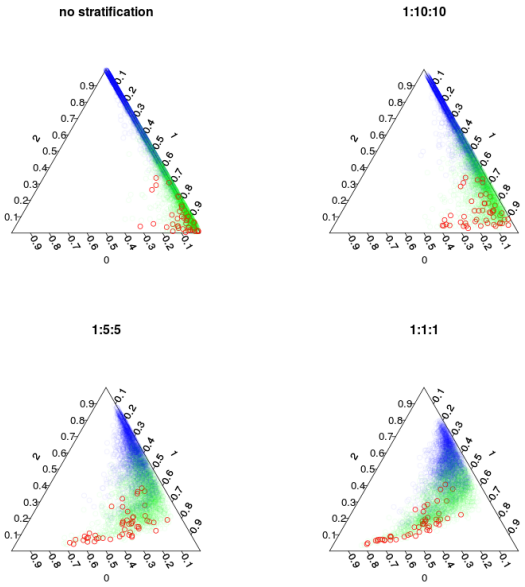
Stratified sampling can be used to increase the influence of the rare class. This is done on the cost on downsampling the other classes. The grown trees will become less deep as much fewer samples need to be split therefore limiting the complexity of the potential pattern learned. The number of trees grown should be large e.g. 4000 such that most observations participate in several trees.

In the example below, I have simulated a training data set of 5000 samples with 3 class with prevalences 1%, 49% and 50% respectively. Thus there will 50 samples of class 0. The first figure shows the true class of training set as function of two variables x1 and x2.

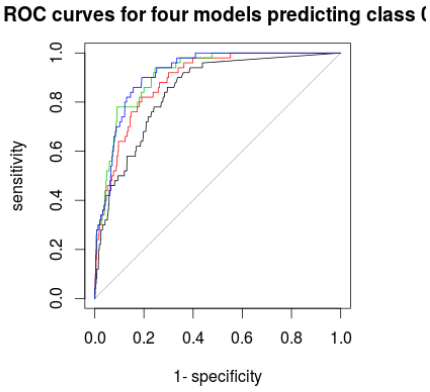


Four models were trained: A default model, and three stratified models with 1:10:10 1:2:2 and 1:1:1 stratification of classes. Main while the number of inbag samples(incl. redraws) in each tree will be 5000, 1050, 250 and 150. As I do not use majority voting I do not need to make a perfectly balanced stratification. Instead the votes on rare classes could be weighted 10 times or some other decision rule. Your cost of false negatives and false positives should influence this rule.

The next figure shows how stratification influences the vote-fractions. Notice the stratified class ratios always is the centroid of predictions.



Lastly you can use a ROC-curve to find a voting rule which gives you a good trade-off between specificity and sensitivity. Black line is no stratification, red 1:5:5, green 1:2:2 and blue 1:1:1. For this data set 1:2:2 or 1:1:1 seems best choice.



By the way, vote fractions are here out-of-bag crossvalidated.

And the code:

```
library(plotrix)
library(randomForest)
library(AUC)

make.data = function(obs=5000,vars=6,noise.factor = .2,smallGroupFraction=.01) {
X = data.frame(replicate(vars,rnorm(obs)))
yValue = with(X,sin(X1*pi)+sin(X2*pi*2)+rnorm(obs)*noise.factor)
yQuantile = quantile(yValue,c(smallGroupFraction,.5))
yClass = apply(sapply(yQuantile,function(x) x<yValue),1,sum)
yClass = factor(yClass)
print(table(yClass)) #five classes, first class has 1% prevalence only
Data=data.frame(X=X,y=yClass)
}

plot.separation = function(rf,...) {
triax.plot(rf$votes,...,col.symbols = c("#FF0000FF",
                                        "#00FF0010",
                                        "#0000FF10")[as.numeric(rf$y)])
}

#make data set where class "0"(red circles) are rare observations
#Class 0 is somewhat separateble from class "1" and fully separateble from class "2"
Data = make.data()
par(mfrow=c(1,1))
```

```
plot(Data[,1:2],main="separation problem: identify rare red circles",
     col = c("#FF0000FF","#00FF0020","#0000FF20")[as.numeric(Data$y)])

#train default RF and with 10x 30x and 100x upsumpling by stratification
rf1 = randomForest(y~.,Data,ntree=500, sampsize=5000)
rf2 = randomForest(y~.,Data,ntree=4000,sampsize=c(50,500,500),strata=Data$y)
    rf3 = randomForest(y~.,Data,ntree=4000,sampsize=c(50,100,100),strata=Data$y)
rf4 = randomForest(y~.,Data,ntree=4000,sampsize=c(50,50,50)  ,strata=Data$y)

#plot out-of-bag pluralistic predictions(vote fractions).
par(mfrow=c(2,2),mar=c(4,4,3,3))
plot.separation(rf1,main="no stratification")
plot.separation(rf2,main="1:10:10")
plot.separation(rf3,main="1:5:5")
plot.separation(rf4,main="1:1:1")

par(mfrow=c(1,1))
plot(roc(rf1$votes[,1],factor(1 * (rf1$y==0))),main="ROC curves for four models predicting
class 0")
plot(roc(rf2$votes[,1],factor(1 * (rf1$y==0))),col=2,add=T)
plot(roc(rf3$votes[,1],factor(1 * (rf1$y==0))),col=3,add=T)
plot(roc(rf4$votes[,1],factor(1 * (rf1$y==0))),col=4,add=T)
```

edited Oct 5 '15 at 17:57

Antoine
1,454 7 25

answered Jun 21 '15 at 21:06

Soren Havelund Welling
2,871 5 17

oups one figure caption says 1:5:5 instead of 1:2:2 – Soren Havelund Welling Jun 21 '15 at 21:36

thank you very much for your detailed answer, that will definitely help me a lot in my daily work. There is one sentence that I don't understand: "Main while the number of inbag samples(incl. redraws) in each tree will be 5000,1050, 250 and 150" .Could you please explain me where does the numbers come from? – Matemattica Jun 22 '15 at 12:26

1 my pleasure ;) in this example the rare class had 50 members. If stratifying 1:10:10 we would need to specify sampsize=c(50,500,500). 50+500+500 = 1050. A fully grown tree of 1050 samples will have 1050x2 nodes in total. – Soren Havelund Welling Jun 22 '15 at 13:07

Sorry if my question is idiot, but what is the meaning of 1:10:10, 1:2:2 and 1:1:1 stratification here? And when you said "the votes on rare classes could be weighted 10 times". Which part of the code represents that? Is it 1:10:10? Thank you very much! – Matemattica Jun 24 '15 at 10:35

1 1:10:10 are the ratios between the classes. The simulated data set was designed to have the ratios 1:49:50. These ratios were changed by down sampling the two larger classes. By choosing e.g. sampsize=c(50,500,500) the same as c(1,10,10) * 50 you change the class ratios in the trees. 50 is the number of samples of the rare class. If you furthermore set keep.inbag=TRUE and inspect rf$inbag, you will see that samples of the rare classes is inbag in ~2/3 trees whereas each non-rare class sample is included in very few trees because of down sampling. – Soren Havelund Welling Jun 24 '15 at 14:46

So, if I understand correctly, it's "down sampling" technique, not weighted random forest? – Matemattica Jun 24 '15 at 14:55

...as stated in third sentence of answer. – Soren Havelund Welling Jun 24 '15 at 14:57

Ok, perfect. Thank you very much for your kind answer:) – Matemattica Jun 24 '15 at 16:07

Add Another Answer

A.2.2 Random forest and outliers I: Outlier Islands

Fully grown decision trees in random forests

Several sources suggest it's ok to fully grow the decision trees in a RF (e.g., Leo Breiman's article and Elements of Statistical Learning, p. 596).

I don't understand the following. Suppose that due to noise, a single data point x of class A ended up somewhere deep inside class B (in terms of its position in the space of features). Every tree that includes x will have a leaf that contains just x alone (because it's fully grown, so it won't stop until each node can precisely identify the class on the training data set; and because x is so isolated from other points of class A, that it can't ever be joined with them in a contiguous region of space carved by a tree). Roughly two-thirds of the trees will include x. Therefore, it seems the majority vote would always be to classify any points close x as if they belong to class A - even though this is clearly overfitting.

A similar argument can be made for any noise that caused a few points of one class to end up in the region that should be assigned to another class.

How is it, then, that fully grown trees inside a RF aren't causing major overfitting?

random-forest cart overfitting

asked Nov 14 '15 at 11:52

max

369 1 12

1 I will admit to being confused by the first sentence of this question. RFs don't "grow" a single tree but many trees. The structure that results from a single tree is lost in the process of developing the forest and aggregating the results across lots of "mini-trees." Therefore, the ensemble nature of the RF answer doesn't represent a "model" that would even be vulnerable to "overfitting." – DJohnson Nov 14 '15 at 12:25

Yes, RF has multiple trees, and each of these trees is fully grown. The structure isn't lost at all, each tree works like usual, the ensemble just counts how many trees predicted each category, and selects the one with the highest count. – max Nov 14 '15 at 18:09

...for practical use of RF, I agree with DJohnson – Soren Havelund Welling Nov 16 '15 at 10:50

1 Answer

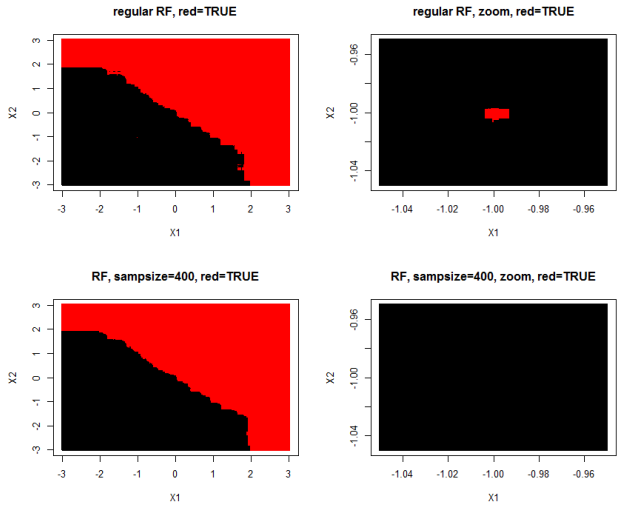
Yes, even a single A-outlier (sample of class A) placed in the middle of many B examples (in a feature space) would affect the structure of the forest. The trained forest model will predict new samples as A, when these are placed very close to the A-outlier. But the density of neighboring B examples will decrease size of this "predict A"-island in the "predict B"-ocean. But the "predict A"-island will not disappear. For noisy classification, e.g. Contraceptive Method Choice, the default random forest can be improved by lowering tried varaibles in each split(mtry) and bootstrap sample size(sampsize). If sampsize is e.g. 40% of training size, then any '*single sample prediction island*' will completely drown if surrounded by only counter examples, as it only will be present in 40% of the trees.

EDIT: If sample replacement is true, then more like 33% of trees.

```
mean(replicate(1000,length(unique(sample(1:1000,400,rep=T)))))
```

I made a simulation of the problem (A=TRUE,B=FALSE) where one (A/TRUE)sample is injected within many (B/FALSE) samples. Hereby is created a tiny A-island in the B ocean. The area of the A-island is so small, it has no influence on the overall prediction performance. Lowering sample size makes the island disappear.

1000 samples with two features X_1 and X_2 are of class "true/A" if $y_i = X_1 + X_2 >= 0$
Features are drawn from $N(0, 1)$




```
library(randomForest)
par(mfrow=c(2,2))
set.seed(1)

#make data
X = data.frame(replicate(2,rnorm(1000)))
y = factor(apply(X,1,sum) >=0) #create 2D class problem
X[1,] = c(-1,-1); y[1]='TRUE' #insert one TRUE outlier inside 'FALSE-land'

#train default forest
rf = randomForest(X,y)

#make test grid(250X250) from -3 to 3,
Xtest = expand.grid(replicate(2,seq(-3,3,le=250),simplify = FALSE))
Xtest[1,] = c(-1,-1) #insert the exact same coordinate of train outlier
Xtest = data.frame(Xtest); names(Xtest) = c("X1","X2")
plot(Xtest,col=predict(rf,Xtest),pch=15,cex=0.5,main="regular RF, red=TRUE")

#zoom in on area surrouding outlier
Xtest = expand.grid(replicate(2,seq(-1.05,-.95,le=250),simplify = FALSE))
Xtest = data.frame(Xtest); names(Xtest) = c("X1","X2")
plot(Xtest,col=predict(rf,Xtest),pch=15,cex=0.5,main= "regular RF, zoom, red=TRUE")

#train extra robust RF
rf = randomForest(X,y,sampsize = 400)
Xtest = expand.grid(replicate(2,seq(-3,3,le=250),simplify = FALSE))
Xtest[1,] = c(-1,-1)
Xtest = data.frame(Xtest); names(Xtest) = c("X1","X2")
plot(Xtest,col=predict(rf,Xtest),pch=15,cex=0.5,main="RF, sampsize=400, red=TRUE")

Xtest = expand.grid(replicate(2,seq(-1.05,-.95,le=250),simplify = FALSE))
Xtest = data.frame(Xtest); names(Xtest) = c("X1","X2")
plot(Xtest,col=predict(rf,Xtest),pch=15,cex=0.5,main="RF, sampsize=400, zoom, red=TRUE")
```

edited Nov 16 '15 at 10:46

answered Nov 16 '15 at 10:27

Soren Havelund Welling
2,871 5 17

Lowering sample size, but still with replacements, correct? In that case, only 40% * ~2/3, or roughly 25% of the original sample is used. I would have assumed that once you use less than 75% of the original sample, due to replacement less than half the trees will see the point, and so the island will disappear - but maybe I'm missing some detail. And about lowering `mtry` : it won't make the island disappear, but would reduce the noise in general, right? – max Nov 16 '15 at 10:31

40% without replacement, 33% with replacement. try simulate with mean(replicate(1000,length(unique(sample(1:1000,400,rep=T))))) – Soren Havelund Welling Nov 16 '15 at 10:39

lowering mtry will not make islands disappear. But for high dimensional data I guess it will make the islands smaller. – Soren Havelund Welling Nov 16 '15 at 10:42

In the case of binary classification, let's say the number of trees that see any given observation falls below 50% (so if you use replacements, you can start with maybe ~60%). Wouldn't that be guaranteed to remove the island if it was formed by a single observation? It would seem the majority vote would be for the other class at that point? – max Nov 16 '15 at 18:20

So in practice don't worry about it :) If you have 1000 samples and 500 trees and sample 600 for each tree with replacement. You are likely to have still some ~10 samples who happend to get picked for more than 250 trees. You can simulate the distribution with this one-liner: plot(table(table(unlist(replicate(500,unique(sample(1:1000,600,rep=T))))))) – Soren Havelund Welling Nov 16 '15 at 22:15

Add Another Answer

A.2.3 Random forest and outliers II: Robustness

How are Random Forests not sensitive to outliers?

I've read in a few sources, including this one, that Random Forests are not sensitive to outliers (in the way that Logistic Regression and other ML methods are, for example).

However, two pieces of intuition tell me otherwise:

1. Whenever a decision tree is constructed, all of the points must be classified. This means that even outliers will get classified, and hence will affect the decision trees where they were selected during boosting.
2. Bootstrapping is a part of how a RandomForest does sub-sampling. Bootstrapping is susceptible to outliers.

Is there any way to reconcile my intuition about its sensitivity to outliers, with sources that disagree?

random-forest bootstrap outliers cart

edited Dec 19 '15 at 16:58 m0nhawk 150 2 8

asked Dec 17 '15 at 6:23 Hunle 87 1 9

The answer, below, is very good. The intuitive answer is that a decision tree works on splits and splits aren't sensitive to outliers: a split only has to fall anywhere between two groups of points to split them. – Wayne Dec 20 '15 at 15:15

So I suppose if the `min_samples_leaf_node` is `1`, then it could be susceptible to outliers. – Hunle Dec 21 '15 at 0:13

yes min_samples and bootstrap sample can completely remove the influence of 1b outliers in RF regression – Soren Havelund Welling Dec 21 '15 at 10:44

Some statisticians obtain a tunnel vision on those inliers, that one can predict and understand. Cherish the outliers as 'known unknowns' and wonder if your business model is fragile towards them. Some outliers are fundamentally unpredictable, but their impact is very real ... a paraphrase of N. Taleb's, 'Black Swan' – Soren Havelund Welling Dec 21 '15 at 10:52

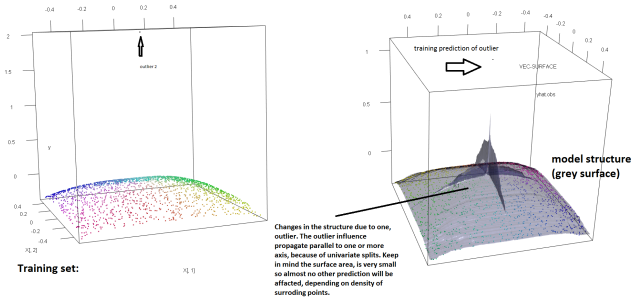
3 Answers

outlier 1a: This outlier has one or more extreme feature values and is placed distant to any other sample. The outlier will influence the initial splits of the trees as any other sample, so no strong influence. It will have low *proximity* to any other sample, and will only define the model structure in a remote part of feature space. During prediction most new samples are likely not to be similar to this outlier, and will rarely end up in the same terminal node. Moreover decision trees regards features as if they were ordinal(ranking). The value is either smaller/equal to or larger than break point, thus it does not matter if a feature value is an extreme outlier.

outlier 1b: For classification one single sample may be regarded as an outlier, when embedded in the middle of many sample of a different class. I described earlier how a default RF model will get influenced by this one sample of odd class, but only very close to the sample.

outlier 2: This outlier has an extreme target value perhaps many times higher than any other values, but the feature values are normal. A .631 fraction of the trees will have a terminal node with this sample. The model structure will get affected locally close to the outlier. Notice the model structure is affected mainly parallel to the feature axis, because nodes are split uni-variately.

I included a RF-regression simulation of outlier_2. 1999 points drawn from a smooth rounded structure $y = (x_1^4 + x_2^4)^{\frac{1}{2}}$ *and one outlier with a much higher target value(y=2,* x_1 *=0,*x_2 *=0). The training set is shown to the left. The learned RF model-structure is shown the the right.*




```
library(forestFloor)
library(randomForest)
```

```
library(rgl)
set.seed(1)

X = data.frame(replicate(2,runif(2000)-.5))
y = -sqrt((X[,1])^4+(X[,2])^4)^1
Col = fcol(X,1:2) #make colour pallete by x1 and x2
#insert outlier2 and colour it black
X[1,] = c(0,0);y[1]=2 ;Col[1] = "#000000FF" #black

#plot training set
plot3d(X[,1],X[,2],y,col=Col)

rf = randomForest(X,y)
vec.plot(rf,X,1:2,col=Col,grid.lines = 400)
```

EDIT: comment to user603

Yes for extreme outliers on target scale, one should consider to transform target scale before running RF. I added below a *robustModel()* function which tweaks randomForest. Another solutions would be to log transform before training.

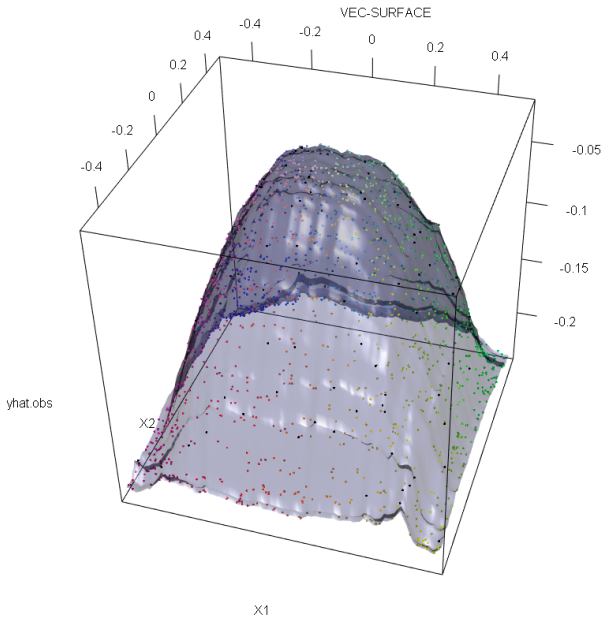
```
.
##---code by user603
library(forestFloor)
library(randomForest)
library(rgl)
set.seed(1)

X<-data.frame(replicate(2,runif(2000)-.5))
y<--sqrt((X[,1])^4+(X[,2])^4)
Col<-fcol(X,1:2) #make colour pallete by x1 and x2

#insert outlier2 and colour it black
y2<-y;Col2<-Col
y2[1:100]<-rnorm(100,200,1);    #outliers
Col2[1:100]="#000000FF" #black
##---

#function to make models robust
robustModel = function(model,keep.outliers=TRUE) {
  f = function(X,y,lim=c(0.1,.9),keep.outliers="dummy",...) {
  limits = quantile(y,lim)
  if(keep.outliers) {#keep but reduce outliers
  y[limits[1]>y] = limits[1] #lower limit
  y[limits[2]<y] = limits[2] #upper limit
  } else {#completely remove outliers
    thrashThese = mapply("||",limits[1]>y,limits[2]>y)
    y = y[thrashThese]
    X = X[thrashThese,]
  }
  obj = model(x=X,y=y,...)
  class(obj) = c("robustMod",class(obj))
  return(obj)
  }
  formals(f)$keep.outliers = keep.outliers
  return(f)
}

robustRF = robustModel(randomForest) #make RF robust
rh = robustRF(X,y2,sampsize=250)     #train robustRF
vec.plot(rh,X,1:2,col=Col2)          #plot model surface
mean(abs(rh$predict[-c(1:100)]-y2[-c(1:100)]))
```



edited Dec 21 '15 at 15:58

answered Dec 19 '15 at 16:38

Soren Havelund Welling
2,871 5 17

You write "no other prediction will be affected". If you shift your single outlier to put `y[1]=200` you will see that it single handedly causes the prediction error *on the uncontaminated observations* to jump by a factor of 20! – user603 Dec 21 '15 at 14:08

@user603 True that, In such cases the target scale can be transformed monotonically before handed to RF. I added a 'robustModel: makes models robust' to my answer.....of course to predict such random target outlier(s) (type 2) remains impossible, but the remaining model structure do not have to suffer – Soren Havelund Welling Dec 21 '15 at 16:00

The Log transform is not, in general, a solution against outliers (it merely hides the problem). The robustification of RF you propose is essentially the approach advocated in Galimberti, G., Pillati, M., & Soffritti, G. (see my answer). The main difference is that your "robustModel" approach has a maximum breakdown point of 25% on the response space (it can withstand 25% or arbitrary 'y'-outliers) whereas theirs has a bdp of 50%. Note that neither approach is robust to outliers in the design space. – user603 Dec 21 '15 at 16:06

Your intuition is correct. This answer merely illustrates it on an example.

It is indeed a common misconception that CART/RF are somehow robust to outliers.

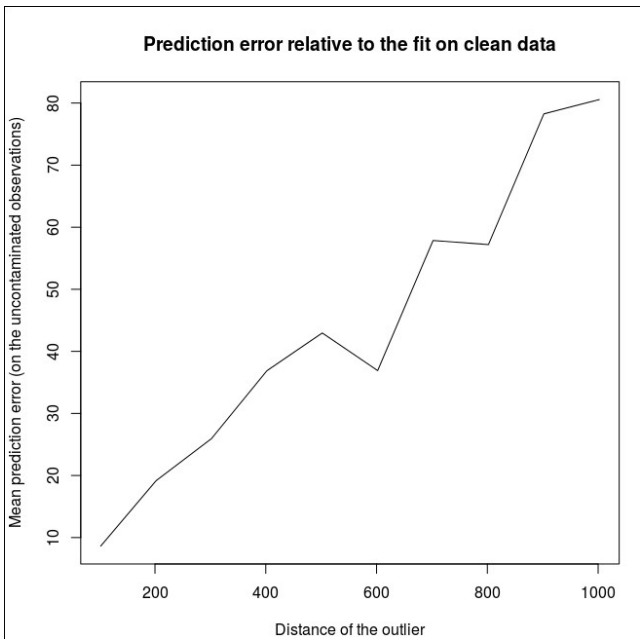
To illustrate the lack of robustness of RF to the presence of a single outliers, we can (lightly) modify the code used in Soren Havelund Welling's answer above to show that a **single** 'y'-outliers suffices to completely sway the fitted RF model. For example, if we compute the mean prediction error *of the uncontaminated observations* as a function of the distance between the outlier and the rest of the data, we can see (image below) that introducing *a single* outlier (by replacing one of the original observations by an arbitrary value on the 'y'-space) suffices to pull the predictions of the RF model arbitrarily far away from the values they would have had if computed on the original (uncontaminated) data:

```
 library(forestFloor)
library(randomForest)
library(rgl)
set.seed(1)

X = data.frame(replicate(2,runif(2000)-.5))
y = -sqrt((X[,1])^4+(X[,2])^4)
X[1,]=c(0,0);
y2<-y
rg<-randomForest(X,y)   #RF model fitted without the outlier
outlier<-rel_prediction_error<-rep(NA,10)

for(i in 1:10){
    y2[1]=100*i+2
    rf=randomForest(X,y2)   #RF model fitted with the outlier
    rel_prediction_error[i]<-mean(abs(rf$predict[-1]-y2[-1]))/mean(abs(rg$predict[-1]-
y[-1]))
    outlier[i]<-y2[1]
}
plot(outlier,rel_prediction_error,type='l',ylab="Mean prediction error (on the
```

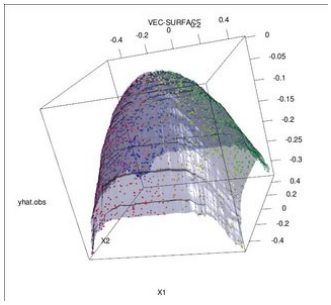
```
uncontaminated observations) \\\ relative to the fit on clean data",xlab="Distance of the
outlier")
```

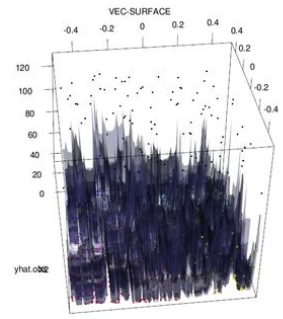


How far? In the example above, the single outlier has changed the fit so much that the mean prediction error (on the uncontaminated) observations is now *1-2 orders of magnitude* bigger than it would have been, had the model been fitted on the uncontaminated data.

So it is not true that a single outlier cannot affect the RF fit.

Furthermore, as I point out elsewhere, outliers are much harder to deal with when there are potentially *several* of them (though they don't need to be a large *proportion* of the data for their effects to show up). Of course, contaminated data can contain more than one outlier; to measure the impact of several outliers on the RF fit, compare the plot on the left obtained from the RF on the uncontaminated data to the plot on the right obtained by arbitrarily shifting 5% of the responses values (the code is below the answer).





Finally, in the regression context, it is important to point out that outliers can stand out from the bulk of the data in both the design and response space (1). In the specific context of RF, design outliers will affect the estimation of the hyper-parameters. However, this second effect is more manifest when the number of dimension is large.

What we observe here is a particular case of a more general result. The extreme sensitivity to outliers of multivariate data fitting methods based on convex loss functions has been rediscovered many times. See (2) for an illustration in the specific context of ML methods.

Edit.

Fortunately, while the base CART/RF algorithm is emphatically not robust to outliers, it is possible (and quiet easy) to modify the procedure to impart it robustness to "y"-outliers. I will now focus on regression RF's (since this is more specifically the object of the OP's question). More precisely, writing the splitting criterion for an arbitrary node t as:

$$s^* = \arg\max_s [p_L \mathrm{var}(t_L(s)) + p_R \mathrm{var}(t_R(s))]$$

where t_L and t_R are emerging child nodes dependent on the choice of s^* (t_L and t_R are implicit functions of s) and p_L denotes the fraction of data that falls to the left child node t_L and $p_R = 1 - p_L$ is the share of data in t_R. Then, one can impart "y"-space robustness to regression trees (and thus RF's) by replacing the variance functional used in the original definition by a robust alternative. This is in essence the approach used in (4) where the variance is replaced by a robust M-estimator of scale.

- (1) Unmasking Multivariate Outliers and Leverage Points. Peter J. Rousseeuw and Bert C. van Zomeren Journal of the American Statistical Association Vol. 85, No. 411 (Sep., 1990), pp. 633-639
- (2) Random classification noise defeats all convex potential boosters. Philip M. Long and Rocco A. Servedio (2008). http://dl.acm.org/citation.cfm?id=1390233
- (3) C. Becker and U. Gather (1999). The Masking Breakdown Point of Multivariate Outlier Identification Rules.
- (4) Galimberti, G., Pillati, M., & Soffritti, G. (2007). Robust regression trees based on M-estimators. Statistica, LXVII, 173–190.


```
library(forestFloor)
library(randomForest)
library(rgl)
set.seed(1)

X<-data.frame(replicate(2,runif(2000)-.5))
y<--sqrt((X[,1])^4+(X[,2])^4)
Col<-fcol(X,1:2) #make colour pallete by x1 and x2
#insert outlier2 and colour it black
y2<-y;Col2<-Col
y2[1:100]<-rnorm(100,200,1);    #outliers
Col2[1:100]="#000000FF" #black

#plot training set
plot3d(X[,1],X[,2],y,col=Col)
rf=randomForest(X,y)    #RF on clean data
rg=randomForest(X,y2)   #RF on contaminated data
vec.plot(rg,X,1:2,col=Col2,grid.lines=200)
mean(abs(rf$predict[-c(1:100)]-y[-c(1:100)]))
    mean(abs(rg$predict[-c(1:100)]-y2[-c(1:100)]))
```

edited Jan 2 at 1:58

answered Dec 20 '15 at 17:09

user603
14.1k 1 37 74

Thanks for your detailed answer. If there are several outliers in the same high dimensional space, it begs the question what is our criteria for calling an "outlier"? In that case, I wonder what hyper parameters may be set so that I can specify some kind of criteria for an outlier a priori? – Hunle Dec 21 '15 at 0:53

Right, so like I'm trying to say, I guess it's a matter of definition of "outlier". How do you build a definition of an "outlier" into the hyperparameters of an RF algorithm? – Hunle Dec 21 '15 at 1:19

1 I have added my earlier comments to my answer. I hope it now does a better job of answering your question! – user603 Dec 21 '15 at 1:46

1 Thanks. What are `p` and `s` in the formula? – Hunle Dec 21 '15 at 2:34

1 Why are combined outliers (1a+2) bad? In your example, the RF model fit the data structure perfectly, 99,99% OOB MSE. The model structure of the middle land between the two clusters is pretty rough, yes, and more a product of the model than of the data. But, no inference and/or predictions should be in this unknown area, so it does not matter. Absolute robustness toward outliers is inevitably to ignore rare but perhaps important possible events. Most ML algos would by default take a middle ground stance between robustness and 'flexibility' but can be tweaked to increase robustness. – Soren Havelund Welling Dec 21 '15 at 11:13

@SorenHavelundWelling: Thanks! fixed the example to show that the outliers can make the mean absolute error arbitrarily large. – user603 Dec 21 '15 at 16:15

When I first pondered this question, I assumed that the way an RF treats outliers would be the same regardless of classification or regression. I'm actually interested in how this would be different if, say, I were using RF for binary classification. – Hunle Dec 24 '15 at 19:42

If the response is binary, the only type of outliers that matter are those on the design space. Outliers on the design are not handled by the robustification strategy outlined in the edit of my answer and I do not know how to modify RF to handle them. – user603 Dec 26 '15 at 12:24

Is there any reason that in your code, @user603, that you eliminate the last value in the array in this line: `rel_prediction_error[i]<-mean(abs(rf$predict[-1]-y2[-1]))/mean(abs(rg$predict[-1]-y[-1]))` . Why do you use `[-1]` to eliminate the last value? – Hunle Jun 7 at 6:55

@Hunle: yes y[1] is the outlier. rel_prediction_error is the vector of prediction errors on the *non outlying observations*. This corresponds to the sentence "For example, if we compute the mean prediction error of the uncontaminated observations" in y text. – user603 Jun 7 at 7:15

This example seems a bit unfair, since the function you use is \-sqrt{x^4+x^4}\ and the outliers are positive. You say you're measuring the "distance" of the outler - distance from what? – Hunle Jun 8 at 7:35

@Hunle: a) why would this example be unfair? Could you elaborate? I do not understand the relevance of your argument ' the function...the outliers are positive'. b) distance to where the original, uncontaminated observation was. We measure the sensitivity of the RF to the presence of a single outlier in the data by showing that the *whole* fitted RF surface (at all values $\boldsymbol{x} \in \mathbb{R}^2$) can be affected at will by just moving one of the original data point ($\boldsymbol{x}_i$) to an arbitrary location in the design space ($\boldsymbol{x}_i'$). The distance is $||\boldsymbol{x}_i' - \boldsymbol{x}_i||$. – user603 Jun 8 at 8:01

It is not the Random Forest algorithm itself that is robust to outliers, but the base learner it is based on: the **decision tree**. Decision trees isolate atypical observations into small leaves (i.e., small subspaces of the original space). Furthermore, decision trees are **local** models. Unlike linear regression, where the same equation holds for the entire space, a very simple model is fitted locally to each subspace (i.e., to each leaf).

- In the case of regression, it is generally a very low-order regression model (usually only the average of the observations in the leaf).
- For classification, it is majority voting.

Therefore, for regression for instance, extreme values do not affect the entire model because they get averaged locally. So the fit to the other values is not affected.

Actually, this desirable property carries over to other tree-like structures, like dendograms. Hierarchical clustering, for instance, has long been used for data cleaning because it automatically isolates aberrant observations into small clusters. See for instance Loureiro et al. (2004). Outlier detection using clustering methods: a data cleaning application.

So, in a nutshell, RF inherits its insensitivity to outliers from **recursive portioning** and **local model fitting**.

Note that decision trees are low bias but high variance models: their structure are prone to changing upon a small modification of the training set (removal or addition of a few observations). But this should not be mistaken with sensitivity to outliers, this is a different matter.

edited Jan 1 at 19:33

answered Jan 1 at 19:27

Antoine
1,454 7 25

I actually considered using a clustering method, as you suggest, for the detection of outliers. But then, I'm unsure where to apply the clustering. Should it be applied to `labeled` or `unlabeled` data? And how would this clustering be achieved on heterogenous data that contains both categorical and numerical features? – Hunle Jan 6 at 6:33

Add Another Answer

A.2.4 Simple check that random forest can fit interactions

2,871 5 17 1 review help

Do random forest variable importance measures take into account the interactions?

Do random forest measures of variable importance (mean change of accuracy, mean change of Gini index) take the interactions into account? I think I know how we come up with the variable importance plot (by permuting each of the predictors), and it doesn't seem that random forest captures the interaction. Does anybody have another point of view? Thanks.

random-forest importance

edited Jul 10 '14 at 4:39
Nick Stauner
8,550 5 24 53

asked Jul 10 '14 at 3:50
peace7
23 3

2 Answers

The variable importance obtained by permutations is computed only by permuting values for a single variable. Thus, it computes some importance measure of the given variable in the context that all other data is fixed. I think it is reasonable to state that the importance measure includes in the measurement also interactions, if such interactions exists. I mean that I see VI as an impure measure, a measure influenced by the main effect of that variable and also interaction with others.

Gini importance is found often to be in concordance with permutation importance, and I see it as a similar measure.

There is however something called interaction which is measured in random forests, and this measures if a split on a given variable increase or decrease splits on other measure. This can be computed for each pair of measures. It looks like a 2 measure interactions. If one want to measure interactions with more than 2 variables than I suppose it is possible extending the given procedure, but soon becomes too computer intensive.

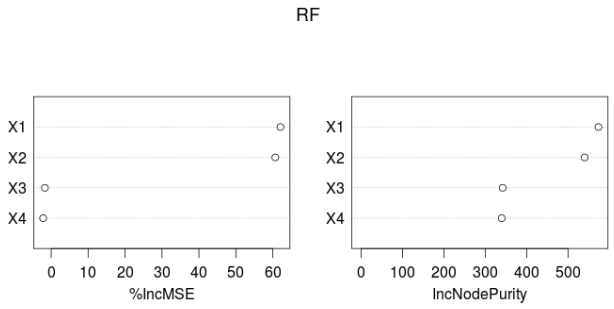
Last thing called *interactions* is not implemented in R package *randomForests* as far as I know. Take a look on the brief description from the Breiman's page on RF here, and check for Interactions section.

answered Jul 10 '14 at 9:18
rapaio
2,926 7 26

Run this code and assert that RF variable importance do incorporate interactions.

```
library(randomForest)
obs=1000
vars =4
X = data.frame(replicate(vars,rnorm(obs)))
ysignal = with(X,sign(X1*X2))
ynoise  = 0.1 * rnorm(obs)
y = ysignal + ynoise
RF = randomForest(X,y,importance=T)
varImpPlot(RF)
```

You should see X1 and X2 are found the important and X3 and X4 are not. y is only explained as the interaction between X1 and X2, alone both variables are useless.



answered Apr 1 '15 at 14:38
Soren Havelund Welling
2,871 5 17

Two thumbs up! Nice work! – rolando2 Apr 1 '15 at 23:45

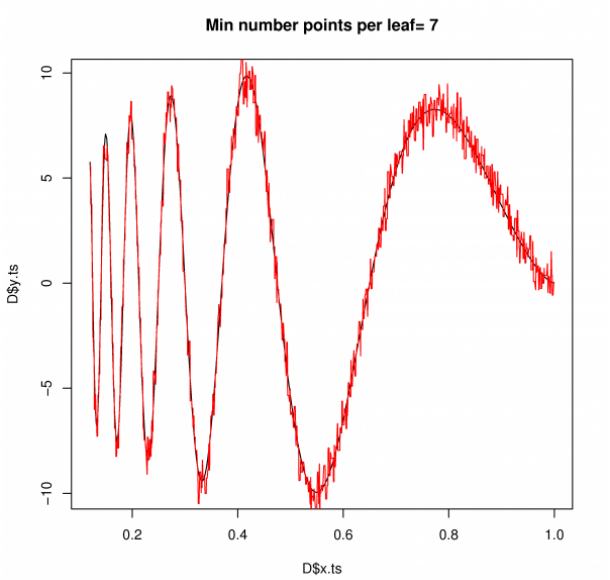
`X = data.frame(replicate(vars,rnorm(obs))) ysignal = with(X,sign(X1*X2))` very tight code +1 – Qbik Apr 26 at 7:11

Add Another Answer

A.2.5 Interpolation with RF and SVM is very similar. Extrapolation is not.

When to use regression trees/forests?

As I was looking for a fine regression algorithm for my problem. I found out one can do that with simple decision trees as well, which is usually used for classification. The output would be something like:



The red *noise* would be the prediction states of such a tree or forest.

Now my question is, why at all to use this method, when there are alternatives, that really try to figure out the underlying equation (such as the famous *support vector machines* SVM). Are there any positive / unique aspects, or was a regression tree more a nice-to-have-algorithm?

regression machine-learning svm random-forest

asked Jan 23 at 11:46

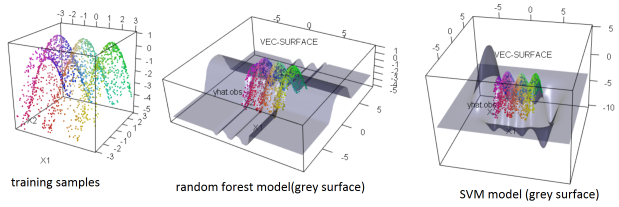
user3085931
108 4

1 Answer

As your figure exemplifies, single decision trees would under perform SVM in most problems. But an ensemble of trees as random forest, is certainly a useful tool to have. Gradient boosting is another great tree derived tool.

SVM and random forest(RF) algorithms are not alike, of course. But both are useful for the same kind of problems and provide similar model structures. Of course the explicitly stated model structures of forest/trees as hierarchical ordered binary step-functions are quite different from SVM regression in a Hilbert space. But if focusing on the actual learned structure of the mapping connecting a feature space with a prediction space the two algorithms produce models with similar predictions. But, when extrapolating outside the feature space region represented with training examples, the *"personality"* of the model takes over and SVM and RF predictions would strongly disagree. See the example below. That's because both SVM and RF are predictive models, your can see that both SVM and RF did a terrible job extrapolating.

$y = sin(x_1\pi) - .5{x_2}^2$



training samples | random forest model(grey surface) | SVM model (grey surface)

So No SVM is not trying to figure out *the underlying true equation* and certainly not anymore than RF. I disagree with your platonist view-point, expecting real life problems to be governed by some algebra/calculus math, that we humans coincidentally happen to teach each other in high school/uni. Yes in some cases, such simple stated equations are fair approximations of the underlying system. We see that in classic physics and accounting... But that does not mean the equation is the true hidden reality. The *"all models are wrong, but some are useful"* would be one statement in a conversation going further from here...

I does not matter if you use SVM, RF or any other appropriate estimator. You can always inspect the model structures and perhaps realize the problem can be described by some simple equation or even develop some theory explaining the observations. It becomes a little tricky in high dimensional spaces, but it is possible.

In general rather consider RF over SVM, when:

- You have more than 1 million samples
- Your features are categorical with many levels(not more than 10 though)
- You would like to distribute the training on several computers
- Simply when a cross-validated test suggests RF works better then SVM for a given problem.

.

```
library(randomForest) #randomForest
library(e1071) #svm
library(forestFloor) #vec.plot and fcol
library(rgl) #plot3d

#generate some data
X = data.frame(replicate(2,(runif(2000)-.5)*6))
y = sin(X[,1]*pi)-.5*X[,2]^2
plot3d(data.frame(X,y),col=fcol(X))

#train a RF model (default params is nearly always, quite OK)
rf=randomForest(X,y)
vec.plot(rf,X,1:2,zoom=3,col=fcol(X))

#train a SVM model (with some resonable params)
sv = svm(X, y,gamma = 1, cost = 50)
vec.plot(sv,X,1:2,zoom=3,col=fcol(X))
```

edited Jan 25 at 0:26

answered Jan 24 at 23:59
Soren Havelund Welling
2,871 5 17

thanks for your precise answer. Concluding (and making it very short) you want to tell, there is no rule for let's say *use this category of algorithms, for these kind of problems etc.*, since their drawbacks are not predictable for every case. Means whatever I'm trying, I have to evalute in the end and if possible, compare with every available algorithm evaluation there is. – user3085931 Jan 26 at 14:22

However I don't get the meaning from one of your points: *You would like to distribute the training on several computers*. In the end I want to have a very accurate prediction. If I can distribute my training, but the result is distinctively worse than my alternatives, why should I prefer then ? – user3085931 Jan 26 at 15:12

1 To answer the first comment, you typically start with simple ML models (linear regression / logistic regression) and see if it works good enough. If not you, try to realize whats missing and try a more complex models or some data fix. With experience you may realize typical pro's and con's pick a good model first time and/or you just learn to brute force running thousands of model candidates including grid search and nested cross-validation to assist model selecting. – Soren Havelund Welling Jan 26 at 15:52

1 The biggest data set so far I used RF on was 16GB. 5000 rows some 500.000 columns. To compute 500 trees on one PC took a day. To compute 3 trees on each of 80 nodes in a cluster took minutes. Each tree can be combined afterwards. You cannot distribute SVM as easily. Lot of times big is not needed, to do good enough. Just dump 90% of samples or features. *"but the result is distinctively worse than my alternatives, why should I prefer then ?"* -If you are a **server nerd**, you go with bigger for the fun :) , otherwise you go with what get you the best results of course. – Soren Havelund Welling Jan 26 at 16:02

1 If you expect your data structure to be some kind of smooth yet complex polynomial surface, and you have limited data points to train on. SVM would probably be quite superior. – Soren Havelund Welling Jan 26 at 16:10

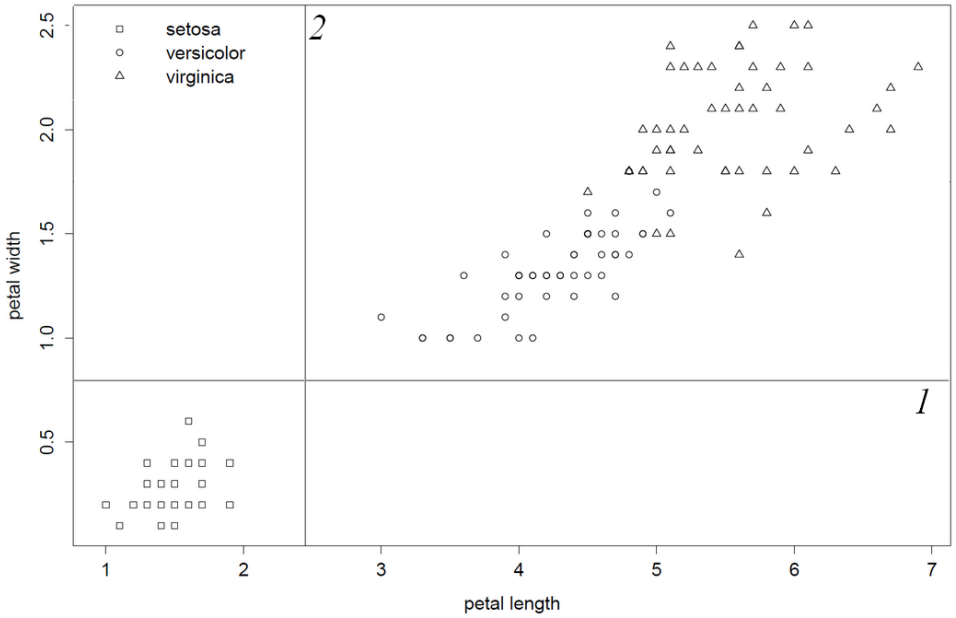
One last question: if my function to learn is e.g. the velocity of an object to track (in image processing), then pretty sure I'm clearly getting a polynominal function - which is more or less what I am looking for in this special case. However, doesn't everything you want to figure out by regression underlie a polynominal structur ? For a better understanding, might you could give a simple example where this wouldn't be the case? – user3085931 Jan 27 at 7:28

Add Another Answer

A.2.6 How does CART break ties

CART: Selection of best predictor for splitting when gains in impurity decrease are equal?

My question deals with **Classification** trees. Consider the following example from the Iris data set:



I want to manually select the best predictor for the first split. According to the CART algorithm, the best feature to make a split is the one that maximizes the decrease in partition impurity, also called Gini gain:

$$GiniGain(N, X) = Gini(N) - \frac{|N_1|}{|N|}Gini(N_1) - \frac{|N_2|}{|N|}Gini(N_1)$$

Where X is a given feature, N is the node on which the split is to be made, and N_1 and N_2 are the two child nodes created by splitting N. $|.|$ is the number of elements in a node.

And $Gini(N) = 1 - \sum_{k=1}^{K} p_k^2$, where K is the number of categories in the node

Now, since making a split based on *petal width* (axis #1) and *petal length* (axis #2) yields the same partition (all Setosa flowers are separated from the non-Setosa), the GinGain scores will be exactly the same for each predictor. **So how does the CART algorithm decide which one is best?**

Intuitively, one can see that splitting on *petal length* (2) is associated with the greatest "margin", hence *petal length* should be chosen (it is actually what happens when implementing `rpart` in R), but nothing in $GiniGain$ measures margin, so the decision must be based on something else.

Related thread but without the answer to my question.

Related thread without any answer.

r | machine-learning | classification | data-mining | cart

edited Aug 13 '15 at 10:23

asked Aug 10 '15 at 21:49

Antoine
1,454 7 25

1 Answer

I confess to be a mediocre c-code interpreter and this old code is not not user-friendly. That said I went through the source code and made these observations which makes me quite sure to say: "rpart literally picks the first and best variable column". As column 1 and 2 produce inferior splits, petal.length will be first split-variable because this column is before petal.width in data.frame/matrix. Lastly, I show this by inverting column order such that petal.with will be first split-variable.

In in the c source file "bsplit.c" in source code for rpart I quote from line 38:

```
 * test out the variables 1 at at time
me->primary = (pSplit) NULL;
for (i = 0; i < rp.nvar; i++) {
```

... thus iterating in a for loop starting from i=1 to rp.nvar a loss function will be called to scan all split by one variable, inside gini.c for "non-categorical split" line 230 the best found split is updated if a new split is better. (This could also be a user defined loss-function)

```
if (temp < best) {
        best = temp;
        where = i;
        direction = lmean < rmean ? LEFT : RIGHT;
}
```

and last line 323, the improvement for best split by a variable is calculated...

```
*improve = total_ss - best
```

...back in bsplit.c the improvement is checked if larger than what previously seen, and only updated if larger.

```
if (improve > rp.iscale)
rp.iscale = improve;        /* largest seen so far */
```

My impression on this is that the first and best (of possible ties will be chosen), because only if new break point have a better score it will be saved. This concerns both the first best break point found and the first best variable found. Break points seems not to be scanned simply left to right in gini.c so the first found tying break point may be tricky to predict. But variables are very predictable scanned from first column to last column.

This behavior is different from the randomForest implementation where in classTree.c the following solution is used:

```
/* Break ties at random: */
if (crit == critmax) {
    if (unif_rand() < 1.0 / ntie) {
        *bestSplit = j;
        critmax = crit;
        *splitVar = mvar;
    }
    ntie++;
}
```

very lastly I confirm this behaviour by flipping the columns of iris, such that petal.width is chosen first

```
library(rpart)
data(iris)
iris = iris[,5:1]  #flip/flop", invert order of columns columns
obj = rpart(Species~.,data=iris)
print(obj) #now petal width is first split


1) root 150 100 setosa (0.33333333 0.33333333 0.33333333)
  2) Petal.Width< 0.8 50   0 setosa (1.00000000 0.00000000 0.00000000) *
  3) Petal.Width>=0.8 100  50 versicolor (0.00000000 0.50000000 0.50000000)
    6) Petal.Width< 1.75 54   5 versicolor (0.00000000 0.90740741 0.09259259) *
    7) Petal.Width>=1.75 46   1 virginica (0.00000000 0.02173913 0.97826087) *
```

...and flip back again

```
iris = iris[,5:1]  #flop/flip", revert order of columns columns
obj = rpart(Species~.,data=iris)
print(obj) #now petal length is first split
1) root 150 100 setosa (0.33333333 0.33333333 0.33333333)
  2) Petal.Length< 2.45 50   0 setosa (1.00000000 0.00000000 0.00000000) *
  3) Petal.Length>=2.45 100  50 versicolor (0.00000000 0.50000000 0.50000000)
    6) Petal.Width< 1.75 54   5 versicolor (0.00000000 0.90740741 0.09259259) *
    7) Petal.Width>=1.75 46   1 virginica (0.00000000 0.02173913 0.97826087) *
```

edited Aug 13 '15 at 0:30

answered Aug 13 '15 at 0:10

Soren Havelund Welling

2,871 5 17

Thank you very much for your answer. So, when more than one predictor corresponds to the optimal split, the first one is selected, this has nothing to do with margins whatsoever. Kind of makes sense after all. – Antoine Aug 13 '15 at 9:15

It was fun to figure out :) I guess margins are not natively implemented in many tree models as binary splits are natively non-parametric – Soren Havelund Welling Aug 13 '15 at 9:51

it might be helpful to mention that the source code for rpart can also be obtained from the R console via `untar(download.packages(pkgs = "rpart",destdir = ".",type = "source")[,2])` , and then opening the `src` folder in the current working directory (from this SO thread). Then the code for one particular function can be viewed with Notepad++. – Antoine Aug 13 '15 at 9:58

And the algorithm stops when splitting does not lead to any improvement anymore for all nodes, right? – Antoine Aug 13 '15 at 10:11

yep. in partition.c line 80 isch: " This is rather rare -- but I couldn't find a split worth doing" ...said the impersonated recursive function. Hereafter the node is sealed off and the recursive algorithm return to previous node be calling return(0). – Soren Havelund Welling Aug 13 '15 at 11:06

Awesome. Thanks for your confirmation. When all nodes are sealed off, the tree has reached maximum size. – Antoine Aug 13 '15 at 11:52

yup - btw copied your formula to answer this related question: stats.stackexchange.com/questions/144818/… – Soren Havelund Welling Aug 14 '15 at 18:40

Add Another Answer

A.2.7 Variable importance for other models than random forest

Is Random Forest the only algorithm to measure the importance of input variables …?

I have three time series say (Stock price open, Stock price high, Stock price low) and one output (Stock price close) and I need to know which of the 3 inputs has a greater effect on my output. R's Random Forests' (importance) ' %IncMSE yields the importance for this case.

Are there any other algorithms apart from Random Forest, to measure the importance of a input variable?

r | machine-learning | random-forest | predictor

edited Feb 17 at 14:40
General Abrial
15.6k 4 35 74

asked Feb 17 at 11:31
Sudharsan
27 5

Gradient Boosting Machines with the "gbm" R package is other algorithm with similar performance than RF, and showing also the importance of the predictors, but the ranking should be very similar – Jesus Herranz Valera Feb 17 at 11:44

Maybe `Boruta algorithm` will help you: cran.r-project.org/web/packages/Boruta/Boruta.pdf ? – Maju116 Feb 17 at 11:50

1 Ordinary regression coefficients will tell you "importance" in a specific sense. What kind of importance are you interested in? – General Abrial Feb 17 at 14:39

2 @user777 Actually and unless the predictors have been standardized (never a good practice), your statement is not true since regression coefficients are expressed in the unit of the predictor. As such they are *not* scale invariant and sensitive to variations in the moments. A better, quick and dirty heuristic is to rank the standardized metrics such as F-statistics, t-values and/or chi-squares associated with the parameters, as appropriate. For the "state-of-the-art" in deriving relative importance, see Ulrike Groemping's papers on 'relaimpo'...a multivariate approach to relative rankings – DJohnson Feb 17 at 14:52

@Maju116 I just now saw Random uniform forest package in R . I think it does the best with this use case of finding the input variable affecting the response variable . Also if the input is categorical, say region(A,B,C) it also shows the importance of each region . – Sudharsan Apr 26 at 11:32

1 Answer

Any supervised regression/classification model, that I can think of, could be bootstrap aggregated (bagged) and therefore variable importance could be computed. It would just be a little slow to train e.g. svm 50 times compared to growing 50 trees.

> *"I need to know which of the 3 inputs has a greater effect on my output"*

I would abstain from causal interpretation of importance and at most see it as a source of inspiration. Importance only describe usefulness of features to predict, given all other features and one specific model. Two highly redundant features will roughly share the same fixed amount of variable importance. Two features can be complimentary and have a higher variable importance, than if one of them were never included in the training. This happens if an interaction between two features is useful to predict the output.

So importance is not an universal metric, and the answer will depend on your model and and the included features. You may want to ask instead:

"Which overall relationship between input and output has e.g. an RF/SVM/NN model captured?" -You could use some fancy plots exploring the high dimensional model structure. But I can reveal that the effective RF, SVM or NN model will be something very boring like $close_t = (open_t + high_t + low_t)/3$

"Is this relationship trivial or inspiring?" - In this case, quite trivial, as the future absolute price is highly dependent on the past price. If some asset were 10€ yesterday, its probably gonna be priced 9€ or 12€ today, not 1 cent or 5000€.

Try use rolling window to predict the change of price, that is in contrary more challenging. If you were to succeed better than others, then the effective structure of a well performing empirical model could be very inspiring to form new hypotheses.

answered Feb 19 at 16:41
Soren Havelund Welling
2,871 5 17

Add Another Answer

A.2.8 How to combine multiple models in a bootstrap aggregated ensemble

Use of a bagging model or feature engineering?

As a pet project, I have been learning some data analysis and machine learning skills (mainly text analytics) with the Analytics Edge course on edX. I decided to put some of my new skills at use analysing a dataset from UCI Machine Learning: https://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection

I did some analysis already (can be seen at https://github.com/Khaltar/Portfolio/blob/master/R/Machine%20Learning/SMS.R) and computed a LogRegression Model, a RF Model and a CART Model. The Random Forest Model seems to be getting the best results regarding AUC and accuracy but I'm still not happy with it.

A friend of mine suggested using a bagging approach joining the three models and using some kind of "voting" system to classify predictions and achieve better results but I am completely at a loss on how to do that. My doubt is how to actually implement a bootstrapping model in R using RF to raise accuracy of the model. I tried using bagRboostR package (sample code in my sms.R file in github) but I can't figure out how to use it or if there is a simpler solution to implement it.

Thanks in advance

r | machine-learning | text-mining | bagging

edited Jul 9 '15 at 18:17

asked Jul 9 '15 at 16:36

Sarcus
11 2

This seems like a good question, but you'll need to be more specific than "Could I get some help..." in order to get a viable answer here. Can you focus this more & provide a concrete problem? Bear in mind that you can ask a series of questions w/ each springing from the previous as you get clearer on the issues. – gung Jul 9 '15 at 17:14

1 Answer

Bagging a RF model do not normally improve prediction performance(AUC) as RF already is bagged. If it does, probably some parameters in RF training are set suboptimal. So the easy answer is: don't bag the randomForest algorithm. Also bagging RF could be computationally slow.

Bagging CART is a good idea. Infact so good, that some guys did just that plus some extra features and called it e.g. "random forests"! Bagging CART models is highly related to RF, but still inferior as mtry is set to the number of variables. This will lead to a little less decorrelated trees and lower robustness.

Bagging glm is a good idea for sparse and/or noisy datasets, as it will reduce overfitting. If you have a rich dataset 1000 samples and 10 features, don't bother as the unbiased glm will be very stable. Regularization could also be achieved more elegantly with the ridge-regression, elasticnet, lasso, PLS etc.. Check out glmnet package.

If two classifiers would to be directly combined, the voting is either unanimously or equal. This is of course a problem. Instead the classifiers can each be bagged(50 times e.g.) and all votes could be combined. This works fine, unless the dataset is unbalanced.

Many classifiers actual calculate some kind of probability measure and make decision here upon. This is true for logistic regression. For random forest the voting ratio can be understood as a pseudo probability. Remember to either stratify(down sampling) or use a lower weighting for the over-represented class. This will likely give a better model, see this answer and this answer on how to implement. If model do not provide any probability measure (regular SVM) it would be needed to bag the model to achieve a better fusing of models. In case of SVM it might smart to lower regularization to achieve more differentiated votes. As if the model is completely consistent, bagging will only provide the same result many times.

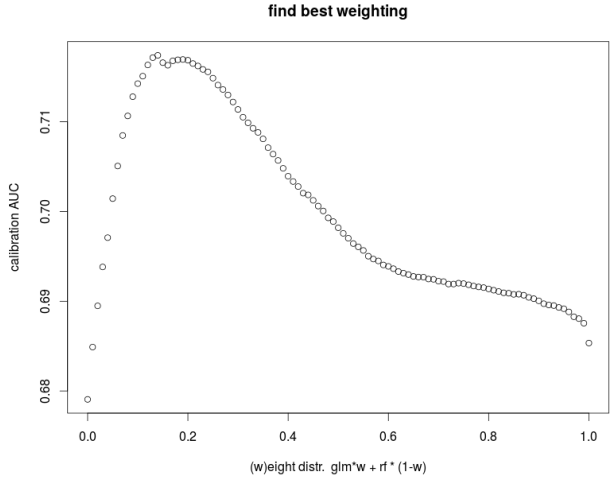
If one classifier is strongly superior, combining models is not likely to improve prediction performance. I tried to design a data-set of a linear component distributed on all variables (regression is good at this) and a non-linear component(RF can fit this) such that combining models is expected to improve performance. In example the models must identify if the sample is "spam" but any real text-analysis is left out. Because I could not make your github script work at first try.

I use a training-set to train, a calibration-set to decide weighting and a large test-set to measure performance. For real data, I would replace calibration and test set with nested n-fold cross-validation. But the example is already 100 lines, so I skip this for now.

Because you asked for bagging, I regularized glm by bagging and not by any other method. Otherwise I would have used elastic-net and RF and combined the probabilistic predictions directly.

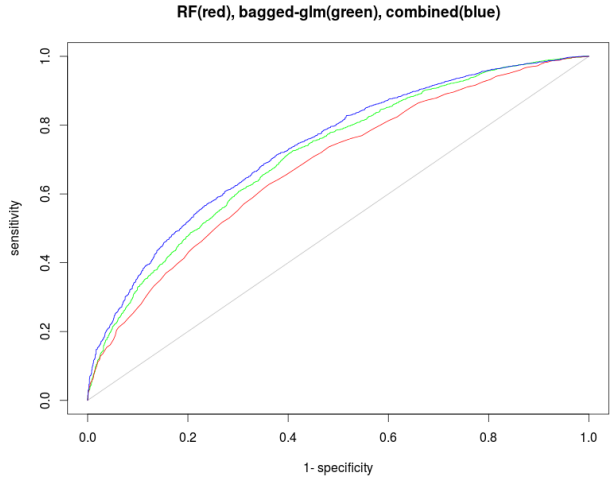
I have tried to write all bagging function so general that you easily can use them for your project even with completely different models. There is a makeSimData, trainBag, predictBag and a combine function for glm. I train a RF model and bagged glm. I use the probabilistic voting of either models and find best weighting with calibration-set. Other parameters could be tuned also. Again, model weights and model parameters could also be found directly by cross

validation on training set.



To speed things up a little bagging can easily be parallelized. But this implementation will not work in windows, if running in windows set parallel=FALSE.

Lastly I post roc-AUC on test set. Of either RF and bagged-glm alone or in combination with the best found linear weighting rule.



and lastly the code:

```
rm(list=ls())
library(parallel) #for linux&mac only
#for windows use parallel=F below
library(randomForest)
library(AUC)

std = function(x) x/sd(x) #scaling function
makeSimData = function(
  N = 500,
  var = 250,
  linW = .7,
  noiseW = 1,
  spamRatio = .5,
  hidden.function = function(X) {
    y.linear = apply(X,1,sum) #something glm is best at
    y.quadratic = X[,1]^2     #something glm cannot fit
    y.value = std(y.linear) * linW +
              std(y.quadratic) * (1-linW) +
              rnorm(N) * noiseW * 2 #noise component
    y.class = (y.value<=quantile(y.value,spamRatio))+1
  }
) {
```

```
  X = replicate(var,rnorm(N))
  y = c("regular","spam")[hidden.function(X)]
  Data = data.frame(y=y,X=X)
}

trainBag = function(
  Data,  #defined data structure as makeSimData
  model_func,  #defined model function
  trainPars,      #what we
  reps=11,
  sampsizeRatio= 1, #
  parallel=TRUE) {

  if(parallel) nCore=detectCores() else nCore=1
  Nsamples = dim(Data)[1]
  sampsize = ceiling(sampsizeRatio * Nsamples)
  preds =  mclapply(1:reps,function(r) {
    bootstrapData = Data[sample(Nsamples,sampsize,replace=T),]
    do.call(model_func,trainPars)
  },mc.cores=nCore)
}

predictBag = function(
  myBag,
  newData,
  predPars,
  combine = c,
  parallel=TRUE
) {
  if(parallel) nCore=detectCores() else nCore=1
  preds = mclapply(myBag, function(model.fit) do.call(predict,predPars),
                   mc.cores=nCore)
  out = combine(preds)
}

#make data
trainData = makeSimData(500,spamRatio=.5,noiseW=.2)
calibData = makeSimData(500,spamRatio=.5,noiseW=.2) #could also have used 10fold-CV
testData  = makeSimData(5000,spamRatio=.5,noiseW=.2)


#train glm.bag and rf
glm.trainPars = alist(formula=y~., #define how glm should be run
                data = bootstrapData,
                family = "binomial")

glm.fit = do.call(glm,alist(formula=y~.,data=trainData,family="binomial"))
glm.bag = trainBag(trainData,glm,glm.trainPars)
rf.fit = randomForest(y~.,trainData)

#predict bag and rf
glm.predPars = alist(object = model.fit,
                     newdata=newData,
                     type="response")

#combine predicted probs, #could also be true/false ratio but that would stupid
glm_combine = function(bagPred) {
  aMatrix = do.call(cbind,bagPred)
  out = apply(aMatrix,1,mean) #combine by probabilistic average
  #out = apply(aMatrix,1,x>0.5) #alternative combine by booleen voting
  attributes(out) = NULL
  return(out)
}

glmPred.calib = predictBag(glm.bag,
                     calibData,
                     glm.predPars,
                     glm_combine,
                     parallel=T)
rfPred.calib = predict(rf.fit,calibData,type="prob")[,2]

#tune weighting with calibration set (could also be some nFold-CV more lines...)
getBestWeight = function(Data,pred1,pred2) {
  with(Data,{
    weights = seq(0,1,le=101)
  AUCbyWeight = sapply(weights, function(weight) {
    auc( roc(pred1 *    weight +
             pred2 * (1-weight)
            ,y))
  })
  plot(
    x=weights,
    y=AUCbyWeight,
    xlab = "(w)eight distr.  glm*w + rf * (1-w)",
    ylab = "calibration AUC",
    main = "find best weighting",
  )

  #return best weight
  weights[which(AUCbyWeight==max(AUCbyWeight))[1]]

  })
}

#find best weight
bestWeight = getBestWeight(calibData,glmPred.calib,rfPred.calib)

#combine predictions with weighting, e.g. the best weighting
weightedPredict = function(Data,w,pred1,pred2,...) {
  with(Data,{

    weightedPred = pred1 * w + pred2 * (1-w)
    plot(roc(pred1,y),add=F,col="green",...)
    plot(roc(pred2,y),add=T,col="red")
```

```
    plot(roc(weightedPred,y),add=T,col="blue")
    return(weightedPred)
  })
}

glmPred.test  = predictBag(glm.bag,
                           testData,
                           glm.predPars,
                           glm_combine,
                           parallel=T)
rfPred.test = predict(rf.fit,testData,type="prob")[,2]

#apply
weightedPred.test = weightedPredict(testData,bestWeight,glmPred.test,rfPred.test,
               main="RF(red), bagged-glm(green), combined(blue)")
```

edited Jul 20 '15 at 21:11

answered Jul 19 '15 at 23:07

Soren Havelund Welling
2,871 5 17

Add Another Answer

A.2.9 Bootstrapping process of random forest: Sampling proability as function of hyperparameters.

number of trees that were built without minority class?

Lets assume that my random forest has 500 trees. My data is imbalance with 90% of class A and 10% of class B. I am wonder if there is any way to calculate roughly the number of trees that are built with only samples from class A.

Thanks

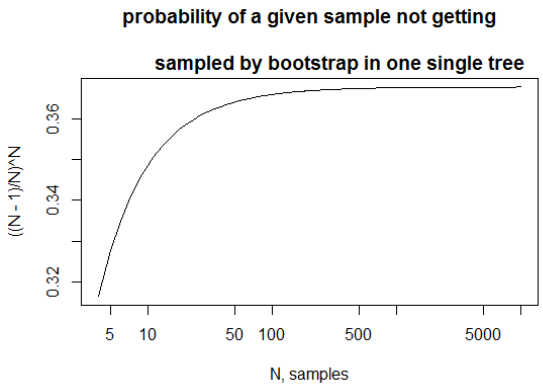
random-forest cart

asked Feb 4 at 20:31
rudky martin
16 2

1 Answer

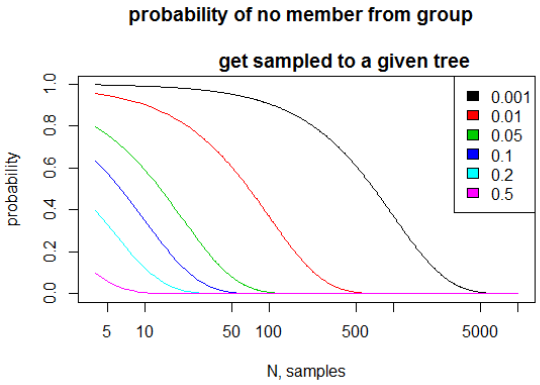
You can also have the exact answer. Besides sizes of groups it depends on size of training set (*N*). First I compute the probability that any given sample *x* is not selected for a tree. For a given tree *N* samples are drawn and the probability *(one draw not x)* $= \frac{N-1}{N}$ and the probability of *(all draws not x) = (one draw not x)* ^{N}.

```
N=unique(ceiling(10^((seq(.5,4,len=500)))))
plot(N,((N-1)/N)^N,log="x",type="l",xlab="N, samples",
     main="probability of a given sample not getting\n
           sampled by bootstrap in one single tree")
PnotSample = (((N-1)/N)^N)
```



Next I define 6 scenarios where the small group *X* make out 0.1%, 1%, 5%, 10%, 20% and 50% for any number of training set size. *(all draws not group X) = (all draws not x)* ^{NgroupRatio}

```
#probability of group not getting sampled to a tree
groupRatio=c(0.001,.01,.05,.1,.2,.5)
for(i in 1:length(groupRatio)) {
  PnotGroup = PnotSample^(N*groupRatio[i])
  if(i==1) {
    plot(N,PnotGroup,log="x",type="l",col=i,
         xlab="N, samples",ylab="probability",
         main="probability of no member from group  \n
               get sampled to a given tree")
  } else {
    points(N,PnotGroup,type="l",col=i)
  }
}
legend("topright",legend=groupRatio,fill=1:length(groupRatio))
```



So in your case (blue line), not selecting any sample from minor group would be quite rare if you have more than 50 samples in your training set. The expected number of trees not having any from minor group is simply the probability for a single tree multiplied with number of trees in forest. Anyhow, there could be a number of good reasons to modify these odds, you can read about why and how in this answer.

edited Feb 7 at 14:47

answered Feb 7 at 14:34

Soren Havelund Welling
2,871 5 17

Add Another Answer

A.2.10 Simple tutorial on log transformation before PCA

2,921 6 19 review help

Why log-transforming the data before performing principal component analysis?

Im following a tutorial here: http://www.r-bloggers.com/computing-and-visualizing-pca-in-r/ to gain a better understanding of PCA.

The tutorial uses the Iris dataset and applies a log transform prior to PCA:

> Notice that in the following code we apply a log transformation to the continuous variables as suggested by [1] and set `center` and `scale` equal to `TRUE` in the call to `prcomp` to standardize the variables prior to the application of PCA.

Could somebody explain to me in plain English why you first use the log function on the the first four columns of the Iris dataset. I understand it has something to do with making data relative but am confused what's exactly the function of log, center and scale.

r pca

edited Aug 2 '15 at 22:31 amoeba 24.8k 5 88 148

asked Aug 2 '15 at 16:05 Marc van der Peet 53 3

2 Answers

The iris data set is a fine example to learn PCA. That said, the first four columns describing length and width of sepals and petals are not an example of strongly skewed data. Therefore log-transforming the data does not change the results much, since the resulting rotation of the principal components is quite unchanged by log-transformation.

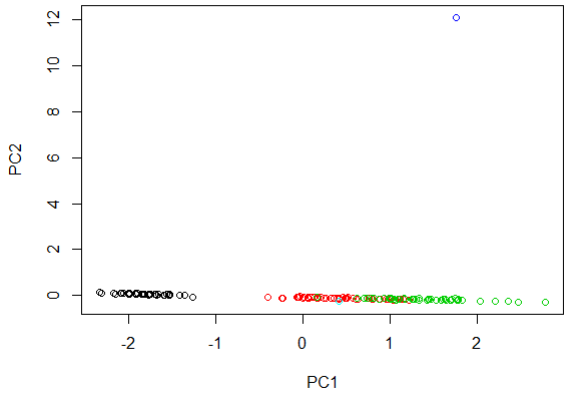
In other situations log-transformation is a good choice.

We perform PCA to get insight of the general structure of a data set. We center, scale and sometimes log-transform to filter off some trivial effects, which could dominate our PCA. The algorithm of a PCA will in turn find the rotation of each PC to minimize the squared residuals, namely the sum of squared perpendicular distances from any sample to the PCs. Large values tend to have high leverage.

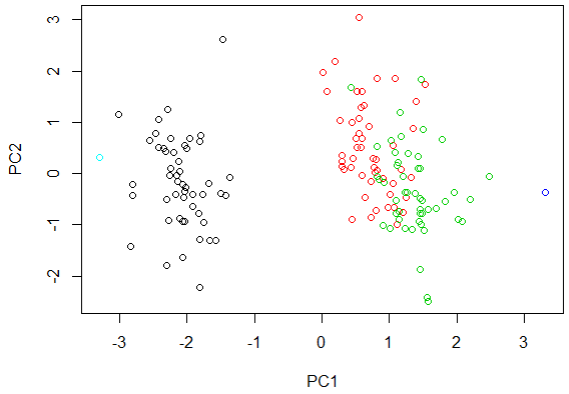
Imagine injecting two new samples into the iris data. A flower with 430 cm petal length and one with petal length of 0.0043 cm. Both flowers are very abnormal being 100 times larger and 1000 times smaller respectively than average examples. The leverage of the first flower is huge, such that the first PCs mostly will describe the differences between the large flower and any other flower. Clustering of species is not possible due to that one outlier. If the data are log-transformed, the absolute value now describes the relative variation. Now the small flower is the most abnormal one. Nonetheless it is possible to both contain all samples in one image and provide a fair clustering of the species. Check out this example:

```
data(iris) #get data
#add two new observations from two new species to iris data
levels(iris[,5]) = c(levels(iris[,5]),"setosa_gigantica","virginica_brevis")
iris[151,] = list(6,3,  430  ,1.5,"setosa_gigantica") # a big flower
iris[152,] = list(6,3,.0043,1.5  ,"virginica_brevis") # a small flower

#Plotting scores of PC1 and PC" without log transformation
plot(prcomp(iris[,-5],cen=T,sca=T)$x[,1:2],col=iris$Spec)
```




```
#Plotting scores of PC1 and PC2 with log transformation
plot(prcomp(log(iris[,-5]),cen=T,sca=T)$x[,1:2],col=iris$Spec)
```



edited 39 mins ago

answered Aug 2 '15 at 17:25

Soren Havelund Welling

2,921 6 19

2 Nice demo and plots. – ssdecontrol Aug 3 '15 at 0:55

Well, the other answer gives an example, when the log-transform is used to reduce the influence of extreme values or outliers.
Another general argument occurs, when you try to analyze data which are *multiplicatively* composed instead of *addititively* - PCA and FA model by their math such additive compositions. *Multiplicative* compositions occur in the most simple case in physical data like the surface and the volume of bodies (functionally) dependent on (for instance) the three parameters lenght, width, depth. One can reproduce the compositions of an historic example of the early PCA, I think it is called "Thurstone's Ball- (or 'Cubes'-) problem" or the like. Once I had played with the data of that example and had found that the log-transformed data gave a much nicer and clearer model for the composition of the measured volume and surface data

with the three one-dimensional measures.

Besides of such simple examples, if we consider in social research data *interactions* , then we ususally think them as well as multiplicatively composed measurements of more elementary items. So if we look specifically at interactions, a log-transform might be a special helpful tool to get a mathematical model for the de-composition.

answered Jul 2 at 8:48

Gottfried Helms
793 6 13

Add Another Answer

A.2.11 Efficient implementation gini loss function in random forest

Does Breiman's random forest use information gain or Gini index?

I would like to know if Breiman's random forest (random forest in R randomForest package) uses as a splitting criterion (criterion for attribute selection) information gain or Gini index? I tried to find it out on http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm and in documentation for randomForest package in R. But the only thing I found is that Gini index can be used for variable importance computing.

r | random-forest | entropy | gini

edited Apr 4 '15 at 16:36 Nick Cox 27.6k 3 54 82

asked Apr 4 '15 at 16:17 somebody 41 2

I also wonder if trees of random forest in randomForest package are binary or not. – somebody Apr 4 '15 at 16:51

1 Answer

The randomForest package in R by A. Liaw is a port of the original code being a mix of c-code(translated) some remaining fortran code and R wrapper code. To decide the overall best split across break points and across mtry variables, the code uses a scoring function similar to gini-gain:

$$GiniGain(N,X) = Gini(N) - \frac{|N_1|}{|N|}Gini(N_1) - \frac{|N_2|}{|N|}Gini(N_2)$$

Where X is a given feature, N is the node on which the split is to be made, and N_1 and N_2 are the two child nodes created by splitting N. $|.|$ is the number of elements in a node.

And $Gini(N) = 1 - \sum_{k=1}^{K} p_k^2$, where K is the number of categories in the node

But the applied scoring function is not the exactly same, but instead a equivalent more computational efficient version. $Gini(N)$ and |N| are constant for all compared splits and thus omitted.

Also lets inspect the part if the sum of squared prevalence in a node(1) is computed as
$\frac{|N_2|}{|N|}Gini(N_2) \propto |N_2|Gini(N_2) = |N_2|(1 - \sum_{k=1}^{K} p_k^2) = |N_2| \sum \frac{nclass_{2,k}^2}{|N_2|^2}$

where $nclass_{1,k}$ is the class count of target-class k in daughter node 1. Notice $|N_2|$ is placed both in nominator and denominator.

removing the trivial constant $1-$ from equation such that best split decision is to maximize nodes size weighted sum of squared class prevalence...

score= $|N_1| \sum_{k=1}^{K} p_{1,k}^2 + |N_2| \sum_{k=1}^{K} p_{2,k}^2 = |N_1| \sum_{k=1}^{K} \frac{nclass_{1,k}^2}{|N_1|^2} + |N_2| \sum_{k=1}^{K} \frac{nclass_{2,k}^2}{|N_2|^2}$
$= \sum_{k=1}^{K} \frac{nclass_{2,k}^2}{1}|N_1|^{-1} + \sum_{k=1}^{K} \frac{nclass_{2,k}^2}{1}|N_1|^{-2}$
$= nominator_1/denominator_1 + nominator_2/denominator_2$

The implementation also allows for classwise up/down weighting of samples. Also very important when the implementation update this modified gini-gain, moving a single sample from one node to the other is very efficient. The sample can be substracted from nominators/denominators of one node and added to the others. I wrote a prototype-RF some months ago, ignorantly recomputing from scratch gini-gain for every break-point and that was slower :)

If several splits scores are best, a random winner is picked.

This answer was based on inspecting source file ***"randomForest.x.x.tar.gz/src/classTree.c"*** line 209-250

edited Aug 14 '15 at 18:33

answered Aug 14 '15 at 14:00 Soren Havelund Welling 2,921 6 19

Add Another Answer

A.2.12 Limiting bootstrap sample size versus limiting maxnodes

random forest tuning - tree depth and number of trees

I have basic question about tuning a random forest classifier. Is there any relation between the number of trees and the tree depth? Is it necessary that the tree depth should be smaller than the number of trees?

random-forest

asked Jan 25 at 16:11

Vysh

52 7

Are you sure you have the right forum? Should this maybe be in "The Great Outdoors" instead, or is this really a programming question? – B. Clay Shannon Jan 25 at 16:15

2 @B.ClayShannon Random forests is a machine learning method. His question totally belongs here. – Tim Biegeleisen Jan 25 at 16:16

1 I have never heard of a rule of thumb ratio between the number of trees and tree depth. Generally you want as many trees as will improve your model. The depth of the tree should be enough to split each node to your desired number of observations. – Tim Biegeleisen Jan 25 at 16:20

@TimBiegeleisen here's my thumb rule :) – Soren Havelund Welling Jan 26 at 12:06

2 Answers

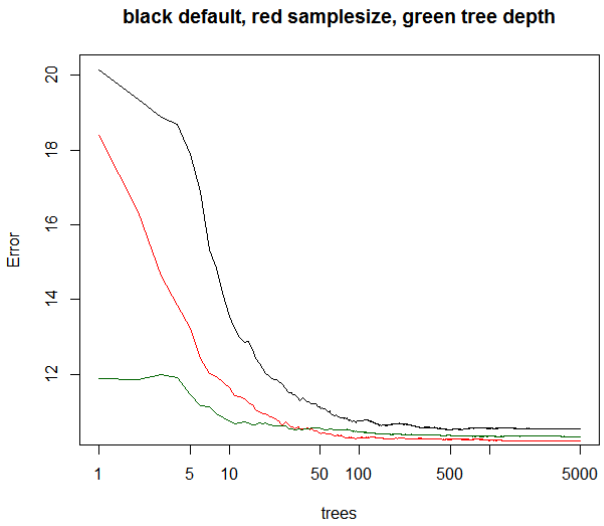
For most practical concerns, I agree with Tim.

Yet, other parameters do affect when the ensemble error converges as a function of added trees. I guess limiting the tree depth typically would make the ensemble converge a little earlier. I would rarely fiddle with tree depth, as though computing time is lowered, it does not give any other bonus. Lowering bootstrap sample size both gives lower run time and lower tree correlation, thus often a better model performance at comparable run-time. A not so mentioned trick: When RF model explained variance is lower than 40%(seemingly noisy data), one can lower samplesize to ~10-50% and increase trees to e.g. 5000(usually unnecessary many). The ensemble error will converge later as a function of trees. But, due to lower tree correlation, the model becomes more robust and will reach a lower OOB error level converge plateau.

You see below samplesize gives the best long run convergence, whereas maxnodes starts from a lower point but converges less. For this noisy data, limiting maxnodes still better than default RF. For low noise data, the decrease in variance by lowering maxnodes or sample size does not make the increase in bias due to lack-of-fit.

For many practical situations, you would simply give up, if you only could explain 10% of variance. Thus is default RF typically fine. If your a quant, who can bet on hundreds or thousands of positions, 5-10% explained variance is awesome.

the green curve is maxnodes which kinda tree depth but not exactly.




```
library(randomForest)

X = data.frame(replicate(6,(runif(1000)-.5)*3))
ySignal = with(X, X1^2 + sin(X2) + X3 + X4)
yNoise = rnorm(1000,sd=sd(ySignal)*2)
y = ySignal + yNoise
plot(y,ySignal,main=paste("cor=",),cor(ySignal,y))

#std RF
rf1 = randomForest(X,y,ntree=5000)
print(rf1)
plot(rf1,log="x",main="black default, red samplesize, green tree depth")

#reduced sample size
rf2 = randomForest(X,y,sampsize=.1*length(y),ntree=5000)
print(rf2)
points(1:5000,rf2$mse,col="red",type="l")

#limiting tree depth (not exact )
rf3 = randomForest(X,y,maxnodes=24,ntree=5000)
print(rf2)
points(1:5000,rf3$mse,col="darkgreen",type="l")
```

edited Jan 26 at 12:00

answered Jan 26 at 10:44

Soren Havelund Welling
621 2 9

Thank you so much for the explanation. I could understand to some extent what you mean, however, since I am still getting used to this whole concept of developing random forest models, I have a few more questions based on your answer. What exactly is the tree correlation and how do you measure it? Is the OOB estimate and ensemble error the same things? Since these could be very basic, you could let me know if there is an article if I can read up to understand the terms better.Thanks a lot! – Vysh Jan 30 at 3:09

It is true that generally more trees will result in better accuracy. However, more trees also mean more computational cost and after a certain number of trees, the improvement is negligible. An article from Oshiro et al. (2012) pointed out that, based on their test with 29 data sets, after 128 of trees there is no significant improvement(which is inline with the graph from Soren).

Regarding the tree depth, standard random forest algorithm grow the full decision tree without pruning. A single decision tree do need pruning in order to overcome over-fitting issue. However, in random forest, this issue is eliminated by random selecting the variables and the OOB action.

Reference: Oshiro, T.M., Perez, P.S. and Baranauskas, J.A., 2012, July. How many trees in a random forest?. In MLDM (pp. 154-168).

answered Jan 28 at 23:24

Sharp Yan

35 7

Add Another Answer

A.3 Manual: R CRAN package forestFloor 1.9.5

Package 'forestFloor'

June 1, 2016

Type Package

Title Visualizes Random Forests with Feature Contributions

Version 1.9.5

Date 2016-06-01

Author Soeren Havelund Welling

Maintainer Soeren Havelund Welling <SOWE@DTU.DK>

Depends

Suggests randomForest, utils, devtools, tools

Description Form visualizations of high dimensional mapping structures of random forests and feature contributions.

SystemRequirements OpenGL, GLU Library, zlib

License GPL-2

URL http://forestFloor.dk

Imports Rcpp (>= 0.11.3), rgl, kknn

LinkingTo Rcpp

NeedsCompilation yes

Repository CRAN

Date/Publication 2016-06-01 14:11:11

R topics documented:

`forestFloor-package` *forestFloor: visualize the random forest model structure*

Description

forestFloor visualizes randomForests models(RF). Package enables users to understand a non-linear, regression problem or a binary classification problem through RF. Any model can be separated into a series of main effect and interactions with the concept of feature contributions.

Details

Package:	forestFloor
Type:	Package
Version:	1.9
Date:	2015-12-25
License:	GPL-2

Author(s)

Soren Havelund Welling

References

Interpretation of QSAR Models Based on Random Forest Methods, http://dx.doi.org/10.1002/minf.201000173
Interpreting random forest classification models using a feature contribution method, http://arxiv.org/abs/1312.1121

`append.overwrite.alists`
Combine two argument lists

Description

First argument list is master, second list slave

Usage

```
append.overwrite.alists(masterArgs,slaveArgs)
```

Arguments

masterArgs	List of arguments, of which will stay unchanged
slaveArgs	List of arguments, conflicts with masterArgs will be deleted. Additional args will be appended.
	s

Details

This function combines to lists of arguments. Conflicts will be resolved by masterArgs.

Value

List of arguments, being masterArgs appended by slaveArgs

Author(s)

Soren Havelund Welling

Examples

```
arglist1 = alist(monkey="happy",telephone.no=53)
arglist2 = alist(monkey="sad",house.no=12)

#this should yield a alist(monkey="happy", telephone.no=53, house.no=12)
forestFloor:::append.overwrite.alists(arglist1,arglist2)
```

`as.numeric.factor` *Convert a factor to numeric.vector.*

Description

Internal function which will drop unused levels and convert remaining to a number from 1 to n.levels.

Usage

```
as.numeric.factor(x,drop.levels=TRUE)
```

Arguments

x	Normally a factor, can be a numeric vector(will be output unchanged)
drop.levels	Boolean, should unused levels be dropped?

Details

Simple internal function, used to direct categorical variables to a 1 dimensional scale.

Value

A vector of same length, where each category/level is replaced with number from 1 to n

Author(s)

Soren Havelund Welling

Examples

```
as.numeric.factor = forestFloor:::as.numeric.factor #import to environment
some.factor = factor(c("dog","cat","monkey")[c(1,3,2,1,3,2,1,1)]) #make factor
a.numeric.vector = as.numeric.factor(some.factor) #convert factor representation.
```

---

box.outliers *Box Outliers*

Description

Squeeze all outliers onto standard.dev-limits and/or normalize to [0;1] scale

Usage

```
box.outliers(x, limit = 1.5, normalize = TRUE)
```

Arguments

x	numeric vector, matrix, array, data.frame
limit	limit(SD,standard deviation) any number deviating more than limit from mean is an outlier
normalize	TRUE/FALSE should output range be normalized to [0;1]?

Details

Can be used to squeeze high dimensional data into a box, hence the name box.outliers. Box.outliers is used internally in forestFloor-package to compute colour gradients without assigning unique colours to few outliers. It's a box because the borders uni-variate/non-interacting.

Value

matrix(n x p) of normalized values

Author(s)

Soren Havelund Welling, 2014

See Also

scale()

Examples

```
box.outliers = function (x, limit = 1.5)  {
  x = scale(x)
  x[ x > limit] =  limit
  x[-x > limit] = -limit
  x = x - min(x)
  x = x/(limit * 2)
  return(x)
}
n=1000 #some observations
p = 5  #some dimensions
X = data.frame(replicate(p,rnorm(n))) # a dataset
Xboxed =box.outliers(X,limit=1.5) #applying normalization
plot(Xboxed[,1],Xboxed[,2],col="#00000088") #plot output for first two dimensions
```

`convolute_ff` *Cross-validated main effects interpretation for all feature contributions.*

Description

convolute_ff estimates feature contributions of each feature separately as a function of the corresponding variable/feature. The estimator is a k-nearest neighbor function with Gaussian distance weighting and LOO cross-validation see `train.kknn`.

Usage

```
convolute_ff(ff,
             these.vars=NULL,
             k.fun=function() round(sqrt(n.obs)/2),
             userArgs.kknn = alist(kernel="gaussian"))
```

Arguments

ff	forestFloor object "forestFloor_regression" or "forestFloor_multiClass" consisting of at least ff$X and ff$FCmatrix with two matrices of equal size
these.vars	vector of col.indices to ff$X. Convolution can be limited to these.vars
k.fun	function to define k-neighbors to consider. n.obs is a constant as number of observations in ff$X. Hereby k neighbors is defined as a function k.fun of n.obs. To set k to a constant use e.g. k.fun = function() 10. k can also be overridden with userArgs.kknn = alist(kernel="Gaussian",kmax=10).
userArgs.kknn	argument list to pass to train.kknn function for each convolution. See (link) kknn.args. Conflicting arguments to this list will be overridden e.g. k.fun.

Details

convolute_ff uses train.kknn from kknn package to estimate feature contributions by their corresponding variables. The output inside a ff$FCfit will have same dimensions as ff$FCmatrix and the values will match quite well if the learned model structure is relative smooth and main effects are dominant. This function is e.g. used to estimate fitted lines in plot.forestFloor function "plot(ff,...)". LOO cross validation is used to quantify how much of feature contribution variation can be explained as a main effect.

Value

ff$FCfit a matrix of predicted feature contributions has same dimension as ff$FCmatrix. The output is appended to the input "forestFloor" object as $FCfit.

Author(s)

Soren Havelund Welling

Examples

```
## Not run:
library(forestFloor)
library(randomForest)

#simulate data
obs=1000
vars = 6
X = data.frame(replicate(vars,rnorm(obs)))
Y = with(X, X1^2 + 2*sin(X2*pi) + 8 * X3 * X4)
Yerror = 5 * rnorm(obs)
cor(Y,Y+Yerror)^2
Y= Y+Yerror

#grow a forest, remeber to include inbag
rfo=randomForest(X,Y,keep.inbag=TRUE)

ff = forestFloor(rfo,X)
```

```
ff = convolute_ff(ff) #return input oject with ff$FCfit included

#the convolutions correlation to the feature contribution
for(i in 1:6) print(cor(ff$FCmatrix[,i],ff$FCfit[,i])^2)

#plotting the feature contributions
pars=par(no.readonly=TRUE) #save graphicals
par(mfrow=c(3,2),mar=c(2,2,2,2))
for(i in 1:6) {
  plot(ff$X[,i],ff$FCmatrix[,i],col="#00000030",ylim=range(ff$FCmatrix))
  points(ff$X[,i],ff$FCfit[,i],col="red",cex=0.2)
}
par(pars) #restore graphicals

## End(Not run)
```

`convolute_ff2` *Low-level function to estimate a specific set of feature contributions by corresponding features with kknn-package. Used to estimate goodness-of-fit of surface in show3d.*

Description

Low-level function to estimate a selected combination feature contributions as function of selected features with leave-one-out k-nearest neighbor.

Usage

```
convolute_ff2(ff,
              Xi,
              FCi = NULL,
              k.fun=function() round(sqrt(n.obs)/2),
              userArgs.kknn = alist(kernel="gaussian")            )
```

Arguments

`ff`	forestFloor object class "forestFloor_regression" or "forestFloor_multiClass" consisting of at least ff$X and ff$FCmatrix with two matrices of equal size
`Xi`	integer vector, of column indices of ff$X to estimate by.
`FCi`	integer vector, column indices of features contributions in ff$FCmatrix to estimate. If more than one , these columns will be summed by samples/rows. If NULL then FCi will match Xi.
`k.fun`	function to define k-neighbors to consider. n.obs is a constant as number of observations in ff$X. Hereby k neighbors is defined as a function k.fun of n.obs. To set k to a constant use e.g. k.fun = function() 10. k can also be overridden with userArgs.kknn = alist(kernel="Gaussian",kmax=10).
`userArgs.kknn`	argument list passed to train.kknn function for each convolution, see `train.kknn`. Arguments in this list have priority of any arguments passed by default by this wrapper function. See argument merger `train.kknn`

Details

convolute_ff2 is a wrapper of train.kknn to estimate feature contributions by a set of features. This function is e.g. used to estimate the visualized surface layer in show3d function. LOO CV is used to quantify how much of a feature contribution variation can by explained by a given surface. Can in theory also be used to quantify higher dimensional interaction effects, but randomForest do not learn much 3rd order (or higher) interactions. Do not support orderByImportance, thus Xi and FCi points to column order of training matrix X.

Value

an numeric vector with one estimated feature contribution for any observation

Author(s)

Soren Havelund Welling

Examples

```
## Not run:
library(forestFloor)
library(randomForest)
library(rgl)
#simulate data
obs=2500
vars = 6
X = data.frame(replicate(vars,rnorm(obs)))
Y = with(X, X1^2 + 2*sin(X2*pi) + 8 * X3 * X4)
Yerror = 15 * rnorm(obs)
cor(Y,Y+Yerror)^2  #relatively noisy system
Y= Y+Yerror

#grow a forest, remeber to include inbag
rfo=randomForest(X,Y,keep.inbag=TRUE,ntree=1000,sampsize=800)

#obtain
ff = forestFloor(rfo,X)

#convolute the interacting feature contributions by their feature to understand relationship
fc34_convoluted = convolute_ff2(ff,Xi=3:4,FCi=3:4,  #arguments for the wrapper
                userArgs.kknn = alist(kernel="gaussian",k=25)) #arguments for train.kknn

#plot the joined convolution
plot3d(ff$X[,3],ff$X[,4],fc34_convoluted,
       main="convolution of two feature contributions by their own vaiables",
       #add some colour gradients to ease visualization
       #box.outliers squese all observations in a 2 std.dev box
       #univariately for a vector or matrix and normalize to [0;1]
       col=rgb(.7*box.outliers(fc34_convoluted),
               .7*box.outliers(ff$X[,3]),
               .7*box.outliers(ff$X[,4]))
       )
```

```
## End(Not run)
```

`convolute_grid` *Model structure grid estimated by feature contributions*

Description

Low-level n-dimensional grid wrapper of `kknn` (not `train.kknn`). Predicts a grid structure on the basis of estimated feature contributions. Is used to draw one 2D surface in a 3D plot (`show3d`) on basis of feature contributions.

Usage

```
convolute_grid          (ff,
                         Xi,
                         FCi = NULL,
                         grid = 30,
                         limit = 3,
                         zoom = 3,
                         k.fun=function() round(sqrt(n.obs)/2),
                         userArgs.kknn = alist(kernel="gaussian") )
```

Arguments

`ff`	the forestFloor object of class "forestFloor_regression" or "forestFloor_multiClass" at least containing ff$X and ff$FCmatrix with two matrices of equal size
`Xi`	the integer vector, of col indices of ff$X to estimate by, often of length 2 or 3. Note total number of predictions is a equal grid^"length of this vector".
`FCi`	the integer vector, of col indices of ff$FCmatrix. Those feature contributions to combine(sum) and estimate. If FCi=NULL, will copy Xi vector, which is the trivial choice.
`grid`	Either, an integer describing the number of grid.lines in each dimension(trivial choice) or, a full defined matrix of any grid position as defined by this function.
`limit`	a numeric scalar, number of standard deviations away from mean by any dimension to disregard outliers when spanning observations with grid. Set to limit=Inf outliers never should be disregarded.
`zoom`	numeric scalar, the size of the grid compared to the uni-variate range of data. If zoom=2 the grid will by any dimension span the double range of the observations. Outliers are disregarded with limit argument.
`k.fun`	function to define k-neighbors to consider. n.obs is a constant as number of observations in ff$X. Hereby k neighbors is defined as a function k.fun of n.obs. To set k to a constant use e.g. k.fun = function() 10. k can also be overridden with userArgs.kknn = alist(kernel="Gaussian",kmax=10).
`userArgs.kknn`	argument list to pass to train.kknn function for each convolution, see `kknn` for possible args. Arguments in this list will have priority of any passed by default by this wrapper function, see argument merger `append.overwrite.alists`

Details

This low-level function predicts feature contributions in a grid with train.kknn which is k-nearest neighbor + Gaussian weighting. This wrapper is used to construct the transparent grey surface in show3d.

Value

a data frame, 1 + X variable columns. First column is the predicted summed feature contributions as a function of the following columns feature coordinates.

Author(s)

Soren Havelund Welling

Examples

```
## Not run:
## avoid testing of rgl 3D plot on headless non-windows OS
## users can disregard this sentence.
if(!interactive() && Sys.info()["sysname"]!="Windows") skip=TRUE

library(rgl)
library(randomForest)
library(forestFloor)

#simulate data
obs=1500
vars = 6
X = data.frame(replicate(vars,runif(obs)))*2-1
Y = with(X, X1*2 + 2*sin(X2*pi) + 3* (X3+X2)^2 )
Yerror = 1 * rnorm(obs)
var(Y)/var(Y+Yerror)
Y= Y+Yerror

#grow a forest, remember to include inbag
rfo=randomForest::randomForest(X,Y,
                               keep.inbag=TRUE,
                               ntree=1000,
                               replace=TRUE,
                               sampsize=500,
                               importance=TRUE)

#compute ff
ff = forestFloor(rfo,X)

#print forestFloor
print(ff)

#plot partial functions of most important variables first
Col=fcol(ff,1)
plot(ff,col=Col,orderByImportance=TRUE)
```

```
#the pure feature contributions
rgl::plot3d(ff$X[,2],ff$X[,3],apply(ff$FCmatrix[,2:3],1,sum),
            #add some colour gradients to ease visualization
            #box.outliers squese all observations in a 2 std.dev box
            #univariately for a vector or matrix and normalize to [0;1]
            col=fcol(ff,2,orderByImportance=FALSE))

#add grid convolution/interpolation
#make grid with current function
grid23 = convolute_grid(ff,Xi=2:3,userArgs.kknn= alist(k=25,kernel="gaus"),grid=50,zoom=1.2)
#apply grid on 3d-plot
rgl::persp3d(unique(grid23[,2]),unique(grid23[,3]),grid23[,1],alpha=0.3,
col=c("black","grey"),add=TRUE)
#anchor points of grid could be plotted also
rgl::plot3d(grid23[,2],grid23[,3],grid23[,1],alpha=0.3,col=c("black"),add=TRUE)

## and we se that their is almost no variance out of the surface, thus is FC2 and FC3
## well explained by the feature context of both X3 and X4

### next example show how to plot a 3D grid + feature contribution
## this 4D application is very experimental

#Make grid of three effects, 25^3 = 15625 anchor points
grid123 = convolute_grid(ff,
                         Xi=c(1:3),
                         FCi=c(1:3),
                         userArgs.kknn = alist(
                           k= 100,
                           kernel = "gaussian",
                           distance = 1),
                         grid=25,
                         zoom=1.2)

#Select a dimension to place in layers
uni2 = unique(grid123[,2])  #2 points to X1 and FC1
uni2=uni2[c(7,9,11,13,14,16,18)] #select some layers to visualize

## plotting any combination of X2 X3 in each layer(from red to green) having different value of X1
count = 0
add=FALSE
for(i in uni2) {
  count = count +1
  this34.plane = grid123[grid123[,2]==i,]
  if (count==2) add=TRUE

  #  plot3d(ff$X[,1],ff$X[,2]
  persp3d(unique(this34.plane[,3]),
          unique(this34.plane[,4]),
          this34.plane[,1], add=add,
          col=rgb(count/length(uni2),1-count/length(uni2),0),alpha=0.1)
}
```

```
## plotting any combination of X1 X3 in each layer(from red to green) having different value of X2
uni3 = unique(grid123[,4])  #2 points to X1 and FC1
uni3=uni3[c(7,9,11,13,14,16,18)] #select some layers to visualize
count = 0
add=FALSE
for(i in uni3) {
  count = count +1
  this34.plane = grid123[grid123[,4]==i,]
  if (count==2) add=TRUE


  #plot3d(ff$X[,1],ff$X[,2])
  persp3d(unique(this34.plane[,2]),
          unique(this34.plane[,3]),
          this34.plane[,1], add=add,
          col=rgb(count/length(uni3),1-count/length(uni3),0),alpha=0.1)

}

## End(Not run)
```

`fcol` *Generic colour module for forestFloor objects*

Description

This colour module colour observations by selected variables. PCA decomposes a selection more than three variables. Space can be inflated by random forest variable importance, to focus coloring on influential variables. Outliers(>3std.dev) are automatically suppressed. Any colouring can be modified.

Usage

```
fcol(ff, cols = NULL, orderByImportance = NULL, plotTest=NULL, X.matrix = TRUE,
     hue = NULL, saturation = NULL, brightness = NULL,
     hue.range  = NULL, sat.range  = NULL, bri.range  = NULL,
     alpha = NULL, RGB = NULL, byResiduals=FALSE, max.df=3,
     imp.weight = NULL, imp.exp = 1,outlier.lim = 3,RGB.exp=NULL)
```

Arguments

`ff`	a object of class "forestFloor_regression" or "forestFloor_multiClass" or a matrix or a data.frame. No missing values. X.matrix must be set TRUE for "forestFloor_multiClass" as colouring by multiClass feature contributions is not supported.

`cols` vector of indices of columns to colour by, will refer to ff$X if X.matrix=T and else ff$FCmatrix. If ff itself is a matrix or data.frame, indices will refer to these columns

`orderByImportance` logical, should cols refer to X column order or columns sorted by variable importance. Input must be of forestFloor -class to use this. Set to FALSE if no importance sorting is wanted. Otherwise leave as is.

`plotTest` NULL(plot by test set if available), TRUE(plot by test set), FALSE(plot by train), "andTrain"(plot by both test and train)

`X.matrix` logical, true will use feature matrix false will use feature contribution matrix. Only relevant if input is forestFloor object.

`hue` value within [0,1], hue=1 will be exactly as hue = 0 colour wheel settings, will skew the colour of all observations without changing the contrast between any two given observations.

`saturation` value within [0,1], mean saturation of colours, 0 is grey tone and 1 is maximal colourful.

`brightness` value within [0,1], mean brightness of colours, 0 is black and 1 is lightly colours.

`hue.range` value within [0,1], ratio of colour wheel, small value is small slice of colour wheel those little variation in colours. 1 is any possible colour except for RGB colour system.

`sat.range` value within [0,1], for colouring of 2 or more variables, a range of saturation is needed to obtain more degrees of freedom in the colour system. But as saturation of is preferred to be >.75 the range of saturation cannot here exceed .5. If NULL sat.range will set widest possible without exceeding range.

`bri.range` value within [0,1], for colouring of 3 or more variables, a range of brightness is needed to obtain more degrees of freedom in the colour system. But as brightness of is preferred to be >.75 the range of saturation cannot here exceed .5. If NULL bri.range will set widest possible without exceeding range.

`alpha` value within [0;1] transparency of colours.

`RGB` logical TRUE/FALSE,
RGB=NULL: will turn TRUE if one variable selected RGB=TRUE: Red-Green-Blue colour: a system with fewer colours(~3) but more contrast. Can still be altered by hue, saturation, brightness etc.
RGB=FALSE: True-colour-system: Maximum colour detail. Sometimes more confusing.

`byResiduals` logical, should coloring be residuals of main effect fit(overrides X.matrix=). If no fit has been computed "is.null(ff$FCfit)", a temporarily main effect fit will be computed. Use ff = convolute_ff(ff) to only compute once and/or to modify fit parameters.

`max.df` integer 1, 2, or 3 only. Only for true-colour-system, the maximal allowed degrees of freedom in a colour scale. If more variables selected than max.df, PCA decompose to request degrees of freedom. max.df = 1 will give more simple colour gradients

`imp.weight` Logical?, Should importance from a forestFloor object be used to weight selected variables? obviously not possible if input ff is a matrix or data.frame. If

randomForest(importance=TRUE) during training, variable importance will be used. Otherwise the more unreliable gini_importance coefficient.

`imp.exp` exponent to modify influence of imp.weight. 0 is not influence. -1 is counter influence. 1 is linear influence. .5 is square root influence etc..

`outlier.lim` number from 0 to Inf. Any observation which univariately exceed this limit will be suppressed, as if it actually where on this limit. Normal limit is 3 standard deviations. Extreme outliers can otherwise reserve alone a very large part of a given linear colour gradient. This leads to visualization where outlier have one colour and any other observation another but same colour.

`RGB.exp` value between]1;>1]. Defines steepness of the gradient of the RGB colour system Close to one green middle area is missing. For values higher than 2, green area is dominating

Details

fcol produces colours for any observation. These are used plotting.

Value

a character vector specifying the colour of any observations. Each elements is something like "#F1A24340", where F1 is the hexadecimal of the red colour, then A2 is the green, then 43 is blue and 40 is transparency.

Author(s)

Soren Havelund Welling

Examples

```
## Not run:
#example 1 - fcol used on data.frame or matrix
library(forestFloor)
X = data.frame(matrix(rnorm(1000),nrow=1000,ncol=4))
X[] = lapply(X,jitter,amount = 1.5)

#single variable gradient by X1 (Unique colour system)
plot(X,col=fcol(X,1))
#double variable gradient by X1 and X2 (linear colour system)
plot(X,col=fcol(X,1:2))
#triple variable gradient (PCA-decomposed, linear colour system)
plot(X,col=fcol(X,1:3))
#higher based gradient    (PCA-decomposed, linear colour system)
plot(X,col=fcol(X,1:4))


#force linear col + modify colour wheel
plot(X,col=fcol(X,
                cols=1, #colouring by one variable
                RGB=FALSE,
                hue.range = 4, #cannot exceed 1, if colouing by more than one var
```

```
                        #except if max.df=1 (limits to 1D gradient)
                saturation=1,
                brightness = 0.6))

#colour by one dimensional gradient first PC of multiple variables
plot(X,col=fcol(X,
                cols=1:2, #colouring by multiple
                RGB=TRUE, #possible because max.df=1
                max.df = 1, #only 1D gradient (only first principal component)
                hue.range = 2, #can exceed 1, because max.df=1
                saturation=.95,
                brightness = 0.8))

##example 2 - fcol used with forestFloor objects
library(forestFloor)
library(randomForest)

X = data.frame(replicate(6,rnorm(1000)))
y = with(X,.3*X1^2+sin(X2*pi)+X3*X4)
rf = randomForest(X,y,keep.inbag = TRUE,sampsize = 400)
ff = forestFloor(rf,X)

#colour by most important variable
plot(ff,col=fcol(ff,1))

#colour by first variable in data set
plot(ff,col=fcol(ff,1,orderByImportance = FALSE),orderByImportance = FALSE)

#colour by feature contributions
plot(ff,col=fcol(ff,1:2,order=FALSE,X.matrix = FALSE,saturation=.95))

#colour by residuals
plot(ff,col=fcol(ff,3,orderByImportance = FALSE,byResiduals = TRUE))

#colour by all features (most useful for colinear variables)
plot(ff,col=fcol(ff,1:6))

#disable importance weighting of colour
#(important colours get to define gradients more)
plot(ff,col=fcol(ff,1:6,imp.weight = FALSE)) #useless X5 and X6 appear more colourful

#insert outlier in data set in X1 and X2
ff$X[1,1] = 10; ff$X[1,2] = 10

plot(ff,col=fcol(ff,1)) #colour not distorted, default: outlier.lim=3
plot(ff,col=fcol(ff,1,outlier.lim = Inf)) #colour gradient distorted by outlier
plot(ff,col=fcol(ff,1,outlier.lim = 0.5)) #too little outlier.lim


## End(Not run)
```

`forestFloor` *Compute out-of-bag cross-validated feature contributions to visualize model structures of randomForest models.*

Description

Computes a cross validated feature contribution matrix from a randomForest model-fit and outputs a forestFloor S3 class object (a list), including unscaled importance and the original training set. The output object is the basis for all visualizations.

Usage

```
forestFloor(rf.fit, X, Xtest=NULL, calc_np = FALSE, binary_reg = FALSE,
            bootstrapFC = FALSE, ...)
```

Arguments

`rf.fit`	rf.fit, a random forest object as the output from randomForest::randomForest
`X`	data.frame of input variables, numeric(continuous), discrete(treated as continuous) or factors(categorical). n_rows observations and n_columns features X MUST be the same data.frame as used to train the random forest, see above item.
`Xtest`	data.frame of input variables, numeric(continuous), discrete(treated as continuous) or factors(categorical). n_rows test_examples and n_columns features Xtest MUST have same number and order of columns(variables) as X. Number of rows can vary.
`calc_np`	TRUE/FALSE. Calculate Node Predictions(TRUE) or reuse information from rf.fit(FALSE)? Slightly faster when FALSE for regression. calc_np=TRUE will only take effect for rf.fit of class "randomForest" and type="regression". This option, is only for developmental purposes. Just set =FALSE always, as function will override this choice if not appropriate.
`binary_reg`	boolean, if TRUE binary classification can be changed to "percentage votes" of class 1, and thus be treated as regression.
`bootstrapFC`	boolean, if TRUE an extra column is added to FCmatrix or one extra matrix to FCarray accounting for the minor feature contributions attributed to random bootstraps or stratifications. Mainly useful to check FC row sums actually are equal to OOB-CV predictions, or to tweak randomForest into a "probability forest"-like model.
`...`	For classification it is possible to manually set majorityTerminal=FALSE. For the randomForest classification implementation majorityTerminal is by default set to TRUE, as each tree uses majority vote within terminal nodes. In other implemenations terminal nodes are not neccesarily reduced by majority voting before aggregetion on ensemble level. majorityTerminal, does not apply to random forest regressions.

Details

forestFloor computes out-of-bag cross validated feature contributions for a "randomForest" class object. Other packages will be supported in future, mail me a request. forestFloor guides you to discover the structure of a randomForest model fit. Check examples of how latent interactions can be identified with colour gradients.

What is FC?: Feature contributions are the sums over all local increments for each observation for each feature divided by the number of trees. A local increment is the change of node prediction from parent to daughter node split by a given feature. Thus a feature contribution summarizes the average outcome for all those times a given sample was split by a given feature. forestFloor use inbag samples to calculate local increments, but only sum local increments over out-of-bag samples divided with OOBtimes. OOBtimes is the number of times a given observation have been out-of-bag. which is roundly ntrees / 3. In practice this removes a substantial self-leverage of samples to the corresponding feature contributions. Hereby visualizations becomes less noisy.

What is FC used for?: Feature contributions is smart way to decompose a RF mapping structure into additive components. Plotting FC's against variables values yields at first glance plots similar to marginal-effect plots, partial dependence plots and vector effect characteristic plots. This package forsetFloor, make use of feature contributions to separate main effects and identify plus quantify latent interactions. The advantages of forestFloor over typical partial.dependence plots are: (1) Easier to identify interactions. (2) Training samples is a part of plot, such that extrapolated model structure can be disregarded. (3) The "goodness of visualization" (how exactly the plot represent the higher dimensional model structure) can be quantified. (4) Cheerful colours and 3D graphics thanks to the rgl package.

RF regression takes input features and outputs a target value. RF classification can output a pseudo probability vector with predicted class probability for each sample. The RF mapping topology of classification is different than for regression as the output is no longer a scalar, the output is a vector with predicted class probability for each class. For binary classification this topology can be simplified to a regression-like scalar as the probability of class_1 = 1 - class_2. Set binary_reg=TRUE for a binary RF classification to get regression like visualizations. For multi-class the output space is probability space where any point is a probability prediction of each target class.

To plot forestFloor objects use plot-method plot.forestFloor and function show3d. Input parameters for classification or regression are not entirely the same. Check help-file `plot.forestFloor` and `show3d`. For 3-class problems the special function `plot_simplex3` can plot the probability predictions in a 2D phase diagram (K-1 simplex).

Value

the forestFloor function outputs(depending on type rf.fit) an object of either class "forestFloor_regression" or "forestFloor_multiClass" with following elements:

`X`	a copy of the training data or feature space matrix/data.frame, X. The copy is passed unchanged from the input of this function. X is used in all visualization to expand the feature contributions over the features of which they were recorded.
`Y`	a copy of the target vector, Y.
`importance`	The gini-importance or permutation-importance a.k.a variable importance of the random forest object (unscaled). If rfo=randomForest(X,Y,importance=FALSE), gini-importance is used. Gini-importance is less reproducible and more biased. The extra time used to compute permutation-importance is negligible.

imp_ind	the importance indices is the order to sort the features by descending importance. imp_ind is used by plotting functions to present most relevant feature contributions first. If using gini-importance, the order of plots is more random and will favor continuous variables. The plots themselves will not differ.
FC_matrix	[ONLY forestFloor_regression.] feature contributions in a matrix. n_row observations and n_column features - same dimensions as X.
FC_array	[ONLY forestFloor_multiClass.] feature contributions in a array. n_row observations and n_column features and n_layer classes. First two dimensions will match dimensions of X.

Note

check out more guides at forestFloor.dk

Author(s)

Soren Havelund Welling

References

Interpretation of QSAR Models Based on Random Forest Methods, http://dx.doi.org/10.1002/minf.201000173
Interpreting random forest classification models using a feature contribution method, http://arxiv.org/abs/1312.1121

See Also

plot.forestFloor, show3d,

Examples

```
## Not run:
## avoid testing of rgl 3D plot on headless non-windows OS
## users can disregard this sentence.
if(!interactive() && Sys.info()["sysname"]!="Windows") skipRGL=TRUE

#1 - Regression example:
set.seed(1234)
library(forestFloor)
library(randomForest)

#simulate data y = x1^2+sin(x2*pi)+x3*x4 + noise
obs = 5000 #how many observations/samples
vars = 6   #how many variables/features
#create 6 normal distr. uncorr. variables
X = data.frame(replicate(vars,rnorm(obs)))
#create target by hidden function
Y = with(X, X1^2 + sin(X2*pi) + 2 * X3 * X4 + 0.5 * rnorm(obs))

#grow a forest
rfo = randomForest(
  X, #features, data.frame or matrix. Recommended to name columns.
```

```
  Y, #targets, vector of integers or floats
  keep.inbag = TRUE,  # mandatory,
  importance = TRUE,  # recommended, else ordering by giniImpurity (unstable)
  sampsize = 1500 ,   # optional, reduce tree sizes to compute faster
  ntree = if(interactive()) 500 else 50 #speedup CRAN testing
)

#compute forestFloor object, often only 5-10% time of growing forest
ff = forestFloor(
  rf.fit = rfo,       # mandatory
  X = X,              # mandatory
  calc_np = FALSE,    # TRUE or FALSE both works, makes no difference
  binary_reg = FALSE  # takes no effect here when rfo$type="regression"
)

#print forestFloor
print(ff) #prints a text of what an 'forestFloor_regression' object is
plot(ff)

#plot partial functions of most important variables first
plot(ff,                       # forestFloor object
     plot_seq = 1:6,           # optional sequence of features to plot
     orderByImportance=TRUE    # if TRUE index sequence by importance, else by X column
)

#Non interacting features are well displayed, whereas X3 and X4 are not
#by applying color gradient, interactions reveal themself
#also a k-nearest neighbor fit is applied to evaluate goodness-of-fit
Col=fcol(ff,3,orderByImportance=FALSE) #create color gradient see help(fcol)
plot(ff,col=Col,plot_GOF=TRUE)

#feature contributions of X3 and X4 are well explained in the context of X3 and X4
# as GOF R^2>.8

show3d(ff,3:4,col=Col,plot_GOF=TRUE,orderByImportance=FALSE)

#if needed, k-nearest neighbor parameters for goodness-of-fit can be accessed through convolute_ff
#a new fit will be calculated and saved to forstFloor object as ff$FCfit
ff = convolute_ff(ff,userArgs.kknn=alist(kernel="epanechnikov",kmax=5))
plot(ff,col=Col,plot_GOF=TRUE) #this computed fit is now used in any 2D plotting.


###
#2 - Multi classification example:   (multi is more than two classes)
set.seed(1234)
library(forestFloor)
library(randomForest)

data(iris)
X = iris[,!names(iris) %in% "Species"]
Y = iris[,"Species"]

rf = randomForest(
```

```
  X,Y,
  keep.forest=TRUE,  # mandatory
  keep.inbag=TRUE,   # mandatory
  samp=20,           # reduce complexity of mapping structure, with same OOB%-explained
  importance  = TRUE # recommended, else ordering by giniImpurity (unstable)
)

ff = forestFloor(rf,X)

plot(ff,plot_GOF=TRUE,cex=.7,
     colLists=list(c("#FF0000A5"),
                   c("#00FF0050"),
                   c("#0000FF35")))

#...and 3D plot, see show3d
show3d(ff,1:2,1:2,plot_GOF=TRUE)

#...and simplex plot (only for three class problems)
plot_simplex3(ff)
plot_simplex3(ff,zoom.fit = TRUE)

#...and 3d simplex plots (rough look, Z-axis is feature)
plot_simplex3(ff,fig3d = TRUE)

###
#3 - binary regression example
#classification of two classes can be seen as regression in 0 to 1 scale
set.seed(1234)
library(forestFloor)
library(randomForest)
data(iris)
X = iris[-1:-50,!names(iris) %in% "Species"] #drop third class virginica
Y = iris[-1:-50,"Species"]
Y = droplevels((Y)) #drop unused level virginica

rf = randomForest(
  X,Y,
  keep.forest=TRUE,  # mandatory
  keep.inbag=TRUE,   # mandatory
  samp=20,           # reduce complexity of mapping structure, with same OOB%-explained
  importance  = TRUE # recommended, else giniImpurity
)

ff = forestFloor(rf,X,
                 calc_np=TRUE,    #mandatory to recalculate
                 binary_reg=TRUE) #binary regression, scale direction is printed
Col = fcol(ff,1) #color by most important feature
plot(ff,col=Col)   #plot features

#interfacing with rgl::plot3d
show3d(ff,1:2,col=Col,plot.rgl.args = list(size=2,type="s",alpha=.5))

## End(Not run)
```

Description

A method to plot an object of forestFloor-class. Plot partial feature contributions of the most important variables. Colour gradients can be applied to show possible interactions. Fitted function(plot_GOF) describe FC only as a main effect and quantifies 'Goodness Of Fit'.

Usage

```
## S3 method for class 'forestFloor_regression'
 plot(
  x,
  plot_seq=NULL,
  plotTest = NULL,
  limitY=TRUE,
  orderByImportance=TRUE,
  cropXaxes=NULL,
  crop_limit=4,
  plot_GOF = TRUE,
  GOF_args = list(col="#33333399"),
  speedup_GOF = TRUE,
  ...)

## S3 method for class 'forestFloor_multiClass'
 plot(
  x,
  plot_seq = NULL,
  label.seq = NULL,
  plotTest  = NULL,
  limitY = TRUE,
  col = NULL,
  colLists = NULL,
  orderByImportance = TRUE,
  fig.columns = NULL,
  plot_GOF = TRUE,
  GOF_args = list(),
  speedup_GOF = TRUE,
  jitter_these_cols = NULL,
  jitter.factor = NULL,
  ...)
```

Arguments

x	forestFloor-object, also abbreviated ff. Basically a list of class="forestFloor" containing feature contributions, features, targets and variable importance.

plot_seq	a numeric vector describing which variables and in what sequence to plot. Ordered by importance as default. If orderByImportance = F, then by feature/column order of training data.
label.seq	[only classification] a numeric vector describing which classes and in what sequence to plot. NULL is all classes ordered is in levels in x$Y of forestFloor_mulitClass object x.
plotTest	NULL(plot by test set if available), TRUE(plot by test set), FALSE(plot by train), "andTrain"(plot by both test and train)
fig.columns	[only for multiple plotting], how many columns per page. default(NULL) is 1 for one plot, 2 for 2, 3 for 3, 2 for 4 and 3 for more.
limitY	TRUE/FLASE, constrain all Yaxis to same limits to ensure relevance of low importance features is not over interpreted
col	Either a colur vector with one colour per plotted class label or a list of colour vectors. Each element is a colour vector one class. Colour vectors in list are normally either of length 1 with or of length equal to number of training observations. NULL will choose standard one colour per class.
colLists	Deprecetated, will be replaced by col input
jitter_these_cols	vector to apply jitter to x-axis in plots. Will refer to variables. Useful to for categorical variables. Default=NULL is no jitter.
jitter.factor	value to decide how much jitter to apply. often between .5 and 3
orderByImportance	TRUE / FALSE should plotting and plot_seq be ordered after importance. Most important feature plot first(TRUE)
cropXaxes	a vector of indices of which zooming of x.axis should look away from outliers
crop_limit	a number often between 1.5 and 5, referring limit in sigmas from the mean defining outliers if limit = 2, above selected plots will zoom to +/- 2 std.dev of the respective features.
plot_GOF	Boolean TRUE/FALSE. Should the goodness of fit be plotted as a line?
GOF_args	Graphical arguments fitted lines, see points for parameter names.
speedup_GOF	Should GOF only computed on reasonable sub sample of data set to speedup computation. GOF estimation leave-one-out-kNN becomes increasingly slow for +1500 samples.
...	... other arguments passed to par or plot . e.g. mar=, mfrow=, is passed to par, and cex= is passed to plot. par() arguments are reset immediately as plot function returns.

Details

The method plot.forestFloor visualizes partial plots of the most important variables first. Partial dependence plots are available in the randomForest package. But such plots are single lines(1d-slices) and do not answer the question: Is this partial function(PF) a fair generalization or subject to global or local interactions.

Author(s)

Soren Havelund Welling

Examples

```
## Not run:
  ## avoid testing of rgl 3D plot on headless non-windows OS
  ## users can disregard this sentence.
  if(!interactive() && Sys.info()["sysname"]!="Windows") skipRGL=TRUE

  ###
  #1 - Regression example:
  set.seed(1234)
  library(forestFloor)
  library(randomForest)

  #simulate data y = x1^2+sin(x2*pi)+x3*x4 + noise
  obs = 5000 #how many observations/samples
  vars = 6   #how many variables/features
  #create 6 normal distr. uncorr. variables
  X = data.frame(replicate(vars,rnorm(obs)))
  #create target by hidden function
  Y = with(X, X1^2 + sin(X2*pi) + 2 * X3 * X4 + 0.5 * rnorm(obs))

  #grow a forest
  rfo = randomForest(
    X, #features, data.frame or matrix. Recommended to name columns.
    Y, #targets, vector of integers or floats
    keep.inbag = TRUE,  # mandatory,
    importance = TRUE,  # recommended, else ordering by giniImpurity (unstable)
    sampsize = 1500 ,   # optional, reduce tree sizes to compute faster
    ntree = if(interactive()) 1000 else 25 #speedup CRAN testing
  )

  #compute forestFloor object, often only 5-10% time of growing forest
  ff = forestFloor(
    rf.fit = rfo,       # mandatory
    X = X,              # mandatory
    calc_np = FALSE,    # TRUE or FALSE both works, makes no difference
    binary_reg = FALSE  # takes no effect here when rfo$type="regression"
  )

  #print forestFloor
  print(ff) #prints a text of what an 'forestFloor_regression' object is
  plot(ff)

  #plot partial functions of most important variables first
  plot(ff,                       # forestFloor object
       plot_seq = 1:6,           # optional sequence of features to plot
       orderByImportance=TRUE    # if TRUE index sequence by importance, else by X column
  )
```

```
#Non interacting features are well displayed, whereas X3 and X4 are not
#by applying color gradient, interactions reveal themself
#also a k-nearest neighbor fit is applied to evaluate goodness-of-fit
Col=fcol(ff,3,orderByImportance=FALSE) #create color gradient see help(fcol)
plot(ff,col=Col,plot_GOF=TRUE)

#feature contributions of X3 and X4 are well explained in the context of X3 and X4
# as GOF R^2>.8

show3d(ff,3:4,col=Col,plot_GOF=TRUE,orderByImportance=FALSE)

#if needed, k-nearest neighbor parameters for goodness-of-fit can be accessed through convolute_ff
#a new fit will be calculated and saved to forstFloor object as ff$FCfit
ff = convolute_ff(ff,userArgs.kknn=alist(kernel="epanechnikov",kmax=5))
plot(ff,col=Col,plot_GOF=TRUE) #this computed fit is now used in any 2D plotting.


###
#2 - Multi classification example:   (multi is more than two classes)
set.seed(1234)
library(forestFloor)
library(randomForest)

data(iris)
X = iris[,!names(iris) %in% "Species"]
Y = iris[,"Species"]

rf = randomForest(
  X,Y,
  keep.forest=TRUE,  # mandatory
  keep.inbag=TRUE,   # mandatory
  samp=20,          # reduce complexity of mapping structure, with same OOB%-explained
  importance  = TRUE, # recommended, else ordering by giniImpurity (unstable)
  ntree = if(interactive()) 1000 else 25 #speedup CRAN testing
)

ff = forestFloor(rf,X)

plot(ff,plot_GOF=TRUE,cex=.7,
  col=c("#FF0000A5","#00FF0050","#0000FF35") #one col per plotted class
)

#...and 3D plot, see show3d
show3d(ff,1:2,1:2,plot_GOF=TRUE)

#...and simplex plot (only for three class problems)
plot_simplex3(ff)
plot_simplex3(ff,zoom.fit = TRUE)

#...and 3d simplex plots (rough look, Z-axis is feature)
plot_simplex3(ff,fig3d = TRUE)

###
```

```
#3 - binary regression example
#classification of two classes can be seen as regression in 0 to 1 scale
set.seed(1234)
library(forestFloor)
library(randomForest)
data(iris)
X = iris[-1:-50,!names(iris) %in% "Species"] #drop third class virginica
Y = iris[-1:-50,"Species"]
Y = droplevels((Y)) #drop unused level virginica

rf = randomForest(
  X,Y,
  keep.forest=TRUE,  # mandatory
  keep.inbag=TRUE,   # mandatory
  samp=20,          # reduce complexity of mapping structure, with same OOB%-explained
  importance  = TRUE, # recommended, else giniImpurity
   ntree = if(interactive()) 1000 else 25 #speedup CRAN testing
)

ff = forestFloor(rf,X,
                 calc_np=TRUE,    #mandatory to recalculate
                 binary_reg=TRUE) #binary regression, scale direction is printed
Col = fcol(ff,1) #color by most important feature
plot(ff,col=Col)   #plot features

#interfacing with rgl::plot3d
show3d(ff,1:2,col=Col,plot.rgl.args = list(size=2,type="s",alpha=.5))

## End(Not run)
```

`plot_simplex3` *3-class simplex forestFloor plot*

Description

3-class forestFloor plotted in a 2D simplex. The plot describes with feature contributions the change of predicted class probability for each sample due a single variable given all other variables. This plot is better than regular multiclass plots (plot.forestFloor_multiClass) to show the change of class probabilities, but the feature values can only be depcited as a colour gradient. But (fig3d=TRUE) allows the feature value to be depicted by the Z-axis as a extra pop-up 3D plot.

Usage

```
plot_simplex3(ff,
              Xi          = NULL,
              includeTotal = TRUE,
              label.col    = NULL,
              fig.cols     = 3,
              fig.rows     = NULL,
```

```
                auto.alpha   = 0.25,
                fig3d        = FALSE,
                restore_par  = TRUE,
                set_pars     = TRUE,
                zoom.fit     = NULL,
                var.col      = NULL,
                plot.sep.centroid = TRUE)
```

Arguments

ff	x also abbrivated ff, forestFloor_mulitClass the output from the forestFloor function. Must have 3 classes exactly.
Xi	vector of integer indices (refeering to column order of trainingset) to what feature contributions should be plotted in individual plots.
includeTotal	TRUE / FALSE. Combined separation of all feature contributions, which is equal to the separation of the entire model can be included.
label.col	a colour vector of K classes length defining the colour of each class for plotting. NULL is auto.
fig.cols	How many columns should be plotted sideways, is passed to par(mfrow=c(fig.rows,fig.cols))
fig.rows	How many rows should be plotted, is passed to par(mfrow=c(fig.rows,fig.cols)) NULL is auto
auto.alpha	a scalar between 0.5 to 1 most often. Low values increase transparancy of points used to avoid overplotting. auto.alpha is alpha corrected of samplesize such that less adjustment is needed.
fig3d	TRUE/FALSE, a 3D plot including the variable as an axis can be co-plotted with rgl.
restore_par	TRUE/FALSE, calls to graphics par() will be reset
set_pars	TRUE/FALSE, if FALSE plot function will rather inherrit plot settings global pars. USeful for multi plotting loops.
zoom.fit	NULL/TRUE, if TRUE zooming on samples will be applied. Do not set to FALSE.
var.col	a single colour or a colour vector of N samples length. Samples will be coloured accordingly. use function fcol to make colour gradient e.g. by the variable values themselves. See example fcol.
plot.sep.centroid	TRUE/FALSE. Should the average bootstrap prediction be plotted? If no bootstrap stratification, the average bootstrap prediction is equal to class distribution training set. RF model probalistic predictions is equal to average bootstrap prediction plus all feature contributions.

Details

Random forest 3 class maps from a feature space to a 3 dimensional (K-1) probability simplex space, which can be plotted in 2D because class probabilities sum to one, and class feature contributions sum to zero. The centroid these plots is the prior of the random forest model. The prior, unless modified with statification is the target class distribution. Default majority voting lines would run from middle to the corners.

Author(s)

Soren Havelund Welling

Examples

```
## Not run:
  library(randomForest)
  library(forestFloor)
  require(utils)

  data(iris)

  X = iris[,!names(iris) %in% "Species"]
  Y = iris[,"Species"]
  as.numeric(Y)
  rf.test42 = randomForest(X,Y,keep.forest=TRUE,
    replace=FALSE,keep.inbag=TRUE,samp=15,ntree=100)
  ff.test42 = forestFloor(rf.test42,X,calc_np=FALSE,binary_reg=FALSE)

  plot(ff.test42,plot_GOF=TRUE,cex=.7,
       colLists=list(c("#FF0000A5"),
                     c("#00FF0050"),
                     c("#0000FF35")))

  show3d(ff.test42,1:2,3:4,plot_GOF=TRUE)

  #plot all effect 2D only
  pars = plot_simplex3(ff.test42,Xi=c(1:3),restore_par=FALSE,zoom.fit=NULL,
    var.col=NULL,fig.cols=2,fig.rows=1,fig3d=FALSE,includeTotal=TRUE,auto.alpha=.4
    ,set_pars=TRUE)

  pars = plot_simplex3(ff.test42,Xi=0,restore_par=FALSE,zoom.fit=NULL,
    var.col=alist(alpha=.3,cols=1:4),fig3d=FALSE,includeTotal=TRUE,
    auto.alpha=.8,set_pars=FALSE)

  for (I in ff.test42$imp_ind[1:4])  {
    #plotting partial OOB-CV separation(including interactions effects)
    #coloured by true class
    pars = plot_simplex3(ff.test42,Xi=I,restore_par=FALSE,zoom.fit=NULL,
    var.col=NULL,fig.cols=4,fig.rows=2,fig3d=TRUE,includeTotal=FALSE,label.col=1:3,
    auto.alpha=.3,set_pars = (I==ff.test42$imp_ind[1]))

    #coloured by varaible value
    pars = plot_simplex3(ff.test42,Xi=I,restore_par=FALSE,zoom.fit=TRUE,
    var.col=alist(order=FALSE,alpha=.8),fig3d=FALSE,includeTotal=(I==4),
    auto.alpha=.3,set_pars=FALSE)
  }

## End(Not run)
```

print.forestFloor *print summary of forestFloor.Object*

Description

This function simply states the obvious and returns the elements inside the object list.

Usage

```
## S3 method for class 'forestFloor_regression'
    print(x,...)
## S3 method for class 'forestFloor_multiClass'
print(x,...)
```

Arguments

x	x also abbreviated ff, forestFloor_Object the output from the forestFloor function
...	... other arguments passed to generic print function

Details

prints short help text for usage of a forestFloor_object

Author(s)

Soren Havelund Welling

Examples

```
## Not run:
#simulate data
obs=1000
vars = 6
X = data.frame(replicate(vars,rnorm(obs)))
Y = with(X, X1^2 + sin(X2*pi) + 2 * X3 * X4 + 0.5 * rnorm(obs))

#grow a forest, remeber to include inbag
rfo=randomForest::randomForest(X,Y,keep.inbag=TRUE)

#compute topology
ff = forestFloor(rfo,X)

#print forestFloor
print(ff)

## End(Not run)
```

`recTree` *recursiveTree: cross-validated feature contributions*

Description

internal C++ functions to compute feature contributions for a random Forest

Usage

```
recTree(  vars, obs, ntree, calculate_node_pred, X,Y,majorityTerminal, leftDaughter,
    rightDaughter, nodestatus, xbestsplit, nodepred, bestvar,
    inbag, varLevels, OOBtimes, localIncrements)

multiTree(vars, obs, ntree, nClasses,          X,Y,majorityTerminal, leftDaughter,
    rightDaughter, nodestatus, xbestsplit, nodepred, bestvar,
    inbag, varLevels, OOBtimes, localIncrements)
```

Arguments

`vars`	number of variables in X
`obs`	number of observations in X
`ntree`	number of trees starting from 1 function should iterate, cannot be higher than columns of inbag
`nClasses`	number of classes in classification forest
`calculate_node_pred`	should the node predictions be recalculated(true) or reused from nodepred-matrix(false & regression)
`X`	X training matrix
`Y`	target vector, factor or regression
`majorityTerminal`	bool, majority vote in terminal nodes? Default is FALSE for regression. Set only to TRUE when binary_reg=TRUE.
`leftDaughter`	a matrix from a the output of randomForest rf$forest$leftDaughter the node.number/row.number of the leftDaughter in a given tree by column
`rightDaughter`	a matrix from a the output of randomForest rf$forest$rightDaughter the node.number/row.number of the rightDaughter in a given tree by column
`nodestatus`	a matrix from a the output of randomForest rf$forest$nodestatus the nodestatus of a given node in a given tree
`xbestsplit`	a matrix from a the output of randomForest rf$forest$xbestsplit. The split point of numeric variables or the binary split of categorical variables. See help file of randomForest::getTree for details of binary expansion for categorical splits.
`nodepred`	a matrix from a the output of randomForest rf$forest$xbestsplit. The inbag target average for regression mode and the majority target class for classification

`bestvar` a matrix from a the output of randomForest rf$forest$xbestsplit the inbag target average for regression mode and the majority target class for classification

`inbag` a matrix as the output of randomForest rf$inbag. Contain counts of how many times a sample was selected for a given tree.

`varLevels` the number of levels of all variables, 1 for continuous or discrete, >1 for categorical variables. This is needed for categorical variables to interpret binary split from xbestsplit.

`OOBtimes` number of times a certain observation was out-of-bag in the forest. Needed to compute cross-validated feature contributions as these are summed local increments over out-of-bag observations over features divided by this number. In previous implementation(rfFC), articles(see references) feature contributions are summed by all observations and is divived by ntrees.

`localIncrements`
an empty matrix to store localIncrements during computation. As C++ function returns, the input localIncrement matrix contains the feature contributions.

Details

This is function is excuted by the function forestFloor. This is a c++/Rcpp implementation computing feature contributions. The main differences from this implementation and the rfFC-package(Rforge), is that these feature contributions are only summed over out-of-bag samples yields a cross-validation. This implementation allows sample replacement, binary and multi-classification.

Value

no output, the feature contributions are writtten directly to localIncrements input

Author(s)

Soren Havelund Welling

References

Interpretation of QSAR Models Based on Random Forest Methods, http://dx.doi.org/10.1002/minf.201000173
Interpreting random forest classification models using a feature contribution method, http://arxiv.org/abs/1312.1121

`show3d` *make forestFloor 3D-plot of random forest feature contributions*

Description

2 features features(horizontal XY-plane) and one combined feature contribution (vertical Z-axis). Surface response layer will be estimated(kknn package) and plotted alongside the data points. 3D graphic device is rgl. Will dispatch methods show3d.forestFloor_regression for regression and show3d_forestFloor_multiClass for classification.

Usage

```
## S3 method for class 'forestFloor_regression'
 show3d(
      x,
      Xi  = 1:2,
      FCi = NULL,
      col = "#12345678",
      plotTest = NULL,
      orderByImportance = TRUE,
      surface=TRUE,
      combineFC = sum,
      zoom=1.2,
      grid.lines=30,
      limit=3,
      cropPointsOutSideLimit = TRUE,
      kknnGrid.args = alist(),
      plot.rgl.args = alist(),
      surf.rgl.args = alist(),
      user.gof.args = alist(),
      plot_GOF = TRUE,
      ...)

## S3 method for class 'forestFloor_multiClass'
show3d(
      x,
      Xi,
      FCi=NULL,
      plotTest = NULL,
      label.seq=NULL,
      kknnGrid.args=list(NULL),
      plot.rgl.args=list(),
      plot_GOF=FALSE,
      user.gof.args=list(NULL),
      ...)
```

Arguments

x	forestFloor" class object
Xi	integer vector of length 2 indices of feature columns
FCi	integer vector of length 1 to p variables indices of feature contributions columns
col	a colour vector. One colour or colour palette(vector).
plotTest	NULL(plot by test set if available), TRUE(plot by test set), FALSE(plot by train), "andTrain"(plot by both test and train)
orderByImportance	should indices order by 'variable importance' or by matrix/data.frame order?
surface	should a surface be plotted also?

`combineFC` a row function applied on selected columns(FCi) on $FCmatrix or $FCarray. How should feature contributions be combined? Default is `sum`.

`zoom` grid can be expanded in all directions by a factor

`grid.lines` how many grid lines should be used. Total surface anchor points in plot is grid.lines^2. May run slow above 200-500 depending on hardware.

`limit` a number. Sizing of grid does not consider outliers outside this limit of e.g. 3 SD deviations univariately.

`cropPointsOutSideLimit`
#if points exceed standard deviation limit, they will not be plotted

`kknnGrid.args` argument list, any possible arguments to kknnkknn
These default wrapper arguments can hereby be overwritten:
wrapper = alist(formula=fc~., # do not change
train=Data, # do not change
k=k, # integer < n_observations. k>100 may run slow.
kernel="gaussian", #distance kernel, other is e.g. kernel="triangular"
test=gridX #do not change
)
see kknnkknn to understand parameters. k is set by default automatically to a half times the square root of observations, which often gives a reasonable balance between robustness and adeptness. k neighbors and distance kernel can be changed be passing kknnGrid.args = alist(k=5,kernel="triangular",scale=FALSE), hereby will default k and default kernel be overwritten. Moreover the scale argument was not specified by this wrapper and therefore not conflicting, the argument is simply appended.

`plot.rgl.args` pass argument to rgl::plot3d, can override any argument of this wrapper, defines plotting space and plot points. See plot3d for documentation of graphical arguments.
wrapper_arg = alist(x=xaxis, #do not change, x coordinates
y=yaxis, #do not change, y coordinates
z=zaxis, #do not change, z coordinates
col=col, #colouring evaluated within this wrapper function
xlab=names(X)[1], #xlab, label for x axis
ylab=names(X)[2], #ylab, label for y axis
zlab=paste(names(X[,FCi]),collapse=" - "), #zlab, label for z axis
alpha=.4, #points transparency
size=3, #point size
scale=.7, #z axis scaling
avoidFreeType = T, #disable freeType=T plug-in. (Postscript labels)
add=FALSE #do not change, should graphics be added to other rgl-plot?
)

`surf.rgl.args` wrapper_arg = alist(x=unique(grid[,2]), #do not change, values of x-axis
y=unique(grid[,3]), #do not change, values of y-axis
z=grid[,1], #do not change, response surface values
add=TRUE, #do not change, surface added to plotted points
alpha=0.4 #transparency of surface, [0;1]
)

	see rgl::persp3d for other graphical arguments notice the surface is added onto plotting of points, thus can e.g. labels not be changed from here.
`label.seq`	a numeric vector describing which classes and in what sequence to plot. NULL is all classes ordered is in levels in x$Y of forestFloor_mulitClass object x.
`user.gof.args`	argument list passed to internal function ff2, which can modify how goodness-of-fit is computed. Number of neighbors and kernel can be set manually with e.g. list(kmax=40,kernel="gaussian"). Default pars should work already in most cases. Function ff2 computed leave-one-out CV prediction the feature contributions from the chosen context of the visualization.
`plot_GOF`	Boolean TRUE/FALSE. Should the goodness of fit be computed and plotted is main of 3D plot? If false, no GOF input pars are useful.
`...`	not used at the moment

Details

show3d plot one or more combined feature contributions in the context of two features with points representing each data point. The input object must be a "forestFloor_regression" or "forestFloor_multiClass" S3 class object , and should at least contain $X the data.frame of training data, $FCmatrix the feature contributions matrix. Usually this object are formed with the function forestFloor having a random forest model fit as input. Actual visualization differs for each class.

Value

no value

Author(s)

Soren Havelund Welling

Examples

```
## Not run:
## avoid testing of rgl 3D plot on headless non-windows OS
## users can disregard this sentence.
if(!interactive() && Sys.info()["sysname"]!="Windows") skipRGL=TRUE

library(forestFloor)
library(randomForest)
#simulate data
obs=2500
vars = 6

X = data.frame(replicate(vars,rnorm(obs)))
Y = with(X, X1^2 + sin(X2*pi) + 2 * X3 * X4 + 1 * rnorm(obs))


#grow a forest, remeber to include inbag
rfo=randomForest(X,Y,keep.inbag = TRUE,sampsize=1500,ntree=500)

#compute topology
```

```
ff = forestFloor(rfo,X)


#print forestFloor
print(ff)

#plot partial functions of most important variables first
plot(ff)

#Non interacting functions are well displayed, whereas X3 and X4 are not
#by applying different colourgradient, interactions reveal themself
Col = fcol(ff,3)
plot(ff,col=Col)

#in 3D the interaction between X3 and X reveals itself completely
show3d(ff,3:4,col=Col,plot.rgl=list(size=5))

#although no interaction, a joined additive effect of X1 and X2
Col = fcol(ff,1:2,X.m=FALSE,RGB=TRUE) #colour by FC-component FC1 and FC2 summed
plot(ff,col=Col)
show3d(ff,1:2,col=Col,plot.rgl=list(size=5))

#...or two-way gradient is formed from FC-component X1 and X2.
Col = fcol(ff,1:2,X.matrix=TRUE,alpha=0.8)
plot(ff,col=Col)
show3d(ff,1:2,col=Col,plot.rgl=list(size=5))

## End(Not run)
```

`vec.plot` *Compute and plot vector effect characteristics for a given multivariate model*

Description

vec.plot visualizes the vector effect characteristics of a given model. Geometrically it corresponds to a specific 2D or 3D slice of a higher dimensional mapping structure. One variable (2D plot) or two variables (3D plot) are screened within the range of the training data, while remaining variables are fixed at the univariate means (as default). If remaining variables do not interact strongly with plotted variable(s), vec.plot is a good tool to break up a high-dimensional model structure into separate components.

Usage

```
vec.plot(model,X,i.var,grid.lines=100,VEC.function=mean,
         zoom=1,limitY=F,moreArgs=list(),...)
```

Arguments

`model`	model_object which has a defined method predict.model, which can accept arguments as showed for randomForest e.g. library(randomForest) model = randomForest(X,Y) predict(model,X) where X is the training features and Y is the training response vector(numeric)
`X`	matrix or data.frame being the same as input to model
`i.var`	vector, of column_numbers of variables to scan. No plotting is available for more than two variables.
`grid.lines`	scalar, number of values by each variable to be predicted by model. Total number of combinations = grid.lines^length(i_var).
`VEC.function`	function, establish one fixed value for any remaining variables(those not chosen by i.var). Default is to use the mean of variables.
`zoom`	scalar, number defining the size.factor of the VEC.surface compared to data range of scanned variables. Bigger number is bigger surface.
`limitY`	boolean, if TRUE Y-axis is standardized for any variable. Useful for composite plots as shown in example.
`moreArgs`	any lower level graphical args passed to rgl::surface3d or points depending on number of variables(length of i.var)
`...`	any lower level graphical args passed to rgl::plot3d or plot depending on number of variables(length of i.var)

Details

vec.plot visualizes the vector effect characteristics of a given model. One(2D plot) or two(3D plot) variables are screened within the range of the training data, while remaining variables are fixed at the univariate means of each them(as default). If remaining variables do not interact strongly with plotted variable(s), vec.plot is a good tool to break up a high-dimensional model topology in separate components.

Value

no value

Author(s)

Soren Havelund Welling

Examples

```
## Not run:
## avoid testing of rgl 3D plot on headless non-windows OS
## users can disregard this sentence.
if(!interactive() && Sys.info()["sysname"]!="Windows") skipRGL=TRUE
library(randomForest)
library(forestFloor)

#simulate data
```

```
obs=2000
vars = 6
X = data.frame(replicate(vars,rnorm(obs)))
Y = with(X, X1^2 + 2*sin(X2*pi) + 2 * X3 * (X4+.5))
Yerror = 1 * rnorm(obs)
var(Y)/var(Y+Yerror)
Y= Y+Yerror

#grow a forest, remeber to include inbag
rfo2=randomForest(X,Y,keep.inbag=TRUE,sampsize=800)

#plot partial functions of most important variables first
pars=par(no.readonly=TRUE) #save previous graphical paremeters
par(mfrow=c(2,3),mar=c(2,2,1,1))
for(i in 1:vars) vec.plot(rfo2,X,i,zoom=1.5,limitY=TRUE)
par(pars) #restore

#plot partial functions of most important variables first
for(i in 1:vars) vec.plot(rfo2,X,i,zoom=1.5,limitY=TRUE)

#plotvariable X3 and X4 with vec.plot
Col = fcol(X,3:4)
vec.plot(rfo2,X,3:4,zoom=1,grid.lines=100,col=Col)

## End(Not run)
```

`Xtestmerger` *merge training set (X) and (test) set*

Description

... and expand inbag matrix and training target vector to compute FC for a test set.

Usage

```
Xtestmerger(X,test,inbag=NULL,y=NULL)
```

Arguments

`X`	X , training set data.frame used to train a random forest model
`test`	test, a test set data.frame which feature contributions should be computed for
`inbag`	matrix of inbag sampling to expande with training set, which is set OOB for any tree
`y`	random forest target vector, which is set to first value for observation

Details

Xtestmerger is a low-level function to merge a test set with X training set. There can be no names, column class, column number mismatch. Moreover any level in any factor of test must be present in X, as RF/forestFloor cannot score a unknown factor level / category.

Value

List of merged bigX, bigInbag and bigy. The two latter may be NULL if not provided.

Author(s)

Soren Havelund Welling

Examples

```
library(randomForest)
library(forestFloor)
#X y could be a training set
X = data.frame(numeric = c(1,5,2,7,-4.3),
               factor1 = factor(c("jim","freddy","marley","marley","alfred")),
               factor2 = factor(c("jill","ann","liz","leila","vicky")))
y = factor(1:5)
set.seed(1)
rf = randomForest(X,y,keep.inbag=TRUE,ntree=7)
#should not raise any error
test = data.frame(numeric = rnorm(5),
                  factor1 = factor(c("jim","jim","jim","freddy","freddy")),
                  factor2 = factor(c("jill","jill","vicky","leila","vicky"))
                  )
out = Xtestmerger(X,test,inbag=rf$inbag,y=y)
```

Index

Bibliography

[AA93] Eva Karin Anderberg and Per Artursson. "Epithelial transport of drugs in cell culture. VIII: Effects of sodium dodecyl sulfate on cell membrane and tight junction permeability in human intestinal epithelial (Caco-2) cells". In: *Journal of pharmaceutical sciences* 82.4 (1993), pages 392–398.

[Abu12] Yasar Abu-Mostafa. *Learning from Data: The learning problem*. August 2012. URL: `http://work.caltech.edu/lectures.html`.

[Agu+13] Florencia Aguiree et al. "IDF diabetes atlas". In: (2013).

[Agu+16] TAS Aguirre et al. "Current status of selected oral peptide technologies in advanced preclinical development and in clinical trials". In: *Advanced drug delivery reviews* (2016).

[AM90] Per Artursson and Christine Magnusson. "Epithelial transport of drugs in cell culture. II: Effect of extracellular calcium concentration on the paracellular transport of drugs of different lipophilicities across monolayers of intestinal epithelial (Caco-2) cells". In: *Journal of pharmaceutical sciences* 79.7 (1990), pages 595–600.

[ANA92] Eva Karin Anderberg, Christer Nyström, and Per Artursson. "Epithelial transport of drugs in cell culture. VII: Effects of pharmaceutical surfactant excipients and bile acids on transepithelial permeability in monolayers of human intestinal epithelial (Caco-2) cells". In: *Journal of pharmaceutical sciences* 81.9 (1992), pages 879–887.

[Arn+10] Sabine Arnolds et al. "How pharmacokinetic and pharmacodynamic principles pave the way for optimal basal insulin therapy in type 2 diabetes". In: *International journal of clinical practice* 64.10 (2010), pages 1415–1424.

[Art+94] Per Artursson et al. "Effect of chitosan on the permeability of monolayers of intestinal epithelial cells (Caco-2)". In: *Pharmaceutical research* 11.9 (1994), pages 1358–1361.

[Ber98] Andreas Bernkop-Schnürch. "The use of inhibitory agents to overcome the enzymatic barrier to perorally administered therapeutic peptides and proteins". In: *Journal of controlled release* 52.1 (1998), pages 1–16.

[BML13] Benjamin J Bruno, Geoffrey D Miller, and Carol S Lim. "Basics and recent advances in peptide and protein drug delivery". In: *Therapeutic delivery* 4.11 (2013), pages 1443–1467.

[Bou+11] Rémy Boussageon et al. "Effect of intensive glucose lowering treatment on all cause mortality, cardiovascular death, and microvascular events in type 2 diabetes: meta-analysis of randomised controlled trials". In: *Bmj* 343 (2011), page d4169.

[Bou+14] Anne-Laure Boulesteix et al. "Letter to the Editor: On the term 'interaction'and related phrases in the literature on Random Forests". In: *Briefings in bioinformatics* (2014), bbu012.

[Bre+84] Leo Breiman et al. *Classification and regression trees*. CRC press, 1984.

[Bre01] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pages 5–32.

[Bre12] Sarah Øster Brebbia. *Application of Text Analytics and Multivariate Statistics for Characterization of Absorption Enhancers - Introducing the Drug Delivery Problem*. Thesis. Supervised by Professor Per Brockhoff, pbb@imm.dtu.dk, DTU Informatics. Thesis not publicly available. Asmussens Alle, Building 305, DK-2800 Kgs. Lyngby, Denmark, 2012. URL: `http://www.imm.dtu.dk/English.aspx`.

[CL04] Andrea Caumo and Livio Luzi. "First-phase insulin secretion: does it exist in real life? Considerations on shape and function". In: *American Journal of Physiology-Endocrinology And Metabolism* 287.3 (2004), E371–E385.

[CM99] Gerardo P Carino and Edith Mathiowitz. "Oral insulin delivery". In: *Advanced drug delivery reviews* 35.2 (1999), pages 249–257.

[Cor+95] Bernard Corvilain et al. "Effect of short-term starvation on gastric emptying in humans: relationship to oral glucose tolerance". In: *American Journal of Physiology-Gastrointestinal and Liver Physiology* 269.4 (1995), G512–G517.

[Cow90] Stachura ME Cowart SL. "Glucosuria". In: *Clinical Methods: The History, Physical, and Laboratory Examinations. 3rd Edition*. Edited by Hurst JW Walker HK Hall WD. Boston: Butterworths, 1990. Chapter 139.

[Del04] John S Delaney. "ESOL: estimating aqueous solubility directly from molecular structure". In: *Journal of chemical information and computer sciences* 44.3 (2004), pages 1000–1005.

[FHT01] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. 3rd edition. Volume 1. Springer series in statistics Springer, Berlin, 2001.

[Fri01] Jerome H Friedman. "Greedy function approximation: a gradient boosting machine". In: *Annals of statistics* (2001), pages 1189–1232.

[FUJ+85] SETSURO FUJII et al. "Promoting effect of the new chymotrypsin inhibitor FK-448 on the intestinal absorption of insulin in rats and dogs". In: *Journal of pharmacy and pharmacology* 37.8 (1985), pages 545–549.

[Gab+10] Franz Gabor et al. "Improving oral delivery". In: *Drug Delivery*. Springer, 2010, pages 345–398.

[Gæd+08] Peter Gæde et al. "Effect of a multifactorial intervention on mortality in type 2 diabetes". In: *New England Journal of Medicine* 358.6 (2008), pages 580–591.

[Gar+13] Luis-Emilio García-Pérez et al. "Adherence to therapies in patients with type 2 diabetes". In: *Diabetes Therapy* 4.2 (2013), pages 175–194.

[Har+84] Frank E Harrell et al. "Regression modelling strategies for improved prognostic prediction". In: *Statistics in medicine* 3.2 (1984), pages 143–152.

[Hol+08] Rury R Holman et al. "10-year follow-up of intensive glucose control in type 2 diabetes". In: *New England Journal of Medicine* 359.15 (2008), pages 1577–1589.

[Hua+14] Elbert S Huang et al. "Rates of complications and mortality in older patients with diabetes mellitus: the diabetes and aging study". In: *JAMA internal medicine* 174.2 (2014), pages 251–258.

[Joh+15] Gareth Johnson et al. "Keeping Your Thesis Legal". In: (2015).

[Kah11] Daniel Kahneman. *Thinking, fast and slow*. Macmillan, 2011.

[Kli+16] Kyrylo Klimenko et al. "Novel enhanced applications of QSPR models: Temperature dependence of aqueous solubility". In: *Journal of Computational Chemistry* (2016).

[Kor02] M Korytkowski. "When oral agents fail: practical barriers to starting insulin." In: *International Journal of Obesity & Related Metabolic Disorders* 26 (2002).

[Krs+14] Damjan Krstajic et al. "Cross-validation pitfalls when selecting and assessing regression and classification models". In: *Journal of cheminformatics* 6.1 (2014), page 1.

[Kuh15] Max Kuhn. "A Short Introduction to the caret Package". In: *R Project website. cran. r-project. org/web/packages/caret/vignettes/caret. pdf. Published August* 6 (2015).

[Le 10] Jean-Yves Le Boudec. *Performance evaluation of computer and communication systems*. Epfl Press, 2010. Chapter 2.

[Lic13] M. Lichman. *UCI Machine Learning Repository*. 2013. URL: `http://archive.ics.uci.edu/ml`.

[LLT99] Dong-Zhou Liu, Edward L Lecluyse, and Dhiren R Thakker. "Dodecylphosphocholine-mediated enhancement of paracellular permeability and cytotoxicity in Caco-2 cell monolayers". In: *Journal of pharmaceutical sciences* 88.11 (1999), pages 1161–1168.

[LS00] Yong-Hee Lee and Patrick J Sinko. "Oral delivery of salmon calcitonin". In: *Advanced drug delivery reviews* 42.3 (2000), pages 225–238.

[Mah+09] Sam Maher et al. "Safety and efficacy of sodium caprate in promoting oral drug absorption: from in vitro to the clinic". In: *Advanced drug delivery reviews* 61.15 (2009), pages 1427–1449.

[Mah+14] Sam Maher et al. "Formulation strategies to improve oral peptide delivery". In: *Pharmaceutical patent analyst* 3.3 (2014), pages 313–336.

[MKD80] SNS Murthy, Jay Kostman, and Vicente P Dinoso Jr. "Effect of pH, substrate, and temperature on tryptic activity of duodenal samples". In: *Digestive diseases and sciences* 25.4 (1980), pages 289–294.

[Naw+11] Thomas Nawroth et al. "Liposome formation from bile salt–lipid micelles in the digestion and drug delivery model fassifmod estimated by combined time-resolved neutron and dynamic light scattering". In: *Molecular pharmaceutics* 8.6 (2011), pages 2162–2172.

[Pal+07] David S Palmer et al. "Random forest models to predict aqueous solubility". In: *Journal of chemical information and modeling* 47.1 (2007), pages 150–158.

[Pet+13] Signe Beck Petersen et al. "Colonic absorption of salmon calcitonin using tetradecyl maltoside (TDM) as a permeation enhancer". In: *European Journal of Pharmaceutical Sciences* 48.4 (2013), pages 726–734.

[RK12] Milton J Rosen and Joy T Kunjappu. *Surfactants and interfacial phenomena*. John Wiley & Sons, 2012.

[Ser15] Yanir Seroussi. *10 Steps to Success in Kaggle Data Science Competitions*. March 2015. URL: `http://www.kdnuggets.com/2015/03/10-steps-success-kaggle-data-science-competitions.html`.

[Sil10] Dee Silverthorn. *Human physiology : an integrated approach*. San Francisco: Pearson/Benjamin Cummings, 2010. ISBN: 978-0-321-55980-7.

[TR06] Thomas N Tozer and Malcolm Rowland. *Introduction to pharmacokinetics and pharmacodynamics: the quantitative basis of drug therapy*. Lippincott Williams & Wilkins, 2006.

[TT08] Timothy K Tippin and Dhiren R Thakker. "Biorelevant refinement of the Caco-2 cell culture model to assess efficacy of paracellular permeability enhancers". In: *Journal of pharmaceutical sciences* 97.5 (2008), pages 1977–1992.

[VKB99] Brent Vernon, Sung Wan Kim, and You Han Bae. "Insulin release from islets of Langerhans entrapped in a poly (N-isopropylacrylamide-co-acrylic acid) polymer gel". In: *Journal of Biomaterials Science, Polymer Edition* 10.2 (1999), pages 183–198.

[Wel+14] Søren H. Welling et al. "The role of citric acid in oral peptide and protein formulations: Relationship between calcium chelation and proteolysis inhibition". In: *European Journal of Pharmaceutics and Biopharmaceutics* 86.3 (2014), pages 544–551. ISSN: 0939-6411. DOI: http://dx.doi.org/10.1016/j.ejpb.2013.12.017. URL: http://www.sciencedirect.com/science/article/pii/S0939641113003998.

[Wel+16] Soeren H Welling et al. "Forest Floor Visualizations of Random Forests". In: *arXiv preprint arXiv:1605.09196* (2016).

[Wik16a] Wikipedia. *Lucas critique — Wikipedia, The Free Encyclopedia*. [Online; accessed 17-July-2016]. 2016. URL: https://en.wikipedia.org/w/index.php?title=Lucas_critique&oldid=729149918.

[Wik16b] Wikiquote. *John von Neumann — Wikiquote,* [Online; accessed 12-July-2016]. 2016. URL: https://en.wikiquote.org/w/index.php?title=John_von_Neumann&oldid=2132483.

[WKM08] Kathryn Whitehead, Natalie Karr, and Samir Mitragotri. "Safe and effective permeation enhancers for oral drug delivery". In: *Pharmaceutical research* 25.8 (2008), pages 1782–1788.

[XO00] Wei J Xia and Hayat Onyuksel. In: *Pharmaceutical research* 17.5 (2000), pages 612–618.