Responsible Machine Learning with Insurance Applications

Christian Lorentzen & Michael Mayer

Autumn 2022

Table of Contents

Organisation

Audience

- master students
- SAV students

Setting

- 13 lectures
- exercises are integrated in lectures
- oral exam

Prerequisites

- statistics & probability theory
- machine learning & supervised learning

Responsible ML

Why responsible? Common risks are:

- Model does not solve the original goal or task.
- Missing understanding of the given data.
- Bias (model & data).
- Wrong claims about the ability of a model.

"Statistics is about the honest interpretation of data." (Prof. Simon N. Wood)

This lecture aims to provide a toolbox.

- Model Comparison and Calibration
- Explainability

Not in this lecture:

- Data protection law
- Ethical questions
- (AI) Fairness
- MLOps

Applications of ML Models

Actuarial ML models:

- Pricing models for pure premium and profitability
- Reserving models for the ultimate claim costs (RBNS and IBNR)
- Mortality rates / Life tables
- Fraud detection
- Conversion and lapse rate
- ...

Most methodology works for any kind of model for example:

- Natural catastrophe (NatCat) models for annual loss
- Risk models for loss distribution of the company

Decisions are based on actuarial models.

Some Lessons from History

- IBM Watson image recognition (2015): Predicted tag could initially contain unethical associations ("looser").
- Microsoft's Twitter chatbot Tay (2016) gave inflammatory answers like an extremist.
- Google's neural machine translation (2018) gender bias: "doctor" is assigned a *he*, "lazy" is a *she*.
- Apple Card 2019 did provide men higher credit limits than women despite gender not being used by the model.

Measurement

Time

Distance

Velocity

Measurement

Time

Distance

Velocity

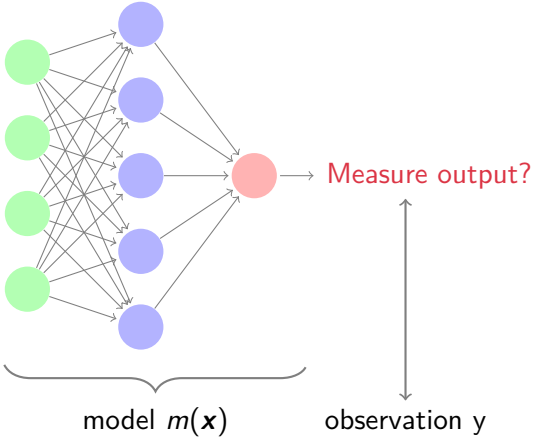
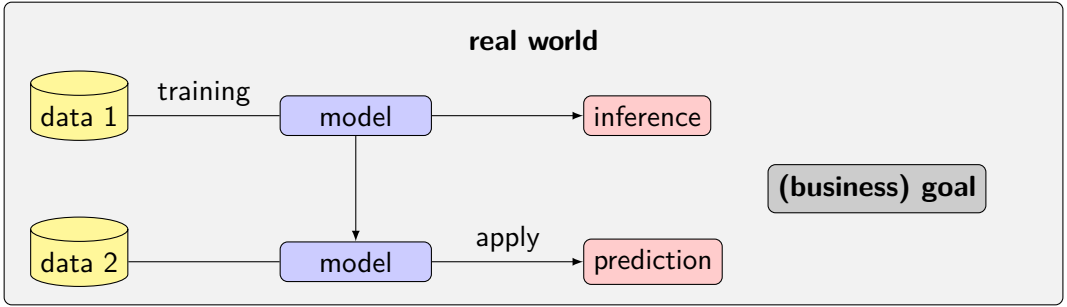


Table of Contents

Picture of ML



Goal of a model

- **inference**—on observations/seen data
- **prediction**—on new, unseen data

Supervised Statistical . . .

Data at population level

- Features $\boldsymbol{X}$ take values in some possibly high dimensional feature space $\mathcal{X}$ such as $\mathbb{R}^p$.
- Output / response / target variable Y takes values in some space $\mathcal{Y}$, which we assume to be a subset of $\mathbb{R}$.

Remark

We consider both $\boldsymbol{X}$ and Y to be *random variables* with joint probability distribution $F_{\boldsymbol{X},Y}$.

Note

In practice, however, $F_{\boldsymbol{X},Y}$ is usually unknown.

... Learning I

Prediction Goal

- **Probabilistic** predictions aim for $F_{Y|\boldsymbol{X}}$.
- **Point** predictions aim for a property / (target) functional $T(F_{Y|\boldsymbol{X}})$.

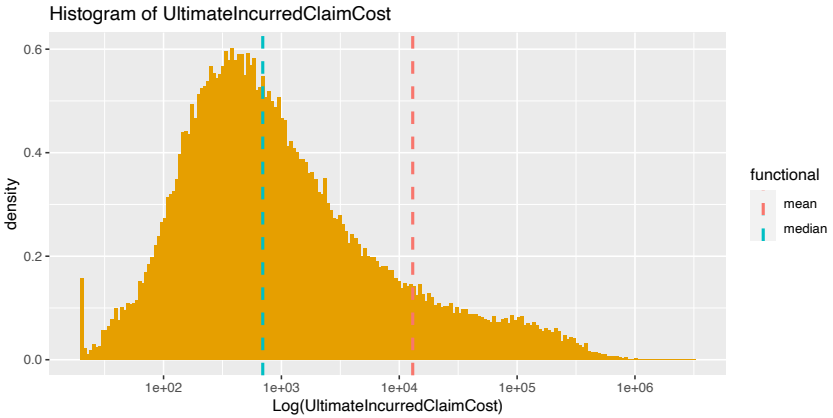
Remark

Y is random, there is no deterministic function $Y = g(\boldsymbol{X})$.
Convention: $T(F_{Y|\boldsymbol{X}}) = T(Y|\boldsymbol{X})$

Example

- expectation $= \frac{1}{2}$-expectile, $T(Y|\boldsymbol{X}) = \mathbb{E}[Y|\boldsymbol{X}]$
- median $= \frac{1}{2}$-quantile
- α-expectile $T(Y|\boldsymbol{X}) = e_\alpha(Y|\boldsymbol{X})$
  solution e of $\alpha \int_e^\infty (y-e)\,\mathrm{d}F(y) = (1-\alpha)\int_{-\infty}^e (e-y)\,\mathrm{d}F(y)$
- α-quantile $T(Y|\boldsymbol{X}) = q_\alpha(Y|\boldsymbol{X})$, any q with $\lim_{y\uparrow q} F(y) \le \alpha \le F(q)$
  lower quantile $q_\alpha(Y|\boldsymbol{X}) = \inf\{t \in \mathbb{R} \mid F_{Y|\boldsymbol{X}}(t) \ge \alpha\}$

Data Set



Workers Compensation data set `https://www.openml.org/d/42876`

... Learning II

Model

We want to find a model $m \in \mathcal{M}$, from some model class $\mathcal{M}$, to predict $T(Y|\boldsymbol{X})$ by $m(\boldsymbol{X})$.

Loss/Scoring function

- We need a **loss** or **scoring function** S to measure the deviation of the model prediction $m(\boldsymbol{X})$ from T using observations Y: $S(m(\boldsymbol{X}), Y)$.
- Convention: The smaller S, the better.
- For model training as well as model comparison.

Example

- squared error $S(z, y) = (z - y)^2$
- absolute error $S(z, y) = |z - y|$

Iterative Optimisation (boosting, GD)

- $\overline{S}(m) = \sum_i S(m(\boldsymbol{x}_i), y_i)$
- $m_{j+1} \approx \arg\min_{m \in \mathcal{M}} \underbrace{\overline{S}(m) - \overline{S}(m_j)}_{\text{model comparison}}$

Statistical Risk

Statistical risk

The *statistical risk* of model m (under distribution $F_{Y,\boldsymbol{X}}$):

$$R(m) = \mathbb{E}\left[S(m(\boldsymbol{X}), Y)\right] = \mathbb{E}\left[\,\mathbb{E}\left[S(m(\boldsymbol{X}), Y)|\boldsymbol{X}\right]\right] \tag{1}$$

Ideal model / Bayes rule

$$m^\star = \arg\min_{m \in \mathcal{M}} R(m) \tag{2}$$

Note

- Use S such that $m^\star = T(Y|\boldsymbol{X}) \Rightarrow$ *consistency*
- $F_{Y,\boldsymbol{X}}$ and therefore $R(m)$ usually not known.
- Neither existence nor uniqueness of $m^\star$ are guaranteed.
- $m = Y$ is almost surely not $m^\star$.

Supervised Learning at Sample Level

Data sample

- Observations of i.i.d. input-output pairs $(\boldsymbol{x}_i, y_i)$, $i = 1, \ldots, n$
  sample of observed y available $\Rightarrow$ term *supervised*
- $D = \{(\boldsymbol{x}_i, y_i), i = 1 \ldots n\}$

Workers Compensation dataset `https://www.openml.org/d/42876`

$y =$ UltimateIncurredClaimCost	InitialCaseEstimate	Age	Gender	WeeklyPay
102	9500	45	M	500
493	1000	18	F	373

Note

- Most results are valid without i.i.d. assumption, e.g. forecast comparison explicitly comes from time series.
- It is good practise to know your data well $\Rightarrow$ exploratory data analysis (EDA).

Empirical Risk

With the inaccessibility of $R(m)$, one resorts to an estimation by given sample data D.

Empirical risik

$$\overline{R}(m; D) = \overline{S}(m; D) = \frac{1}{n} \sum_{(\boldsymbol{x}_i, y_i) \in D} S(m(\boldsymbol{x}_i), y_i) \tag{3}$$

Empirical risk minimisation (ERM) is the actual learning/training step:

$$\widehat{m} = \widehat{m}(\,\cdot\,; D_{\text{train}}) = \underset{m \in \mathcal{M}}{\arg\min}\, \overline{R}(m; D_{\text{train}}) \tag{4}$$

Core of the Learning Problem

1. Estimation error

For small training sample size, $\widehat{m}$ has a high (sample) uncertainty.

2. In-Samples risk is biased

The *in-sample risk* or *training loss* $\overline{R}(\widehat{m}(\,\cdot\,; D_{\text{train}}); D_{\text{train}})$, obtained from training and evaluating $\widehat{m}$ on the same data D_{train}, is a biased estimate for $R(\widehat{m})$, usually over-optimistic.

Definition (Overfitting)

Model $m \in \mathcal{M}$ overfits (w.r.t. model complexity given by Ω) the training data D_{train} if there exists another model $m' \in \mathcal{M}$ with $\Omega(m') < \Omega(m)$ such that $\overline{R}(m; D_{\text{train}}) \leq \overline{R}(m'; D_{\text{train}})$, but $R(m) > R(m')$.

Therefore, we always include the trivial model in $\mathcal{M}$.

Mitigating Overfitting

Adding a penalty

Add a penalty Ω, accounting for model complexity/capacity:

$$\arg\min_{m \in \mathcal{M}} \overline{R}(m; D_{\text{train}}) + \lambda\Omega\,. \tag{5}$$

Ridge regression

$\arg\min_{\beta \in \mathbb{R}} \frac{1}{n} \sum_{(\boldsymbol{x}_i, y_i) \in D} (\boldsymbol{x}_i \beta - y_i)^2 + \lambda \|\beta\|_2^2$

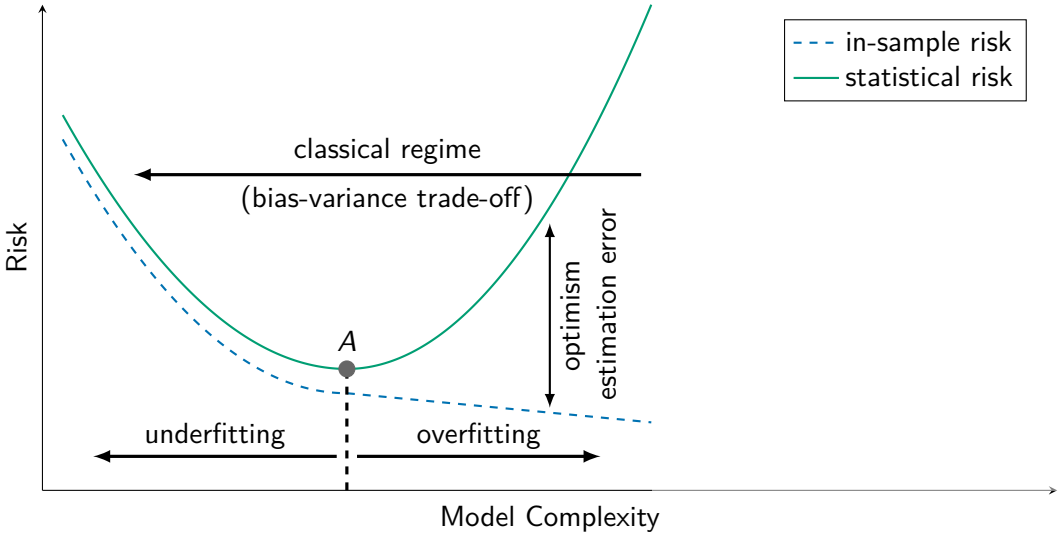
Out-of-sample evaluation

Monitor the *out-of-sample* risk on (ideally)*independent* (and identically distributed) test or validation data D_{test}

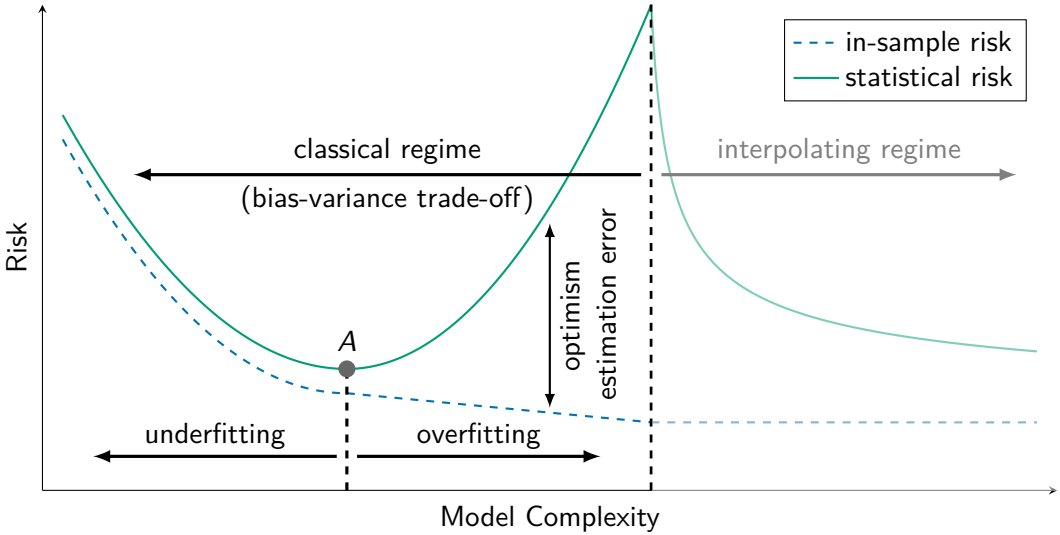
$$\overline{R}(\widehat{m}(\,\cdot\,; D_{\text{train}}); D_{\text{test}})\,. \tag{6}$$

$\Rightarrow$ need more data!

Statistical Risk vs In-Sample Risk



Statistical Risk vs In-Sample Risk



Decomposition of the Statistical Risk

$$
\begin{aligned}
R(m) = & \inf_{g:\mathcal{X}\to\mathcal{Y}} R(g) && \text{inherent unpredictability} \\
& + \inf_{f\in\mathcal{M}} R(f) \quad - \inf_{g:\mathcal{X}\to\mathcal{Y}} R(g) && \text{approximation error} \\
& + \inf_{f\in\mathcal{M}} \overline{R}(f;D) - \inf_{f\in\mathcal{M}} R(f) && \text{estimation error I} \\
& + \overline{R}(m;D) \quad - \inf_{f\in\mathcal{M}} \overline{R}(f;D) && \text{optimisation error} \\
& + R(m) \quad - \overline{R}(m;D) && \text{estimation error II}
\end{aligned}
\tag{7}
$$

Data Split

Train-Validation-Test-Application Split

- **Training set** for model fitting, typically the largest set.
- **Validation set** for model comparison and model selection. Typically, this set is used to tune a model of a given model class while building (fitting) models on the training set.[1] The result is a "final" model for the given model class that is often refit on the joint training and validation sets.
- **Test set** for assessment and comparison of final models. Once the model building phase is finished, this set is used to calculate an unbiased estimate of the statistical risk. It may be used to pick the best one of the (few) final models.
- **Application set**. This is the data the model is used for in production. It consists of feature variables only. If the observations of the response become known after a certain time delay, it can serve to monitor the performance of the model.

[1] Examples are variable selection and specification of terms for linear models, finding optimal architecture and early stopping for neural nets, hyperparameter tuning of boosted trees. This way, the validation set is heavily used and therefore does not provide an unbiased performance estimate anymore.

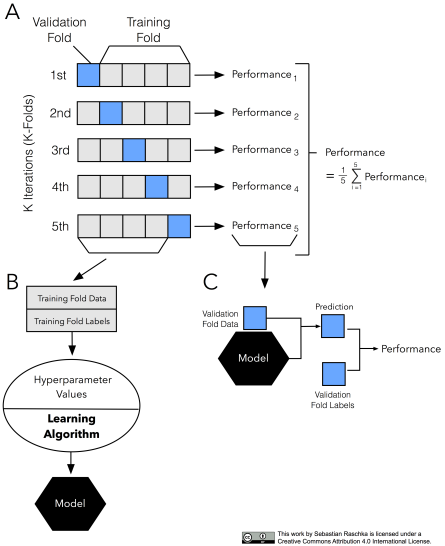
Advice for Data Splits

- *Never ever* look at the test set while still training models.
- The more you use a data set, e.g., the validation set, the less reliable are the results. (analogy: data the new oil $\Rightarrow$ data can be burnt)

Note

A methodological sound train-validation-test split and usage pattern is essential for building good models and for an unbiased assessment of predictive performance.

Cross-Validation



source:

https://sebastianraschka.com/blog/2016/model-evaluation-selection-part3.html

Cross-Validation

CV

- Divide training data into $k = 1, \ldots, K$ disjoint folds D^k_{train}.
- Train model $m_k(\cdot; D^{-k}_{\text{train}})$ on all folds but the kth fold, i.e. on $D^{-k}_{\text{train}} = \bigcup_{j \neq k}^{K} D^j_{\text{train}}$
- Evaluate m_k on fold k as validation set: $\overline{S}_k = \overline{S}(m_k, D^k_{\text{train}})$.
- Repeat for all k and average the scores: $\overline{S}_{CV} = \frac{1}{K} \sum_k \overline{S}_k$.

Disadvantage

This is an average of the scores of K different models m_k and estimates the expected score over all training sets $\mathbb{E}_{D_{\text{train}}}[R(m(\cdot; D_{\text{train}}))] = \mathbb{E}_{D_{\text{train}}}[\mathbb{E}[S(m(\boldsymbol{X}; D_{\text{train}}), Y)]]$.

Further points

- Many different splitting schemes
- Account for (dependency) structure in the data
- Possibly different weighting in the final score average

CV Splitting Schemes

Simple splitting schemes

- K-fold CV: Divide training data into $K \approx 5 \dots 10$ equally large folds.
- Leave-one-out (LOO): Divide into $K = N$ folds, i.e. only one observation in validation set.
  This enables fast algorithms for linear models (and linear smoothers), but is almost infeasible for other model classes.
- Leave-n-out

Ensuring identical distribution

- Random shuffling might help to make the folds more identically distributed.
- Stratifies sampling, in particular for classification problems.
  But this might generate dependencies between train and validation set!

CV Data Dependencies

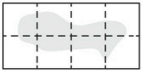
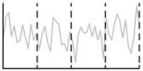
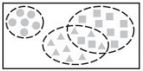
Dependence structure	Parametric solution	Blocking	Blocking illustration
Spatial	Spatial models (e.g.CAR, INLA, GWR)	Spatial	
Temporal	Time-series models (e.g.ARIMA)	Temporal	
Grouping	Mixed effect models (e.g. GLMM)	Group	
Hierarchical / Phylogenetic	Phylogenetic models (e.g. PGLS)	Hierarchical	

Figure: source: doi:10.1111/ecog.02881

Data dependencies

- Blocking strategy / Grouped sampling: Same claim or customer ID should only be in one single fold to prevent data leakage from D^{-k} into D^k.
- (Spatio-) Temporal structure: Usual assumption is that correlation reduces with (spatio/temporal) distance. For time series: out-of-time or forward-validation scheme:
  - Forecasting horizon: 1-time-step or k-time-steps ahead
  - Fixed vs rolling origin
  - Fixed vs rolling windows

Learning Recipe

Ingredients

- Data sample $D = \{(\boldsymbol{x}_i, y_i), i = 1 \dots n\}$
- Chose target functional T.
- Chose a model class $\mathcal{M}$.
- Chose a *loss/scoring function* $S(z_i, y_i) \in \mathbb{R}^p$.

Learn from training data

- Split D into training and validation/test data.
- Train $\widehat{m}$ on D_{train}.
- Evaluate on D_{test}

Loss/Scoring function

The loss function should be chosen in line with the directive T in that it should be *strictly consistent* for T.

Excercises I

Exercise 1

Proof the following properties of the expectile $e_\alpha(X)$, $X \sim F$:

- $e_\alpha(aX + b) = ae_\alpha(X) + b$ for $a, b \in \mathbb{R}$
- $e_\alpha(-X) = -e_{1-\alpha}(X)$

Exercise 2

Given continuous F with finite $\mathbb{E}[Y]$, calculate the Bayes rule for

- Bregman function $S(z, y) = \phi(y) - \phi(z) + \phi'(z)(z - y) + a(y)$ with (strictly) convex ϕ, finite $\mathbb{E}[\phi(Y)]$ and arbitrary a.
- pinball loss $S(z, y) = (\mathbb{1}\{z \geq y\} - \alpha)(z - y)$
- asymmetric piecewise quadratic scoring function (APQSF) $S(z, y) = |\mathbb{1}\{z \geq y\} - \alpha|(z - y)^2$ with finite 2nd moment of F.

Excercises II

Exercise 3

Define optimism as $\mathrm{op} = \mathrm{Err}_{in} - \overline{R}_{in}$ with in-sample risk $\overline{R}_{in} = \overline{R}(\widehat{m}(\,\cdot\,; D_{\text{train}}); D_{\text{train}})$ and *in-sample error* $\mathrm{Err}_{in} = \frac{1}{n}\sum_{(\boldsymbol{x}_i)\in D} \mathbb{E}_{Y_i^0|X=\boldsymbol{x}_i}[S(m(\boldsymbol{x}_i), Y_i^0)]$, where Y_i^0 is a new random response value for each $\boldsymbol{x}_i$. Calculate the expected optimism over the responses of the training set, $\omega = E_{Y_{\text{train}}}[\mathrm{op}]$.

Exercise 4

Give examples of possible penalties Ω.

Exercise 5

Simulate data and try to reproduce the double descent phenomenon (classical regime and interpolating regime).

Table of Contents

GLM

Generalised Linear Models estimate the expectation $\mu = \mathbb{E}[Y|\boldsymbol{X}]$.

Building blocks

- numerical features $\boldsymbol{X} \in \mathbb{R}^{n,p}$ and coefficients (or weights) $\beta \in \mathbb{R}^p$
  They form the linear predictor $\eta = \boldsymbol{X} \cdot \beta$.
- injective inverse link / response function h: $m(\boldsymbol{X}) = h(\eta)$.
- deviance of Exponential Dispersion Family (EDM) as loss function

GLMs are linear in the coefficients β in link space.

Statistical Assumptions

$Y|\boldsymbol{X} \overset{\text{i.i.d}}{\sim} \mathrm{EDM}$

For estimation of β, only mean and variance of Y really matter:

- $\mathbb{E}[Y_i|\boldsymbol{X}] = \mu_i$
- $\mathrm{Var}[Y_i|\boldsymbol{X}] = \sigma_i^2 = \frac{\phi}{w_i} v(\mu_i)$ with dispersion parameter ϕ, weights w_i and variance function $v(\mu)$ given by the EDM.

Canonical Link Functions

The first order condition for estimation is called *score equation*:

$$0 \stackrel{!}{=} \boldsymbol{X}'\mathbf{D}\Sigma^{-1}(y-\mu) \tag{8}$$

with $\mathbf{D} = \text{diag}(h'(\eta))$, $\Sigma = \text{diag}(\sigma^2)$. Canonical link: $h'(\mu) = \frac{1}{v(\mu)} \sim \frac{1}{\sigma^2}$.
The score equation is then called **balance property**:

$$0 \stackrel{!}{=} \boldsymbol{X}'(y-\mu) \tag{9}$$

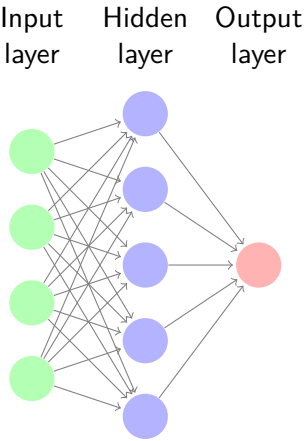
Neural Net

Architecture

- Set of connected neurons, often structured in layers.
- Each neuron
  - gets as input the outputs $\{z_j, j \in \mathcal{I}\}$ from other neurons input layer gets the features as input $z_j = \boldsymbol{x}_j$
  - calculates the propagation function $\eta(\mathbf{z}) = \sum_{j \in \mathcal{I}} z_j \beta_j + b$ with weights (coefficients) β and bias (intercept) b
  - outputs $z = \phi(\eta(\mathbf{z}))$ with some activation function ϕ (sigmoid/logistic, hyperbolic tangent, relu, ...)
- Prediction is a nested function $m(\boldsymbol{x}) = z_{output}(\{z_j\})$.

Learning / Training

- Choose loss/scoring function S.
- Use optimisation methods for $\arg\min_{\beta,b} \overline{R}(m)$



feed forward net

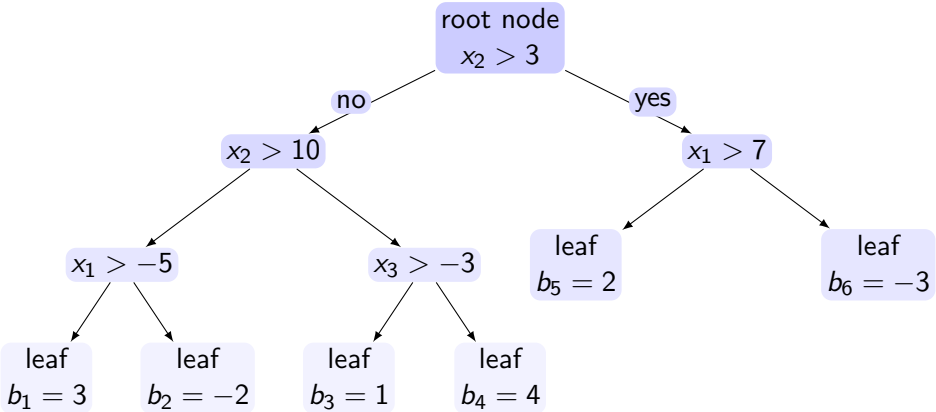
Decision Tree

(Binary) Decision trees are piecewise linear functions. They can model any target T.

Building blocks

- features $\boldsymbol{X}$ with ordering $x_{i_1,j} < x_{i_2,j}$ (observations i_1, i_2, feature j)
- hierarchical structured nodes with feature index j and thresholds θ that partition the feature space $\mathcal{X}$ into disjoint sets Q_k
- loss function S for finding split (j, θ) and node prediction b; plugging in $b = \arg\min_z \overline{S}(z; Q(\theta))$ simplifies loss to entropy, here called splitting criterion, e.g., Gini criterion
- K terminal nodes (aka leaves) with predicted value b_k give tree prediction $m(\boldsymbol{x}) = \mathrm{tree}(\boldsymbol{x}, \{b_k, Q_k\}_1^K) = \sum_{k=1}^K b_k \mathbb{1}\{\boldsymbol{x} \in Q_k\}$

Decision Trees



x_1	x_2	x_3	$m(x)$
-4	2	2	-2
-4	4	2	2

Ensemble: Random Forests

Ensemble

Given K models m_k, pool their predictions as $m(\boldsymbol{x}) = \frac{1}{K}\sum_1^K m_k(\boldsymbol{x})$.

Random forest

Fit K different (independent) decision trees, each only on a subset of observations and/or features and pool them together. Each single tree is usually highly overfitted.

Ensemble: Gradient Boosting

Boosting

- Injective inverse link / response function $m(\boldsymbol{x}) = h(F_k(\boldsymbol{x}))$.
- After k fitting stages/iterations we have $F_k(\boldsymbol{x}) = \sum_{j=1}^{k} f_j(\boldsymbol{x})$.
- In each (learning) iteration, add f_k while keeping F_{k-1} fixed.
- Learn f_k by fitting on the residual loss: $\arg\min_{f_k} \overline{R}(h(F_{k-1} + f_k); D_{\text{train}})$

Gradient boosting

Optimisation in function space.

- $f_k(\boldsymbol{x}) = -\rho_k g_k(\boldsymbol{x})$
- Gradients
  $g_k(\boldsymbol{x}) = \left[\frac{\partial \mathbb{E}[S(h(F(\boldsymbol{x})), Y) | \boldsymbol{X} = \boldsymbol{x}]}{\partial F(\boldsymbol{x})}\right]_{F(\boldsymbol{x}) = F_{k-1}(\boldsymbol{x})} = \mathbb{E}\left[\frac{\partial S(h(F(\boldsymbol{x})), Y)}{\partial F(\boldsymbol{x})} | \boldsymbol{X} = \boldsymbol{x}\right]_{F(\boldsymbol{x}) = F_{k-1}(\boldsymbol{x})}$
- For finite data, fit f_k via squared error on response $-g_k(\boldsymbol{x})$: $\hat{f}_k = -\rho_k \hat{g}_k$
- Line search $\rho_k = \arg\min_{\rho} R(h(F_{k-1}(\boldsymbol{X}) - \rho \hat{g}_k(\boldsymbol{X})))$

Gradient Boosted Trees

Trees as base learner

- Use trees $\hat{g}_k(\boldsymbol{x}) = \mathrm{tree}(\boldsymbol{x}, \{b_j, Q_j\}_1^J)$.
- On leaf Q_j we have $b_j = -\mathrm{mean}_{i \in Q_j} g_k(\boldsymbol{x}_i)$.
- Line search to find ρ_k.
- Update $F_k(\boldsymbol{x}) = F_{k-1} + \rho_k \sum_{j \in \text{leaves}}^J b_{k,j} \mathbb{1}\{\boldsymbol{x} \in Q_j\}$.

As leaves are disjoint, this can be seen as J separate boosting steps:

$$F_k(\boldsymbol{x}) = F_{k-1} + \sum_{j \in \text{leaves}}^{J} \tilde{b}_{k,j} \mathbb{1}\{\boldsymbol{x} \in Q_j\}$$

with $\tilde{b}_{k,j} = \rho_k b_{k,j}$. The values $\tilde{b}$ can be found by a line search on each leaf:

$$\tilde{b}_{k,j} = \arg\min_b \sum_{i \in Q_j} S(h(F_{k-1}(\boldsymbol{x}_i) + b), y_i)$$

Note: This gives optimal line search per leaf instead of a "global" line search for the whole tree.

Modern Gradient Boosting

- Hessian (2. order)
  - $h_k(\boldsymbol{x}) = \mathbb{E}\left[\frac{\partial^2 S(h(F(\boldsymbol{x})), Y)}{\partial^2 F(\boldsymbol{x})} | \boldsymbol{X} = \boldsymbol{x}\right]_{F(\boldsymbol{x})=F_{k-1}(\boldsymbol{x})}$
  - 2. order Taylor of $\overline{R}(F_k; D) \approx const + \frac{1}{2n}\sum_i h_k(\boldsymbol{x}_i)\left(f_k(\boldsymbol{x}_i) + \frac{g_k(\boldsymbol{x}_i)}{h_k(\boldsymbol{x}_i)}\right)^2$
  - Optionally add penalty $\Omega = \frac{\lambda}{2}\sum_{\text{leaves } j} b_j^2$.
  - Fit a tree via weighted least squares on response $-g/h$ with weights h.
  - On leaf Q_j, we have constant prediction $b_j = -\frac{\sum_{i \in Q_j} g_k(\boldsymbol{x}_i)}{\sum_{i \in Q_j} h_k(\boldsymbol{x}_i)}$
- Histogram: Calculate histogram for each feature and accumulate h and g/h.
- Categorical (nominal) features: Reduces splitting from $\mathcal{O}(2^K)$ to $\mathcal{O}(Klog(K) + K)$ by Fisher's algo[2]
  - Sort by g/h
  - Treat them as continuous/ordinal

[2] Fisher, W.D. (1958)"On Grouping for Maximum Homogeneity" Journal of the American Statistical Association, 53, 789-798.

Excercises I

Exercise 6

Given linear models with link function h and prediction $m(\boldsymbol{x}) = h(\boldsymbol{x} \cdot \beta)$, derive the first order optimality condition with Bregman functions as loss. Which condition has to hold to arrive at the balance property (9)? The resulting link function is the *canonical* one.

Exercise 7

Derive the splitting criterion for loss/scoring functions:

1. squared error
2. log loss $S(z, y) = -y \log(z) - (1 - y) \log(1 - z)$, $y \in \{0, 1\}$, $z \in [0, 1]$

Exercise 8

For median regressing gradient boosted trees using the absolute error, derive:

1. the "global" line search ρ_k

Excercises II

2. the per leave line search $\tilde{b}_{k,j}$

Exercise 9

Fit a GLM, a random forest and a gradient boosted tree model on the Workers Compensation dataset `https://www.openml.org/d/42876`. Model goal is $\mathbb{E}[\text{UltimateIncurredClaimCost}|\boldsymbol{X}]$.

Table of Contents