Yandex

SCHOOL OF DATA ANALYSIS

Intro into Machine Learning

Why ML works. Overfitting. Model Selection. Figures of Merits. Linear Models. Regularization. Logistic Regression.

Clermont University,
Nov, 2017

Alexey Artemov[1,3], Andrey Ustyuzhanin[2,3]

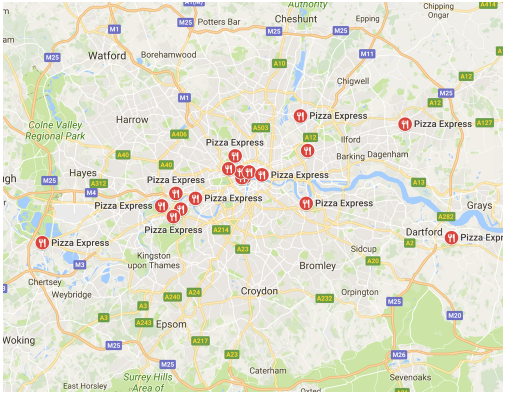
[1] Yandex LLC [2] Yandex School of Data Analysis [3] National Research University Higher School of Econonmics

Lecture overview

- A gentle introduction: ML for the real world
- Linear models: the regression case
- Linear models: the classification case
- Figures of Merits: visual evaluation of solution effectiveness
- Why ML works: a bit of Learning Theory
- Overfitting: how to fool the linear regression
- Regularization

Machine learning for the real world

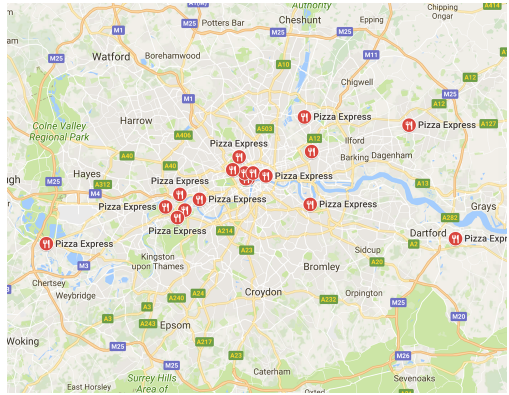
- You own a pizza restaurant chain
- Where to open the next restaurant?
- No revenue laws (of the kind $\vec{\mathbf{F}} = m\vec{\mathbf{a}}$) exist!
- Is there any stats on restaurant revenue?



Machine learning for the real world

- Use stats!
- Properties and revenues of existing restaurants:

Id	Location	Revenue
1	51.472866,-0.2588277	$1.1M
2	51.4865675,-0.2210999	$2.23M
...	...	...
20	51.4334641,-0.5146408	$0.59



Some notation

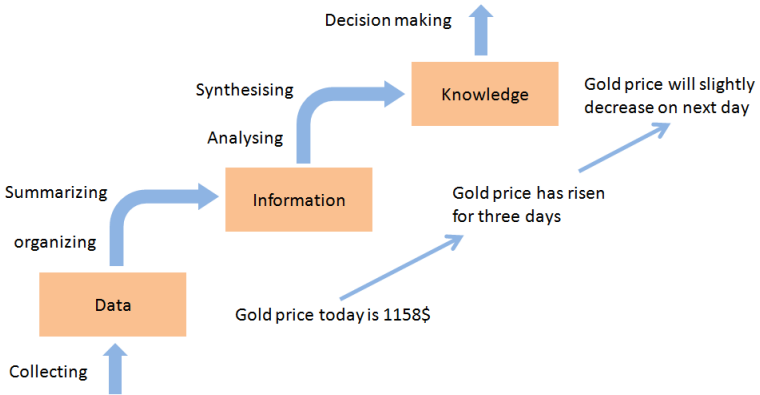
- Object space $\mathbb{X} = \mathbb{R}^2$: restaurant location (also known as the feature space)
- Target space $\mathbb{Y} = \mathbb{R}$: restaurant revenue
- Training set $X^\ell = \left\{(\mathbf{x}_i, y_i)\right\}_{i=1}^{\ell}$ (known locations and revenues for existing restaurants)
- **The goal:** estimate $y \in \mathbb{Y}$ for a new $\mathbf{x} \in \mathbb{X}$ (estimate revenue for a proposed location)

Expert knowledge and physical models VS machine learned models

- Would one want to apply machine learning for . . .
  - . . . Newton's mechanics?
  - . . . suggesting music to radio listeners?
  - . . . sorting integers?
  - . . . controlling steel production?
  - . . . sorting strawberries?
- Criteria for machine learning to be applied in a dependency recovery setting:
  - No prior knowledge for the dependency exists
  - The dependency has a complex form too hard for manual examination
  - A sample from the dependency of sufficiently large size is available

What is machine learning about?

- Inference of statistical dependencies which give us ability to predict
- Data is cheap, knowledge is precious



Picture credit: https://www.packtpub.com/books/content/introduction-data-analysis-and-libraries

Feature vectors

- Computers cannot operate objects of the real world directly
- Need features describing the objects ($\mathbf{x} = (\text{lat}, \text{lon})$ for the restaurant)

- In real applications such as fraud detection, feature engineering is the most complex task
- Many papers on the problem

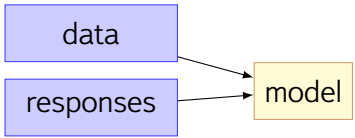
- Machine location (e.g. beside supermarket, latitude, longitude), brand, and age
- Customer demographic (e.g. postcode) and behavioral information
- Card type (e.g. Visa, supplementary, issued yesterday) and usage
- Session/transaction day-of-week, time-of-day, keystrokes (e.g. number of login attempts), transaction type, and amount

Possible kinds of features: some more notation

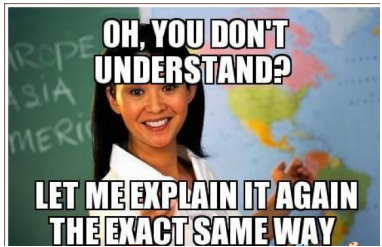
- Binary: $x \in \{0, 1\}$ (are you a member of a terrorist organization?)
- Real-valued: $x \in \mathbb{R}$ (customer income)
- Categorical: $x \in S$ where $|S| = n$ and order of elements in S does not matter (customer postcodes)
- Ordinal: $x \in S$ where $|S| = n$ and order of elements in S does matter (old $<$ renovated $<$ new for house price prediction)
- Structured: images (customer photo in the fraud detection task)

Kinds of machine learning tasks

› Supervised learning

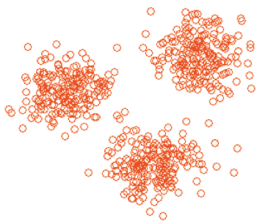


› Regression, (multi-class/multi-label) classification, semi-supervised learning



› Unsupervised learning

› Clustering, density estimation, dimensionality reduction, visualization



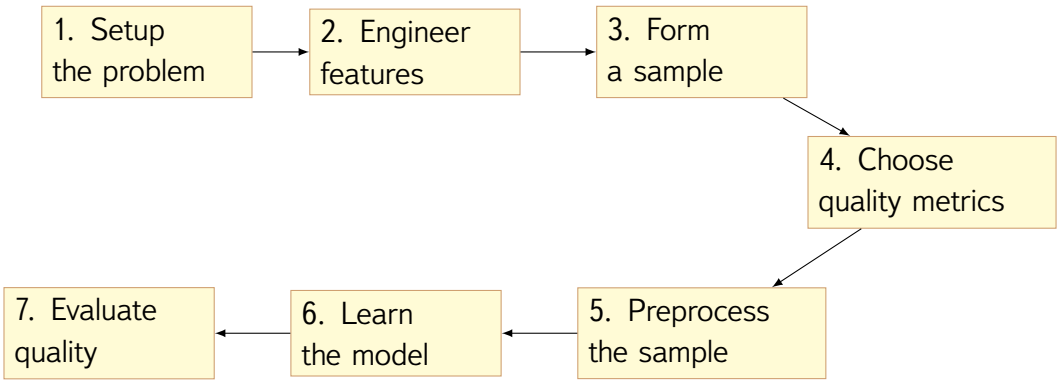
Yet more notation

- An unknown distribution D generates instances ($\mathbf{x}_1, \mathbf{x}_2, \ldots$) independently
- An unknown function $f : \mathbb{X} \to \mathbb{Y}$ generates responses ($y_1, y_2, \ldots$) for them such that $y_i = f(\mathbf{x}_i), i = 1, 2, \ldots$
- The machine learning problem: choose a plausible hypothesis $h : \mathbb{X} \to \mathbb{Y}$ from the hypothesis space $\mathbb{H}$
- The error of a hypothesis h is the deviation from the true f measured by the loss function (an example for regression):

$$Q(h, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} (f(\mathbf{x}_i) - h(\mathbf{x}_i))^2$$

- Learning: the search for the optimal hypothesis $h \in \mathbb{H}$ w. r. t. the fixed loss function

The machine learning pipeline



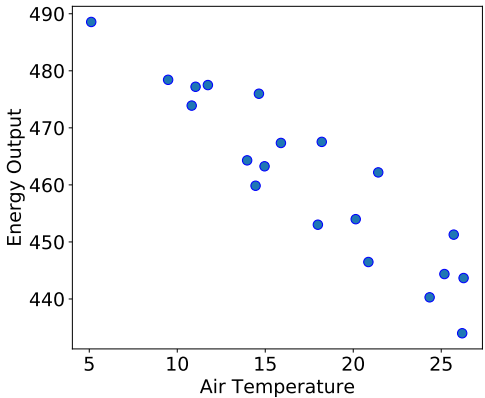
Linear models for regression

Univariate linear regression

- A single feature (regressor) x: Air Temperature
- A single dependent variable y: Energy Output
- Training set $X^\ell = \left\{(x_i, y_i)\right\}_{i=1}^{20}$
- The regression model:

$$y_i = h(x_i; \mathbf{w}) + \varepsilon_i$$

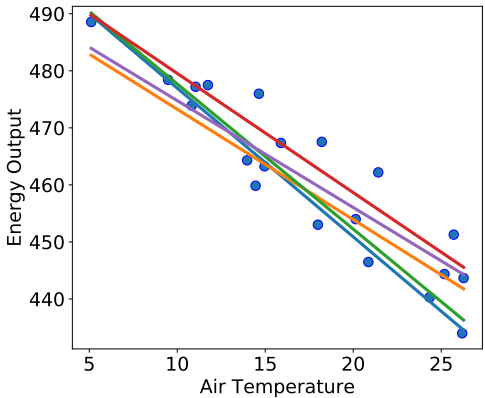
- Linear model: $y_i = w_1 x_i + w_0 + \varepsilon_i$
- The goal: given X^ℓ, find $\mathbf{w} = (w_1, w_0)$



Univariate linear regression

- Which fit to choose?
- With the linear model being fixed, depends on the data and the loss function!
- Mean square (L2) loss (MSE):

$$Q(h, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} (y_i - h(x_i))^2$$



Some other evaluation metrics for regression

- Mean square (L2) loss (MSE): $\mathrm{MSE}(h, X^\ell) = \frac{1}{\ell}\sum_{i=1}^{\ell}(y_i - h(x_i))^2$
- Root MSE: $\mathrm{RMSE}(h, X^\ell) = \sqrt{\frac{1}{\ell}\sum_{i=1}^{\ell}(y_i - h(x_i))^2}$
- Coefficient of determination: $R^2(h, X^\ell) = 1 - \frac{\sum_{i=1}^{\ell}(y_i - h(x_i))^2}{\sum_{i=1}^{\ell}(y_i - \mu_y)^2}$
  with $\mu_y = \frac{1}{\ell}\sum_{i=1}^{\ell} y_i$
- Mean absolute error: $\mathrm{MAE}(h, X^\ell) = \frac{1}{\ell}\sum_{i=1}^{\ell}|y_i - h(x_i)|$

Univariate linear regression

› With the loss fixed, the linear problem reduces to optimization:

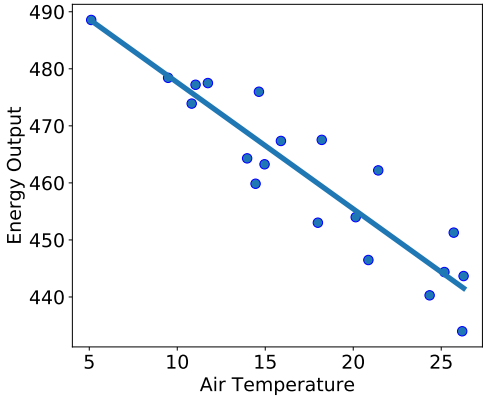
$$\frac{1}{\ell}\sum_{i=1}^{\ell}(y_i - w_1 x_i - w_0)^2 \to \min_{(w_0,w_1)\in\mathbb{R}^2},$$

to which an analytical solution is available

$$\widehat{w}_1 = \frac{\sum_{i=1}^{\ell}(x_i - \mu_x)(y_i - \mu_y)}{\sum_{i=1}^{\ell}(x_i - \mu_x)^2},$$

$$\widehat{w}_0 = \mu_y - \widehat{w}_1\mu_x$$

with $\mu_x = \frac{1}{\ell}\sum_{i=1}^{\ell} x_i, \quad \mu_y = \frac{1}{\ell}\sum_{i=1}^{\ell} y_i$



Multivariate linear regression

- Multiple features (regressors) $\mathbf{x}_i = (x_{1i}, \dots x_{di})$ available for each y_i
- The model:

$$
\begin{aligned}
y_1 &= w_1 x_{11} + \dots w_d x_{d1} + \varepsilon_1, \\
y_2 &= w_1 x_{12} + \dots w_d x_{d2} + \varepsilon_2, \\
&\dots \\
y_\ell &= w_1 x_{1\ell} + \dots w_d x_{d\ell} + \varepsilon_\ell,
\end{aligned}
$$

is often written in matrix-vector form as

$$
\begin{bmatrix} y_1 \\ \vdots \\ y_\ell \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{d1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1\ell} & x_{2\ell} & \dots & x_{d\ell} \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_d \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_\ell \end{bmatrix} \quad \longleftrightarrow \quad \boldsymbol{y} = \boldsymbol{X}\mathbf{w} + \boldsymbol{\varepsilon}
$$

Multivariate linear regression: the solution

› The problem: minimize MSE

$$Q(h, X^l) = \sum_{i=1}^{\ell} \Big(y_i - \sum_{k=1}^{d} w_k x_{ki}\Big)^2 \equiv \|\boldsymbol{y} - \boldsymbol{X}\mathbf{w}\|^2 \to \min_{\mathbf{w}\in\mathbb{R}^d}$$

› Solve analytically via computing the gradient

$$\nabla_{\mathbf{w}} \|\boldsymbol{y} - \boldsymbol{X}\mathbf{w}\|^2 = 2(\boldsymbol{y} - \boldsymbol{X}\mathbf{w})\boldsymbol{X} = 0$$

› The solution

$$\mathbf{w}^* = (\boldsymbol{X}^\intercal \boldsymbol{X})^{-1} \boldsymbol{X}^\intercal \boldsymbol{y}$$

A quick intro into the Numerical Optimization

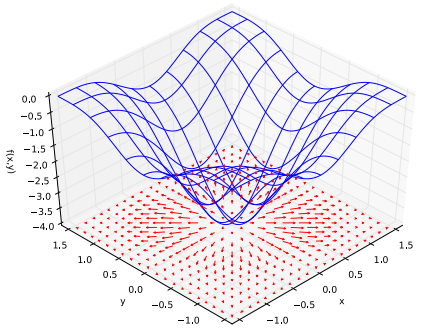
› Consider the optimization problem in $\mathbb{R}^d$

$$f(\mathbf{x}) \to \min_{\boldsymbol{x} \in \mathbb{R}^d} \qquad \text{(such as } \sum_{i=1}^{\ell} \Big(y_i - \sum_{k=1}^{d} w_k x_{ki} \Big)^2 \to \min_{\boldsymbol{w} \in \mathbb{R}^d} \text{)}$$

› In general, solved using the numerical methods such as the gradient descent

› Gradients: directions in $\mathbb{R}^d$ pointing towards steepest function increase

$$\nabla_{\mathbf{x}} f(\mathbf{x}) \equiv \left(\frac{\partial f(\mathbf{x})}{\partial x_1}, \ldots, \frac{\partial f(\mathbf{x})}{\partial x_d} \right)$$



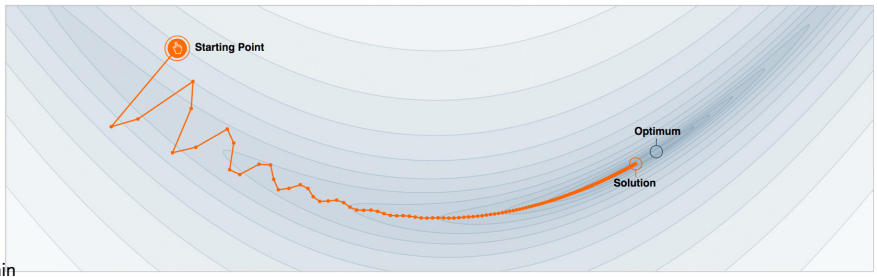
The gradient descent algorithm

› The gradient descent procedure iterates from $\mathbf{x}^{(0)}$ as

$$\mathbf{x}^{(k)} \leftarrow \mathbf{x}^{(k-1)} - \alpha_k \nabla_{\mathbf{x}} f(\mathbf{x}^{(k-1)})$$

with α_k controlling the kth step size

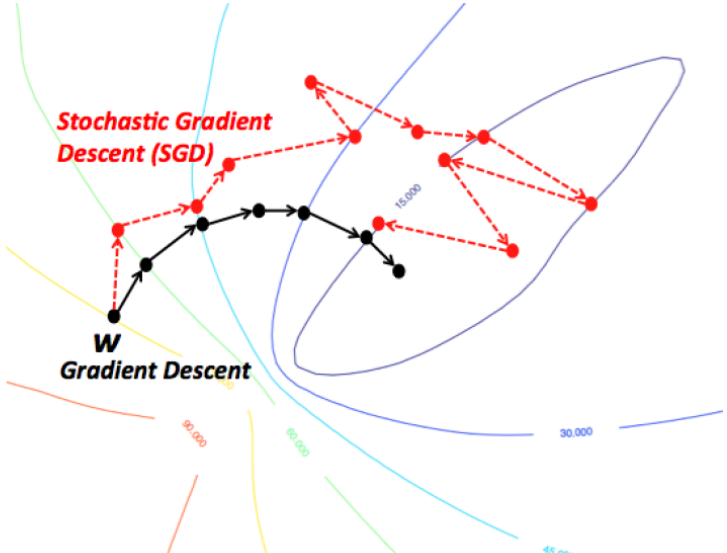
› For smooth convex functions with a single minimum $\mathbf{x}^*$, k steps of gradient descent achieve accuracy $f(\mathbf{x}^{(k)}) - f(\mathbf{x}^*) = \mathcal{O}(1/k)$



The stochastic gradient descent algorithm

- Machine learning: many additive targets $f(\mathbf{w}) = \sum_{i=1}^{\ell} f_i(\mathbf{w})$, computationally inefficient for large ℓ
- Use subsamples for gradient estimation: the Stochastic Gradient Descent (SGD)
  1. Pick $i_k \in \{1, \ldots, \ell\}$ at random;
  2. Compute $\mathbf{w}^{(k)} \leftarrow \mathbf{w}^{(k-1)} - \alpha_k \nabla_{\mathbf{w}} f_{i_k}(\mathbf{w}^{(k-1)})$
- For smooth convex functions with a single minimum $\mathbf{x}^*$, k steps of SGD achieve accuracy $f(\mathbf{w}^{(k)}) - f(\mathbf{w}^*) = \mathcal{O}(1/\sqrt{k})$
- Batch, variance reduction, momentum hacks available to improve the convergence rate to $\mathcal{O}(1/k)$

Gradient descent vs stochastic gradient descent

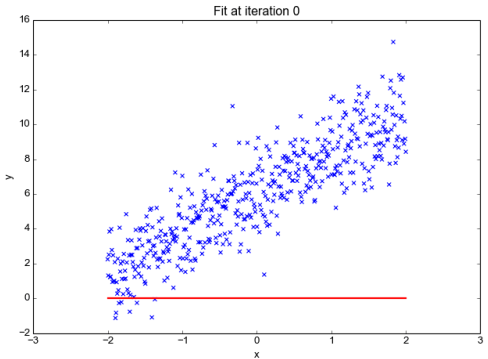


Multivariate linear regression: the numerical solution

- Initialize with some $\mathbf{w}^{(0)}$
- Gradient in i_kth object is

$$\nabla_{\mathbf{x}} f_{i_k}(\mathbf{w}) = 2(y_{i_k} - \mathbf{x}_{i_k}^{\mathsf{T}}\mathbf{w})\mathbf{x}_{i_k} \qquad (\in \mathbb{R}^d)$$

- Compute updates using SGD:
  $\mathbf{w}^{(k)} \leftarrow \mathbf{w}^{(k-1)} - \alpha_k \nabla_{\mathbf{w}} f_{i_k}(\mathbf{w}^{(k-1)})$



Linear models for classification

Linear models for classification

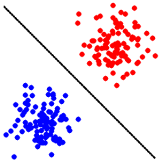
- An unknown distribution D generates instances $(\mathbf{x}_1, \mathbf{x}_2, \ldots)$
- An unknown function $f : \mathbb{X} \to \mathbb{Y}$ generates labels $(y_1, y_2, \ldots)$ for them such that $y_i = f(\mathbf{x}_i)$, and $y_i \in \{-1, +1\}$
- The classification problem: choose a plausible hypothesis (classifier) $h : \mathbb{X} \to \mathbb{Y}$ from the hypothesis space $\mathbb{H}$
- The error of the classifier h is the probability (over D) that it will fail

$$Q(h, D) = \Pr_{\mathbf{x} \sim D}[f(\mathbf{x}) \neq h(\mathbf{x})]$$

usually estimated by the accuracy metric

$$Q(h, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} [f(\mathbf{x}_i) \neq h(\mathbf{x}_i)]$$

Linear models for classification



- Linear model: $h(\mathbf{x}) = \text{sign}\Big(\sum_{i=1}^{d} w_i x_i + w_0\Big) = \text{sign}\big(\mathbf{w}^\intercal \mathbf{x} + w_0\big)$
- The learning problem is discrete over $\mathbf{w} \in \mathbb{R}^d$:

$$Q(h, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} [\text{sign}(\mathbf{w}^\intercal \mathbf{x}_i) \neq h(\mathbf{x}_i)] \to \min_{\mathbf{w} \in \mathbb{R}^d}$$

(cannot optimize using gradient descent)

- The solution: optimize a differentiable upper bound for $Q(h, X^\ell)$!
- $Q(h, X^\ell)$ can be written using $Q(h, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} L(M_i)$

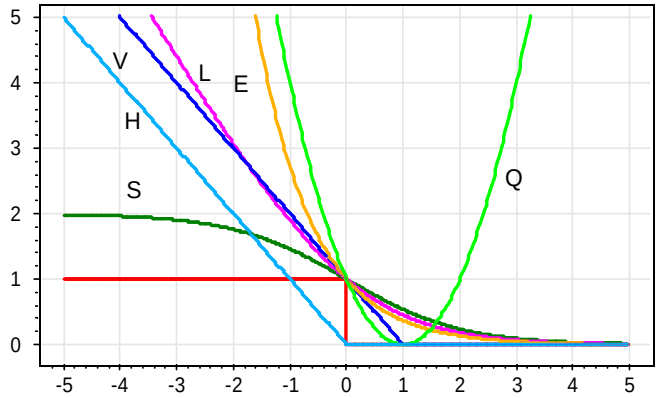
  where $L(M_i) = [M_i < 0] \equiv [y_i \mathbf{w}^\intercal \mathbf{x}_i < 0]$
- Upper-bounding $L(M)$ yields upper bounds for $Q(h, X^\ell)$

Linear models for classification: upper bounds

Multiple approximations to accuracy loss

- $L_{\mathsf{L}}(M) = \log(1 + e^{-M})$
- $L_{\mathsf{H}}(M) = \max(0, 1 - M)$
- $L_{\mathsf{P}}(M) = \max(0, -M)$
- $L_{\mathsf{E}}(M) = e^{-M}$
- $L_{\mathsf{S}}(M) = 2/(1 + e^{M})$

give rise to various learning algorithms



The logistic regression model

- Training set $X^\ell = \left\{(\mathbf{x}_i, y_i)\right\}_{i=1}^{\ell}$ where $y_i \in \{-1, +1\}$
- We seek an algorithm h such that $h(\mathbf{x}) = \mathrm{P}(y = +1|\mathbf{x})$
- A probability that an instance $(\mathbf{x}_i, y_i)$ is encountered in X^ℓ

$$h(\mathbf{x}_i)^{[y_i=+1]} + (1 - h(\mathbf{x}_i))^{[y_i=-1]}$$

- Entire X^ℓ likelihood:

$$L(X^\ell) = \prod_{i=1}^{\ell} h(\mathbf{x}_i)^{[y_i=+1]} + (1 - h(\mathbf{x}_i))^{[y_i=-1]}$$

is often written via log-likelihood (of which the negative is log-loss)

$$\log L(X^\ell) = \sum_{i=1}^{\ell} [y_i = +1] \log h(\mathbf{x}_i) + [y_i = -1] \log(1 - h(\mathbf{x}_i))$$

The logistic regression model

- The choice of h: sigmoid function

$$h(\mathbf{x}) = \sigma(\mathbf{w}^\mathsf{T}\mathbf{x})$$

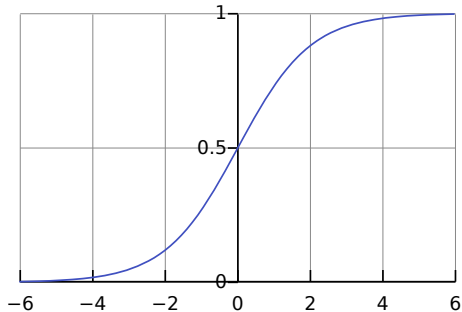
where $\sigma(x) \in [0, 1]$

- Typical choice: the logistic function

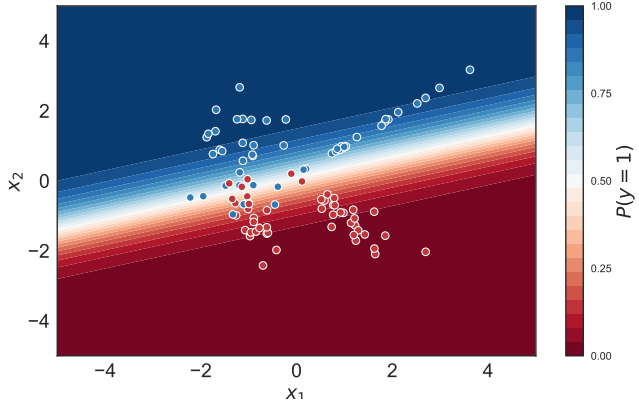
$$\sigma(\mathbf{w}^\mathsf{T}\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^\mathsf{T}\mathbf{x})}$$

- Plugging the logistic function into the loss yields

$$\sum_{i=1}^{\ell}(1 + \exp(\mathbf{w}^\mathsf{T}\mathbf{x})) \to \min_{\mathbf{w}\in\mathbb{R}^d}$$



The logistic regression model



Interactive Logistic Regression Demo

Classification based on the nearest neighbours

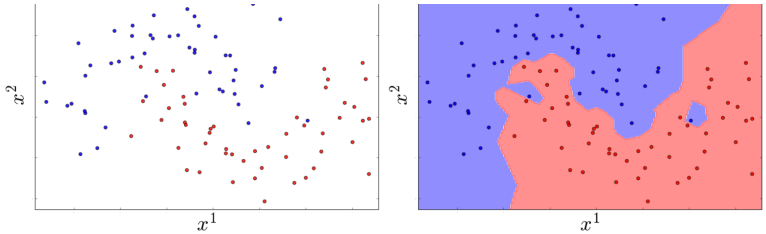
Given training set of objects and their labels x_i, y_i we predict the label for the new observation x:

$$\hat{y} = y_j, \quad j = arg\min_i \rho(x, x_i)$$

Here and after $\rho(x, \tilde{x})$ is the distance in the space of features.

Visualization of decision rule

Consider a classification problem with 2 features:



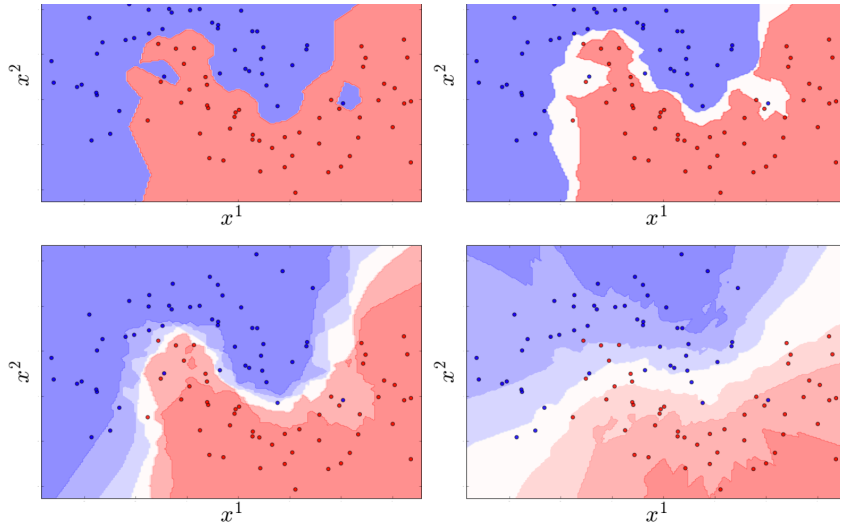
$$x_i = (x_i^1, x_i^2), y \in Y = \{0, 1\}$$

k Nearest Neighbours (kNN)

We can use k neighbours to compute probabilities of observation to belong to each class:

$$p_{\tilde{y}}(x) = \frac{\text{number of knn of point } x \text{ in class } \tilde{y}}{k}$$

Question: why this may be a good idea?



$k = 1, 2, 5, 30$

Overfitting

Question: how accurate is classification on training dataset when $k = 1$?

- answer: it is ideal (closest neighbor is observation itself)
- quality is lower when $k > 1$
- this doesn't mean $k = 1$ is the best, it means we cannot use training observations to estimate quality
- when classifier's decision rule is too complex and captures details from training data that are not relevant to distribution, we call this an overfitting

Figures of Merits

Classification quality evaluation: accuracy

- Given a labeled sample $X^\ell = \left\{(\mathbf{x}_i, y_i)\right\}_{i=1}^{\ell}$, $y_i \in \{-1, +1\}$, and some candidate h, how well does h perform on X^ℓ?
- Let $a(x) = [h(x) > t]$
- Obvious choice: accuracy

$$\text{accuracy}(a, X^\ell) = \frac{1}{\ell}\sum_{i=1}^{\ell}[a(\mathbf{x}_i) = y_i]$$

- Bad for imbalanced data: for $\ell = 1000$,
  $n_- = \sum_{i=1}^{\ell}[y = -1] = 50$, $n_+ = \sum_{i=1}^{\ell}[y = +1] = 950$,
  a trivial rule $h(\mathbf{x}) = +1$ would yield $\text{accuracy}(a, X^\ell) = 0.95$

Classification quality: confusion matrix

	$y = 1$	$y = -1$
$a(x) = 1$	True Positive (TP)	False Positive (FP)
$a(x) = -1$	False negative (FN)	True Negative (TN)

› More informative criteria:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}},$$

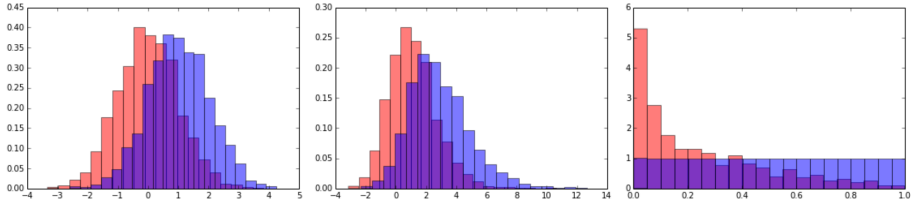
$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

› While accuracy can be expressed, too

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

Measuring quality of binary classification

The classifier's output in binary classification is real variable (say, signal is blue and background is red)



- Which classifier provides better discrimination?
- Discrimination is identical in all three cases

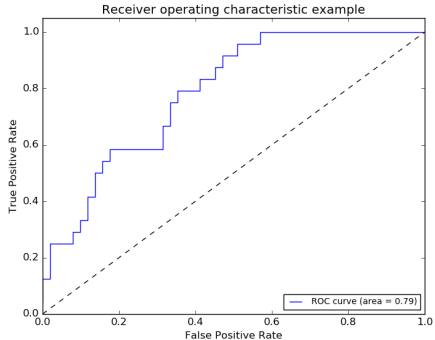
Classification quality: operating curves

- Often $h(\mathbf{x})$ is more valuable than its thresholded version $a(x) = [h(x) > t]$
- Consider two-dimensional space with coordinates

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}, \qquad \text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}},$$

  corresponding to various choices of the threshold t
- The plot TPR(FPR) is called the receiver operating characteristic (or ROC) curve

Receiver operating characteristic curve



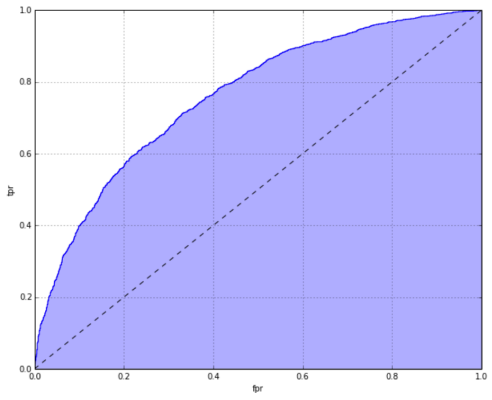
Interactive ROC curve demo

ROC Curve

- Defined only for binary classification
- Contains important information: all possible combinations of signal and background efficiencies you may achieve by setting a threshold
- Particular values of thresholds (and initial pdfs) don't matter, ROC curve doesn't contain this information
- ROC curve = information about order of observations' predictions:
  `b b s b s b ... s s b s s`
- Comparison of algorithms should be based on the information from ROC curve

Area Under ROC Curve - ROC AUC

$ROCAUC = P(r_b < r_s)$ where r_b , r_s are predictions for random background and signal observations.

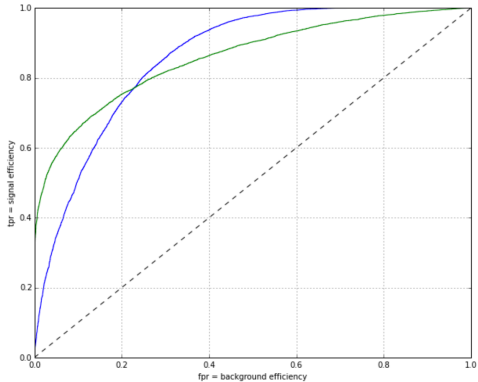


ROC Curve - comparison

Classifier have the same ROC AUC, but which is better ...

- if the problem is spam detection? (putting normal letter in spam is very undesireable)
- for background rejection system at the LHC? (we need to pass very few background)

Applications frequently demand different metric.

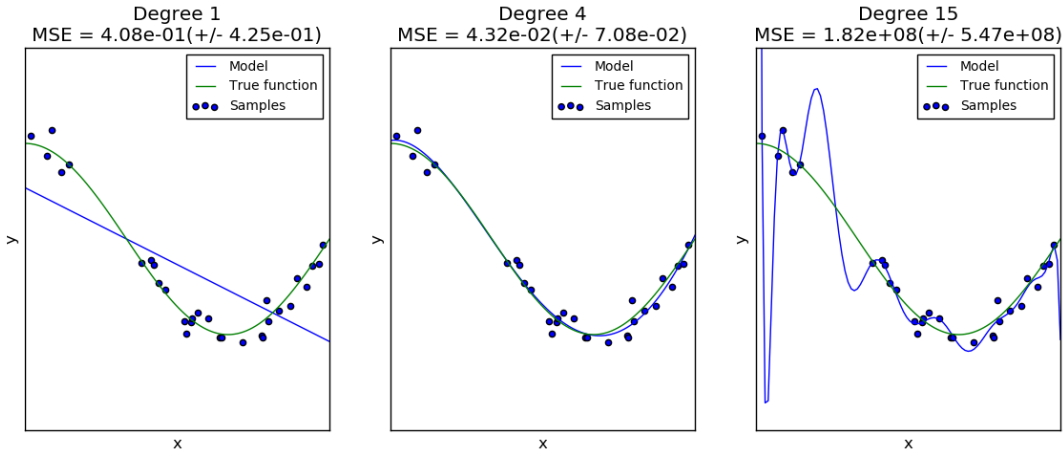


Overfitting: how to fool the linear regression

Generalization and overfitting

- Training set memorization: for seen $(\mathbf{x}, y) \in X^\ell$, $h(\mathbf{x}) = y$
- Generalization: equally good performance on both new and seen instances
- How to assess model's generalization ability?
- Consider an example:
  - $y = \cos(1.5\pi x) + \mathcal{N}(0, 0.01)$, $x \sim \text{Uniform}[0, 1]$
  - Features: $\{x\}$, $\{x, x^2, x^3, x^4\}$, $\{x, \ldots, x^{15}\}$
- How well do the regression models perform?

Polynomial fits of different degrees



Assessing generalization ability: cross-validation

- Split training set into subsets of equal size $X^\ell = X_1^\ell \cup \ldots \cup X_K^\ell$
- Train K models $h_1, \ldots, h_K$ where each model h_k is trained on all subsets but X_k^ℓ
- Assess quality using $\mathrm{CV} = \frac{1}{K} \sum_{k=1}^{K} Q(h_k, X_k^\ell)$ (K-fold)
- Leave-one-out cross-validation: $X_k^\ell = \{(\mathbf{x}_k, y_k)\}$

Regularization

Ad-hoc regularization: motivation

› Consider the multivariate linear regression problem with $\boldsymbol{X} \in \mathbb{R}^{d \times d}$

$$\|\boldsymbol{y} - \boldsymbol{X}\mathbf{w}\|^2 \to \min_{\mathbf{w} \in \mathbb{R}^d}$$

› Analytic solution involves computing the product $\boldsymbol{R} = (\boldsymbol{X}^\intercal \boldsymbol{X})^{-1}\boldsymbol{X}^\intercal$

› If $\boldsymbol{X} = \text{diag}(\lambda_1, \ldots, \lambda_d)$ with $\lambda_1 > \lambda_2 > \ldots > \lambda_d \to 0$
(meaning we're in eigenbasis of $\boldsymbol{X}$) then

$$\begin{aligned}
\boldsymbol{R} &= (\boldsymbol{X}^\intercal \boldsymbol{X})^{-1}\boldsymbol{X}^\intercal = \\
&= \Big(\text{diag}(\lambda_1, \ldots, \lambda_d)\text{diag}(\lambda_1, \ldots, \lambda_d)\Big)^{-1}\text{diag}(\lambda_1, \ldots, \lambda_d) = \\
&= \text{diag}\left(\frac{1}{\lambda_1}, \ldots, \frac{1}{\lambda_d}\right), \quad \text{leading to huge diagonal values in } \boldsymbol{R}
\end{aligned}$$

Ad-hoc regularization: L2

- **Regularization:** replace fit with fit + penalty as in

$$Q(\mathbf{w}) \to Q_\alpha(\mathbf{w}) = Q(\mathbf{w}) + \alpha R(\mathbf{w})$$

- $R(\mathbf{w})$ is called the regularizer, $\alpha > 0$ — the regularization constant
- Regularized multivariate linear regression problem

$$\|\boldsymbol{y} - \boldsymbol{X}\mathbf{w}\|^2 + \alpha\|\mathbf{w}\|_2^2 \to \min_{\mathbf{w}\in\mathbb{R}^d}$$

- Regularized analytic solution available

$$\mathbf{w}^* = (\boldsymbol{X}^\intercal\boldsymbol{X} + \alpha\boldsymbol{I})^{-1}\boldsymbol{X}^\intercal\boldsymbol{y}$$

Why L2 regularization works

› Analytic solution: compute the regularized operator

$$\boldsymbol{R} = (\boldsymbol{X}^\intercal \boldsymbol{X} + \alpha \boldsymbol{I})^{-1} \boldsymbol{X}^\intercal$$

› If $\boldsymbol{X} = \text{diag}(\lambda_1, \ldots, \lambda_d)$ with $\lambda_1 > \lambda_2 > \ldots > \lambda_d \to 0$
(meaning we're in eigenbasis of $\boldsymbol{X}$) then

$$\begin{aligned}
\boldsymbol{R} &= (\boldsymbol{X}^\intercal \boldsymbol{X} + \alpha \boldsymbol{I})^{-1} \boldsymbol{X}^\intercal = \\
&= \Big(\text{diag}(\lambda_1, \ldots, \lambda_d)\text{diag}(\lambda_1, \ldots, \lambda_d) + \text{diag}(\alpha, \ldots, \alpha)\Big)^{-1} \text{diag}(\lambda_1, \ldots, \lambda_d) = \\
&= \text{diag}\left(\frac{\lambda_1}{\lambda_1^2 + \alpha}, \ldots, \frac{\lambda_d}{\lambda_d^2 + \alpha}\right),
\end{aligned}$$

smoothing diagonal values in $\boldsymbol{R}$

More regularizers!

- L2 regularized multivariate linear regression problem

$$\|\boldsymbol{y} - \boldsymbol{X}\mathbf{w}\|^2 + \alpha\|\mathbf{w}\|^2 \to \min_{\mathbf{w}\in\mathbb{R}^d}$$

- L1 regularized regression (LASSO)

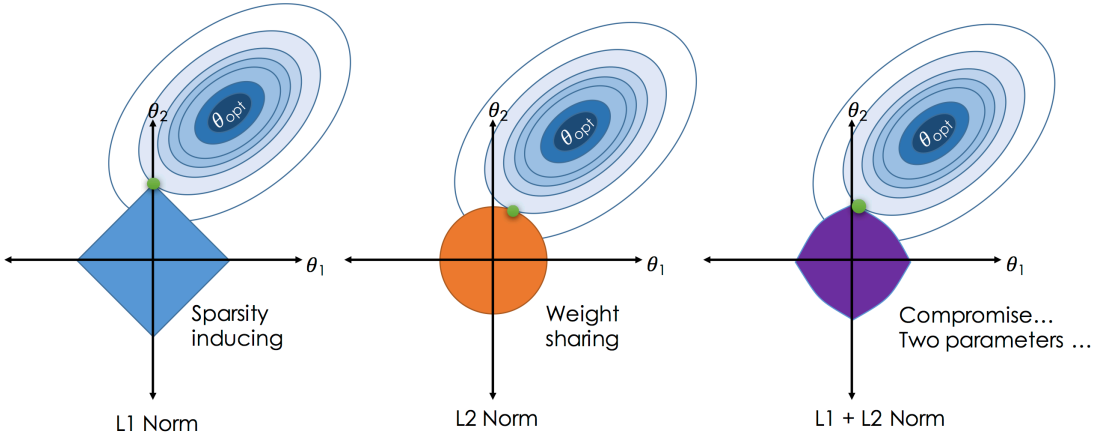
$$\|\boldsymbol{y} - \boldsymbol{X}\mathbf{w}\|^2 + \alpha\|\mathbf{w}\|_1 \to \min_{\mathbf{w}\in\mathbb{R}^d}$$

- L1/L2 regularized regression (Elastic Net)

$$\|\boldsymbol{y} - \boldsymbol{X}\mathbf{w}\|^2 + \alpha_1\|\mathbf{w}\|_1 + \alpha_2\|\mathbf{w}\|_2^2 \to \min_{\mathbf{w}\in\mathbb{R}^d}$$

- Convex $Q(\mathbf{w})$: unconstrained optimization $Q(\mathbf{w}) + \alpha\|\mathbf{w}\|_1$ is equivalent to constrained problem $Q(\mathbf{w})$ s.t. $\|\mathbf{w}\|_1 \leqslant C$

Geometric interpretation of regularizers



Picture credit: http://www.ds100.org/sp17/assets/notebooks/linear_regression/Regularization.html

Another interpretation of regularizers

Figure: Large parameter space

Figure: Regularized models

Why Machine Learning works: a bit of Learning Theory

Empirical Risk Minimization framework

- Given a labeled sample $X^\ell = \left\{(\mathbf{x}_i, y_i)\right\}_{i=1}^{\ell}$ and some candidate h
- Define the empirical error of h as $Q(h, X^\ell) = \frac{1}{\ell}\sum_{i=1}^{\ell}[f(\mathbf{x}_i) \neq h(\mathbf{x}_i)]$ (the proportion of sample points on which h errs)
- $\mathrm{ERM}(X^\ell)$ — find h the minimizes $Q(h, X^\ell)$
- No learning is possible without applying prior knowledge
- A hypothesis class $\mathbb{H}$ is a set of hypotheses
- Re-define the ERM rule by searching only inside such a prescribed $\mathbb{H}$
- $\mathrm{ERM}_{\mathbb{H}}(X^\ell)$ picks a classifier $h \in \mathbb{H}$ that minimizes the empirical error over members of $\mathbb{H}$

Guarantees for ERM

Theorem (Guaranteed success for $\mathrm{ERM}_{\mathbb{H}}$)

- Let $\mathbb{H}$ be a finite class.
- Let the unknown labeling rule, f, be a member of $\mathbb{H}$.
- Then
  - for every $\varepsilon, \delta > 0$, if $m > (\log(|\mathbb{H}|) + \log(1/\delta))/\varepsilon$,
  - with probability $> 1 - \delta$ (over the choice of X^{ℓ}),
  - any $\mathrm{ERM}_{\mathbb{H}}$ hypothesis has error below ε.

Conclusion

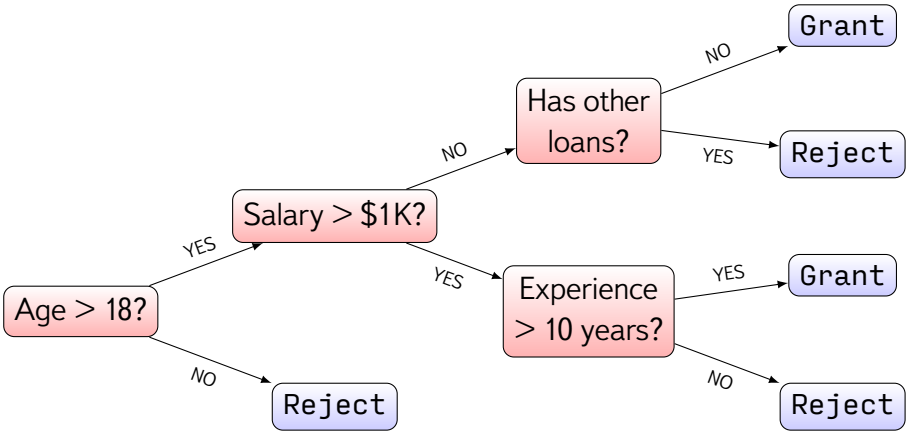
- Linear models - one of the simpliest but quite popular
- Can be used for bot Regression and Classification
- k Nearest Neighbours - another Classification algorithm
- Figure of Merits: meaning of area under ROC curve
- Indicators of overfitting and how to deal with it (by putting cat in a glass)

Part 2

- Decision Trees
- Bagging and Random Forests
- Learning Theory continued
- Stacked generalization

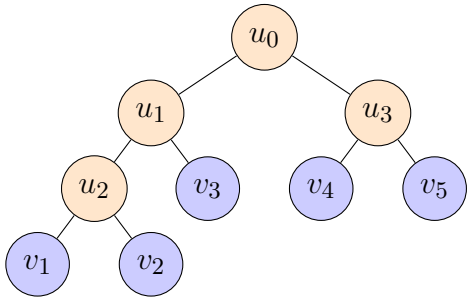
Decision trees

Decision making at a bank



Decision tree formalism

- Decision tree is a binary tree V
- Internal nodes $u \in V$: predicates $\beta_u : \mathbb{X} \to \{0, 1\}$
- Leafs $v \in V$: predictions x
- Algorithm $h(\mathbf{x})$ starts at $u = u_0$
  - Compute $b = \beta_u(\mathbf{x})$
  - If $b = 0$, $u \leftarrow \text{LeftChild}(u)$
  - If $b = 1$, $u \leftarrow \text{RightChild}(u)$
  - If u is a leaf, return b
- In practice: $\beta_u(\mathbf{x}; j, t) = [\mathbf{x}_j < t]$



Greedy tree learning for binary classification

› Input: training set $X^\ell = \left\{(\mathbf{x}_i, y_i)\right\}_{i=1}^{\ell}$

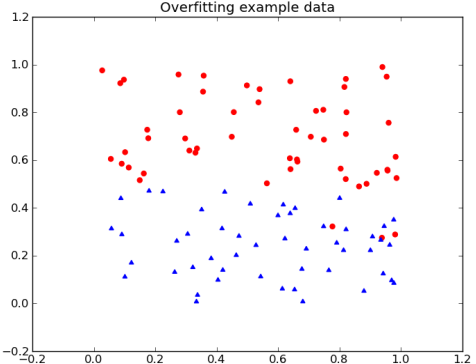
1. Greedily split X^ℓ into R_1 and R_2:

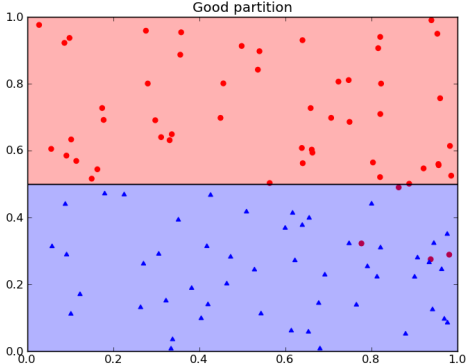
$$R_1(j,t) = \{\mathbf{x} \in X^\ell | \mathbf{x}_j < t\}, \qquad R_2(j,t) = \{\mathbf{x} \in X^\ell | \mathbf{x}_j > t\}$$

   optimizing a given loss: $Q(X^\ell, j, t) \to \min\limits_{(j,t)}$
2. Create internal node u corresponding to the predicate $[\mathbf{x}_j < t]$
3. If a stopping criterion is satisfied for u,
   declare it a leaf, setting some $c_u \in \mathbb{Y}$ as leaf prediction
4. If not, repeat 1–2 for $R_1(j,t)$ and $R_2(j,t)$

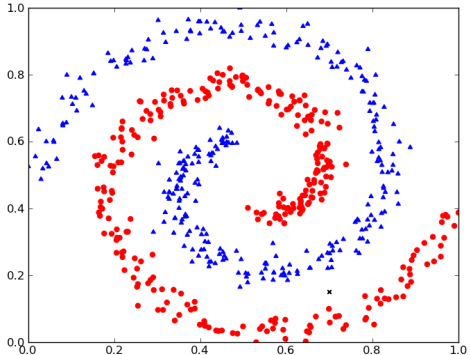
› Output: a decision tree V

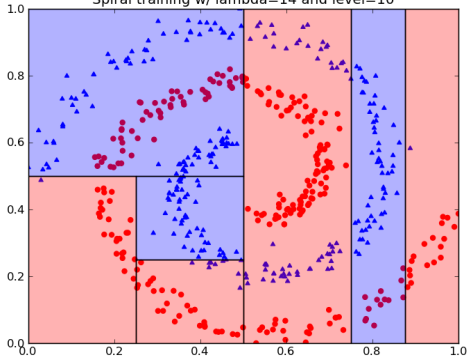
Greedy tree learning for binary classification



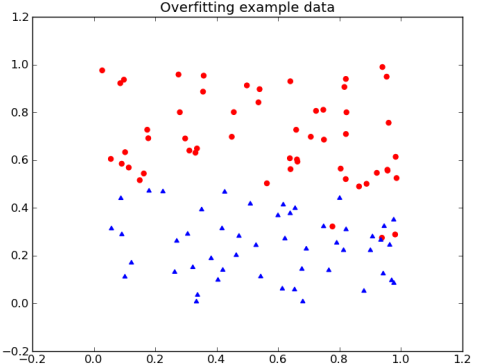


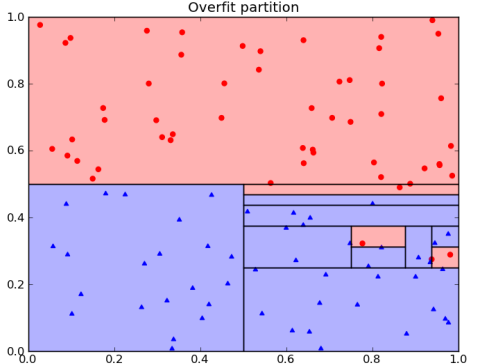
Greedy tree learning for binary classification





With decision trees, overfitting is extra-easy!





Design choices for learning a decision tree classifier

- Type of predicate in internal nodes
- The loss function $Q(X^\ell, j, t)$
- The stopping criterion
- Hacks: missing values, pruning, etc.

- CART, C4.5, ID3

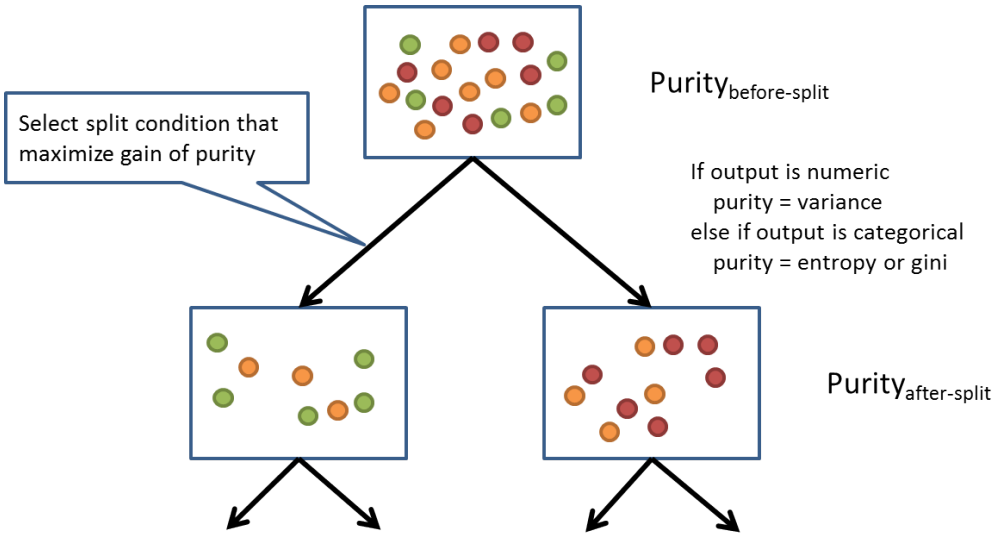
The loss function $Q(X^\ell, j, t)$

- R_m: the subset of X^ℓ at step m
- With the current split, let $R_l \subseteq R_m$ go left and $R_l \subseteq R_m$ go right
- Choose predicate to optimize

$$Q(R_m, j, t) = H(R_m) - \frac{|R_l|}{|R_m|} H(R_l) - \frac{|R_r|}{|R_m|} H(R_r) \to \max$$

- $H(R)$: impurity criterion
- Generally $H(R) = \min\limits_{c \in \mathbb{Y}} \frac{1}{|R|} \sum\limits_{(\mathbf{x}_i, y_i) \in R} L(y_i, c)$

The idea: maximize purity



Picture credit: https://dzone.com/refcardz/machine-learning-predictive

Examples of information criteria

- Regression:
  - $H(R) = \min\limits_{c \in \mathbb{Y}} |R|^{-1} \sum_{(\mathbf{x}_i, y_i) \in R} (y_i - c)^2$
  - Sum of squared residuals minimized by $c = |R|^{-1} \sum_{(\mathbf{x}_j, y_j) \in R} y_j$
  - Impurity $\equiv$ variance of the target
- Classification:
  - Let $p_k = |R|^{-1} \sum_{(\mathbf{x}_i, y_i) \in R} [y_i = k]$ (share of y_i's equal to k)
  - Miss rate: $H(R) = \min\limits_{c \in \mathbb{Y}} |R|^{-1} \sum_{(\mathbf{x}_i, y_i) \in R} [y_i \neq c]$

    Minimizing miss rate $1 - p_{k_*}$,

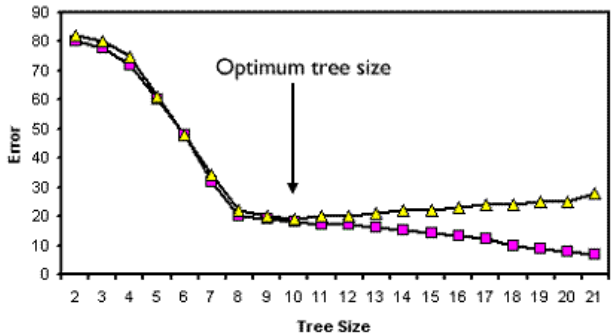
    Gini index $\sum_{k=1}^{K} p_k (1 - p_k)$,

    Cross-entropy $- \sum_{k=1}^{K} p_k \log p_k$

Stopping rules for decision tree learning

- Significantly impacts learning performance
- Multiple choices available:
  - Maximum tree depth
  - Minimum number of objects in leaf
  - Maximum number of leafs in tree
  - Stop if all objects fall into same leaf
  - Constrain quality improvement
    (stop when improvement gains drop below s%)
- Typically selected via exhaustive search and cross-validation

Decision tree pruning

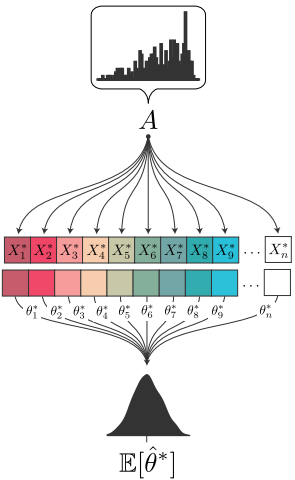


- Learn a large tree (effectively overfit the training set)
- Detect overfitting via K-fold cross-validation
- Optimize structure by removing least important nodes

Bagging and Random Forests

The bootstrapping procedure

- Input: a sample $X^\ell = \{(x_i, y_i)\}_{i=1}^\ell$
- Bootstrapping: generate new samples X_1^m of (x_i, y_i) drawn from X^ℓ uniformly at random with replacement (replicated (x_i, y_i) possible!)
- Ensemble learning idea:
  1. Generate N bootstrapped samples $X_1^m, \ldots, X_N^m$
  2. Learn N hypotheses $h_1, \ldots, h_N$
  3. Average predictions to obtain $h(x) = \frac{1}{N} \sum_{i=1}^N h_i(x)$
  4. Profit!

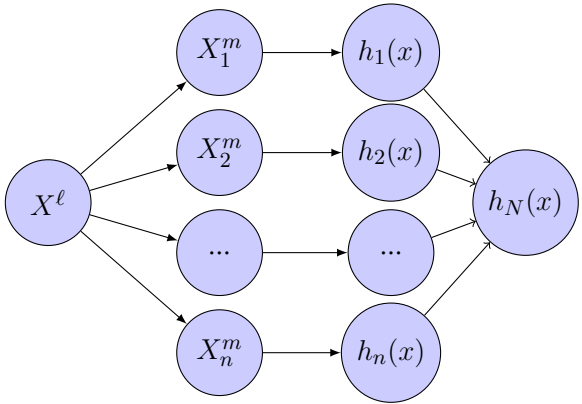


Picture credit: http://www.drbunsen.org/bootstrap-in-picture

Bagging: bootstrap aggregation (+ demo)

- Input: a sample $X^\ell = \{(x_i, y_i)\}_{i=1}^{\ell}$
- Weak learners via bootstrapping $\tilde{\mu}(X^\ell) = \mu(\tilde{X}^\ell)$
- Ensemble average

$$h_N(x) = \frac{1}{N}\sum_{i=1}^{N} h_i(x) = \\ = \frac{1}{N}\sum_{i=1}^{N} \tilde{\mu}(X^\ell)(x)$$



The Random Forest algorithm

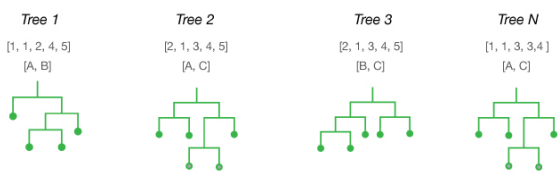
- Bagging over (weak) decision trees
- Reduce error via averaging over instances and features
- Input: a sample $X^\ell = \{(\mathbf{x}_i, y_i)\}_{i=1}^\ell$, where $\mathbf{x}_i \in \mathbb{R}^d, y_i \in \mathbb{Y}$
- The algorithm iterates for $i = 1, \ldots, N$:
  1. Pick p random features out of d
  2. Bootstrap a sample $X_i^m = \{(\mathbf{x}_i, y_i)\}_{i=1}^\ell$ where $\mathbf{x}_i \in \mathbb{R}^p, y_i \in \mathbb{Y}$
  3. Learn a decision tree $h_i(\mathbf{x})$ using bootstrapped X_i^m
  4. Stop when leafs in h_i contain less that n_{min} instances

$\mathbf{x}_i \in \{\text{A}, \text{B}, \text{C}\}$

$X^\ell = \{(\mathbf{x}_i, y_i)\}_{i=1}^5$

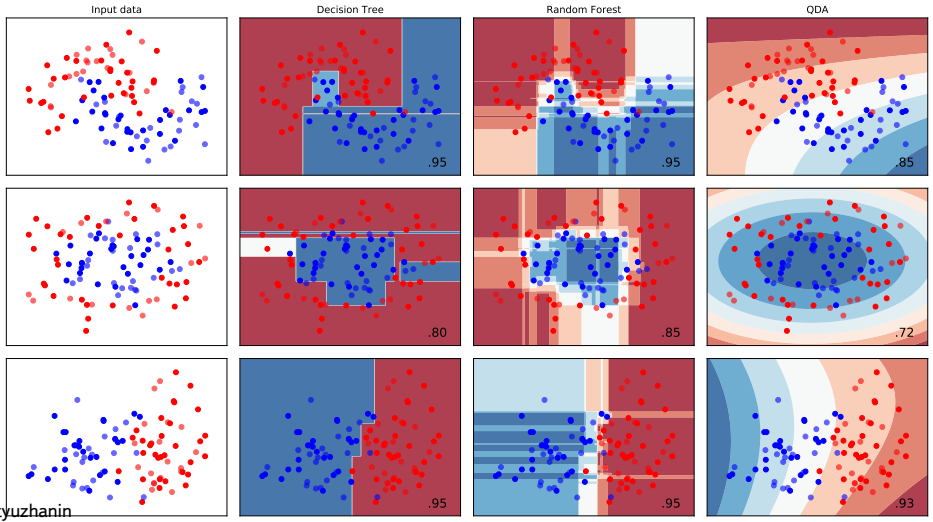
Bootstrap $X_i^m, i \in \{1, 2, 3, 4\}$

Learn Tree$_i(\mathbf{x})$ using X_i^m



Picture credit: http://www.thefactmachine.com/random-forests

Random Forest: synthetic examples



Learning Theory (continued)

Expected risk formalism

- Input: the training set $X^\ell = \{(x_i, y_i)\}_{i=1}^\ell$
- Suppose $(x_i, y_i) \in \mathbb{X} \times \mathbb{Y}$ are generated from a distribution $p(x, y)$
- Consider the MSE loss $Q(y, h) = \big(y - h(x)\big)^2$
- Expected risk: average (over $p(x, y)$) squared loss when using h

$$R(h) = \mathbb{E}_{x,y}\Big[\big(y - h(x)\big)^2\Big] = \int_{\mathbb{X}}\int_{\mathbb{Y}} p(x, y)\big(y - a(x)\big)^2 dxdy.$$

- A statement: the expected risk is minimized by

$$h_*(x) = \mathbb{E}[y \,|\, x] = \int_{\mathbb{Y}} y p(y \,|\, x) dy = \arg\min_h R(h)$$

Bias-variance decomposition

- Input: the training set $X^\ell = \{(x_i, y_i)\}_{i=1}^\ell$
- Learning method $\mu : (\mathbb{X} \times \mathbb{Y})^\ell \to \mathbb{H}$
- Can evaluate quality using average (over possible samples X^ℓ)

$$Q(\mu) = \mathbb{E}_{X^\ell}\left[\mathbb{E}_{x,y}\left[\left(y - \mu(X^\ell)(x)\right)^2\right]\right] =$$

$$= \int_{(\mathbb{X}\times\mathbb{Y})^\ell} \int_{\mathbb{X}\times\mathbb{Y}} \left(y - \mu(X^\ell)(x)\right)^2 p(x,y) \prod_{i=1}^{\ell} p(x_i, y_i) dx dy dx_i dy_i$$

Bias-variance decomposition

› We arrive at the famous **bias-variance**(-noise) **decomposition**

$$Q(\mu) = \underbrace{\mathbb{E}_{x,y}\Big[\big(y - \mathbb{E}[y \mid x]\big)^2\Big]}_{\text{noise}} +$$

$$+ \underbrace{\mathbb{E}_x\Big[\big(\mathbb{E}_{X^\ell}\big[\mu(X^\ell)\big] - \mathbb{E}[y \mid x]\big)^2\Big]}_{\text{bias}} + \underbrace{\mathbb{E}_x\Big[\mathbb{E}_{X^\ell}\Big[\big(\mu(X^\ell) - \mathbb{E}_{X^\ell}\big[\mu(X^\ell)\big]\big)^2\Big]\Big]}_{\text{variance}}$$

› Noise term: an error of an ideal learner (nobody can do better!)

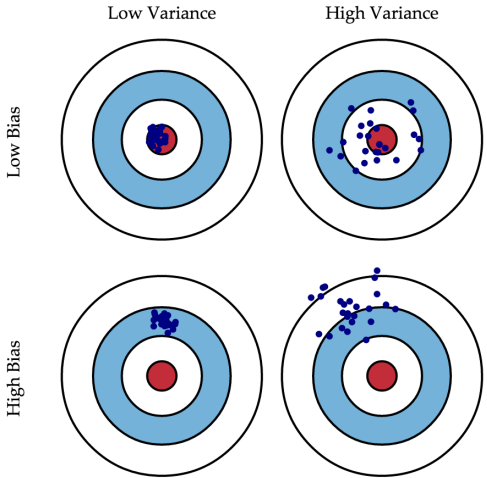
› Bias term: learner's approximation of the ideal algorithm

  › The more complex the learning algorithm, the lower the bias

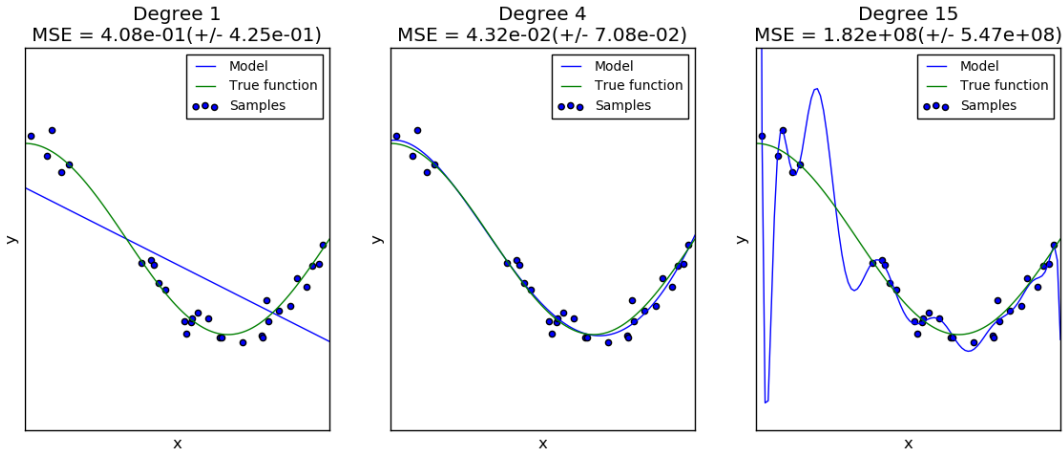
› Variance term: sensitivity to sample replacement

  › Simple algorithms have lower variance

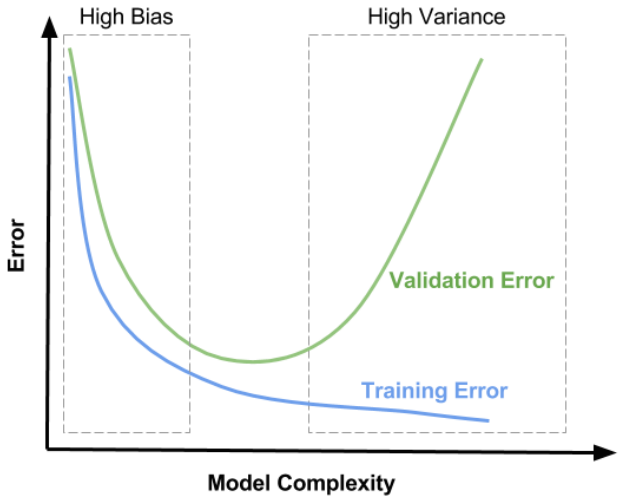
Bias-variance decomposition



Polynomial fits of different degrees



Bias-variance tradeoff



Picture credit: http://www.learnopencv.com/bias-variance-tradeoff-in-machine-learning

An application: statistical analysis of Bagging

› Bias: not made any worse by bagging multiple hypotheses

$$\mathbb{E}_{x,y}\left[\left(\mathbb{E}_{X^\ell}\left[\frac{1}{N}\sum_{n=1}^{N}\tilde{\mu}(X^\ell)(x)\right] - \mathbb{E}[y \mid x]\right)^2\right] =$$

$$= \mathbb{E}_{x,y}\left[\left(\frac{1}{N}\sum_{n=1}^{N}\mathbb{E}_X^\ell[\tilde{\mu}(X^\ell)(x)] - \mathbb{E}[y \mid x]\right)^2\right] =$$

$$= \mathbb{E}_{x,y}\left[\left(\mathbb{E}_{X^\ell}\left[\tilde{\mu}(X^\ell)(x)\right] - \mathbb{E}[y \mid x]\right)^2\right]$$

An application: statistical analysis of Bagging

› Variance: N times lower for uncorrelated hypotheses,
yet not as much an improvement for highly correlated

$$\mathbb{E}_{x,y}\left[\mathbb{E}_{X^\ell}\left[\left(\frac{1}{N}\sum_{n=1}^{N}\tilde{\mu}(X^\ell)(x) - \mathbb{E}_{X^\ell}\left[\frac{1}{N}\sum_{n=1}^{N}\tilde{\mu}(X^\ell)(x)\right]\right)^2\right]\right] =$$

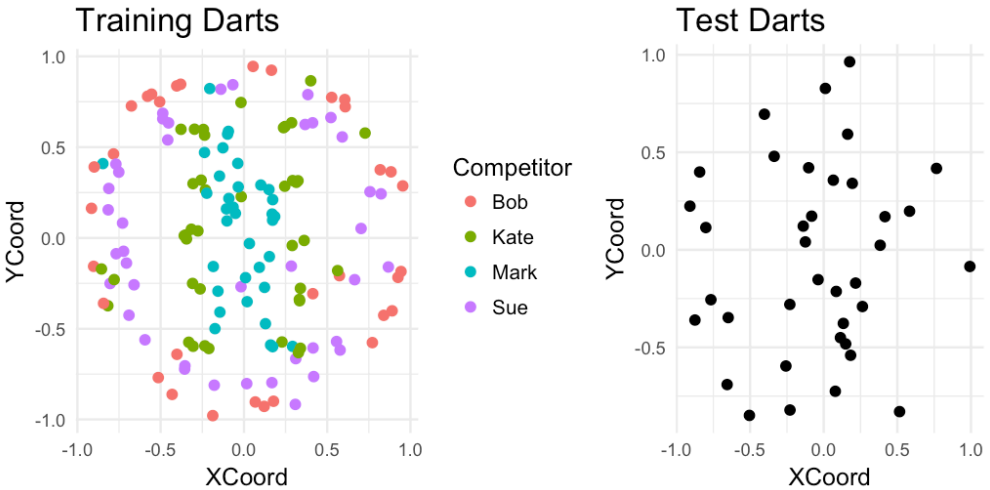
$$= \frac{1}{N}\mathbb{E}_{x,y}\left[\mathbb{E}_{X^\ell}\left[\left(\tilde{\mu}(X^\ell)(x) - \mathbb{E}_{X^\ell}\left[\tilde{\mu}(X)(x)\right]\right)^2\right]\right] +$$

$$+ \frac{N(N-1)}{N^2}\mathbb{E}_{x,y}\left[\mathbb{E}_{X^\ell}\left[\left(\tilde{\mu}(X^\ell)(x) - \mathbb{E}_{X^\ell}\left[\tilde{\mu}(X^\ell)(x)\right]\right)\times\right.\right.$$

$$\left.\left.\times\left(\tilde{\mu}(X^\ell)(x) - \mathbb{E}_{X^\ell}\left[\tilde{\mu}(X^\ell)(x)\right]\right)\right]\right]$$

Stacked generalization

Stacking motivation: the game of Darts



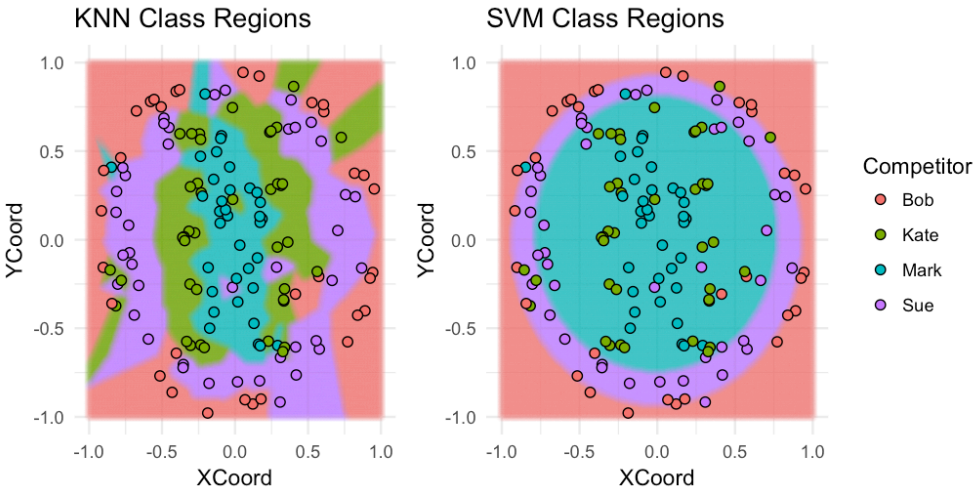
Picture credit: http://blog.kaggle.com/2016/12/27/a-kagglers-guide-to-model-stacking-in-practice

Base model training

- Select k nearest neighbours as base model 1
- Fit base model 1 in the most fancy way possible
  (grid search for optimal k using K-fold cross-validation, etc.)
- k-NN accuracy on Test Darts: 70%

- Select Support Vector Machine as base model 2
- Fit base model 2 in the most fancy way possible
  (different penalizations, grid search for optimal kernel width using K-fold cross-validation, etc.)
- SVM accuracy on Test Darts: 78%

Results for base models



Picture credit: http://blog.kaggle.com/2016/12/27/a-kagglers-guide-to-model-stacking-in-practice

Stacking base models

1. Partition `train` into 5 folds
2. Create `train_meta/test_meta`: same `row/fold Ids` as in `train/test`, empty `M1/M2`
3. For each $\mathrm{Fold}_i \in \{\mathrm{Fold}_1, \ldots, \mathrm{Fold}_5\}$
   3.1 Combine the other 4 folds for training $\to \mathrm{Fold}_{-i}$
   3.2 Fit each base model to Fold_{-i}, predict on Fold_i, save prdections to `M1/M2` in `train_meta`
4. Fit each base model to `train`, predict on `test`, save predictions to `M1/M2` in `test_meta`
5. Fit stacking model S (LinearRegression) to `train_meta`, using `M1/M2` as features
6. Use the stacked model S to make final predictions on `test_meta`

Results for base models

Picture credit: http://blog.kaggle.com/2016/12/27/a-kagglers-guide-to-model-stacking-in-practice

Conclusion

- Decision trees: intuitive and interpretable, yet prone to overfitting
- Bootstrapping: a general statistical technique for computing sample functionals (and their variance)
- Bagging: meta-learner over arbitrary weak algorithms via bootstrap aggregation
- The Random Forest algorithm: Bagging over decision trees
- Stacked generalization: blend output of weak learners (weak signals) with raw features