Machine Learning for Industry

Lecture 3 - Classification and Learning Principles

Mattias Villani

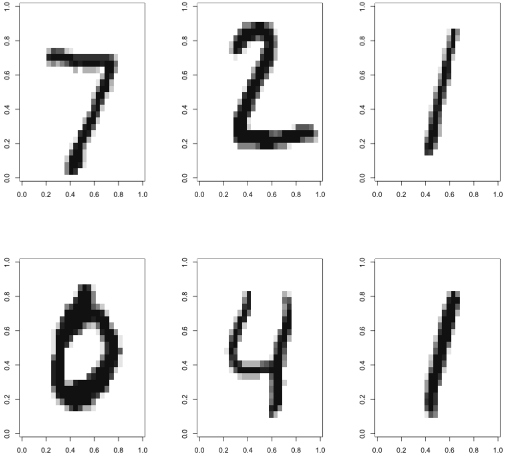
Department of Computer and Information Science
Linköping University

Department of Statistics
Stockholm University

Outline

- Intro to classification
- k-nearest neighbor
- Classification trees and forests
- Logistic regression
- Maximum likelihood
- Naive Bayes
- Bayesian learning and regularization

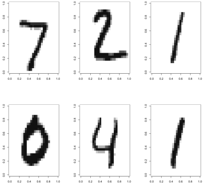
Classifying handwritten digits



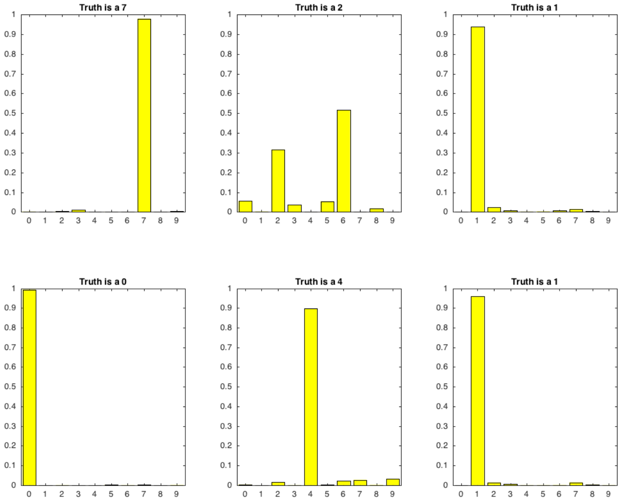
Classifying handwritten digits

- **Raw data**: gray intensity (0-255) in the $28 \times 28 = 784$ pixels.
- **Training** data: 60000 images. **Test** data: 10000 images.
- **Simple features**: 784 pixel intensity covariates.
- **Multi-class** problem: predict class $k \in \{0, 1, ..., 9\}$.
- **Multinomial regression**

$$\Pr(\text{Digit} = k | \text{features}) = \frac{\exp\left(w_{0,k} + w_{1,k}x_1 + ... + w_{784,k}x_{784}\right)}{\sum_{j=0}^{9} \exp\left(w_{0,j} + w_{1,j}x_1 + ... + w_{784,j}x_{784}\right)}$$



Classifying handwritten digits



Handwritten digits 10000 training examples

		Truth									
		0	1	2	3	4	5	6	7	8	9
Decision	0	958	0	8	3	1	7	10	0	7	9
	1	0	1116	3	1	1	5	3	23	9	9
	2	1	2	920	21	5	5	9	22	7	2
	3	0	2	10	915	0	34	1	1	14	10
	4	1	0	16	0	908	9	10	12	11	46
	5	11	3	3	31	2	795	15	1	31	12
	6	6	4	20	2	11	16	909	0	13	1
	7	1	0	15	13	4	5	0	938	9	22
	8	2	8	34	16	2	12	1	5	859	5
	9	0	0	3	8	48	4	0	26	14	893

Handwritten digits 60000 training examples

		Truth									
		0	1	2	3	4	5	6	7	8	9
	0	966	0	8	1	1	7	9	2	4	6
	1	0	1121	1	1	0	2	3	13	7	7
	2	2	2	957	13	5	4	4	21	7	0
	3	0	2	9	947	0	29	1	3	12	10
	4	0	0	12	1	940	5	5	9	8	32
Decision	5	6	1	3	19	1	816	9	1	24	9
	6	4	4	13	1	7	12	926	0	10	1
	7	1	0	9	10	2	2	0	954	5	13
	8	1	4	17	11	2	10	1	3	892	4
	9	0	1	3	6	24	5	0	22	5	927

AI is getting better over time - handwritten digits

Time:	1998	– – – – – – – – – $\longrightarrow$		Today

	Logistic	K-nearest	SVM	3-layer NN	ConvNet
Error rate:	12%	5%	1.4%	1.53%	0.4%

Detecting fraudulent banknotes

- Dataset with 1372 photographed banknotes. 610 fake.
- Raw data: $400 \times 400 = 160000$ gray-scale pixels.
- The 160000 pixel variables are condensed to four features:
  - variance of Wavelet Transformed image
  - skewness of Wavelet Transformed image
  - k´urtosis of Wavelet Transformed image
  - entropy of image
- Deep learning on raw images?

Detecting fraudulent banknotes

- 1000 images for training. Predictions on 372 test images.
- Logistic regression

$$\Pr(\text{Fraud} = \text{True}|\text{features}) = \frac{\exp\left(w_0 + w_1 x_1 + \ldots + w_4 x_4\right)}{1 + \exp\left(w_0 + w_1 x_1 + \ldots + w_4 x_4\right)}$$

- **Decision**: signal fraud if $\Pr(\text{Fraud} = \text{True}|\text{Features}) > 0.5$.
- **Confusion matrix**

		Truth	
		No fraud	Fraud
Decision	No Fraud	208	1
	Fraud	3	160

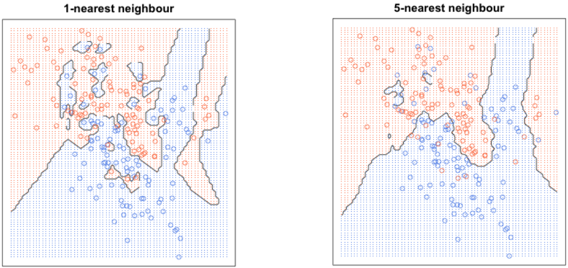
k-nearest neighbor

- *k*-nearest classifier: classify to the local majority vote:

$$\hat{y}(\boldsymbol{x}) = \begin{cases} 0 & \text{if } f(\boldsymbol{x}) \leq 0.5 \\ 1 & \text{if } f(\boldsymbol{x}) > 0.5 \end{cases}$$

where

$$f(\boldsymbol{x}) = \frac{\#\text{class } 1 \text{ among the } k \text{ nearest neighbors}}{k}.$$



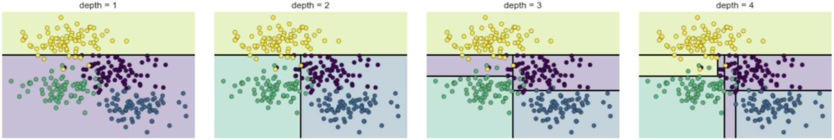
Classification trees

- **Multi-class classification trees**. Probability of class k in rectangle R_m with N_m observations

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I\left(y_i = k\right), \quad k = 1, \ldots, K.$$

- **Predicted class** in rectangle R_m: **majority vote**

$$\hat{k}(m) = \arg\max_k \hat{p}_{mk}$$



Classification trees

- **Prune** the tree by collapsing non-terminal nodes to minimize

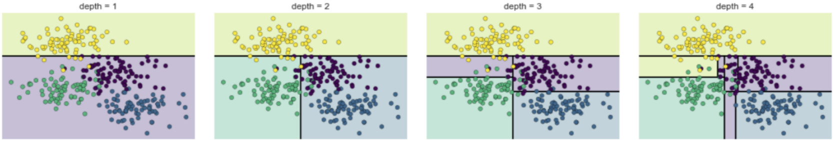
$$\sum_{i=1}^{n} \ell\left(y_i, \hat{k}(m_i)\right) + \eta\,|T|$$

- **Mis-classification rate** as loss function:

$$\sum_{i=1}^{n} \ell\left(y_i, \hat{k}(m_i)\right) = \sum_{m=1}^{|T|} \sum_{x_i \in R_m} I\left(y_i \neq \hat{k}(m_i)\right) = \sum_{m=1}^{|T|} N_m \left(1 - \hat{p}_{m,\hat{k}(m)}\right)$$

- **Cross-entropy** as loss function:

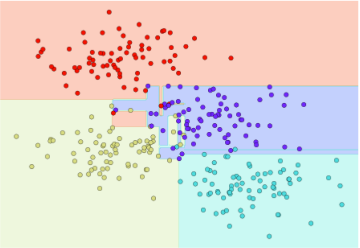
$$\sum_{i=1}^{n} \ell\left(y_i, \hat{k}(m_i)\right) = \sum_{m=1}^{|T|} N_m \left(-\sum_{k=1}^{K} \hat{p}_{m\hat{k}} \log \hat{p}_{m\hat{k}}\right)$$



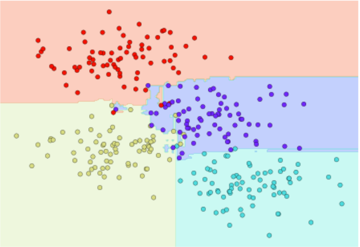
Random forest and XGBoost classification

- **Random Forest** can be directly used for classification.
- **XGBoost**
  - second approximation of a general loss function $\sum_{i=1}^{n} \ell\left(y_i, \hat{y}\right)$
  - loss function must be differentiable (e.g. cross-entropy).

Single tree



Random forest



Classification - logistic regression

- **Logistic classification**

$$\Pr(Y_i = 1|\boldsymbol{x}) = \frac{\exp\left(\boldsymbol{x}_i^T \boldsymbol{w}\right)}{1 + \exp\left(\boldsymbol{x}_i^T \boldsymbol{w}\right)}$$

- **Discriminative** model - direct modeling of $\Pr(Y = 1|\boldsymbol{x})$
- **Multi-class logistic** classification: $c \in \{1, 2, \ldots, C\}$

$$\Pr(Y_i = c|\boldsymbol{x}) = \frac{\exp\left(\boldsymbol{x}_i^T \boldsymbol{w}_c\right)}{\sum_{k=1}^{C} \exp\left(\boldsymbol{x}_i^T \boldsymbol{w}_k\right)} \text{ with } \boldsymbol{w}_1 = 0.$$

.

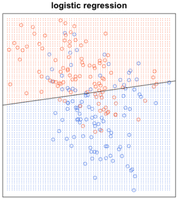
- How to learn the weights, $\boldsymbol{w}$?
  - Quadratic fitting function: $n^{-1} \sum_{i=1}^{n} (y_i - x_i^T \boldsymbol{w})^2$?
  - Fit weights to maximize accuracy on training data?
- **Maximum likelihood**!

Classification - logistic regression

- Logistic regression is a linear model:

$$\log \frac{\Pr(Y_i = 1|\boldsymbol{x})}{\Pr(Y_i = 0|\boldsymbol{x})} = \boldsymbol{x}_i^T \boldsymbol{w}$$

and so has linear decision boundary.



- Get non-linear boundaries by including polynomials in $\boldsymbol{x}$.
- General non-linear logistic regression

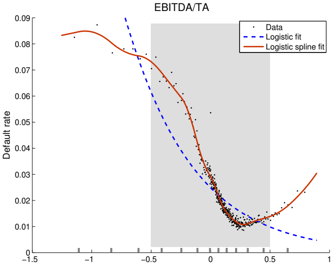
$$\Pr(Y_i = 1|\boldsymbol{x}) = \frac{\exp\left(f(\boldsymbol{x}_i)\right)}{1 + \exp\left(f(\boldsymbol{x}_i)\right)}$$

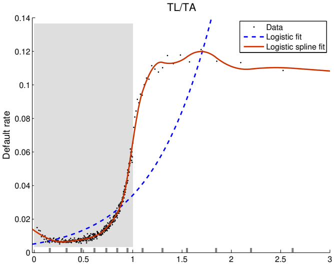
where $f(\boldsymbol{x})$ is some potentially non-linear function, e.g.

- Deep neural nets
- Gaussian Processes

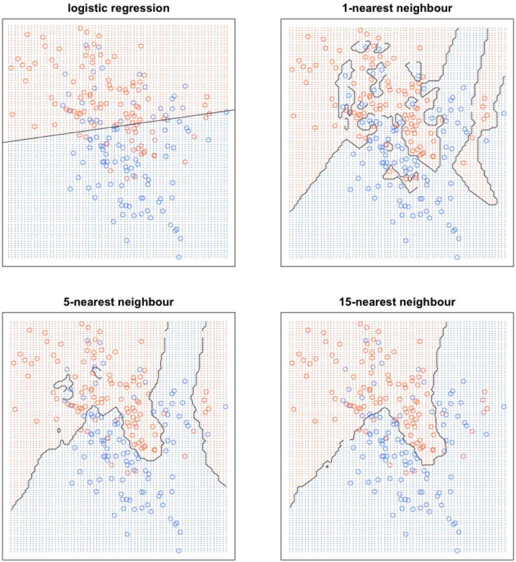
Predicting firm bankruptcy

- Data from $\sim 250,000$ Swedish firms (aktiebolag).
- **Features**: profits, liquidity, debt + macro variables.
- "Big data" $\Rightarrow$ **non-linearities** are visible by the eye.
- Model: **logistic regression** with **additive splines**.
- Substantially improved predictive performance with splines.





Bias-variance trade-off revisited



The likelihood function

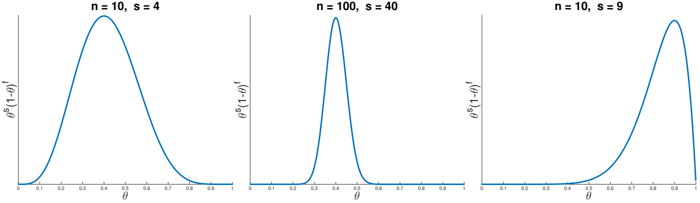
- Let $Y_1, ..., Y_n$ be binary variables (e.g. coin flips) with

$$\Pr(Y = 0) = 1 - \theta \quad \text{and} \quad \Pr(Y = 1) = \theta.$$

- **Data summary**: s successes $(y = 1)$ and f failures $(y = 0)$.
- **Likelihood function**

$$L(\theta) = p(y_1, ..., y_n|\theta) \quad = \quad p(y_1|\theta) \cdots p(y_n|\theta) = \theta^s (1 - \theta)^f$$

- The likelihood, $L(\theta)$, is:
  - **the probability of the observed data**
  - **considered as a function of the parameter**.
- Plot $p(y_1, ..., y_n|\theta)$ as a function of θ for a given dataset.



Maximum likelihood estimation

- Maximum likelihood estimator (MLE) - max the likelihood

$$\hat{\theta}_{ML} = \arg\max_{\theta} L(\theta).$$

- General fitting method. Applies to all probabilistic models.
- Example: binary independent data. Easier on log-scale

$$\log L(\theta) = s \log \theta + f \log(1-\theta)$$

- Compute derivative, set it equal to zero and solve:

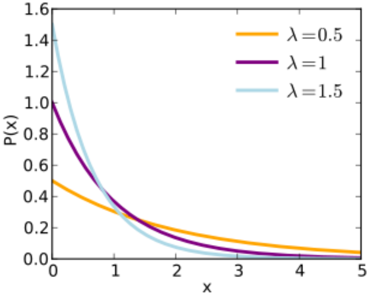
$$\frac{\partial}{\partial \theta} \log L(\theta) = \frac{s}{\theta} - \frac{f}{1-\theta} = 0$$

  gives $\hat{\theta}_{ML} = s/n$, the proportion of successes.
- Regression with Gaussian errors: ML = Least squares.
- MLE minimizes the cross-entropy between data and model.

Exponential distribution

- Exponential random variable $Y \sim \mathrm{Exp}(\lambda)$.
- Continuous variable with positive support $Y \in (0, \infty)$.
- Probability density function (pdf)

$$f(y) = \lambda \exp(-\lambda y).$$



- Mean $\mathbb{E}(Y) = \frac{1}{\lambda}$ and variance $\mathbb{V}(Y) = \frac{1}{\lambda^2}$.

ML for exponential data

- **Likelihood function** based on pdfs:

$$f(y_1, \dots, y_n) = \prod_{i=1}^{n} f(y_i) = \prod_{i=1}^{n} \lambda \exp(-\lambda y_i) = \lambda^n \exp\left(-\lambda \sum_{i=1}^{n} y_i\right)$$

- **Log-likelihood**

$$\log f(y_1, \dots, y_n) = n \log \lambda - \lambda \sum_{i=1}^{n} y_i$$

- **ML estimator**

$$\frac{\partial}{\partial \lambda} \log f(y_1, \dots, y_n) = \frac{n}{\lambda} - \sum_{i=1}^{n} y_i = 0$$

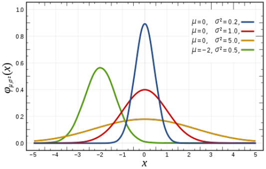
gives

$$\hat{\lambda}_{ML} = \frac{n}{\sum_{i=1}^{n} y_i} = \frac{1}{\bar{y}}.$$

Normal distribution

- Normal random variable $Y \sim N(\mu, \sigma^2)$ for $Y \in (-\infty, \infty)$
- Probability density function

$$f(y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(y-\mu)^2\right).$$



- Mean $\mathbb{E}(Y) = \mu$ and variance $\mathbb{V}(Y) = \sigma^2$.
- ML estimates: $\hat{\mu}_{ML} = \bar{y}$ and $\hat{\sigma}^2_{ML} = \sum_{i=1}^{n}(y_i - \bar{y})^2 / n$.

ML for truncated exponential data

- **Truncated exponential** data: $Y^\star \sim \mathrm{Exp}(\lambda)$, but we observe

$$Y = \begin{cases} Y^\star & \text{if } Y^\star < c \\ c & \text{if } Y^\star \geq c \end{cases}$$

- If $Y \sim \mathrm{Exp}(\lambda)$, then $\Pr(Y_i > c) = 1 - \Pr(Y_i \leq c) = e^{-\lambda c}$.
- **Likelihood function** based on pdfs:

$$f(y_1, \ldots, y_n) = \prod_{i \in \mathcal{U}} f(y_i) \prod_{i \notin \mathcal{U}} \Pr(Y_i > c) = \lambda^{n_u} \exp\left(-\lambda \sum_{i \in \mathcal{U}} y_i\right) e^{-\lambda(n-n_u)c}$$

- **Maximum likelihood** estimator

$$\frac{\partial}{\partial \lambda} \log f(y_1, \ldots, y_n) = \frac{n_u}{\lambda} - \sum_{i \in \mathcal{U}} y_i - (n - n_u)c = 0$$

which gives

$$\frac{1}{\hat{\lambda}_{ML}} = \frac{\sum_{i \in \mathcal{U}} y_i}{n_u} + \frac{n - n_u}{n_u} c.$$

Maximum likelihood for logistic regression

- Logistic regression

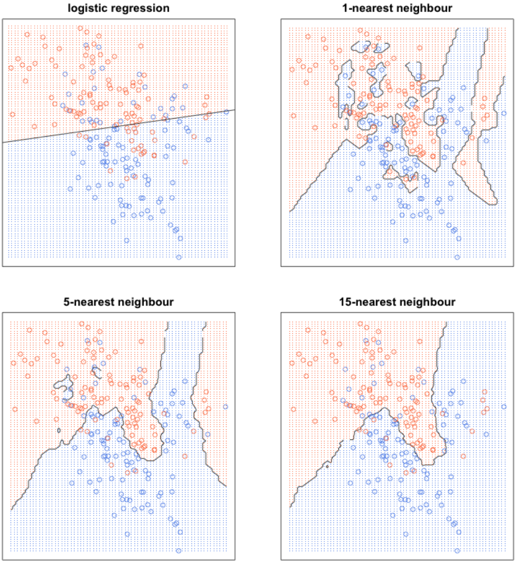
$$\Pr(Y_i = y_i | \boldsymbol{x}) = \frac{\exp\left(\boldsymbol{x}_i^T \boldsymbol{w}\right)^{y_i}}{1 + \exp\left(\boldsymbol{x}_i^T \boldsymbol{w}\right)}$$

- Likelihood

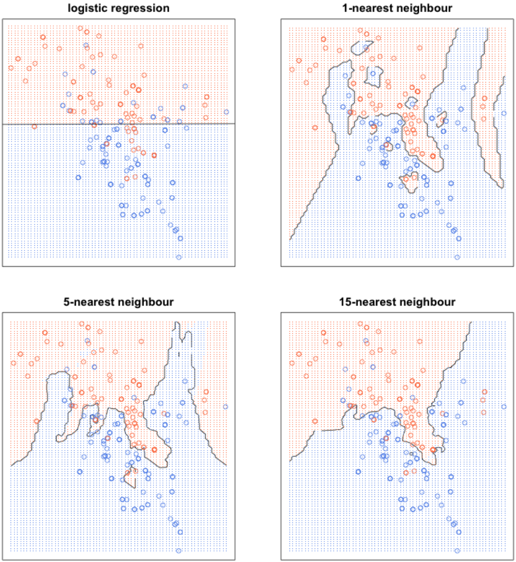
$$p(y_1, \ldots, y_n | \mathbf{x}_1, \ldots, \mathbf{x}_n, \boldsymbol{w}) = \prod_{i=1}^{n} \frac{\exp\left(\boldsymbol{x}_i^T \boldsymbol{w}\right)^{y_i}}{1 + \exp\left(\boldsymbol{x}_i^T \boldsymbol{w}\right)}$$

- No closed form solution. Iteratively reweighted least squares.
- `MLbyOptimization.ipynb` for **numerically** computing $\hat{\boldsymbol{w}}_{ML}$.
- **Gradient** $\nabla_{\boldsymbol{w}} \log p(y_1, \ldots, y_n | \mathbf{x}_1, \ldots, \mathbf{x}_n, \boldsymbol{w})$ easily computed and speeds up optimization (gradient descent).
- Big data: **Stochastic gradient descent** for large n.
- **Automatic differentiation**.

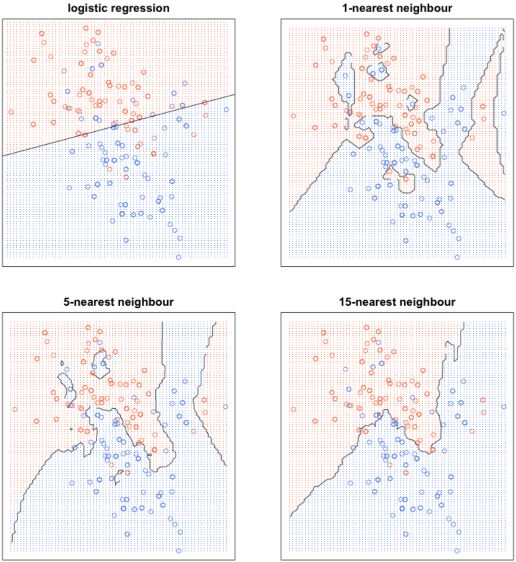
Bias-Variance trade-off revisited



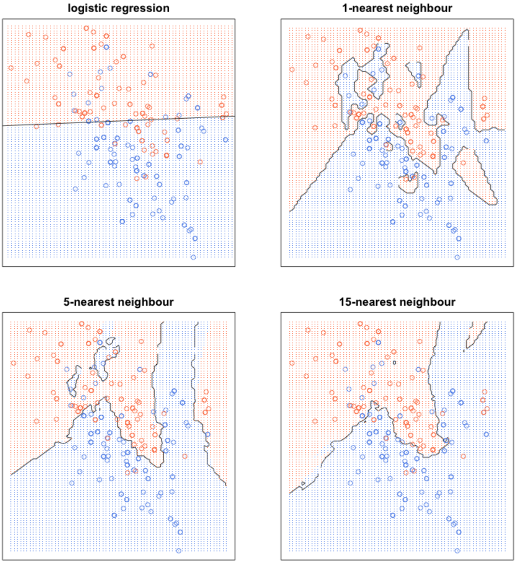
Bootstrap sample no 1



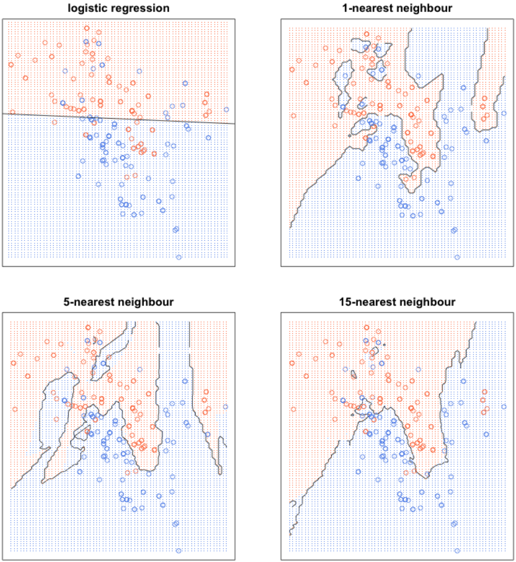
Bootstrap sample no 2



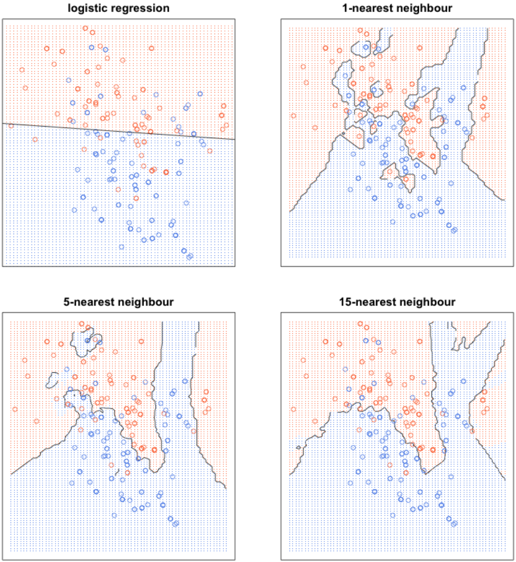
Bootstrap sample no 3



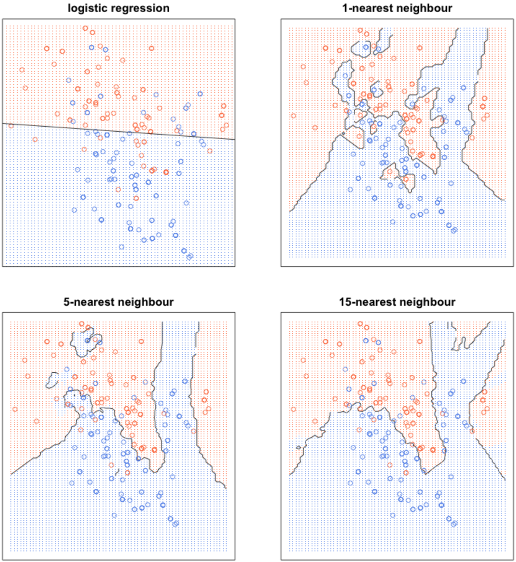
Bootstrap sample no 4



Bootstrap sample no 5



Bootstrap sample no 5



Naive Bayes

- Aim: $p(k|\mathbf{x})$ probability of class k given features $\mathbf{x}$.
- **Bayes' theorem**

$$p(k|\mathbf{x}) \propto p(\mathbf{x}|k)p(k)$$

- $p(k)$ estimated from training data by relative frequencies.
- $p(\mathbf{x}|k)$ can high-dimensional. Hard to estimate.
- **Naive Bayes**: **features are assumed independent**

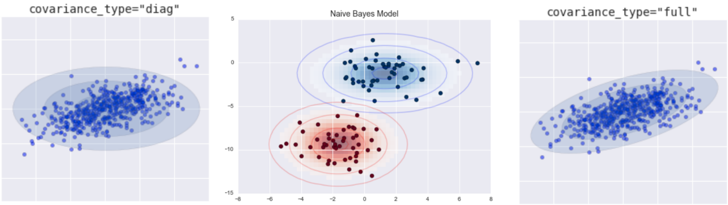
$$p(\mathbf{x}|k) = \prod_{j=1}^{p} p(x_j|k)$$

- Predicting text. Binary features: existence of specific words.

$$\hat{p}\left(\text{has(ball)}|\text{news}\right) = \frac{\text{Number of news articles containing the word 'ball'}}{\text{Number of news articles}}.$$

Naive Bayes for continuous features

- Fit **Gaussian** (normal) distribution to each feature separately.



- More flexible:
  - **kernel density estimate**
  - **mixture of Gaussians** (see Lecture Block 4).

- Alternative: make features **discrete by binning**

$$y = k \text{ if } m_k \leq x < m_{k+1}$$

Bayesian learning

- **Subjective probability**. Subjective degree of belief.
- The statement $\Pr(\text{10th decimal of } \pi = 9) = 0.1$ makes sense.
- Example: Binary data $\Pr(Y_i = 1) = \theta$.
- **Prior distribution**: $p(\theta) = \text{Beta}(\alpha, \beta)$.
- **Likelihood**

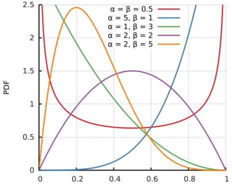
$$p(y_1, ..., y_n|\theta) = \theta^s(1-\theta)^f.$$

- **Posterior distribution** by **Bayes' theorem**:

$$p(\theta|y_1, \ldots, y_n) \propto \underbrace{p(y_1, \ldots, y_n|\theta)}_{\text{likelihood}}\underbrace{p(\theta)}_{\text{prior}}$$

- **Posterior** for Bernoulli data

$$\theta|y_1, \ldots, y_n \sim \text{Beta}(\alpha + s, \beta + f)$$



Bayesian regression

- **Regularization prior**

$$w_i|\sigma^2 \overset{iid}{\sim} N\left(0, \frac{\sigma^2}{\lambda}\right)$$

- Posterior distribution is Gaussian with **ridge regression** as mean

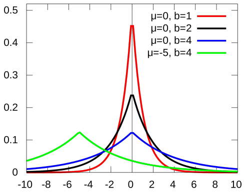
$$\tilde{\boldsymbol{w}} = \left(\mathbf{X}^T\mathbf{X} + \lambda I\right)^{-1}\mathbf{X}^T\mathbf{y}$$

- **Shrinkage** toward zero

$$\text{As } \lambda \to \infty, \ \tilde{\boldsymbol{w}} \to 0$$

- **Lasso** is equivalent to posterior mode under Laplace prior

$$\beta_i|\sigma^2 \overset{iid}{\sim} \text{Laplace}\left(0, \frac{\sigma^2}{\lambda}\right)$$



Evaluating a classifier - confusion matrix

- Confusion matrix:

		Truth	
		Positive	Negative
Decision	Positive	tp	fp
	Negative	fn	tn

- tp = true positive, fp = false positive
  fn = false negative, tn = true negative.
- Example:

		Truth	
		Fraud	No Fraud
Decision	Fraud	tp	fp
	No Fraud	fn	tn

Evaluating a classifier - Accuracy

- **Accuracy** is the proportion of correctly classified items

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fn + fp}$$

		Truth	
		Positive	Negative
Decision	Positive	**tp**	fp
	Negative	fn	**tn**

Evaluating a classifier - Precision

- **Precision** is the proportion of truly positive items among those signaled as positive:

$$\text{Precision} = \frac{tp}{tp + fp}$$

		Truth: Positive	Truth: Negative
Decision	Positive	**tp**	fp
	Negative	fn	tn

- High precision:
  - trustworthy positives
  - people pointed out as fraudulent are almost always frauds.

Evaluating a classifier - Recall

- **Recall** is the proportion of signaled positive items among those that are truly positive:

$$\text{Recall} = \frac{tp}{tp + fn}$$

		Truth: Positive	Truth: Negative
Decision	Positive	tp	fp
	Negative	fn	tn

- High recall:
  - will find the positive items.
  - fraudulents will be caught.
- Recall is also called the **True Positive Rate** (**TPR**)
- There is a trade-off between Precision and Recall.

Evaluating a classifier - False Positive Rate

- False Positive Rate (FPR) is the proportion of signaled positive items among those that are truly negative:

$$\mathrm{FPR} = \frac{fp}{fp + tn}$$

		Truth	
		Positive	Negative
Decision	Positive	tp	fp
	Negative	fn	tn

- Low FPR:
  - will very rarely signal a positive for a negative item.
  - people will not be falsely accused of fraud.

Evaluating a classifier - ROC curve

- Precision and recall depends on the decision threshold.
- $\Pr(\text{Spam}|\text{text in an email}) = 0.9$. Do we send it to the spam-box?
- Is $\Pr(\text{Fraud}|\text{features}) > 0.5$ a good decision threshold?
- Optimal decisions depend on the consequences. Decision theory.
- ROC-curve: Receiver Operating Characteristic.
- ROC: Plots the true positive rate (TPR) against the false positive rate (FPR) **at various thresholds**.
- AUC = Area Under Curve. Area under the ROC curve.

Evaluating a classifier - ROC