Machine Learning: Making Predictions

Dr. Holger Fröhlich

SS 2016

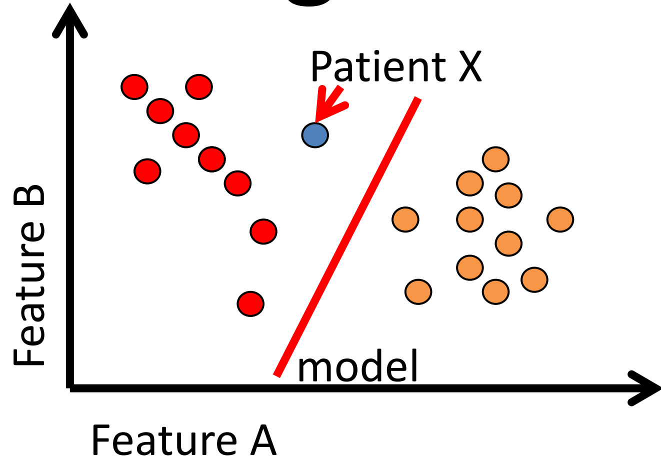
ML and Statistics

- Statistics:
  - Model and parameter estimation
  - Hypothesis tests
  - Classically: #variables << #samples
- Machine Learning:
  - Model prediction: Am I able to predict something new with my model? How well does this work?
  - Often #variables >> #samples (high dimensional data)
- Bioinformatics: Frequent application of statistics <u>and</u> machine learning

Types of Machine Learning

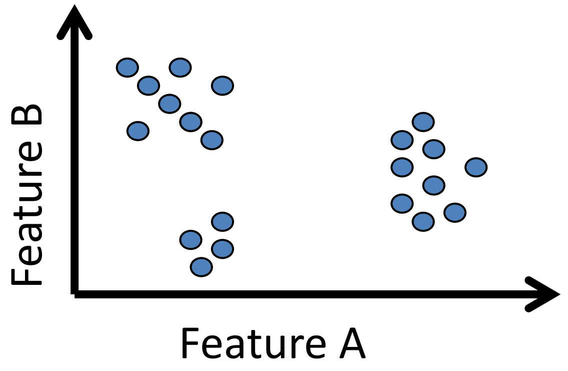
Supervised Learning (Predictive Modeling)

- Learning from fully labeled samples
- **Goal**: making predictions
- Primary goal is **not** understanding of models
- Main examples:
    - Classification, regression



Unsupervised Learning

- Learning without labels
- **Primary goal**: pattern inference
- Example: clustering



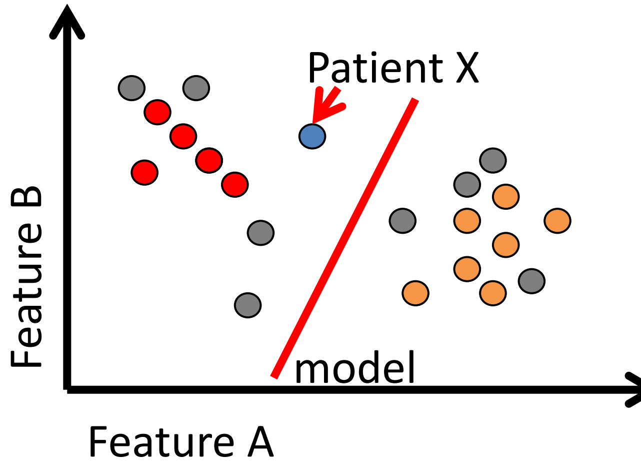
Types of Machine Learning

Re-inforcement Learning

- Learning from criticism (« good », « bad »)
- **Goal**: same as in supervised learning
- Often used in robotics

Semi-supervised Learning

- Learning from labeled as well as unlabeled samples
- **Goals**:
  - Most often same as in supervised learning
  - Sometimes only labeling of unlabeled training samples (**transductive learning**)



Types of Machine Learning

Batch Learning

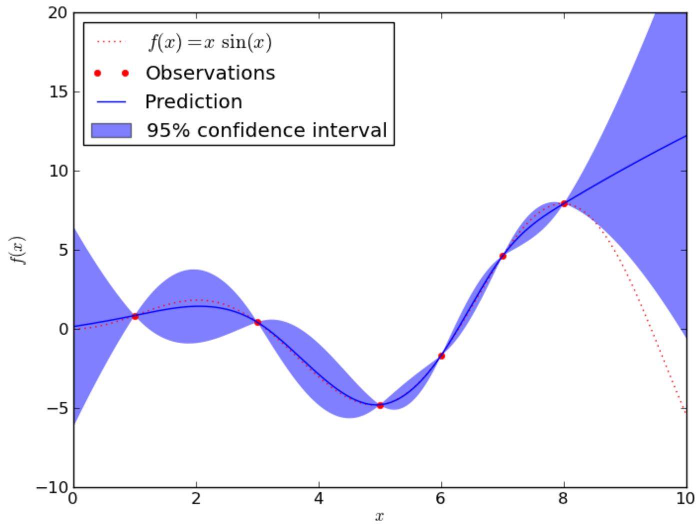
1. Collect sufficiently large training set
2. Build your model

Online Learning

1. Start with small training set
2. Gradually update model while more data is arriving

Active Learning

- Not all training samples are equally worthy
- Actively propose where to sample a new training point



Families of Learning Algorithms

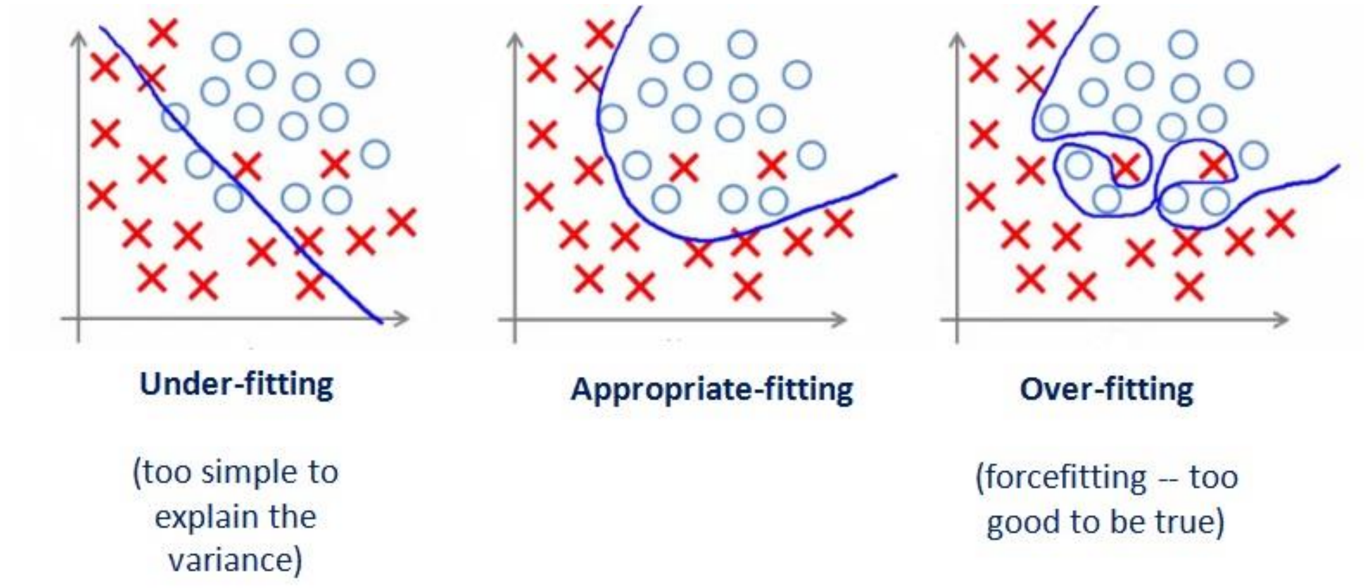
Generative / parametric model

- Mixture models
- Hidden Markov Models
- Bayesian Networks

Non-parametric models

- k nearest neighbors
- Boosting
- Decision trees
- Support Vector Machines
- …

A classification example: How can we decide which model is appropriate?


Under-fitting
(too simple to explain the variance)
Appropriate-fitting
Over-fitting
(forcefitting -- too good to be true)

- We want to have a classifier, which predicts as correct as possible on unseen data (so-called *test data*), i.e. it generalizes well!

- How to construct such a classifier based on available *training data*?

Supervised Learning Formally

- Given: $D = (x_1, y_1), \ldots, (x_n, y_n) \in X \times Y$
- We want to find: $f : X \to Y$
  - **Classification: Y finite**
  - Regression: $Y = \Re^d$

- **Assumption:** Data has been drawn independently and identically distributed (i.i.d.) from an (*unknown*) probability distribution P(x,y)

Empirical Risk (Training Error)

- Find function f minimizing the **empirical risk** (= **training error**)

$$R_{\text{emp}}[f] = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, f(x_i))$$

$\ell : Y \times Y \to \Re$ is called loss function

  - Example: mis-classification loss (see later for other choices)

$$\ell(y, y') = \begin{cases} 1 & y \neq y' \\ 0 & \text{otherwise} \end{cases}$$

  - Regression loss $\ell(y, y') = (y - y')^2$

Why not just minimizing the training error?

- **True risk (generalization error) = expected loss over whole distribution P(x,y)**

$$R[f] = \int_{X \times Y} \ell(y, f(x)) dP(x, y)$$

- Good model should minimize R[f]
- **BUT:** R[f] cannot be computed, since P(x,y) is unknown
- Is empirical risk minimization sufficient instead?

Key Observation (Vladimir Vapnik, 1970s)

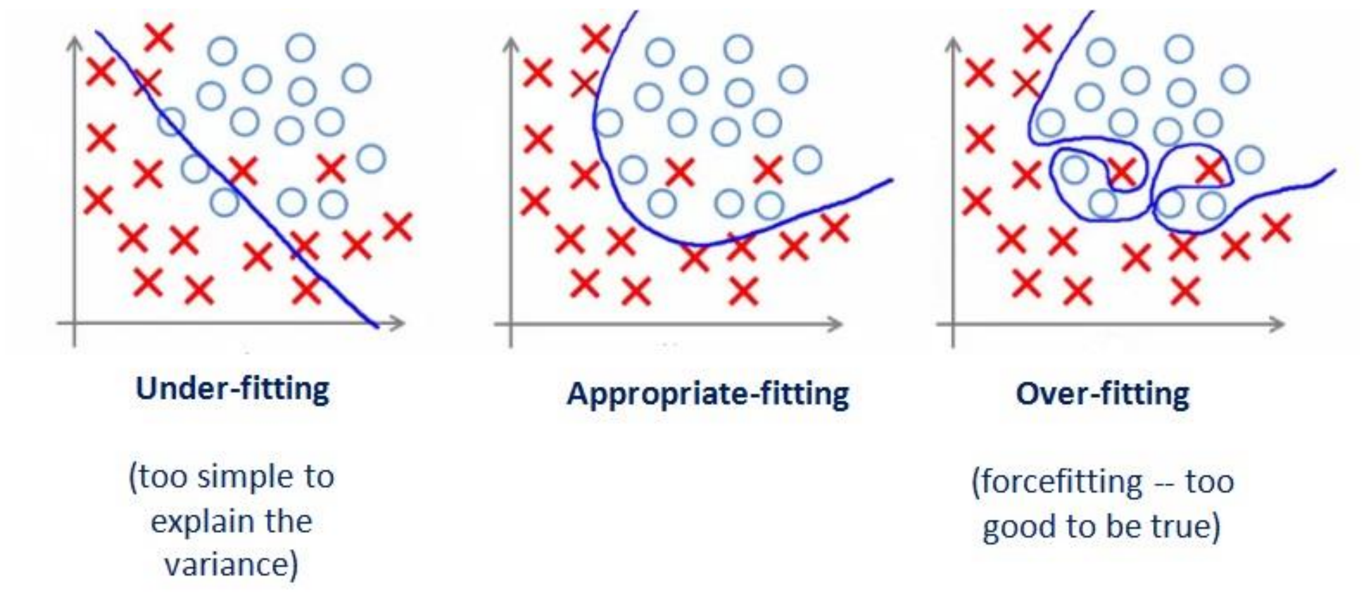
- With probability $1 - \delta$:

$$R[f] \leq \mathrm{R}_{\mathrm{emp}}[f] + \sqrt{\frac{1}{n}\left(h\left(\ln\frac{2n}{h}+1\right)+\ln\frac{4}{\delta}\right)}$$

where h is a measure of the **complexity / capacity** (*VC dimension*) of the function f and n the number of training data

- generalization capability (i.e. expected test error) depends on
  - Training error + *complexity* of the learned model

Classification Functions of Different Complexity

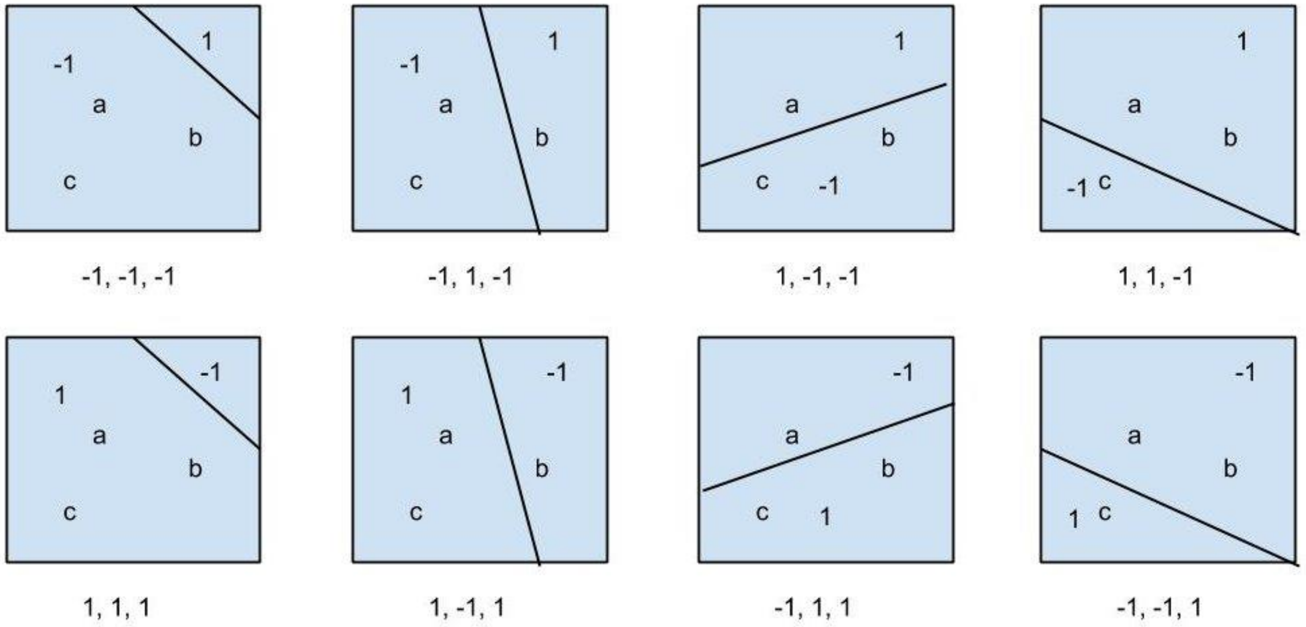

Under-fitting
(too simple to explain the variance)
Appropriate-fitting
Over-fitting
(forcefitting -- too good to be true)

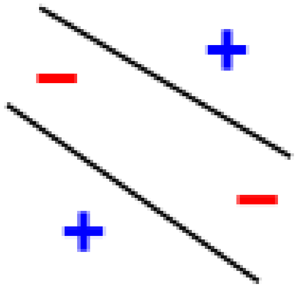
Model complexity

Vapnik-Chervonenkis (VC) Dimension

- VC dimension h = formal definition of function complexity
- A classification model f with parameters ψ is said to *shatter* n data points, if there exists some ψ such that f classifies all points correctly.
- VC dimension = maximum number of points such f can shatter.

Example: VC dimension of a 2D line





- 2D line can shatter at most 3 points ➔ VC dimension = 3

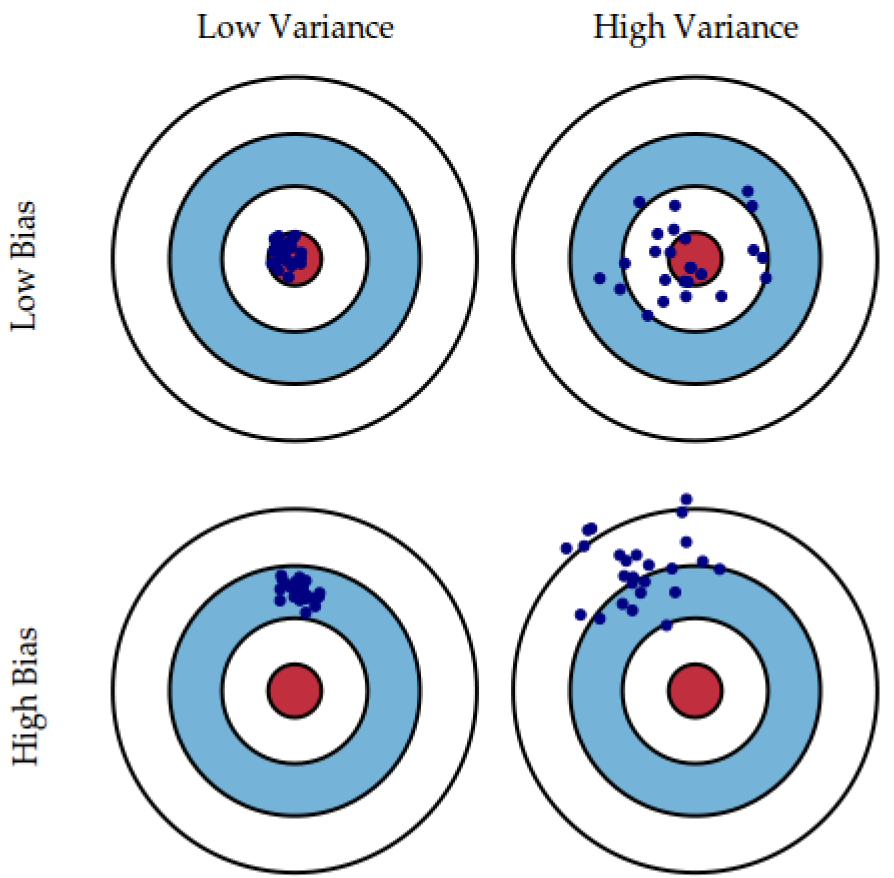
Overfitting

- A low training error (may be even 0!) alone is **not** sufficient!
- It does not guarantee to get a low test error (*overfitting*).
- We must not forget the complexity of the model: Both, the complexity and the training error should be low.
- Hence: Find the simplest possible model explaining the data sufficiently well.

Other Point of View: Bias and Variance of Models

- Instead of talking about complexity one can also talk about the **variance** of the model error
  - How sensitive is the model to small changes in the data (i.e. how much does the model fluctuate on average)?
- Instead of training error one can also talk about model **bias**
  - How accurate is the model on different training sets?

Visualization



Bias and Variance Formally Defined

- Suppose we have a model $\hat{f}$
- Investigate the expected squared error loss on an unseen test data point x:

$$E[(\hat{f}(x) - y)^2] = Bias[\hat{f}(x)]^2 + Var[\hat{f}(x)] + \sigma^2$$

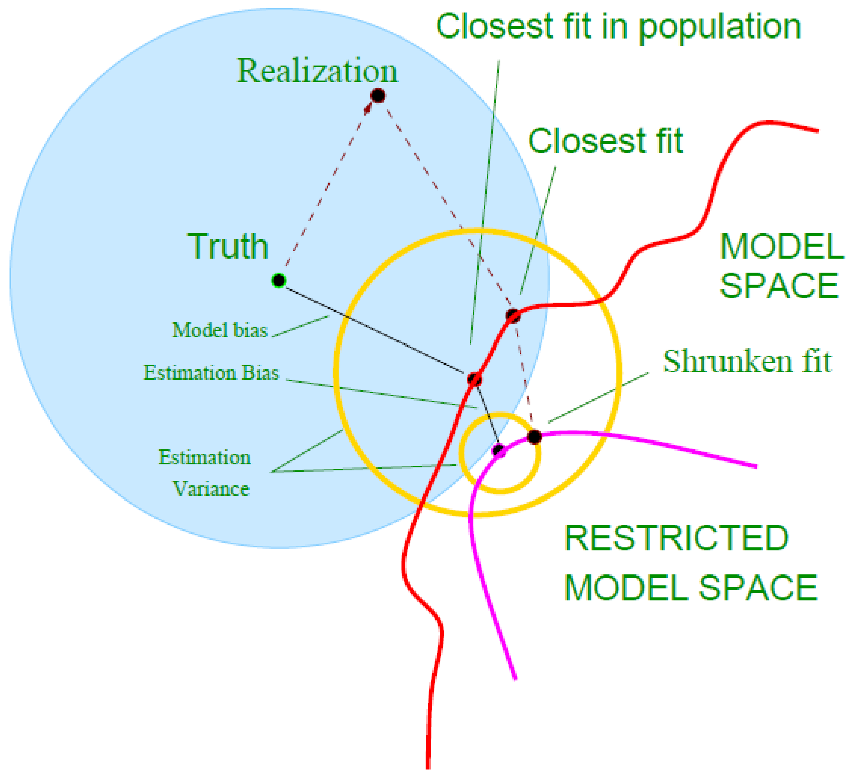
σ^2 = noise variance

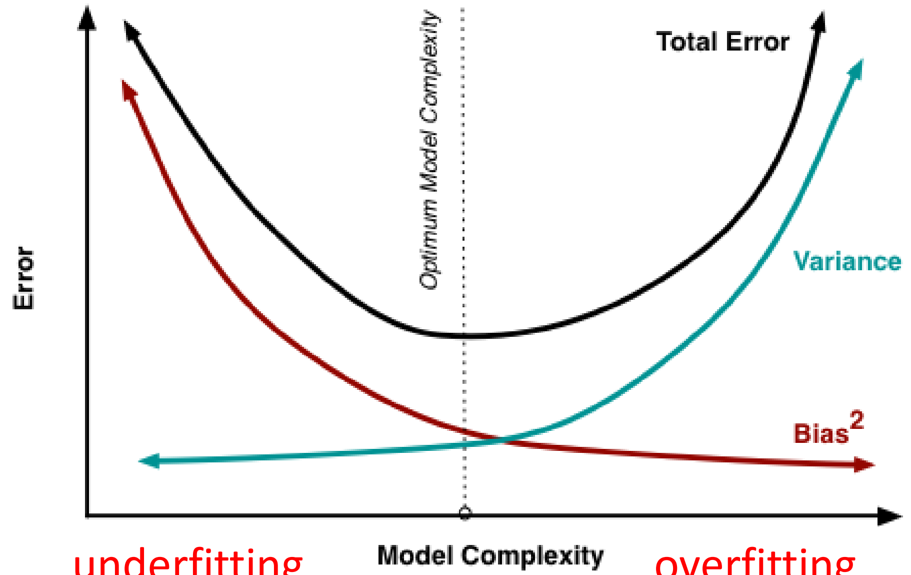
$$Bias[\hat{f}(x)]^2 = E[\hat{f}(x)] - f(x)$$

$$Var[\hat{f}(x)] = E[(\hat{f}(x) - E[\hat{f}(x)])^2]$$

- Expectation is taken over different choices of the training data D from distribution P(x,y)

Schematic Visualization





underfitting overfitting

Bias-Variance Dilemma

- Central insight: bias and variance cannot be minimized at the same time!
  - There is a trade-off between both
- Consequences:
  - Low bias models fit the training data well, but generalize poor
  - Low bias models are usually more complex / have larger variance
  - Lower variance implies larger bias: better generalization comes at the price of a worse fit to the original training data!

Possible Solution: Regularization

- Trade-off between overfitting (f is overly complex) and underfitting (f is too simple / does not explain data)
- **Minimize <u>regularized</u> risk:**

$$R_{\text{reg}}[f] = R_{\text{emp}}[f] + \lambda\Omega[f], \ \lambda > 0$$

Ω[f] = measure of complexity for f

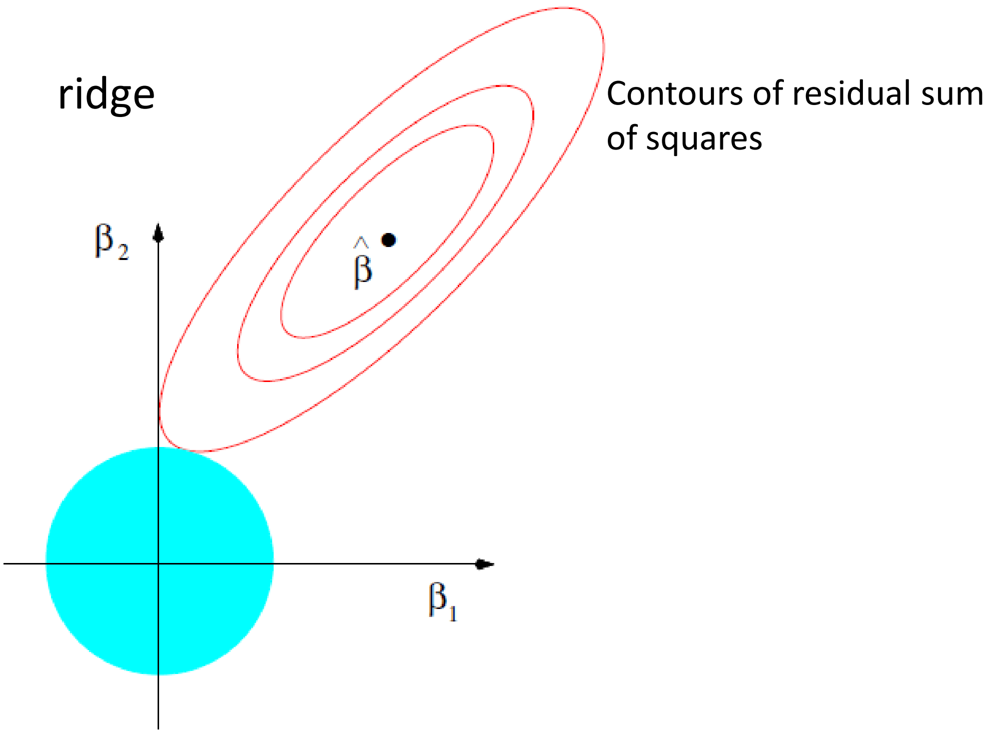
Example: Penalized Regression

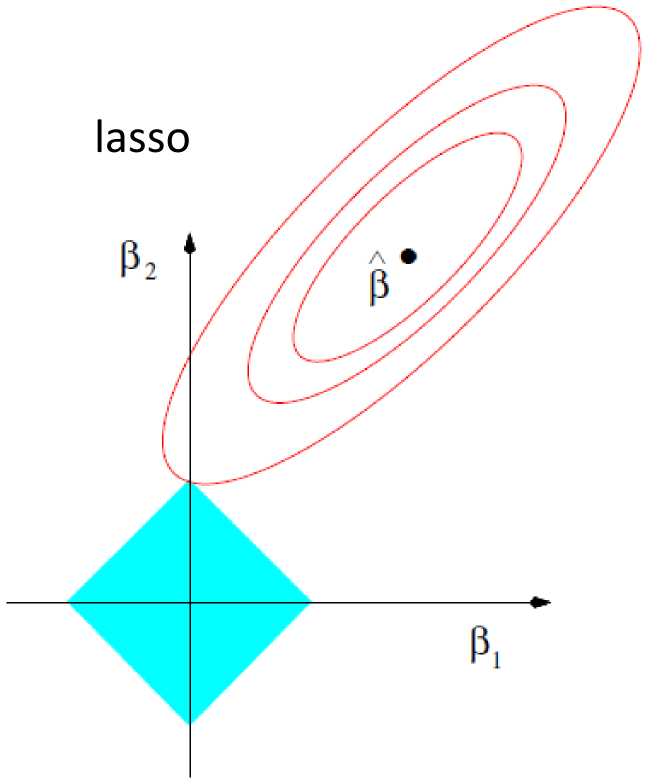
- Enforce models of low complexity/variance by pushing coefficients to a constant (e.g. 0)
  - **Ridge regression**: $\min_{\beta} \sum_{i=1}^{n} (y_i - \beta^T x_i)^2 + \lambda \sum_{i=1}^{n} \beta_i^2$

- *Sparse* regression: force many coefficients to be exactly 0 (i.e. variables do not contribute)
  - **Lasso regression**: $\min_{\beta} \sum_{i=1}^{n} (y_i - \beta^T x_i)^2 + \lambda \sum_{i=1}^{n} |\beta_i|$

    Residual sum of squares penalty

Difference between ridge (L2) and lasso (L1) penalty





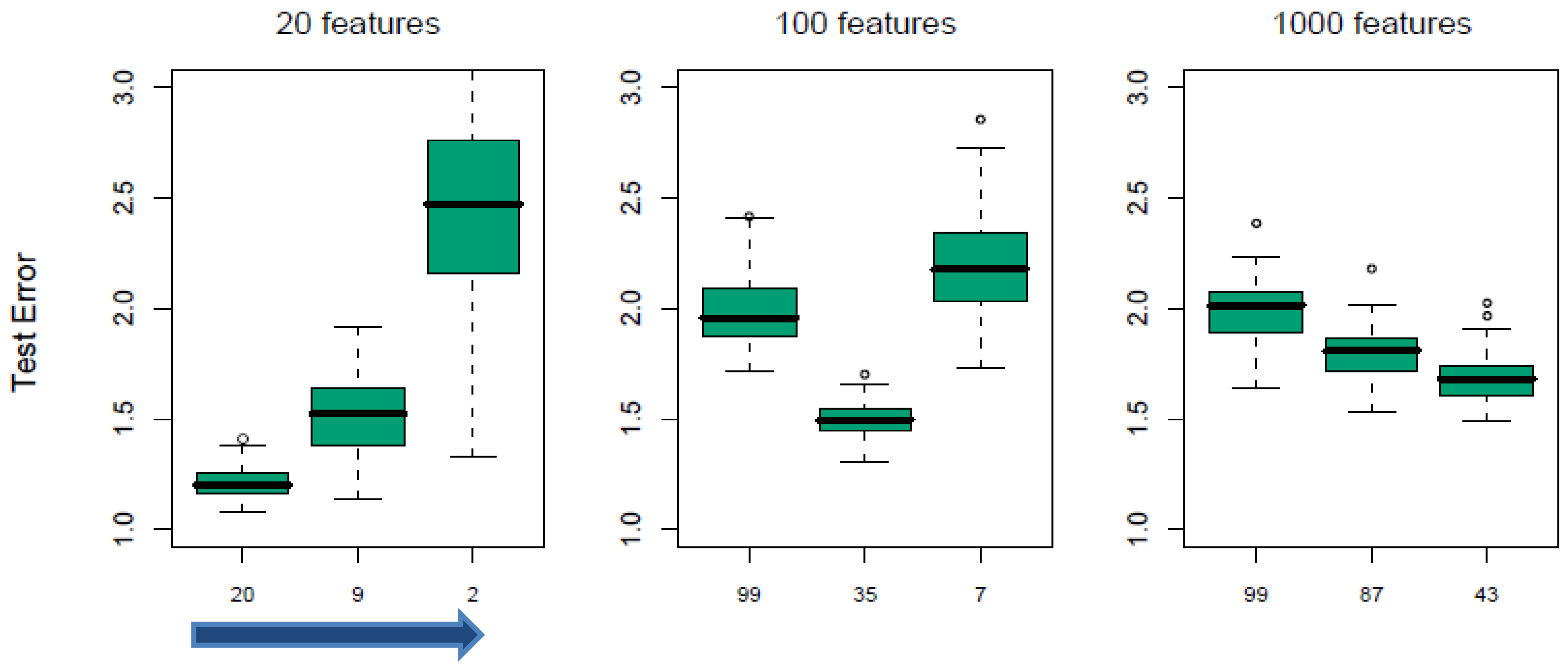
- Lasso penalty enforces sparsity, ridge not
- Ridge tends to shrink all coefficients

Linear Ridge Regression in High Dimensions

Consider #variables (p) >> #samples (n)

- Typical situation in many bioinformatics applications: -omics data

Example: simulation of multivariate normally distributed data (n = 100, pair-wise correlation = 0.2) and linear dependent outcome using 9, 33 and 331 relevant features



Stronger penalty

Hastie et al, The Elements of Statistical Learning, 2008

Why is that happening?

Low dimensional data (p = 20): ridge regression requires little regularization and can exploit covariance structure.

- Be as close as possible to unpenalized situation

Intermediate case (p = 100): moderate shrinkage allows for identifying some relevant features

- Correlated features reveal a similiar magnitude of shrinkage

High dimensional case (p = 1000): little hope to identify the correct covariance structure and features ➔ shrink all features

Consequence: optimal amount of shrinkage is data dependent and needs to be tuned ➔ cross-validation (see later)

Alternative: Bayesian Priors

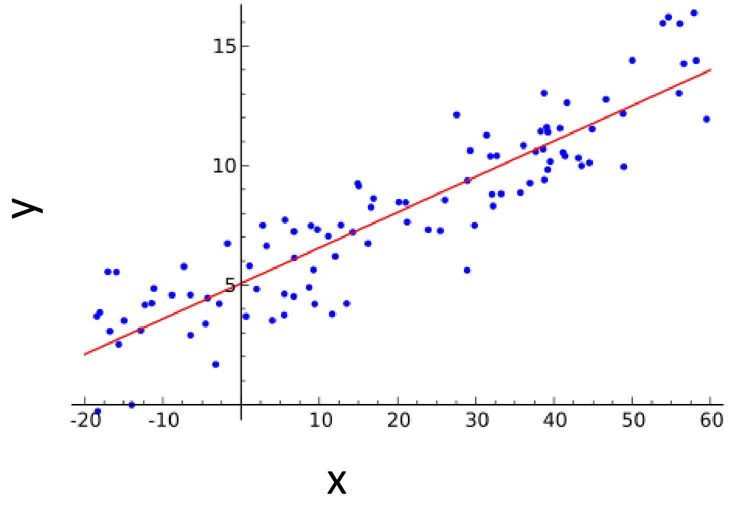
- A complementary formulation is to encode prior assumptions about the model via Bayes‘ law
- Since models are specified by parameters, this (usually) translates into a prior over parameters:

$$p(\theta \mid data) = \frac{p(data \mid \theta)\, p(\theta)}{p(data)}$$

Example: Bayesian Simple Linear Regression

- Denote sample points by $\{x_i, y_i\}$, i=1,...,n.
- We have the following equation for y_i:

$$y_i = \beta_0 + \beta x_i + \varepsilon_i$$
$$\varepsilon_i \sim N(0, \eta^2)$$



- Suppose data to be mean centered: mean(y) = mean(x) = 0
- Recall ML estimate:

$$\hat{\beta} = \frac{\mathrm{cov}(\mathbf{x}, \mathbf{y})}{\mathrm{var}(\mathbf{x})} = \frac{\mathbf{x}^T \mathbf{y}}{\mathbf{x}^T \mathbf{x}}, \beta_0 = 0$$

Posterior Inference

- Assume $\beta \mid \lambda \sim N(0, 1/\lambda^2)$
- Posterior parameter distribution:

$$p(\beta \mid \mathbf{y}, X, \lambda, \eta) = \frac{p(\mathbf{y} \mid X, \beta, \eta)\, p(\beta \mid \lambda)}{p(\mathbf{y} \mid X, \eta, \lambda)}$$

$$= \frac{\prod_i N(y_i \mid \beta x_i, \eta^2) N(\beta \mid 1/\lambda^2)}{\int \prod_i N(y_i \mid \beta x_i, \eta^2) N(\beta \mid 1/\lambda^2) d\beta}$$

$$= N\left(\frac{1}{\lambda + \mathbf{x}^T \mathbf{x}} \mathbf{x}^T \mathbf{y}, \eta^2 \frac{1}{\lambda + \mathbf{x}^T \mathbf{x}} \right)$$

- Expectation looks almost identical to ML estimate, except for additional λ in denominator
- Parameter estimates are intentionally biased

Example: Bayesian vs. Classical Linear Regression

- ML parameters after centering (species richness as a function of lake area):

$$\hat{\beta} = 0.1773$$

- Bayesian estimate:

$$\lambda = 0.1 : E[\beta] \approx 0.1771$$
$$\lambda = 1 : E[\beta] \approx 0.1752$$
$$\lambda = 10 : E[\beta] \approx 0.1579$$

Species Richness	Lake Area
32	2.0
29	0.9
35	3.1
36	3.0
41	3.0

Species Richness	Lake Area
-2.6	-0.4
-5.6	-1.5
0.4	0.7
1.4	0.6
6.4	0.6

Relationship to Ridge Regression

- Posterior log-density:

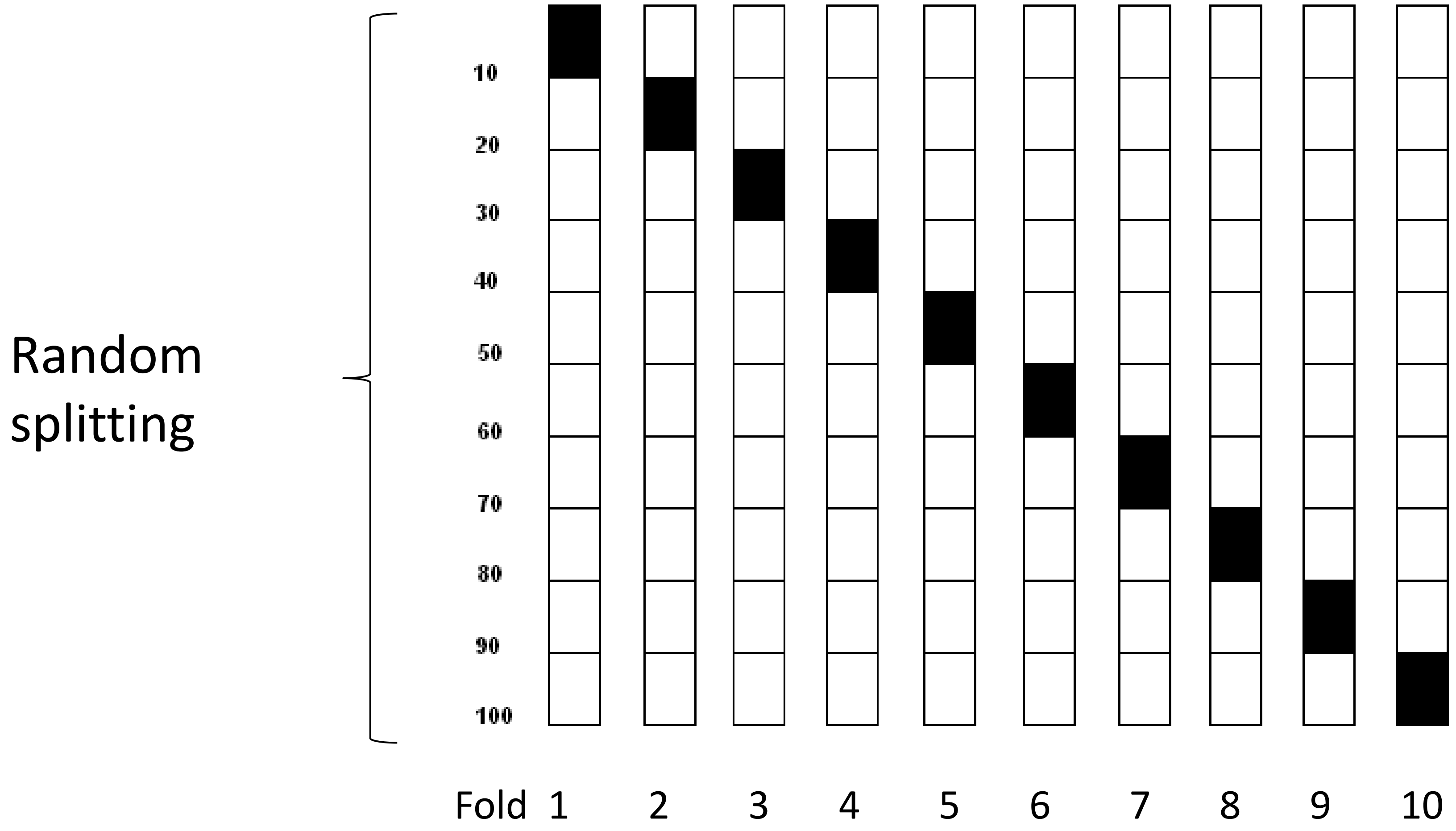
$$-\log p(\beta \mid \mathbf{y}, X, \lambda, \eta) \propto \frac{\sum_{i=1}^{n} (y_i - \beta x_i)^2}{2\eta^2} + \lambda^2 \frac{\beta^2}{2}$$

- Minimization w.r.t. β yields exactly the expectation value of Bayesian linear regression
- Up to constant factors identical to objective of ridge regression
- Consequence: ridge regression yields posterior expectation(mode) of Bayesian linear regression
  - Maximum a posteriori (**MAP**) estimate

Model Evaluation

- Suppose you have trained your favorit model. How can we evaluate, how well it works?
- **Possibility 1**: Take (very large) independent test set with known outcomes and compare predictions with true outcomes
- **Possibility 2**: If you don‘t have enough test data (typical in Bioinformatics), perform **cross-validation**
- How much data is enough?
  - Very difficult question: depends on model complexity and on signal-to-noise ration

K-fold Cross-Validation



K-Fold Cross-Validation Error

- On each CV fold we ask the model to make predictions on the left out test data by **blinding out** the true outcomes (e.g. class memberships)

class	p53 expression	predicted
normal	1.34	tumor
normal	2.78	normal
tumor	10.24	tumor

- K-fold CV error = average over prediction errors in each CV fold.

Cross-Validation Estimator

- Cross-validated prediction error is given by:

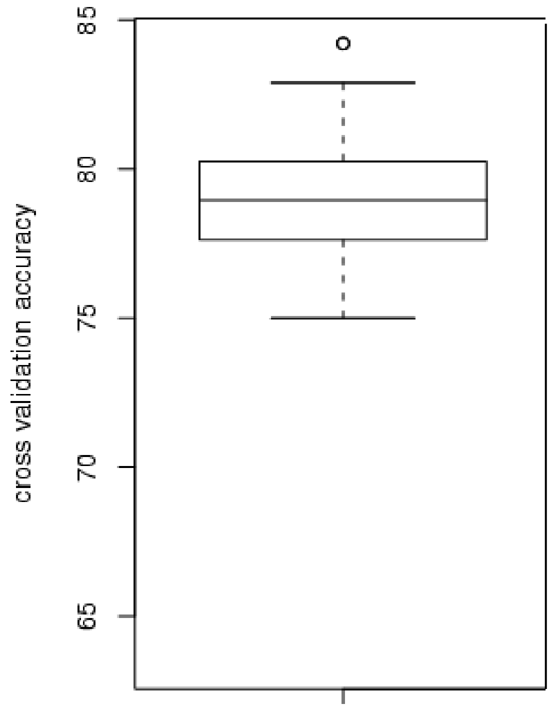
$$CV = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{|D_k|}\sum_{(x,y)\in D_k}\ell(y, f(x))$$

where D_k is the data left out in cross-validation fold k for testing

- Observation: CV depends on the ordering of the data!

Variance of the Cross-Validation

- The CV procedure depends on the initial random splitting of the dataset
- Every split produces a different result
- Repeating the CV procedure gives an estimate of expectation and variance



How to compute the prediction error?

- Depends on the model. E.g.:
  - Classifier: misclassifications rate (e.g. 1/3 in previous example)
  - Regression: squared difference between predicted and correct values
- In machine learning this is called loss function
- It is a function measuring the error we make for a given sample when predicting y‘ instead of y.

$$\ell(y, y') = (y' - y)^2 \quad \text{squared error}$$

$$\ell(y, y') = 1[y' \neq y] = \begin{cases} 1 & y' \neq y \\ 0 & y' = y \end{cases} \qquad \text{misclassification}$$

Measuring Classification Performance

- **Possibility 1:** count number of correctly classified examples
  - Problem:

Class 1	Class -1
500	50

→ correctly classified:

Class 1	Class -1
400/500	0/50

→**accuracy (%):** 400/550 = 72.72%

- **Possibility 2:** *contingency table*

true class	*predicted*	
	Class 1	**Class -1**
Class 1	400/500	100/500
Class -1	50/50	0/50

*true pos. rate (**sensitivity/recall**)* 400/500 = 80%

*true neg. rate (**specificity**)* = 0/50 = 0%

false pos. rate = 50/50 = 100%

false neg. rate = 100/500 = 20%

positive predictive value / **precision** = 400/(400+50) = 89%

Measuring Classification Performance

true class	*predicted*	
	Class 1	**Class -1**
Class 1	400/500	100/500
Class -1	50/50	0/50

$$BAC = 0.5 * (\text{sensitiviy} + \text{specificity}) = (0.8 + 0)/2 = 40\%$$

$$F_1 = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}} = 2 * \frac{0.89 * 0.8}{0.89 + 0.8} \approx 84\%$$

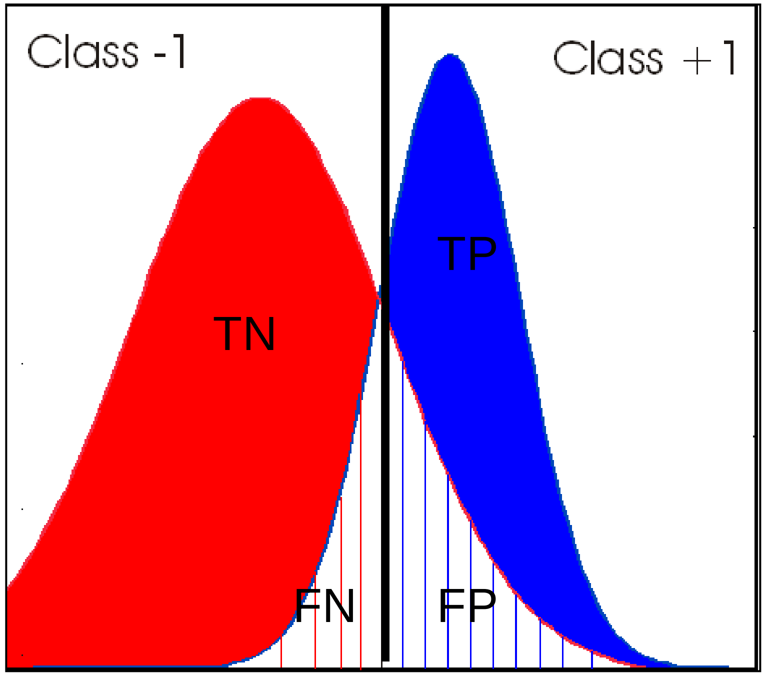
$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

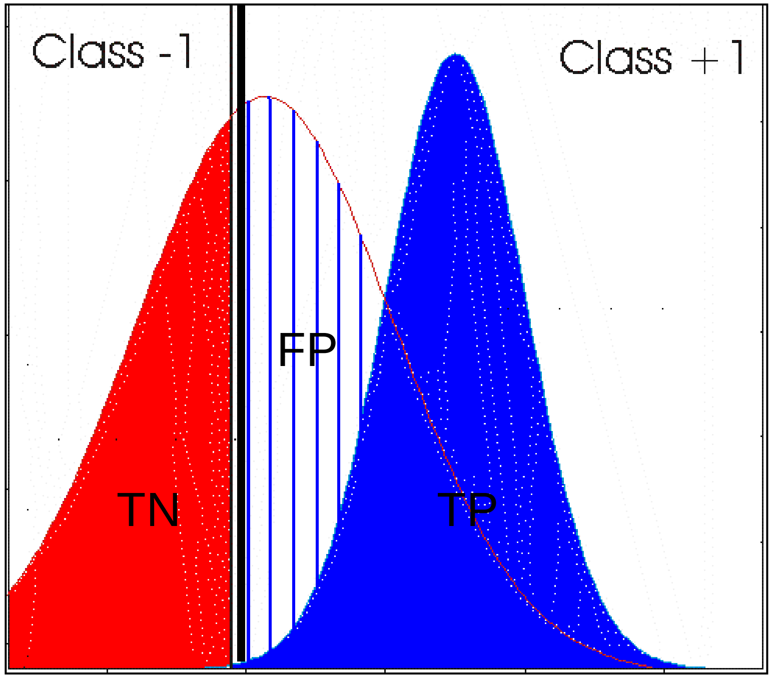
$$= \frac{400 * 0 - 50 * 100}{\sqrt{(400+50)(400+100)(0+50)(0+100)}} = -0.15$$

- The balanced accuracy (**BAC**) is the average of sensitivity and specificity.
- The F1 is the harmonic mean of precision and recall. It is between 0 and 1.
- Matthews correlation coefficient (**MCC**) similarly to a correlation coefficient lies between -1 and 1. A value of 0 indicates the performance of a random classifier.
- .

Measuring Classification Performance

- What happens, if our model produces class probabilities and not a class label?
  - Think about logistic regression!
- Each cutoff t results in a different contingency table!





$$f(x) := \begin{cases} \text{class 1} & \text{if } g(x) \geq t \\ \text{class -1} & \text{otherwise} \end{cases}$$

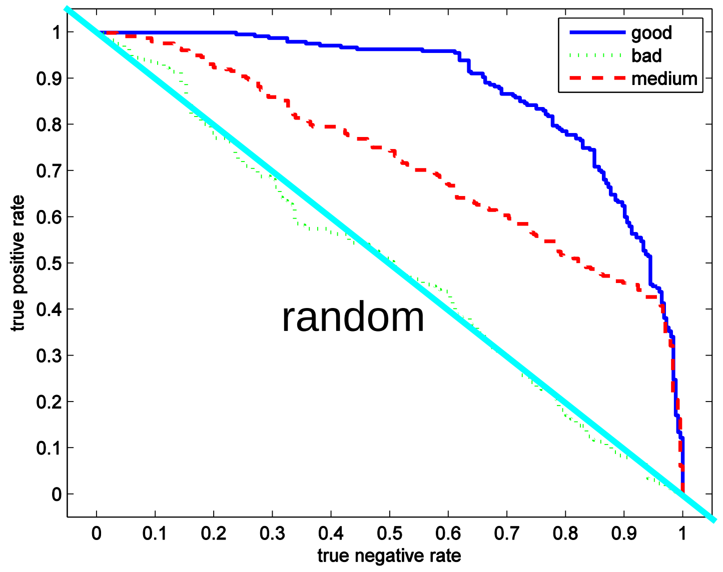
Receiver Operator Characteristic (ROC) Curves

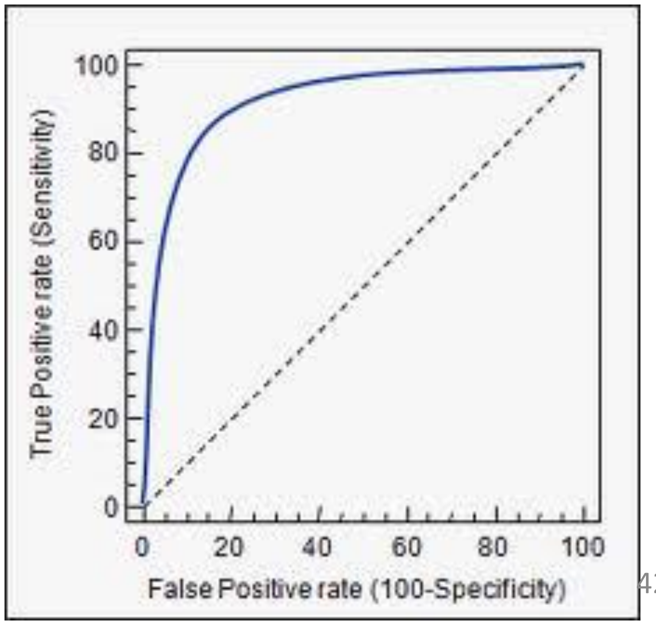
- Each point on the ROC curve corresponds to a specific combination of sensitivity and specificity
- *Area under ROC curve* summarizes performance for all possible classification cutoffs:

$$AUC = \Pr(g(x^+) \geq g(x^-))$$

$$\approx \frac{1}{n*m}\sum_{i=1}^{n}\sum_{j=1}^{m}\mathbf{1}(g(x^+{}_i) \geq g(x^-{}_j))$$

- AUC estimate is identical to Mann-Whitney U-statistic
- It is a ranking essentially a ranking measure





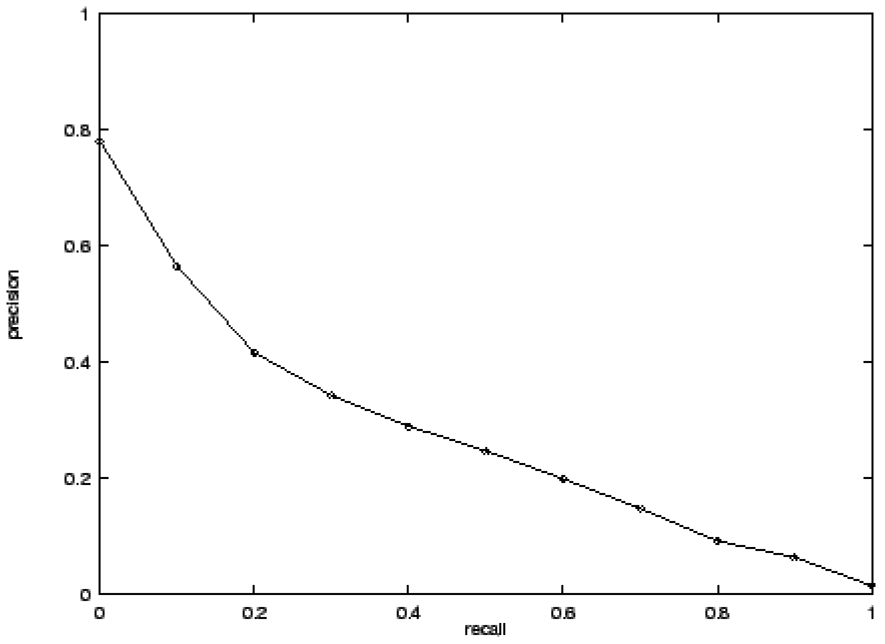
Precision-Recall Curves

- Each point on the PR curve corresponds to a combination of recall and precision.
- Area under PR curve (AUPR) summarizes performance for all possible classifier cutoffs.

$$AUPR = \int_0^1 precision(t)dt$$

$$\approx \sum_{t=1}^{n} precision(t)\Delta recall(t)$$

$$\Delta recall(t) = \text{change in recall from t-1 to t}$$



What you should know and being able to <u>apply</u>

- Which general types of machine learning directions exist (→ unsupervised, supervised, reinforcement)? What are examples?
- Which families of models are there?
- What is the difference between training and testing?
- Generalization and bias-variance dilemma
- Penalized regression
  - Lasso, ridge
- Model validation:
  - cross-validation
  - Measuring classification performance: AUC, AUPR, F1, MCC, BAC