EURECOM
Sophia Antipolis

MALCOM

Machine Learning for Communication Systems

Lecture 3

Recap – Review

Key concepts from Supervised Learning

Slides: Marios Kountouris
Lecturer: Ayşe Ünsal

Spring 2024

Supervised Learning

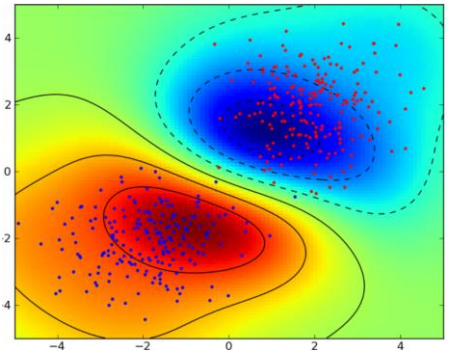
Supervised learning

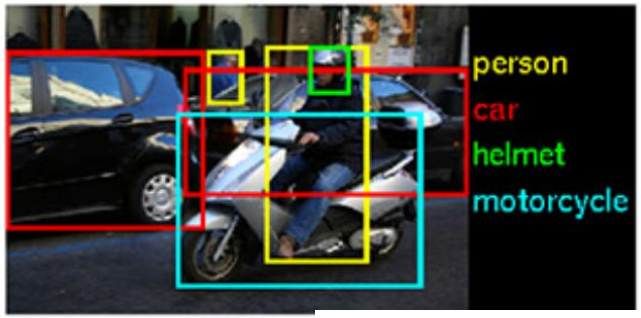
- Training data includes desired outputs
- dataset comprised of labeled examples (a pair (input, label))

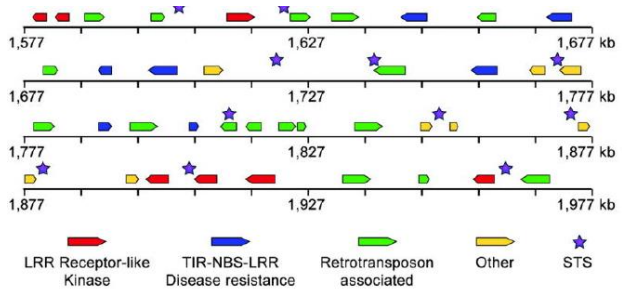
$$y = f(x)$$

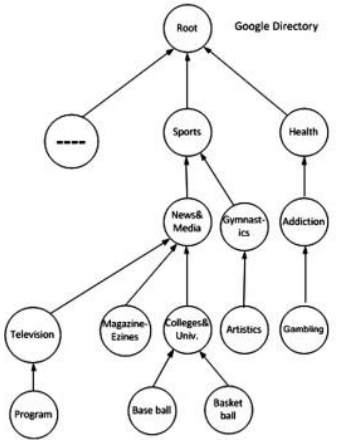
often with loss $\ell(y, f(x))$

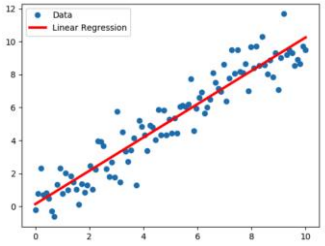
- **Binary classification**
  - Given x find y in $\{-1, 1\}$
- **Multi-class classification**
  - Given x find y in $\{1, \dots\ k\}$
- **Regression**
  - Given x find y in $\mathbb{R}$ (or $\mathbb{R}^d$)
- **Sequence annotation**
  - Given sequence $x_1, \dots, x_n$ find $y_1, \dots, y_n$
- **Hierarchical Categorization (Ontology)**
  - Given x find a point in the hierarchy of y (e.g. a tree)
- **Prediction**
  - Given x_t and $y_{t-1}, \dots, y_1$ find y_t











EURECOM Sophia Antipolis

Supervised Learning

Regression:

- The goal is to generalize the relationship between input and output variables outside of the training set.
- Given a test input (not yet observed), we want to obtained a predicted output via extrapolating from the training set.

Classification:

- The goal is to generalize the relationship between input and label outside of the training set.

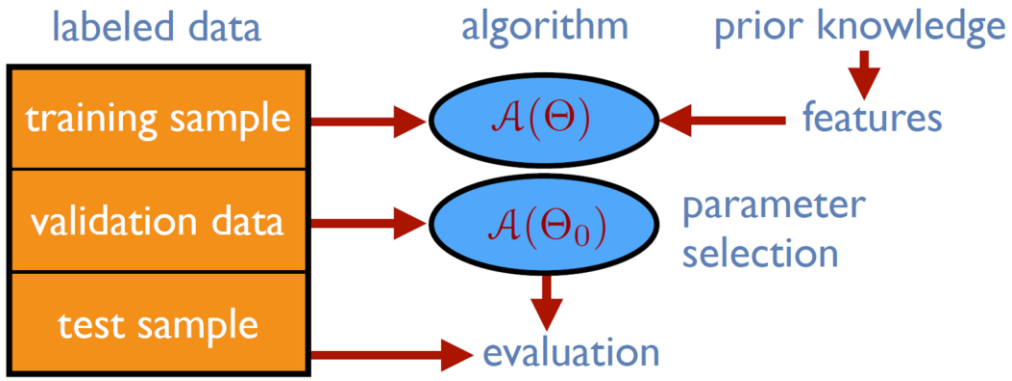
The main goal of learning is generalization!

Target domain	Statistical Inference	Supervised Learning
Discrete and Finite	detection(hypothesis testing)	Classification
Continuous	estimation	Regression

Learning is needed when a model for the data is not available.

Prelude - General Setting

- Data:
  - training data (typically labeled)
  - test data (labeled but labels not seen).
  - validation data (labeled, for tuning parameters)
- *Example*: item, instance of the data used
- *Features*: attributes associated to an item, often represented as a vector
- *Labels*: category (classification) or real value (regression) associated to an item



How do we learn?

- To learns/generalize we need to make prior assumptions about the relationship between input and output

Inductive Bias:

The set of all assumptions made on the relationship between input and output to enable learning

Supervised Learning – Basic Setting

- **Training data:** labelled examples $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$: $(\mathrm{x}_n, \mathrm{y}_n) \sim_{\text{i.i.d.}} p(x, y), n = 1, \dots, N$

where

Space: input space X, output space Y

- each input x_i is a machine-readable description of an instance (e.g., image, sentence, email)
- each corresponding label y_i is an annotation relevant to the task - typically not easy to automatically obtain (e.g. spam/no spam)

- **Test pair**: $(\mathrm{x}_n, \mathrm{y}_n) \sim_{\text{ind.of } \mathcal{D}} p(x, y)$

- **Goal**: find (*learn*) a predictor (a function) $\hat{f} = h(x)$ from labeled examples, which accurately "predicts" the labels of new (previously unseen) inputs.
- Predictor: $y \approx \hat{y}(x) = h(x) = \hat{f}(x_{new})$

- Loss function (quality of prediction): $\ell: Y \times Y \to \mathbb{R}$

  e.g., $\ell(y, \hat{y}(x)) = (y - \hat{y})^2$ quadratic

  $\ell(y, \hat{y}) = |y - \hat{y}|$ absolute

  $\ell(y, \hat{y}) = 1(y \neq \hat{y})$ probability of error (0-1 error)

  Huber loss, ϵ-insensitive loss, ..., etc.

Hypothesis Set: $H \subseteq Y^X$ subset of functions out of which the learner selects her hypothesis.

- depends on *features*
- represents *prior* knowledge about task

- **Goal:** minimize average loss on the test pair (generalization loss)

$$R_p(\hat{y}) = \mathbb{E}_{(\mathrm{x},\mathrm{y}) \sim p_{\mathrm{xy}}}[\ell(y, \hat{y}(x))]$$

Problem: find hypothesis $h \in H$ with small generalization error

Errors

- **Data:** $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$: $(\mathrm{x}_n, \mathrm{y}_n) \sim_{\text{i.i.d.}} p(x, y), n = 1, \dots, N$

- Generalization error: for $h \in H$, it is defined by

$$R(h) = \mathbb{E}_{(\mathrm{x},\mathrm{y})\sim p_{\mathrm{xy}}}[\ell(y, h(x))]$$

- Empirical error: for $h \in H$ and sample $\mathcal{D}$, it is

$$\hat{R}(h) = \frac{1}{N}\sum_{i=1}^{N} \ell(y_i, h(x_i))$$

- Bayes error:

$$R^* = R(h^*) = \inf_{\substack{h \in H \\ h \text{ measurable}}} R(h)$$

Infimum: greatest lower bound of a set S, defined as a quantity m such that no member of the set is less than m, but if ϵ is any positive quantity, however small, there is always one member that is less than $m + \epsilon$

in deterministic case $R^* = 0$

set of all predictors

How easy is this?

What are the difficulties?

- Data may be hard to obtain or is not the right one
- How to clean/improve/augment data?
- How to choose structure/model for $\hat{f}$? How do we pick the right model class?
  - How do we pick magic *hyper-parameters*?
  - How do we perform *feature selection*?
- How to algorithmically fit $\hat{f}$ to data?
- How to ensure $\hat{f}$ does not overfit,

  i.e., it fits $\{(x_i, y_i)\}_{i=1}^{n}$ well, but is useless on future data?

- Impossible task without assuming a model (inductive bias) [*No free lunch theorem*]
  - without specific structural assumptions, no optimization scheme can perform better than blind/random search on the average
  - No model works best for all situations

Generalization

Observations

- the best hypothesis on the sample may not be the best overall
- generalization is not memorization
- complex rules can be poor predictors
- trade-off: complexity of hypothesis set vs sample size (underfitting/overfitting)

Learning ≠ Fitting

- When training an ML model, we do not only want it to learn to model the training data
- We want it to generalize to data that has never seen before=> Test set!
- ➢ Algorithm works well on the training set but fails to generalize=> Overfitting!

Empirical Risk Minimization (ERM)

- Select hypothesis set H
- Find hypothesis $h \in H$ minimizing empirical error:

$$\hat{h} = \underset{h \in H}{\operatorname{argmin}} \hat{R}(h)$$

set of all predictors

$$= \underset{h \in H}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, h(x_i))$$

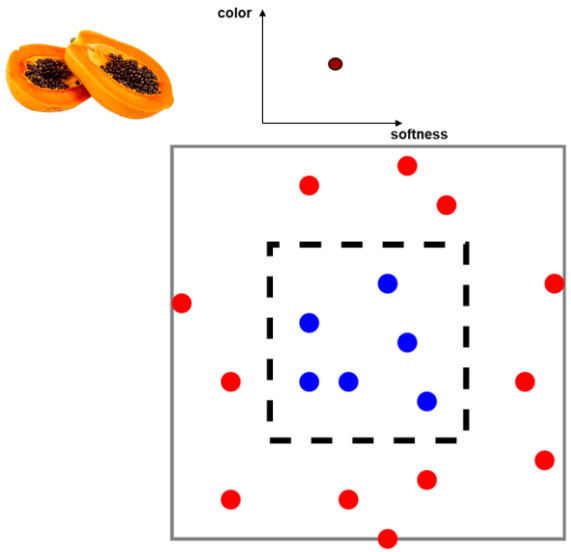
  - but H may be too complex
  - sample size may not be large enough

- This method approximates the generalization error $R(h)$ (expected risk) by the training error (empirical risk) $\hat{R}(h)$ (defined by the average error over the sample)

- This method raises two principal issues:
  - **Statistical problem**: How well does the minimizer of the empirical risk perform?
  - **Optimization problem**: How to minimize the empirical risk $\hat{R}(h)$?

Empirical Risk Minimization (ERM)

- The empirical risk is $\hat{R}(h) = \frac{1}{N}\sum_{i=1}^{N} \ell(y_i, h(x_i))$



- Instances are distributed uniformly within the gray square
- Labeling function: label is 1 if instance is within the inner blue square
  0 otherwise
- Area of gray square is 2 // area of blue square is 1

- Consider the predictor $h(x) = y_i 1_{x_i}(x)$ (so basically y_i if $\exists i$ s.t. $x_i = x$)

- No matter what the sample is, the training error is $\hat{R}(h) = 0$
- This predictor is one of the empirical-minimum-cost hypotheses (no classifier can have smaller error).
- True error of any classifier that predicts the label 1 only on a finite number of instances is $R(h) = 1/2$.
- So, we have a predictor whose performance on the training set is excellent, yet its performance on the true "world" is very poor.

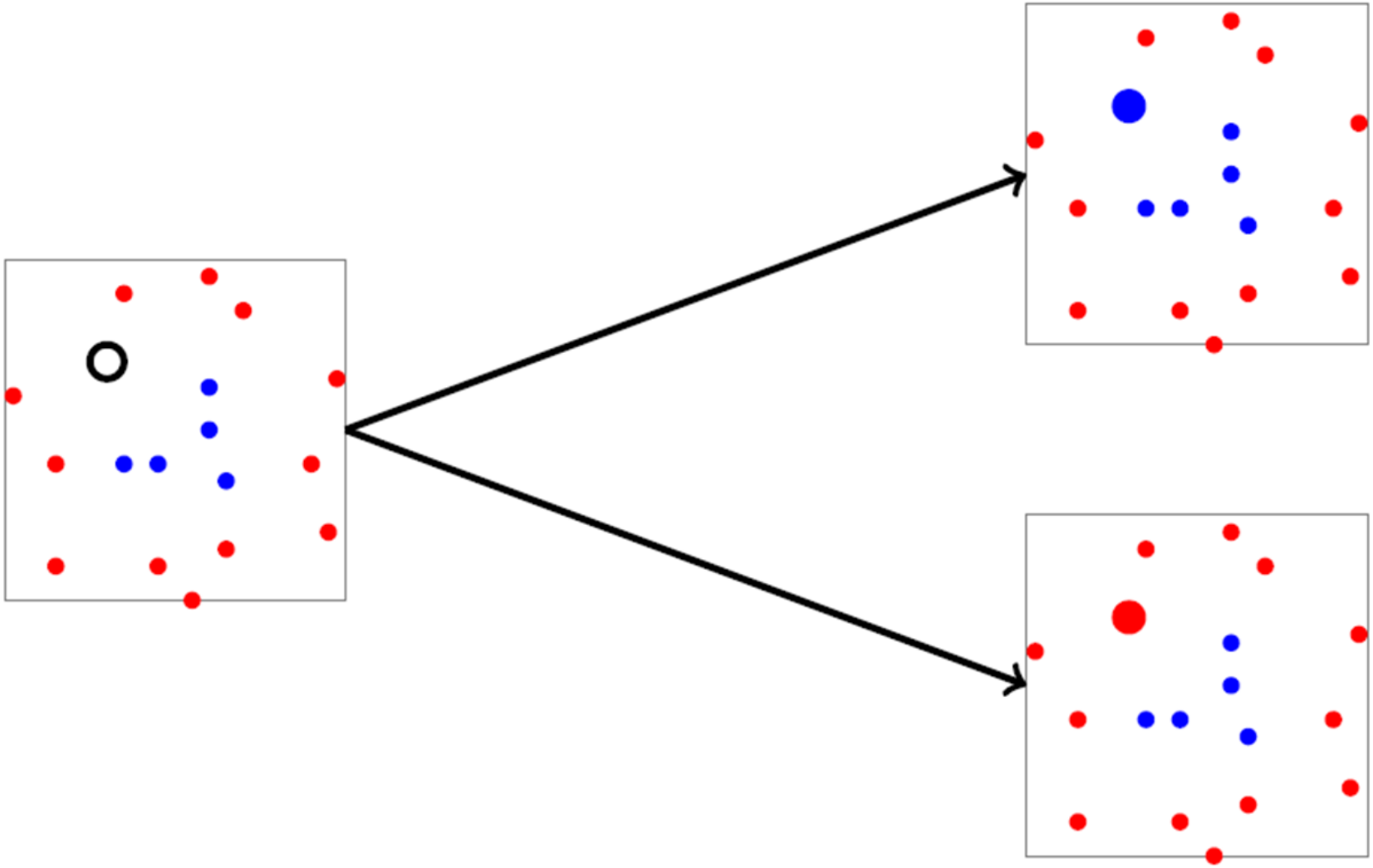
- Overfitting: the prediction fits the data too closely and does not try to generalize enough.

Empirical Risk Minimization (ERM)

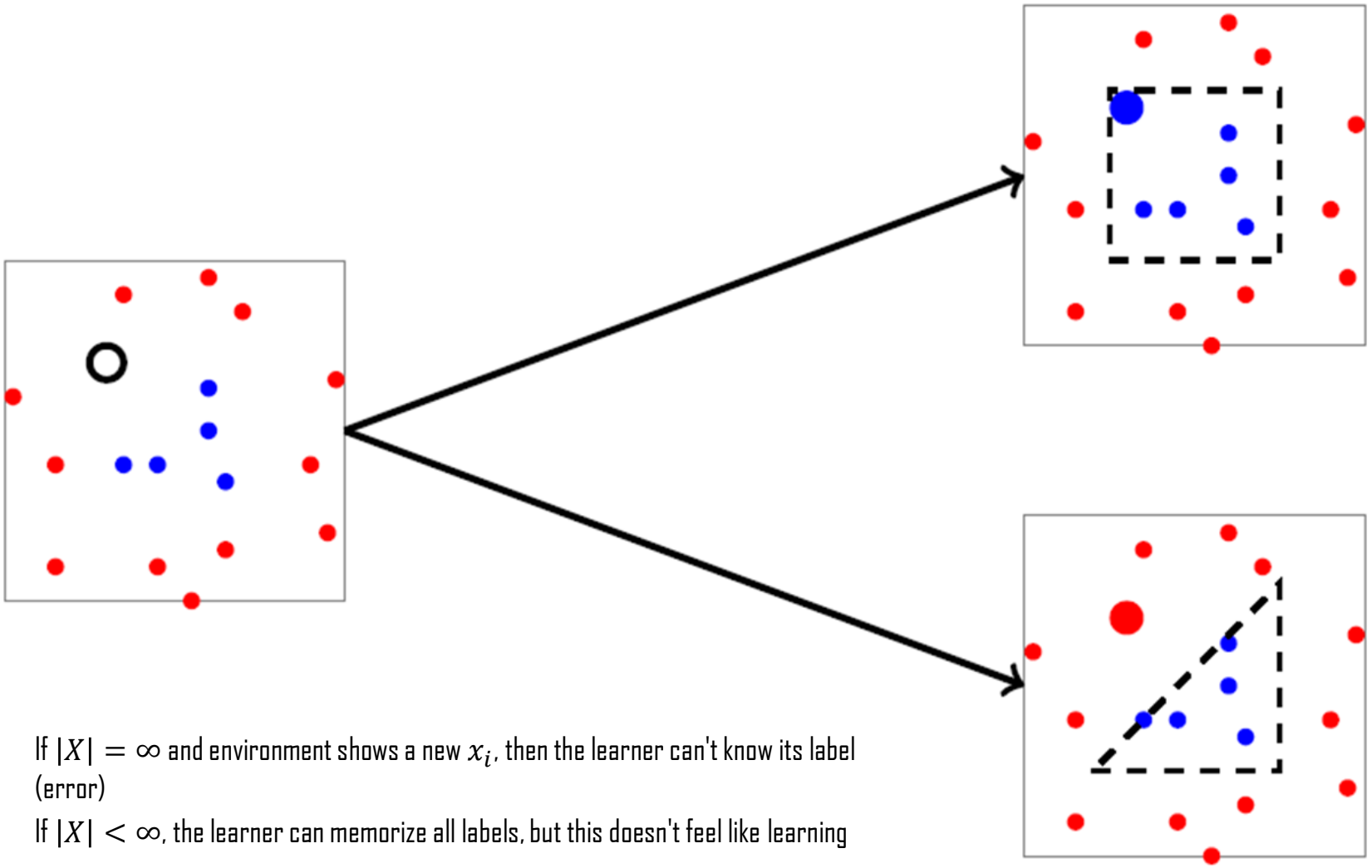
How to prevent overfitting?

- A common solution is to apply the ERM learning rule over a restricted search space.
- Formally, the learner should choose in advance (before seeing the data) a set of predictors. This set is called a hypothesis class H.
- Inductive bias: by restricting the learner to choosing a predictor from H, we bias it toward a particular set of predictors.
- Since the choice of such a restriction is determined before the learner sees the training data, it should ideally be based on some prior knowledge about the problem to be learnt.
- Fundamental tradeoff: choosing a more restricted hypothesis class better protects us against overfitting but at the same time might cause us a larger inductive bias.

Constraint and Prior Knowledge



Constraint and Prior Knowledge



- If $|X| = \infty$ and environment shows a new x_i, then the learner can't know its label (error)
- If $|X| < \infty$, the learner can memorize all labels, but this doesn't feel like learning

...

Empirical Risk Minimization cont'd

- ***The empirical risk*** is the average loss over data points.

- If the empirical risk is small the predictor fits the data well according to the loss function.

  The empirical risk $\Longleftrightarrow$ Performance metric

  related but not the same purpose!

- Example-Least squares in linear regression: ERM is a general method for choosing the model parameter: fitting a parametrized predictor model
- Choose the parameter that minimizes the empirical risk
- The predictor you found depends on the loss you chose (quadratic loss, absolute loss, etc)
- Use validation (performance metric) to choose from among candidate losses (from other hypotheses in your class)

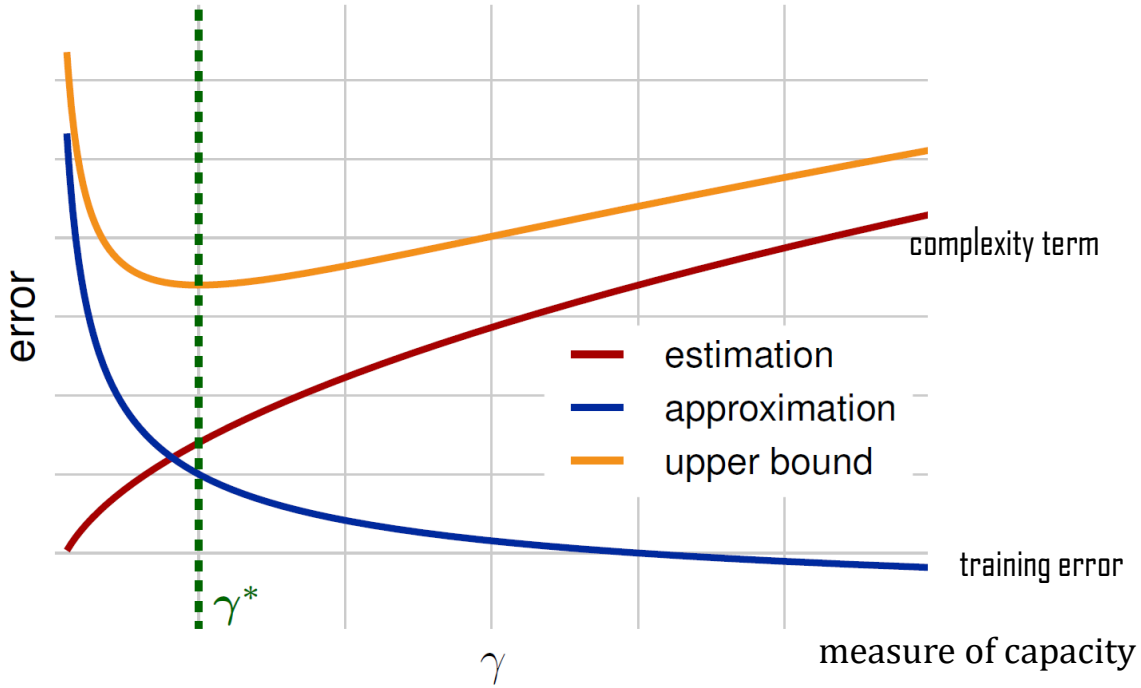
Model Selection

- Excess Risk: can be decomposed for any $h \in H$,

best in hypothesis class $\min_{h \in H} R(h)$

$$R(\hat{h}) - R^* = \underbrace{\left[R(\hat{h}) - R(h^*)\right]}_{\text{estimation}} + \underbrace{\left[R(h^*) - R^*\right]}_{\text{approximation}}$$

- Approximation: deterministic, only depends on H
- Estimation: only term we can hope to bound.



Estimation error: appears bcz. training error is an estimate of generalization error (bcz finite sample is used).
Depends on the size of the training set and on the complexity of the hypothesis class.

Approximation error: measures how much is lost from restricting the set of predictors.
Deterministic and does not depend on the sample size.

Bias-variance tradeoff: choosing a large class results in a small approximation error and a large estimation error since it leads to overfitting.

Relation between training error (bias), complexity (variance) and test error (bias+variance)

Model Selection

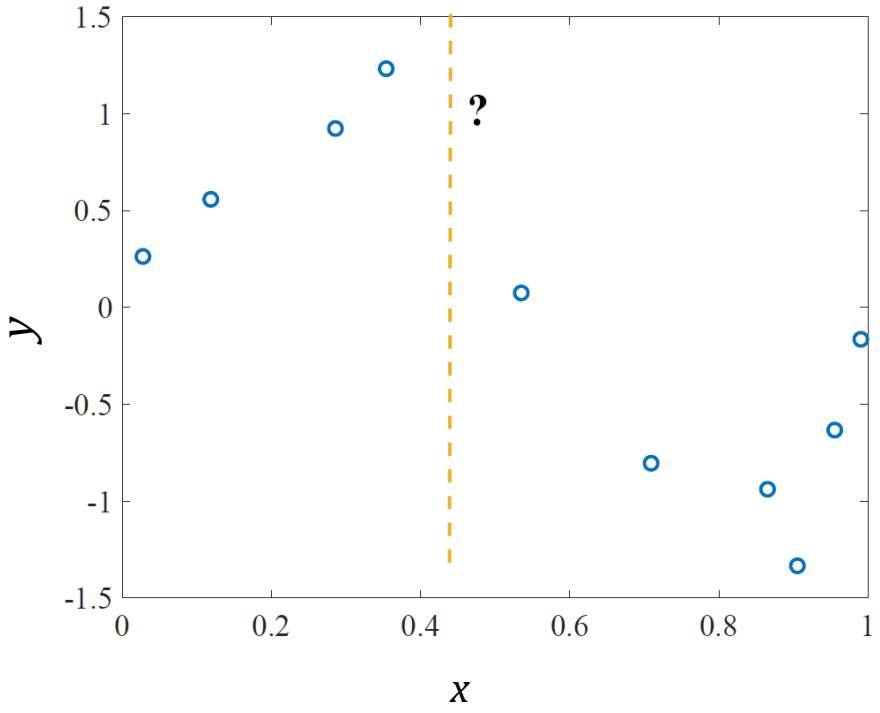
How to select the right model class (inductive bias)?

- Use domain knowledge if available, e.g., to select features
- Focus here on model order selection, i.e., selection of the capacity of the model

- **Example**: Regression using a discriminative model $p(y|x,w)$

$$\underbrace{\sum_{m=1}^{M} w_m x^m}_{\hat{y}(x)} + \mathcal{N}(0,1)$$

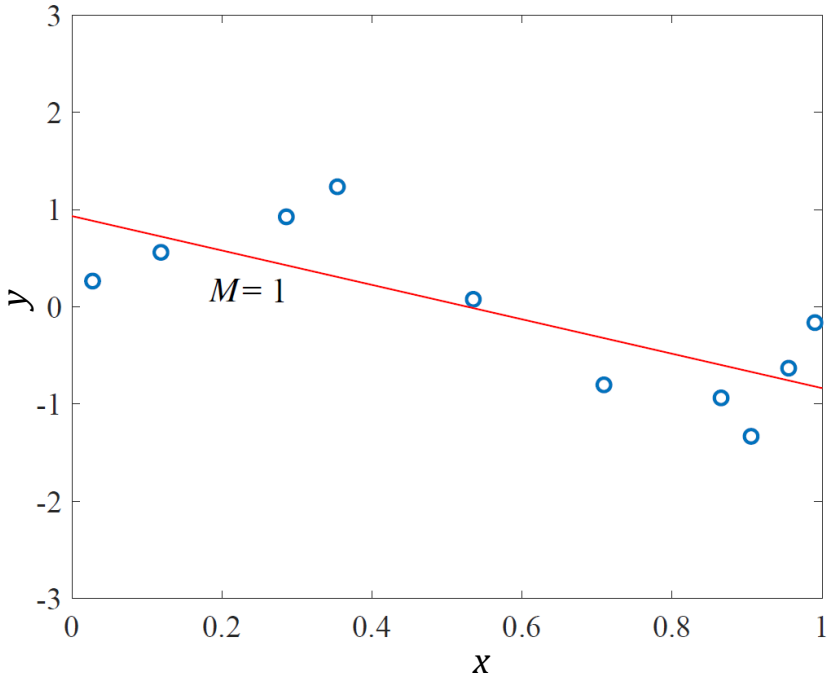
$\hat{y}(x)$: polynomial of order M



Model Selection: Underfitting

- With $M = 1$, the ML predictor $\hat{y}(x)$ *underfits* the data: poor performance on training data, unreliable predictions!
- The model is not rich enough to capture the important variations (signal) present in the data
- Large training loss ⇨ high bias & low variance

$$L_{\mathcal{D}}(\theta) = \frac{1}{N}\sum_{n=1}^{N}(y_n - \hat{y}(x_n))^2$$

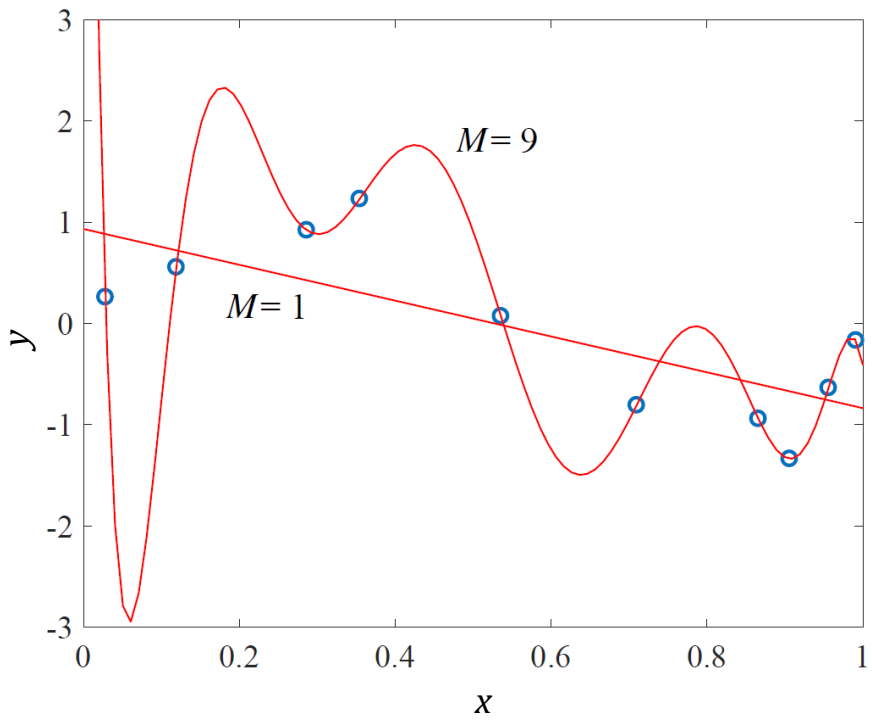


Model Selection: Overfitting

- With $M = 9$, the ML predictor *overfits* the data:
- Model is too rich … to account for the observations in the training set, it appears to yield inaccurate predictions outside it
- Complex models fit not only the signal but also the noise in the data
- Presumably we have a large generalization loss ⇨ low bias and high variance

$$R_p(\hat{y}) = \mathbb{E}_{(x,y)\sim p_{xy}}[(y - \hat{y}(x))^2]$$

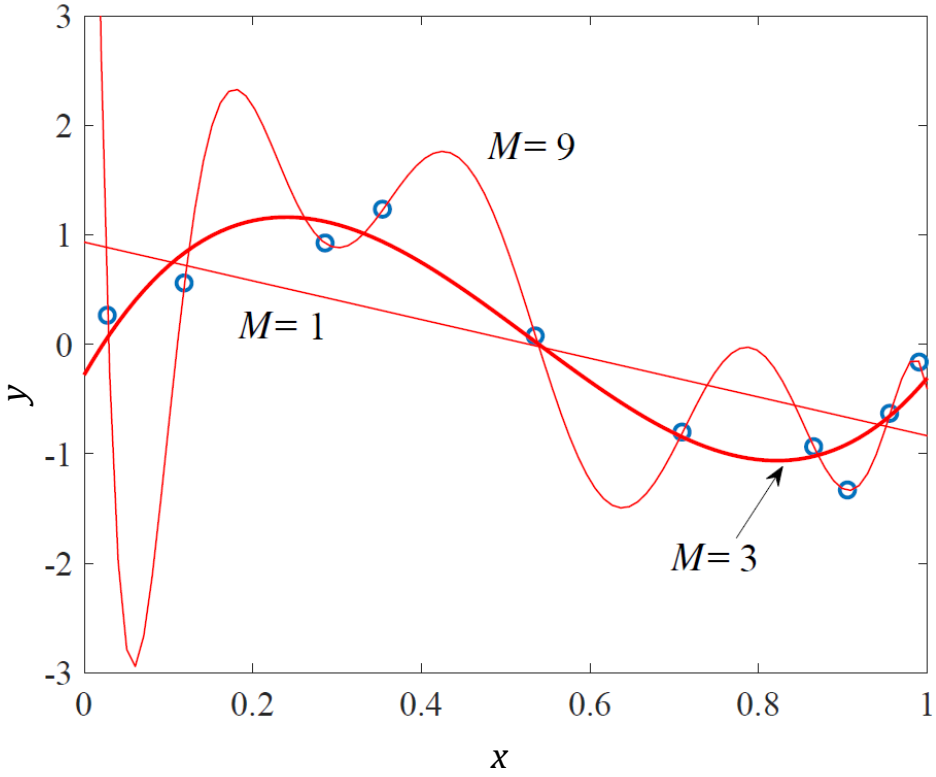
m=9
memorize the data
not good



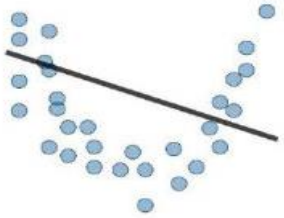
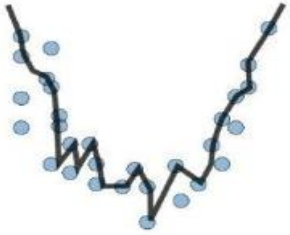
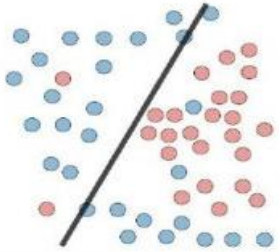
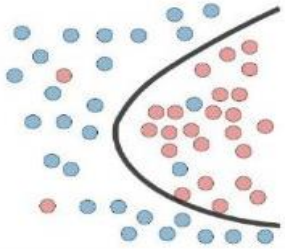
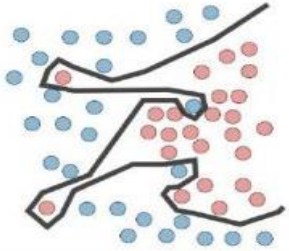
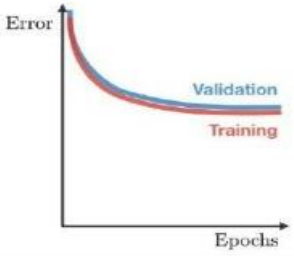
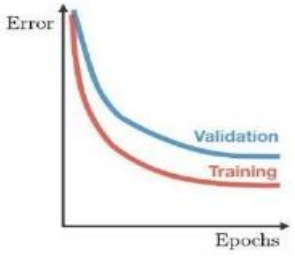
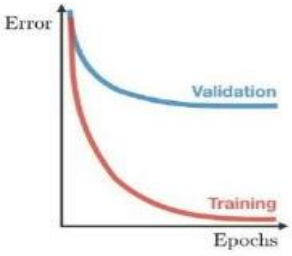
Model Selection

- $M = 3$ seems to be a reasonable choice…

How to pick the model with lowest expected loss / test error?

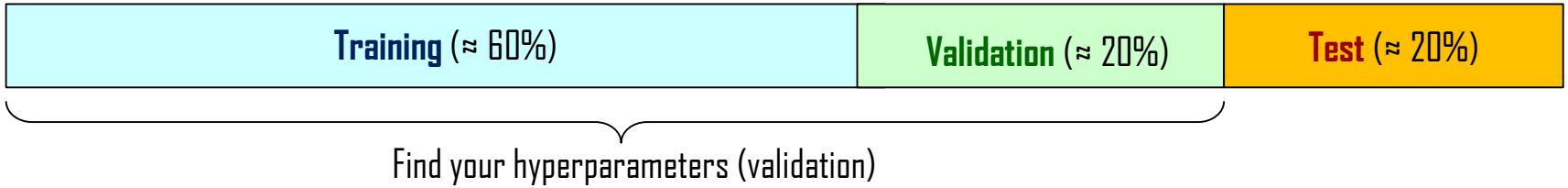


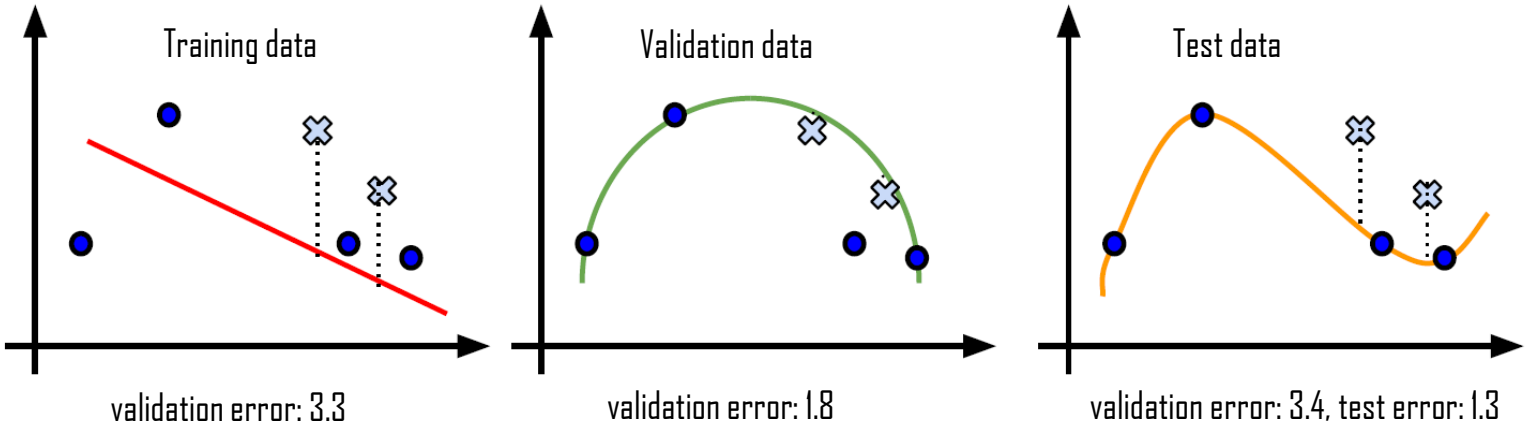
Model Selection

	Underfitting	Just right	Overfitting
Symptoms	- High training error - Training error close to test error - High bias	- Training error slightly lower than test error	- Low training error - Training error much lower than test error - High variance
Regression			
Classification			
Deep learning			
Remedies	- Complexify model - Add more features - Train longer		- Regularize - Get more data

Model Selection

- How to pick the model with lowest expected loss / test error?
- **Validation:** split the labeled data into 3 parts

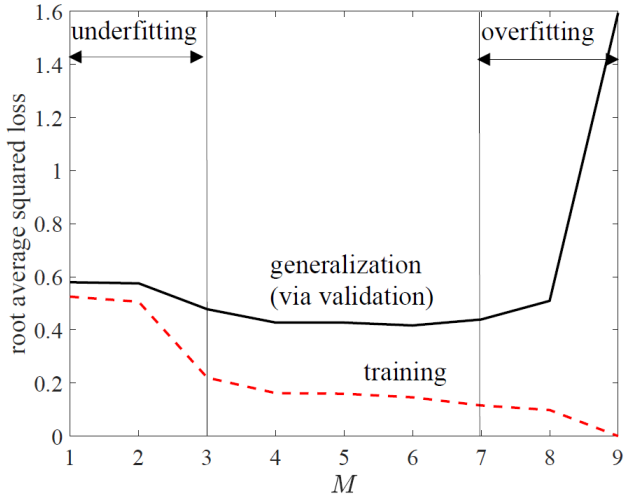


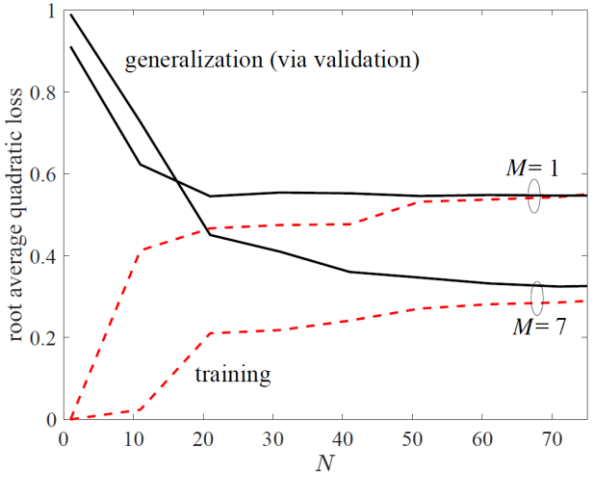


- **Regularization:** bound the test error by bounding
  - training error
  - model complexity

Model Selection: Validation

- Validation allows model order selection.
- Model order selection should depend on the amount of data…
- It is a problem of bias (asymptotic error) versus generalization gap



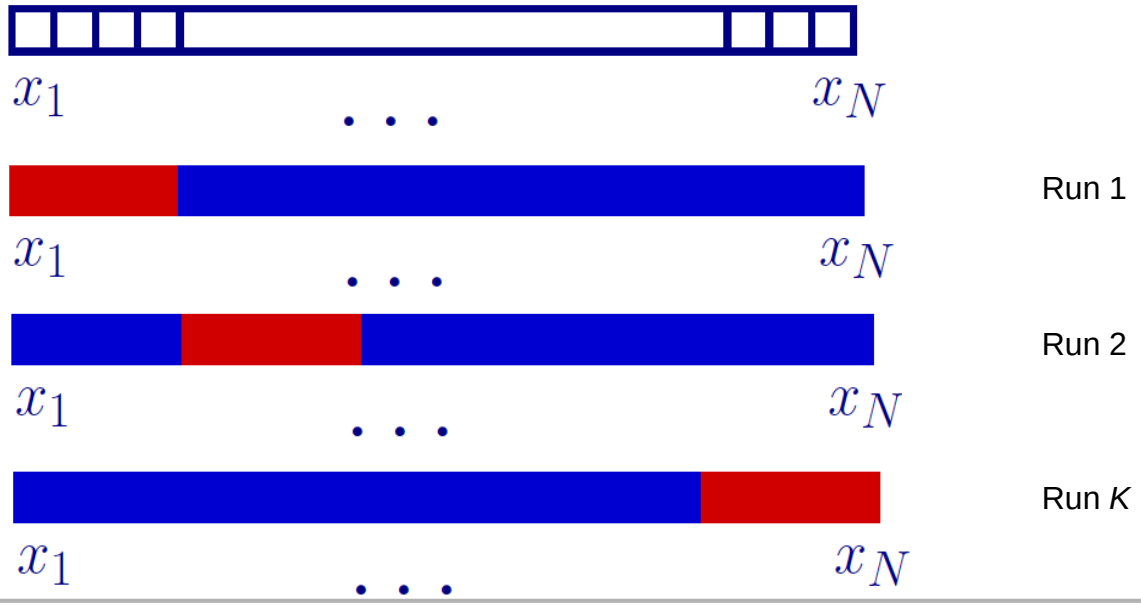


- Validation can also be used to select other hyperparameters (e.g., learning rate).
- **Bias – Variance tradeoff**
  - Relation between training error (bias), complexity (variance) and test error (bias+variance)

Cross-validation

- Split data into two parts (training and testing) is **not** the most efficient way to use the data
- ***K*-fold cross validation:** *split the training data into K folds*
- Randomly partition data into *K* roughly equal parts (groups)
- Train on all but *j*-th part, test on *j*-th part (train *K* times)
- Average the *K* results

- **+** Use all data for training and all data for testing, and use each data point the same number of times.
- Cross-validation returns an unbiased estimate of the generalization error and its variance.
- Leave-one-out cross validation: $K = N$



Generalization Bounds

- Upper bound on

$$\mathbb{P}\left[\sup_{h\in H}\left|R(h) - \hat{R}(h)\right| > \epsilon\right]$$

- Bound on estimation error for hypothesis h_0 given by ERM:

$$\begin{aligned} R(h_0) - R(h^*) &= R(h_0) - \hat{R}(h_0) + \hat{R}(h_0) - R(h^*) \\ &\leq R(h_0) - \hat{R}(h_0) + \hat{R}(h^*) - R(h^*) \\ &\leq 2\sup_{h\in H}\left|R(h) - \hat{R}(h)\right| \end{aligned}$$

- How should we choose H? (model selection problem)

Known True Distribution

- No need for data
- This is a standard **inference problem**, i.e., estimation (regression) or detection (classification)
- The solution can be directly computed from the *posterior* or *predictive* distribution

$$p(y|x) = \frac{p(x,y)}{p(x)}$$

$$\text{as } \hat{y}^*(x) = \arg\min_{\hat{y}} \mathbb{E}_{y\sim p_{y|x}}[\ell(y,\hat{y})|x]$$

Examples

- With quadratic loss, conditional mean: $\hat{y}^*(x) = \mathbb{E}_{y\sim p_{y|x}}[y|x]$

Hint: $R(h) = \frac{1}{2}\mathbb{E}[y - \mathbb{E}[y|x]]^2 + \frac{1}{2}\mathbb{E}[\mathbb{E}[y|x] - \hat{y}]^2$

- With probability of error, maximum a posteriori (MAP): $\hat{y}^*(x) = \arg\max_y p(y|x)$

Known True Distribution

Example

- With joint distribution

x\y	0	1
0	0.05	0.45
1	0.4	0.1

- We have $p(y = 1|x = 0) = 0.9$

- For quadratic loss, we have $\hat{y}^*(x) = \mathbb{E}_{y \sim p_{y|x}}[y|x]$

$$\hat{y}^*(x = 0) = 0.9 \times 1 + 0.1 \times 0 = 0.9$$

- For probability of error, MAP: $\hat{y}^*(x) = \arg\max_y p(y|x)$

$$\hat{y}^*(x = 0) = 1$$

Unknown True Distribution

- Need for data
- This is a **learning problem**

1. Model selection: Define a parametric model (Bayesian vs classical/frequentist)

$$\underbrace{p(x, y|\theta)}_{\text{generative model}} \quad \text{or} \quad \underbrace{p(y|x, \theta)}_{\text{discriminative model}}$$

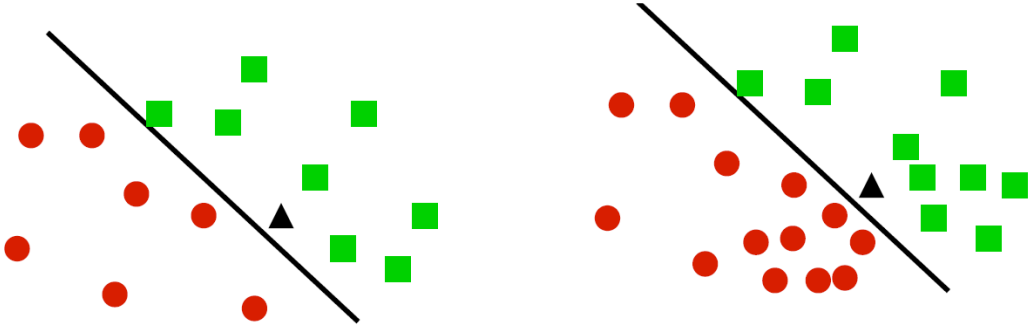
- Estimate joint distribution over $p(x, y)$
- Use Bayes Rule to infer $p(y|x)$
- But, can generate a sample, $p(\mathrm{x}) = \sum_y p(y)p(x|y)$
- Easier to add prior knowledge
- Sensitive to mis-specification

- Estimate $p(y|x)$ directly
- Often better convergence + simpler solutions
- Learn "discriminant" function $h(x)$ (SVM)
- Direct but cannot obtain a data sample, because $p(x)$ is not available

2. Learning: Given data $\mathcal{S}$, optimize a learning criterion to obtain the parameter θ

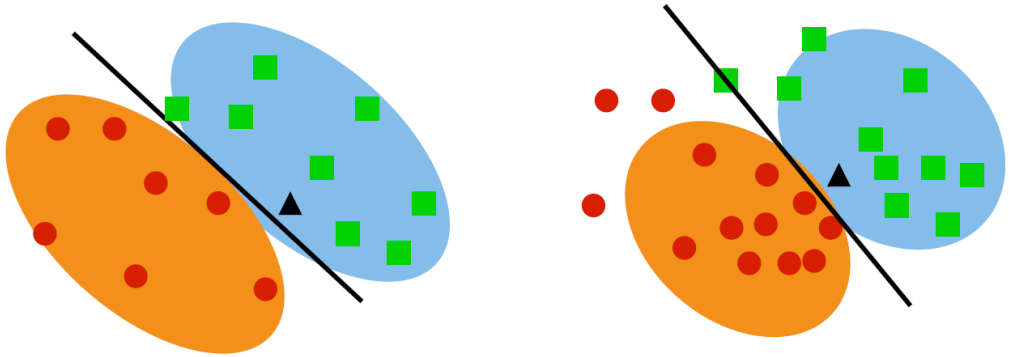
3. Inference: Use model to obtain the predictor $\hat{y}(x)$ (to be tested on new data)

Discriminative vs. Generative Models



Discriminative

- Only supervised, not for unlabeled data
- Very good when underlying distribution of data is really complicated (e.g. texts, images, movies)



Generative

- Model observations (x, y) first hen infer $p(y|x)$
- Good for missing variables, better diagnostics
- Easy to add prior knowledge about data
- Natural use of unlabeled data

Gradient Descent

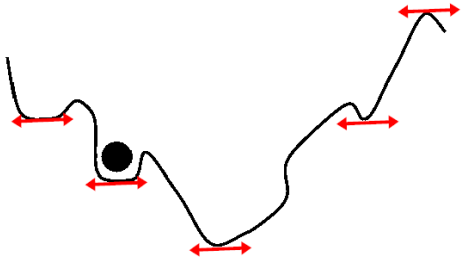
Gradient Descent

ANALYSE MATHÉMATIQUE. — *Méthode générale pour la résolution des systèmes d'équations simultanées; par* M. AUGUSTIN CAUCHY.

- Iterative first order optimization algorithm for finding a local minimum of a differentiable and convex function
- To find values of a function's parameters that minimize the loss function as far as possible, e.g. linear regression

- Start from some $\mathbf{w}_0$
- A step: $\mathbf{w}_{i+1} = \mathbf{w}_i - \eta \, \nabla J(\mathbf{w}_i)$

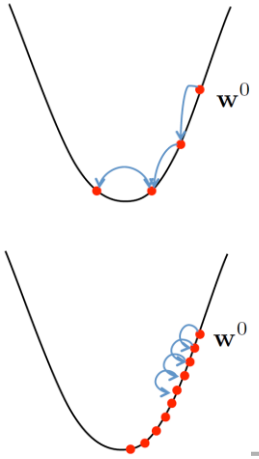
- **GDA: calculate the next point using gradient at the current position, scale it by learning rate, subtract from current position**

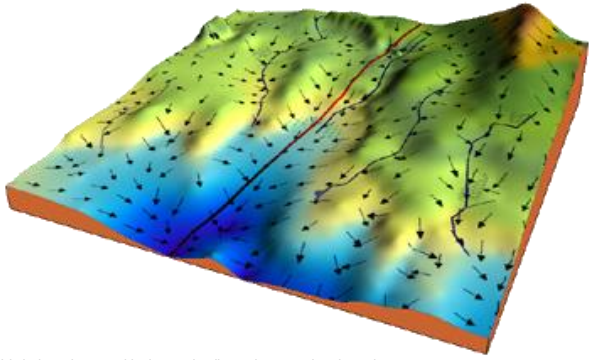


Non convex
(most DNN problems)

Learning rate η

- Large η $\Rightarrow$ Fast convergence but larger residual error.
- Small η $\Rightarrow$ Slow convergence but small residual error





https://ml-cheatsheet.readthedocs.io/en/latest/_images/gradient_descent.png

EURECOM
Sophia Antipolis

Statistical Estimation

True underlying distribution

$$p_{data}(\boldsymbol{y}|\mathbf{X})$$

Parametric family of distributions

$$p_{model}(\boldsymbol{y}|\mathbf{X}, \boldsymbol{\theta})$$

Observations from $p_{data}(\boldsymbol{y}|\mathbf{X})$

Controlled by parameter(s)

Maximum Likelihood Estimate (MLE)

- Method of estimating the parameters of a statistical model given observations, by finding the parameter values θ that maximize the likelihood of making the observations given the parameters.

$$\widehat{\boldsymbol{\theta}}_{ML} = \arg\max_{\boldsymbol{\theta}} p_{model}(\boldsymbol{y}|\mathbf{X}, \boldsymbol{\theta})$$

$$= \arg\max_{\boldsymbol{\theta}} \prod_{n=1}^{N} p_{model}(y_n|\mathbf{x}_n, \boldsymbol{\theta}) \qquad \textit{i.i.d. assumption}$$

$$= \arg\max_{\boldsymbol{\theta}} \sum_{n=1}^{N} p_{model}(y_n|\mathbf{x}_n, \boldsymbol{\theta})$$

- Linear Regression: assuming $y_n = \mathcal{N}(\mathbf{x}_n\boldsymbol{\theta}, \boldsymbol{\sigma^2}) \Rightarrow \widehat{\boldsymbol{\theta}} = (\boldsymbol{X^T X})^{-1}\boldsymbol{X^T y}$ (least square estimate)

Coin Flipping

3/5
frequency of Heads

- Frequency of heads is exactly the MLE for this problem
- Data $D = \{X_n\}_{n=1}^{N}, X_n \in \{\mathrm{H}, \mathrm{T}\}$ with α_H heads and α_T tails outcomes $(\alpha_H + \alpha_T = N)$
- $P(\mathrm{Head}) = \theta,\ P(\mathrm{Tail}) = 1 - \theta$
- i.i.d. flips: independent events & ident. distributed 1's and 0's (Bernoulli)
- α_H and α_T are counts that sum these outcomes (Binomial) – **Hypothesis**
- $P(D|\theta) = P(\alpha_H, \alpha_T|\theta) \sim \theta^{\alpha_H}(1-\theta)^{\alpha_T}$

$$\hat{\theta}_{ML} = \arg\max_{\theta} P(D|\theta) = \arg\max_{\theta} \prod_{n=1}^{N} P(X_n|\theta) \quad \textit{ind. draws}$$

$$= \arg\max_{\theta} \prod_{n:X_n=H} \theta \prod_{n:X_n=T} (1-\theta) \quad \textit{ident. distr.}$$

$$= \arg\max_{\theta} \theta^{\alpha_H}(1-\theta)^{\alpha_T}$$

$$= \arg\max_{\theta} \log[\theta^{\alpha_H}(1-\theta)^{\alpha_T}] \quad \textit{log likelihood}$$

$$\boxed{\hat{\theta}_{ML} = \frac{\alpha_H}{N}}$$

How good is this MLE?

I want to know the parameter θ*, within* $\epsilon = 0.1$ *error with probability at least* $1 - \delta = 0.95$.

Q: How many flips do I need ?

Hoeffding's inequality (1963)

$$\mathbb{P}\left(\left|\hat{\theta} - \theta^*\right| \geq \epsilon\right) \leq 2e^{-2N\epsilon^2}$$

- Sample complexity: how many data samples do I need to achieve a certain level of performance?

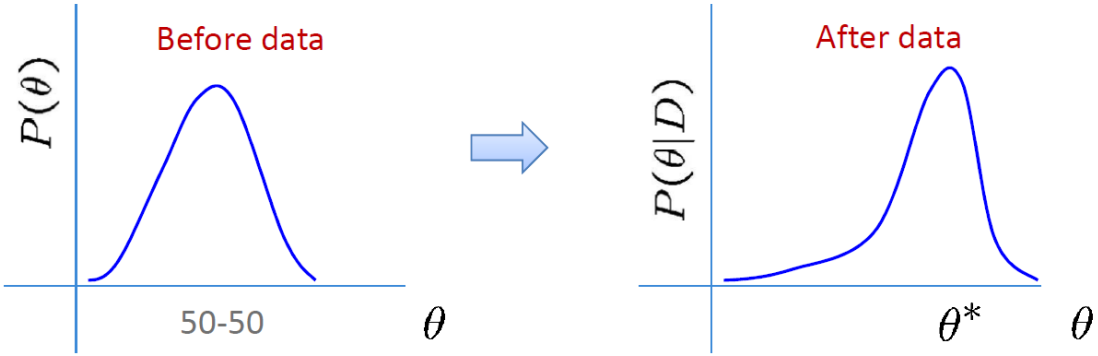
$$N \geq \frac{\log(2/\delta)}{2\epsilon^2}$$

Q: Is this an ML problem?

- improve their **performance** (accuracy of the predicted prob.)
- at some **task** (predicting the probability of heads)
- with **experience** (the more coins we flip the better we are)

Bayesian Learning – MAP Estimation

- Prior knowledge: we know coin is close to "50-50"
- Rather than estimating a single θ, we obtain a distribution over possible values of θ



- Coin flipping: likelihood is Binomial $P(D|\theta) = \binom{N}{\alpha_H}\theta^{\alpha_H}(1-\theta)^{\alpha_T}$
- If prior distribution is Beta: $P(\theta) \sim Beta(\beta_H, \beta_T)$ [as $N \to \infty$, prior is forgotten]
- Posterior distribution: $P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)} \sim Beta(\alpha_H + \beta_H, \alpha_T + \beta_T)$
- Maximum A Posteriori (MAP) Estimation: choose the parameter value that is most probable given observed data and prior belief

$$\hat{\theta}_{MAP} = \arg\max_{\theta} P(\theta|D) = \arg\max_{\theta} P(D|\theta)P(\theta)$$

$$\hat{\theta}_{MAP} = \frac{\alpha_H + \beta_H - 1}{N + \beta_H + \beta_T - 2}$$

Signal Detection

- Questions
  - Is a signal of specific model present in our time series? E.g., detection of noisy sinusoid; beep or no beep?
  - Is the transmitted pulse present at radar signal at time t?
  - Does the mean level of a signal change at time t?
  - After calculating the mean change in pixel values of subsequent frames in video, is there something moving in the scene?
  - Is there a person in this video frame?

- Closely related to *hypothesis testing*

Supervised Classification & Statistical Detection Theory

- Both observe noisy data and output a decision on the discrete unknown it originated from.

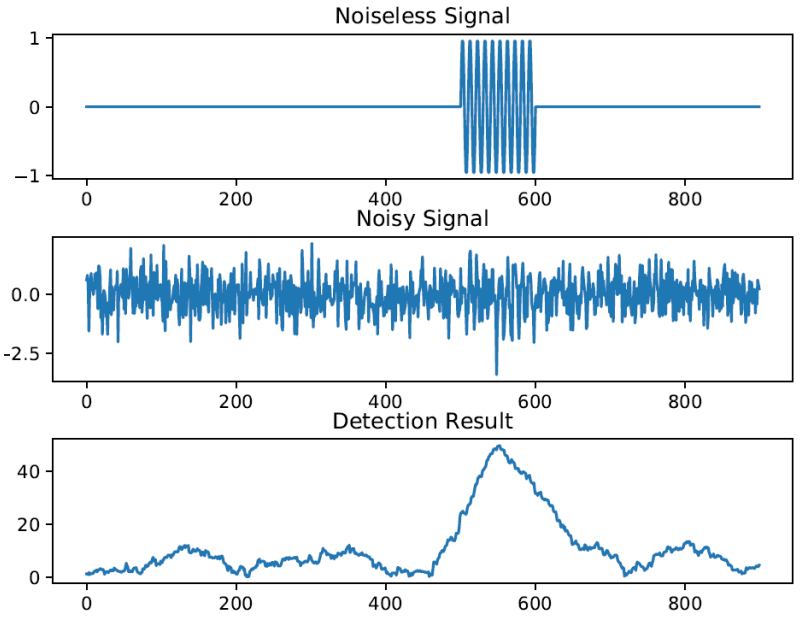
- Detection theory: based on a prior probabilistic model of the environment
- Supervised Classification: data driven and based on examples

Signal Detection

- Detection of a sinusoidal waveform
- Hypothesis testing for detection of noisy sinusoid

$$\mathcal{H}_0 : x[k] = n[k]$$
$$\mathcal{H}_1 : x[k] = A\cos(2\pi f_0 k + \phi) + n[k]$$

noise only

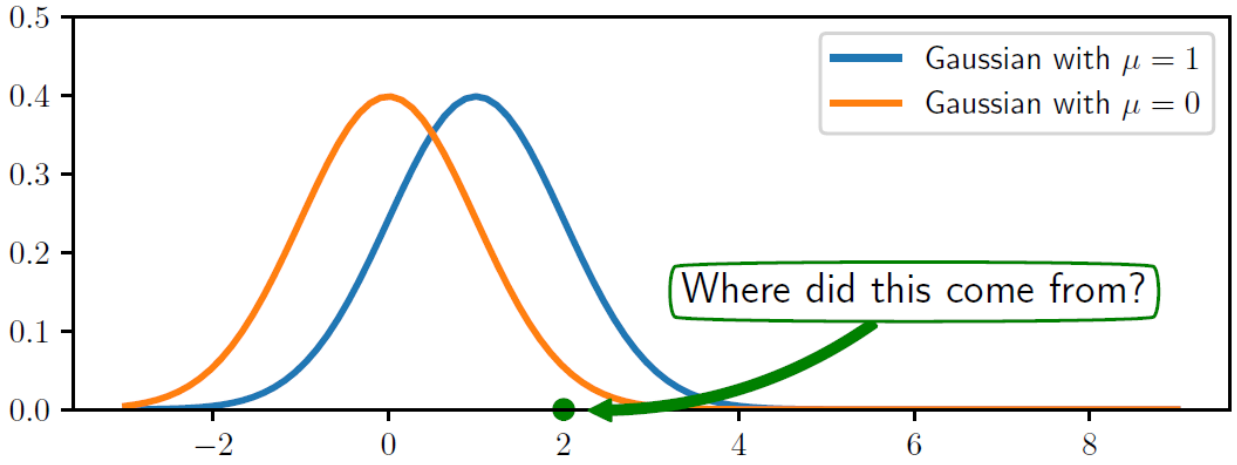


- *Hypothesis* $\mathcal{H}_0$: measurements consists of noise only (*null hypothesis*)
- *Hypothesis* $\mathcal{H}_1$: the sinusoid is present (*alternative hypothesis*)

Signal Detection

Introductory Example

- Consider a very simple detection problem, where we observe one sample $x[0]$ from one of the two densities: $\mathcal{N}(0,1)$ or $\mathcal{N}(1,1)$.
- Goal: choose the correct density in an optimal manner.

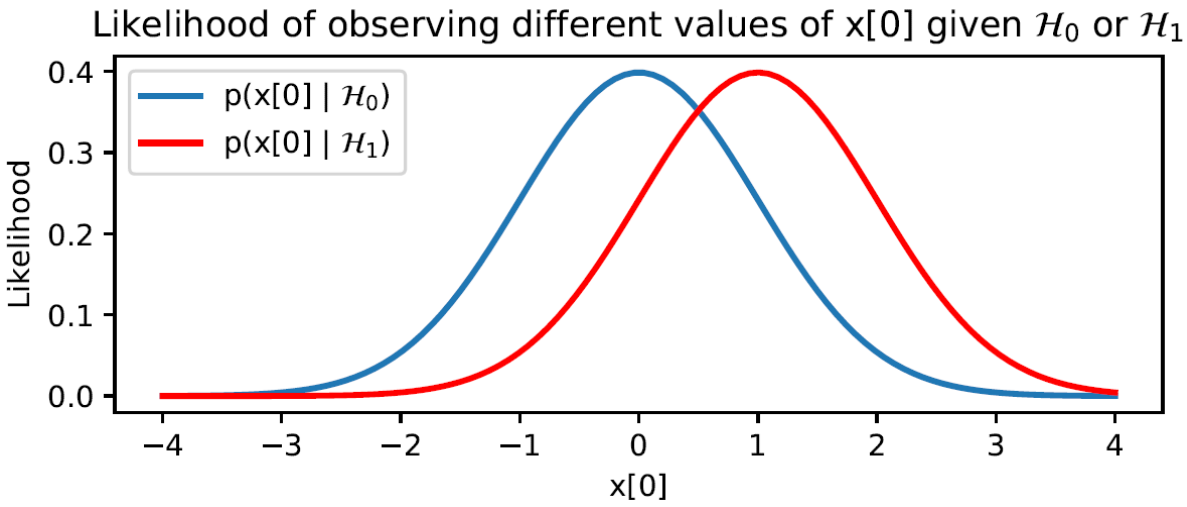


Signal Detection

- The hypotheses are

$$\mathcal{H}_1 : \mu = 1$$
$$\mathcal{H}_0 : \mu = 0$$

and the corresponding likelihoods are plotted below:



Signal Detection

- Obvious approach for deciding the density: choose the one that is higher for a particular $x[0]$.
- More specifically, study the likelihoods and choose the more likely one.
- Likelihoods

$$\mathcal{H}_1 : p(x[0] \mid \mu = 1) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x[0]-1)^2}{2}\right).$$
$$\mathcal{H}_0 : p(x[0] \mid \mu = 0) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x[0])^2}{2}\right).$$

- Select $\mathcal{H}_1$ if "$\mu = 1$" is more likely than "$\mu = 0$".
- In other words, $p(x[0] \mid \mu = 1) > p(x[0] \mid \mu = 0)$.

Signal Detection

- Let's state this in terms of $x[0]$:

$$p(x[0] \mid \mu = 1) > p(x[0] \mid \mu = 0)$$

$$\Leftrightarrow \frac{p(x[0] \mid \mu = 1)}{p(x[0] \mid \mu = 0)} > 1$$

$$\Leftrightarrow \frac{\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x[0]-1)^2}{2}\right)}{\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x[0])^2}{2}\right)} > 1$$

$$\Leftrightarrow \exp\left(-\frac{(x[0]-1)^2 - (x[0])^2}{2}\right) > 1$$

$$\Leftrightarrow (x[0])^2 - (x[0]-1)^2 > 0$$

$$\Leftrightarrow 2x[0] - 1 > 0$$

$$\Leftrightarrow x[0] > \frac{1}{2}.$$

Signal Detection

- In other words, choose $\mathcal{H}_1$ if $x[0] > 0.5$ and $\mathcal{H}_0$ if $x[0] < 0.5$
- *Likelihood Ratio* is a key quantity

$$\frac{p(x[0] \mid \mu = 1)}{p(x[0] \mid \mu = 0)} > \gamma$$

- LRT (likelihood ratio test): comparison to a threshold γ (here $\gamma = 1$).
- Detection threshold γ can be chosen other than 1.

Using the Posterior Distribution

- Also possible to study posterior probability ratios $\frac{p(\mathcal{H}_1|\mathbf{x})}{p(\mathcal{H}_0|\mathbf{x})}$

instead of the likelihood ratio $\frac{p(\mathbf{x}|\mathcal{H}_1)}{p(\mathbf{x}|\mathcal{H}_0)}$

- However, using Bayes rule, this MAP test turns out to be

$$\frac{p(\mathbf{x}|\mathcal{H}_1)}{p(\mathbf{x}|\mathcal{H}_0)} > \frac{p(\mathcal{H}_1)}{p(\mathcal{H}_0)}$$

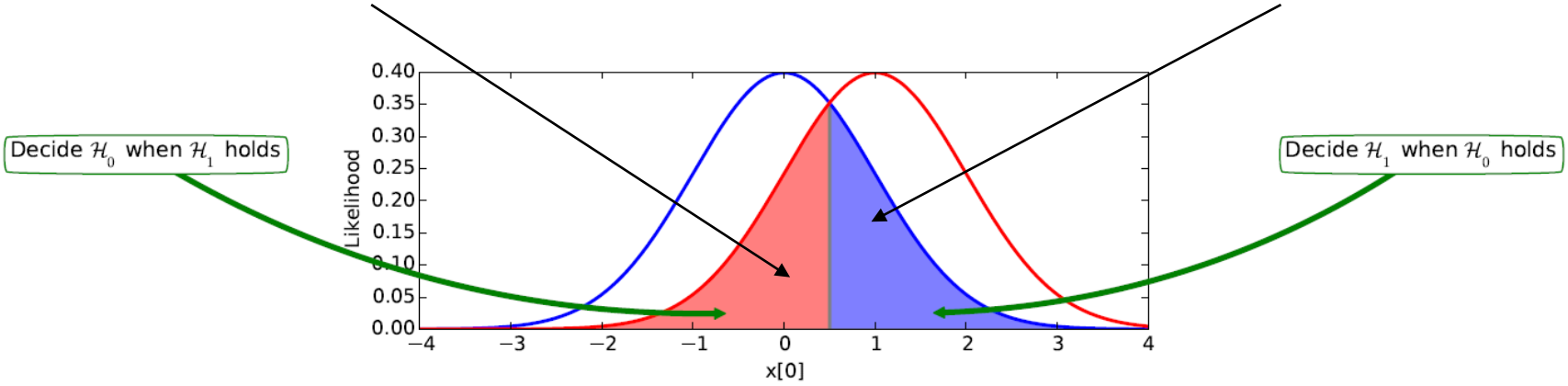
i.e., the only effect of using posterior probability is on the threshold for the LRT. Why?

Error Types

- It might be that the detection problem is not symmetric and some errors are more costly/critical than others.
- Example: in disease detection, a missed detection is more costly than a false alarm.
- The tradeoff between misses and false alarms can be adjusted using the threshold of LRT

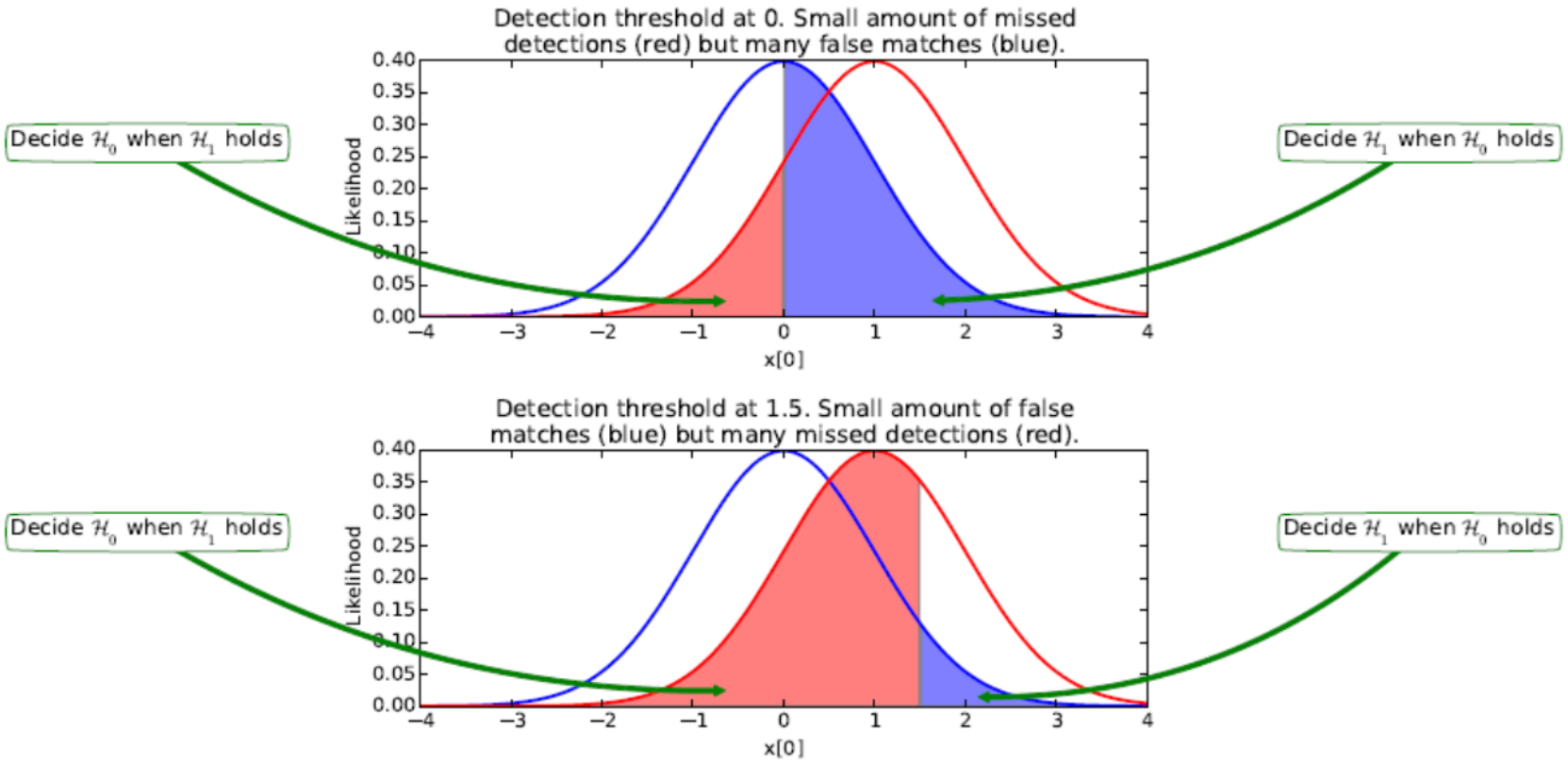
The red area is the probability of choosing $\mathcal{H}_0$ while $\mathcal{H}_1$ would hold (missed detection)

The blue area corresponds to the probability of choosing $\mathcal{H}_1$ while $\mathcal{H}_0$ would hold (false alarm)



Error Types

- We can decrease either probability arbitrarily small by adjusting the detection threshold.



Error Types

- For example, suppose the threshold is $\gamma = 1.5$. What are P_{FA} and P_D?
- Probability of false alarm is found by integrating over the blue area:

$$P_{FA} = P(x[0] > \gamma | \mu = 0) = \int_{1.5}^{\infty} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x[0])^2}{2}\right) dx[0] \approx 0.0668$$

- Probability of missed detection is the area marked in red:

$$P_M = P(x[0] < \gamma | \mu = 1) = \int_{-\infty}^{1.5} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x[0]-1)^2}{2}\right) dx[0] \approx 0.6915$$

- Probability of detection: complement of P_M, equivalent but more useful term (the power of the test!)

$$P_D = 1 - P_M = \int_{1.5}^{\infty} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x[0]-1)^2}{2}\right) dx[0] \approx 0.3085$$

Neyman-Pearson Theorem

- Since P_{FA} and P_D depend on each other, we would like to maximize P_D subject to given maximum allowed P_{FA}.
- **Neyman-Pearson Theorem**: For a fixed P_{FA}, the likelihood ratio test maximizes P_D with the decision rule

$$\mathrm{L}(\mathbf{x}) = \frac{p(\mathbf{x}; \mathcal{H}_1)}{p(\mathbf{x}; \mathcal{H}_0)} > \gamma$$

where threshold γ is the value for which

$$\int_{\mathbf{x}:\mathrm{L}(\mathbf{x})>\gamma} p(\mathbf{x}; \mathcal{H}_0)\, d\mathbf{x} = P_{FA}$$

Threshold Selection

- Often we don't want to define the threshold but the amount of false alarms we can tolerate.
- For example, suppose we want to find the best detector for our introductory example, and we can tolerate 10% false alarms (P_{FA}=0.1)
- According to Neyman-Pearson theorem, the likelihood ratio detection rule is

$$\text{Select } \mathcal{H}_1 \text{ if } \frac{p(x[0] \mid \mu=1)}{p(x[0] \mid \mu=0)} > \gamma$$

- The only thing to find out now is the threshold γ such that

$$\int_{\gamma}^{\infty} p(x[0] \mid \mu = 0)\, dx = 0.1$$

Detection of random signals

- Suppose $\mathbf{s} \sim \mathcal{N}(0, \mathbf{C}_s)$ and $\mathbf{n} \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$
- The detection problem is a hypothesis test

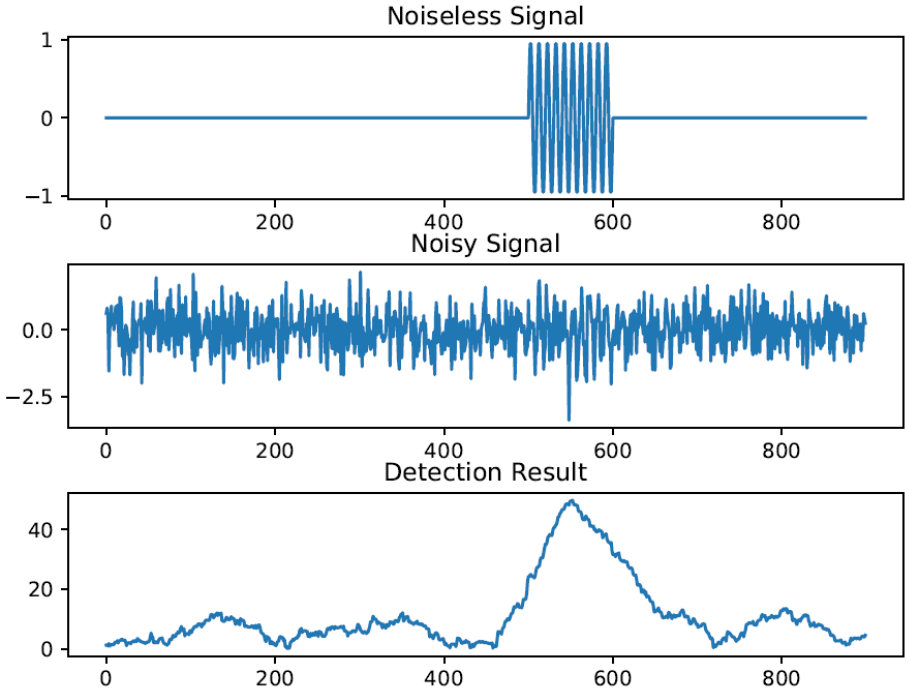
$$\mathcal{H}_1 : \mathbf{x} \sim \mathcal{N}(0, \mathbf{C}_s + \sigma^2 \mathbf{I})$$
$$\mathcal{H}_0 : \mathbf{x} \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$$

- The decision rule becomes

$$\text{Decide } \mathcal{H}_1 \text{ if } \mathbf{x}^T \hat{\mathbf{s}} > \gamma$$

where

$$\hat{\mathbf{s}} = \mathbf{C}_s (\mathbf{C}_s + \sigma^2 \mathbf{I})^{-1} \mathbf{x}$$



Receiver Operating Characteristics (ROC)

- A usual way of illustrating the detector performance is the *Receiver Operating Characteristics* curve (ROC curve).
- This describes the relationship between P_{FA} and P_D for all possible values of the threshold γ.
- The functional relationship between P_{FA} and P_D depends on the problem and the selected detector.

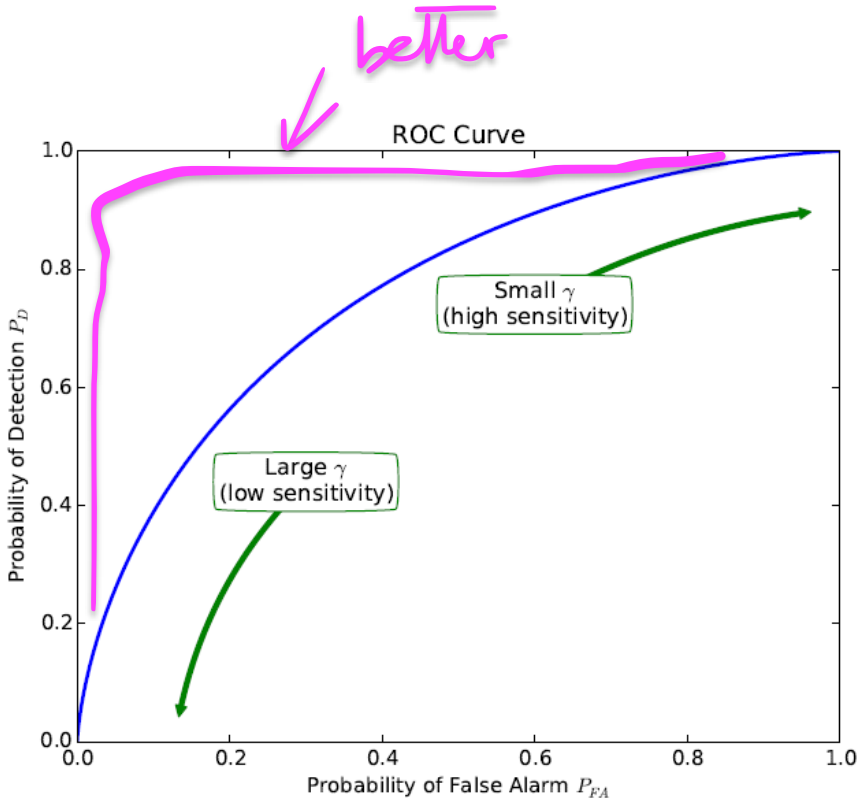
- For example, in the DC level example

$$P_D(\gamma) = \int_{\gamma}^{\infty} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x-1)^2}{2}\right) dx$$

$$P_{FA}(\gamma) = \int_{\gamma}^{\infty} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) dx$$

- It is easy to see the relationship:

$$P_D(\gamma) = \int_{\gamma-1}^{\infty} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) dx = P_{FA}(\gamma - 1)$$

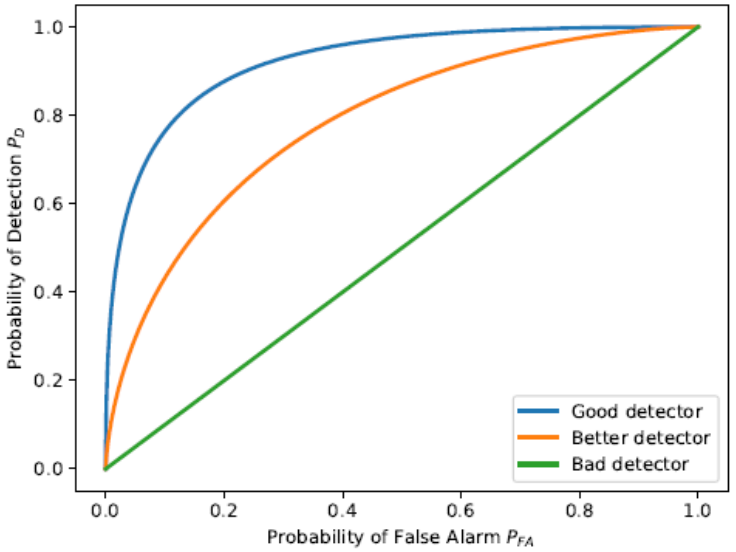


ROC curve for all γ

EURECOM Sophia Antipolis

Receiver Operating Characteristics (ROC)

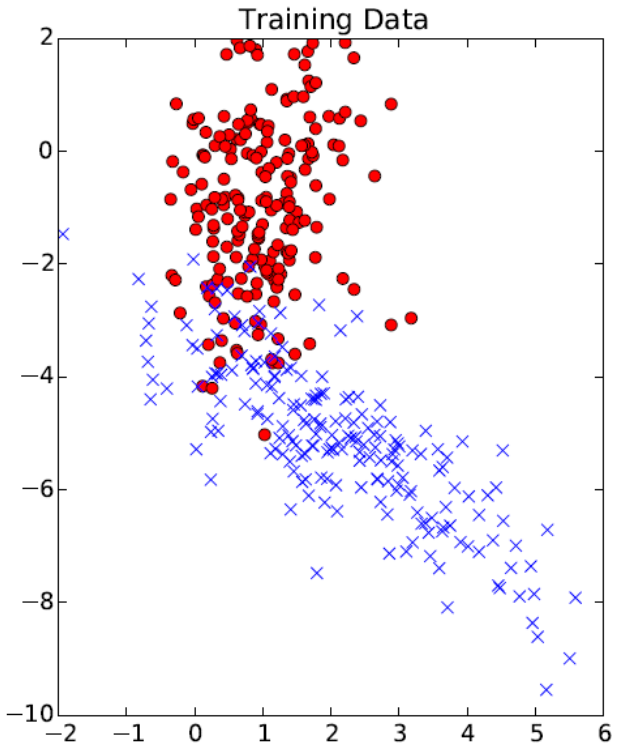
- The higher the ROC curve, the better the performance
- A random guess has diagonal ROC curve
- This gives rise to a widely used measure for detector performance: the *Area Under ROC curve (AUC)* criterion.
- The benefit of AUC is that it is threshold independent, and tests the accuracy for *all* thresholds.

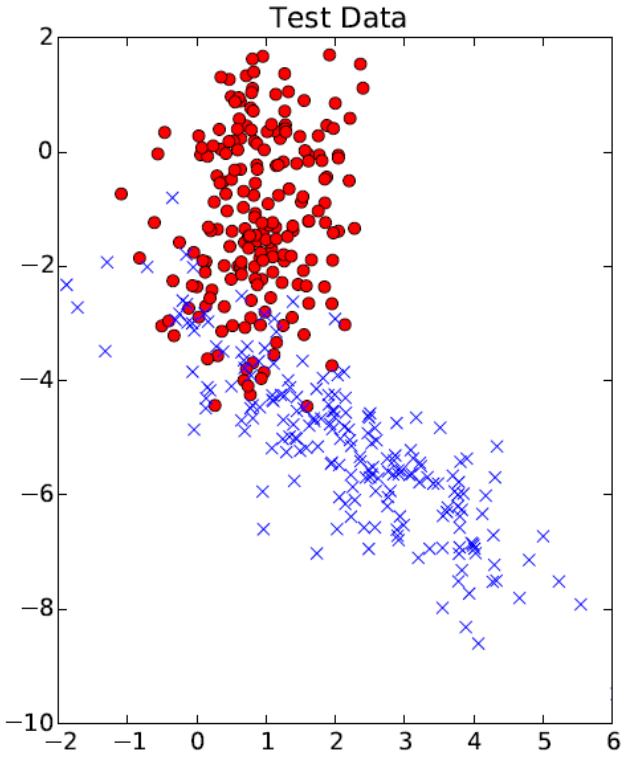


- Initially, AUC and ROC stem from radar and radio detection problems.
- More recently, AUC has become one of the standard measures of classification performance, as well.
- Usually, a closed form expression for P_D and P_{FA} cannot be derived.
- Thus, ROC and AUC are often computed empirically, i.e., by evaluating the prediction results on a holdout test set.

Classification Example – ROC and AUC

- Consider the two-dimensional dataset
- Data is split into *training* and *test* sets, which are *similar* but not exactly the *same*.
- We train four classifiers on the training data and compute the ROC for each on the test data.





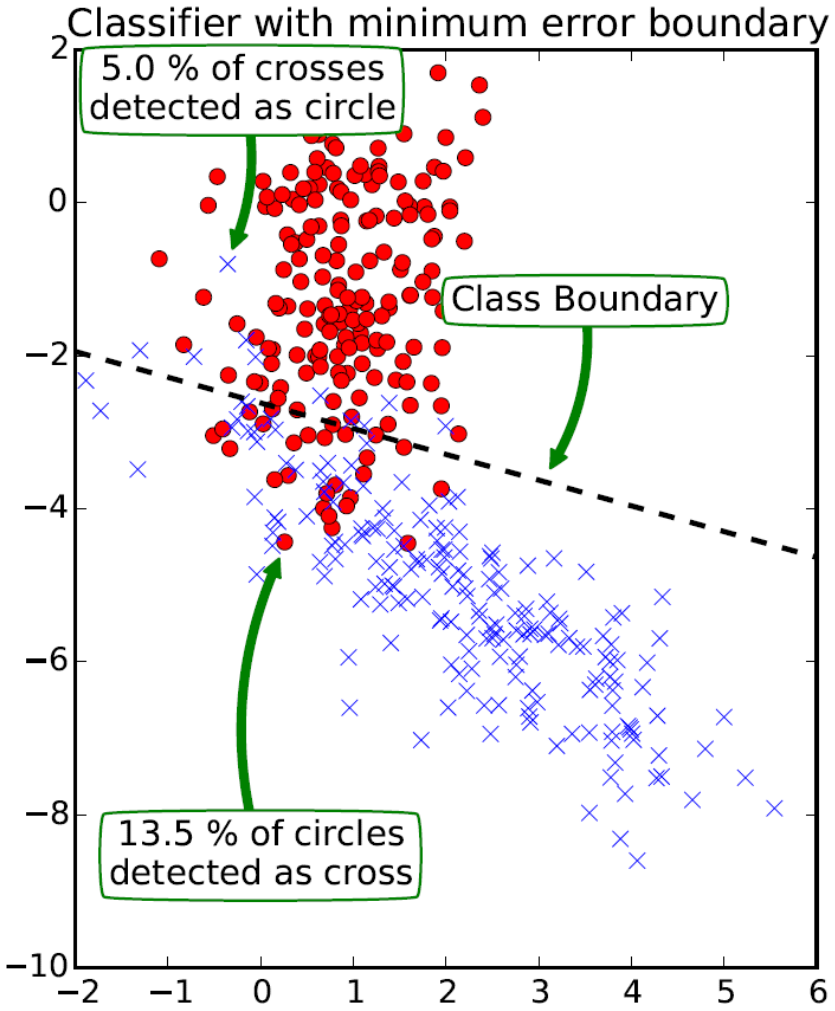
EURECOM
Sophia Antipolis

Classification Example – ROC and AUC

- A linear classifier trained with the training data produces the class boundary shown in the figure.
- The class boundary has the orientation and location that minimize the overall classification error for the training data.
- The boundary is defined by

$$y = c_1 x + c_0$$

with parameters c_1 and c_0 learned from data.

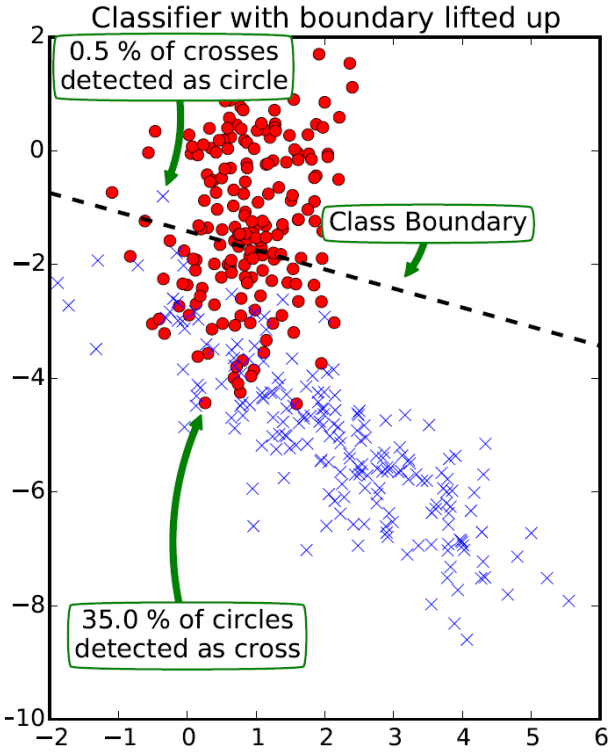


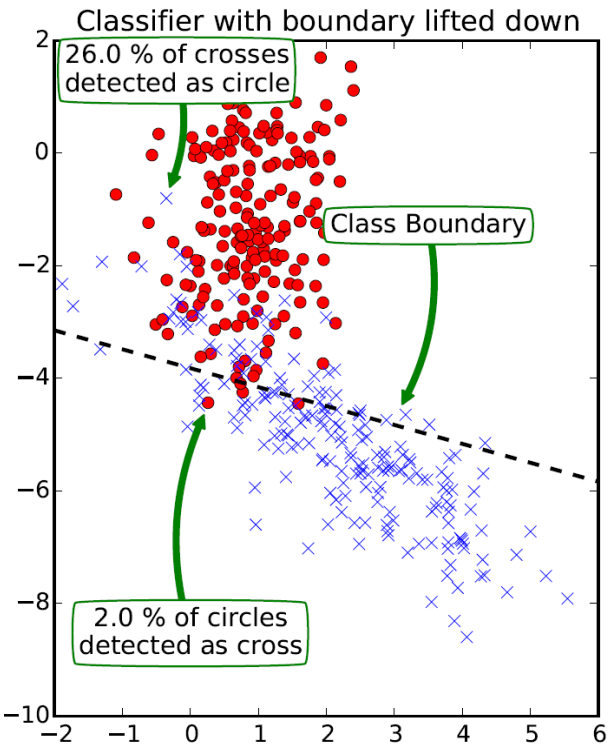
Classification Example – ROC and AUC

- We can adjust the sensitivity of classification by moving the decision boundary up or down.
- In other words, slide the parameter c_0 in

$$y = c_1 x + c_0$$

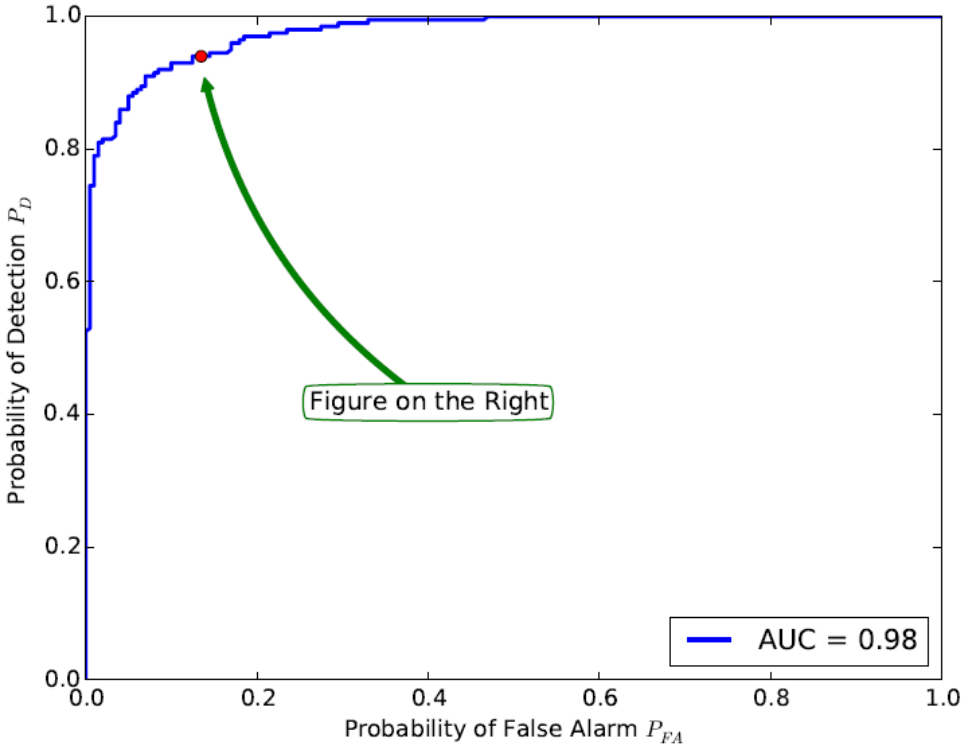
- This can be seen as a *tuning parameter* for plotting the ROC curve

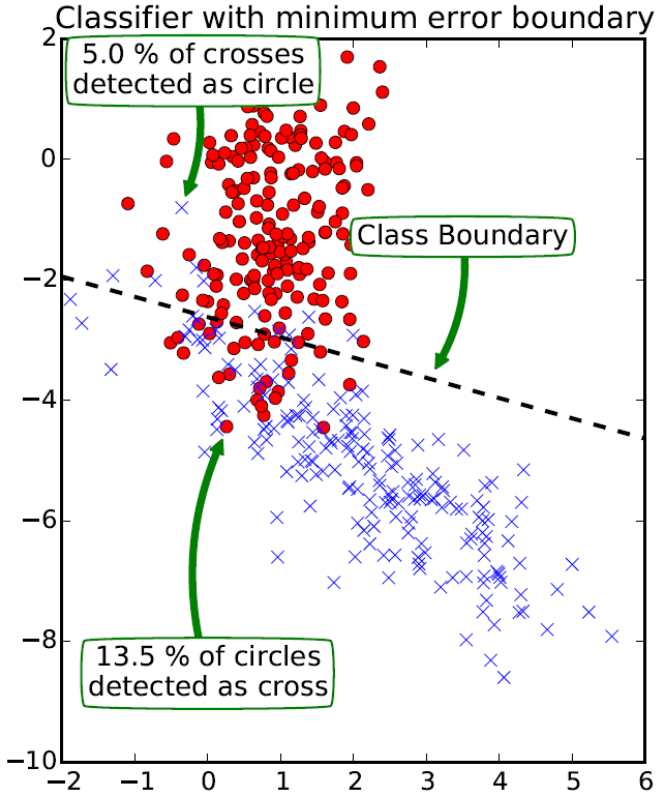




Classification Example – ROC and AUC

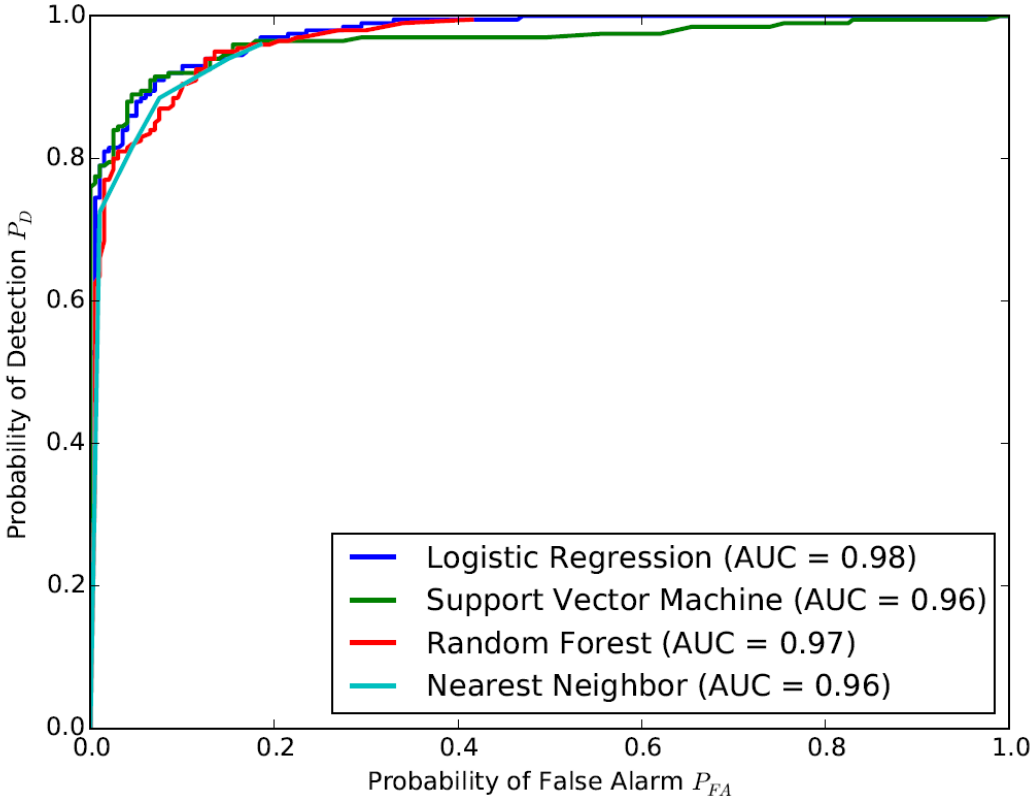
- When the boundary slides from bottom to top, we plot the *empirical ROC curve*
- Plotting starts from upper right corner
- Every time the boundary passes a blue cross, the curve moves left
- Every time the boundary passes a red circle, the curve moves down





Classification Example – ROC and AUC

- Real usage is for comparing classifiers
- Plot of ROC curves for 4 widely used classifiers (below)
- Each classifier produces a class membership score over which the tuning parameter slides



Metrics

Balanced dataset

- TP: true positives // FP: false positives
- TN: true negatives // FN: false negatives

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$F_\beta - score = (1 + \beta^2) \times \frac{Precision \times Recall}{\beta^2 \times Precision + Recall} \qquad \beta \in \mathbb{R}^+$$

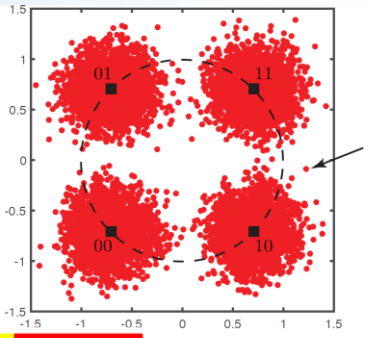
$$True\ Negative\ Rate = \frac{TN}{TN + FP}$$

- F-score is related to the harmonic mean of the precision and recall
- Highest possible F-score is 1 ⇨ perfect precision and recall.

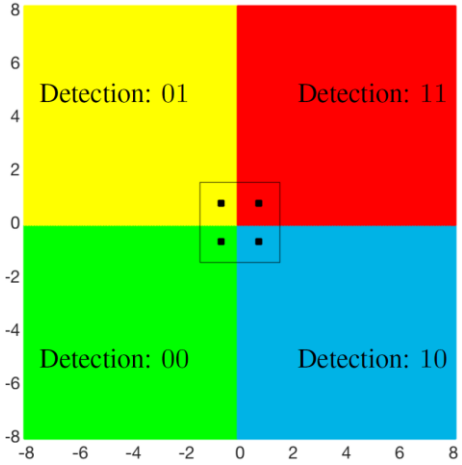
Signal Detection

AWGN Channel

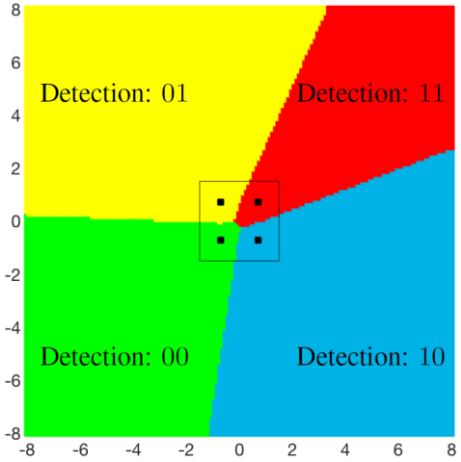
- Received signal: $\boldsymbol{y} = \boldsymbol{s} + \boldsymbol{n}$
  with $\boldsymbol{y} \in \mathbb{R}^2$, $\boldsymbol{s} \in \mathbb{R}^2$, and $\boldsymbol{n} \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$
- Two bits of information encoded into $\boldsymbol{s}$ using QPSK (quadrature phase-shift keying)
- Four possible signal points (equally spaced on the unit circle):



a received signal $\boldsymbol{y}$



Optimal detection regions (detection theory)



Detection regions (neural network)

- NN cannot outperform a known optimal algorithm
- Detection error probability almost the same (most received signals appear within the black box where the NN has a decent behavior).
- Detection regions wrongly shaped - all training examples appeared inside the black box around the signal points. Miscategorizations for RX signals outside the black box
- Domain knowledge could be used for input signal preprocessing.
- E.g., use Euclidean distance between the received signal and each of the four signal constellation points as input (fewer characteristics to learn).