# Statistical Machine Learning

https://cvml.ist.ac.at/courses/SML_W20

Christoph Lampert

Fall Semester 2020/2021
Lecture 2

## Overview (tentative)

| Date | | no. | Topic |
|---|---|---|---|
| Oct 05 | Mon | 1 | A Hands-On Introduction |
| Oct 07 | Wed | 2 | Bayesian Decision Theory, Generative Probabilistic Models |
| Oct 12 | Mon | 3 | Discriminative Probabilistic Models |
| Oct 14 | Wed | 4 | Maximum Margin Classifiers, Generalized Linear Models |
| Oct 19 | Mon | 5 | Estimators; Overfitting/Underfitting, Regularization, Model Selection |
| Oct 21 | Wed | 6 | Bias/Fairness, Domain Adaptation |
| Oct 26 | Mon | - | no lecture (public holiday) |
| Oct 28 | Wed | 7 | Learning Theory I |
| Nov 02 | Mon | 8 | Learning Theory II |
| Nov 04 | Wed | 9 | Deep Learning I |
| Nov 09 | Mon | 10 | Deep Learning II |
| Nov 11 | Wed | 11 | Unsupervised Learning |
| Nov 16 | Mon | 12 | project presentations |
| Nov 18 | Wed | 13 | buffer |

## Refresher from Previous Lecture

**The goal of (supervised) machine learning is**

- use a training set $\mathcal{D} = \{(x^1, y^1), \dots, (x^n, y^n)\} \subset \mathcal{X} \times \mathcal{Y}$
- to find a prediction function $f : \mathcal{X} \to \mathcal{Y}$ (="learning")
- that works well on future data.

## Refresher from Previous Lecture

**The goal of (supervised) machine learning is**

- use a training set $\mathcal{D} = \{(x^1, y^1), \dots, (x^n, y^n)\} \subset \mathcal{X} \times \mathcal{Y}$
- to find a prediction function $f : \mathcal{X} \to \mathcal{Y}$ (="learning")
- that works well on future data.

**Many learning techniques have been developed, such as**

- decision trees
- (k-)nearest neighbor
- Perceptron
- Boosting $\leftarrow$ today
- Artificial Neural Networks $\leftarrow$ today

## Refresher from Previous Lecture

**The goal of (supervised) machine learning is**

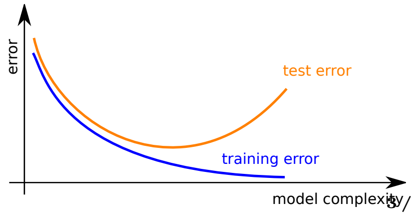
- use a training set $\mathcal{D} = \{(x^1, y^1), \ldots, (x^n, y^n)\} \subset \mathcal{X} \times \mathcal{Y}$
- to find a prediction function $f : \mathcal{X} \to \mathcal{Y}$ (="learning")
- that works well on future data.

**Many learning techniques have been developed, such as**

- decision trees
- (k-)nearest neighbor
- Perceptron
- Boosting $\leftarrow$ today
- Artificial Neural Networks $\leftarrow$ today

**Some phenomena are universal:**

- models with too small complexity underfit the data
  - high training error, high test error
- models with too high complexity overfit the data
  - low training error, high test error



## Boosting [Schapire. 1990]

Given: training examples $\mathcal{D} = \{(x^1, y^1), \dots, (x^n, y^n)\} \subset \mathcal{X} \times \mathcal{Y}$ with $\mathcal{Y} = \{\pm 1\}$.

Problem:

- it's hard to guess a *strong* (=good) classifier.
- it's easy to guess *weak* (=slightly better than random) classifiers.

Question [Kearns, Valiant. 1988/89]:

- Given enough weak classifiers, can one always construct a strong one?

Answer [Schapire. 1990]:

- Yes, by **Boosting**!

## Boosting – Weak Classifiers

For example: if our features are

| property | possible values |
|---|---|
| `eye color` | blue/brown/green |
| `handsome` | yes/no |
| `height` | short/tall |
| `sex` | male (M)/female (F) |
| `soccer fan` | yes/no |

define (weak) classifiers:

$$h_1(x) = \begin{cases} +1 & \text{if } \texttt{eye color} = \texttt{brown} \\ -1 & \text{otherwise.} \end{cases} \qquad h_2(x) = \begin{cases} +1 & \text{if } \texttt{eye color} = \texttt{blue} \\ -1 & \text{otherwise.} \end{cases}$$

$$h_3(x) = \begin{cases} +1 & \text{if } \texttt{eye color} = \texttt{green} \\ -1 & \text{otherwise.} \end{cases} \qquad h_4(x) = \begin{cases} -1 & \text{if } \texttt{eye color} = \texttt{brown} \\ +1 & \text{otherwise.} \end{cases}, \dots$$

$$h_5(x) = \begin{cases} +1 & \text{if } \texttt{handsome} = \text{yes} \\ -1 & \text{otherwise.} \end{cases} \qquad h_6(x) = \begin{cases} -1 & \text{if } \texttt{handsome} = \text{yes} \\ +1 & \text{otherwise.} \end{cases}, \dots$$

Set of all possible combinations: $\mathcal{H} = \{h_1, \dots, h_J\}$.

## AdaBoost – Training

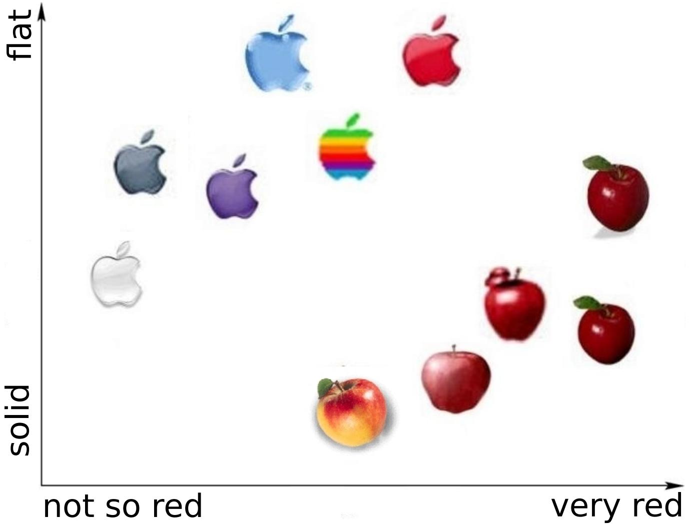
**input** training set $\mathcal{D}$, set of weak classifiers $\mathcal{H}$, number of iterations $T$.

```
  $d_1 = d_2 = \cdots = d_n = 1/n$    (weight for each example)
  for t=1,...,T do
    for $h \in \mathcal{H}$ do $e^t(h) = \sum_{i=1}^{n} d_i \, [\![ h(x^i) \neq y^i ]\!]$    (weighted training error)
    $h_t = \mathbf{argmin}_{h \in \mathcal{H}} \, e^t(h)$    ("best" of the weak classifiers)
    $\alpha_t = \frac{1}{2} \log(\frac{1-e_t(h_t)}{e_t(h_t)})$    (classifier importance, $\alpha_t = 0$ if $e_t(h_t) = \frac{1}{2}$)
    for $i = 1, \dots, n$ do $\tilde{d}_i \leftarrow d_i \times \begin{cases} e^{\alpha_t} & \text{if } h_t(x^i) \neq y^i, \\ e^{-\alpha_t} & \text{otherwise.} \end{cases}$
    for $i = 1, \dots, n$ do $d_i \leftarrow \tilde{d}_i / \sum_i \tilde{d}_i$
  end for
```

**output** classifier: $f(x) = \operatorname{sign} \sum_{t=1}^{T} \alpha_t h_t(x)$

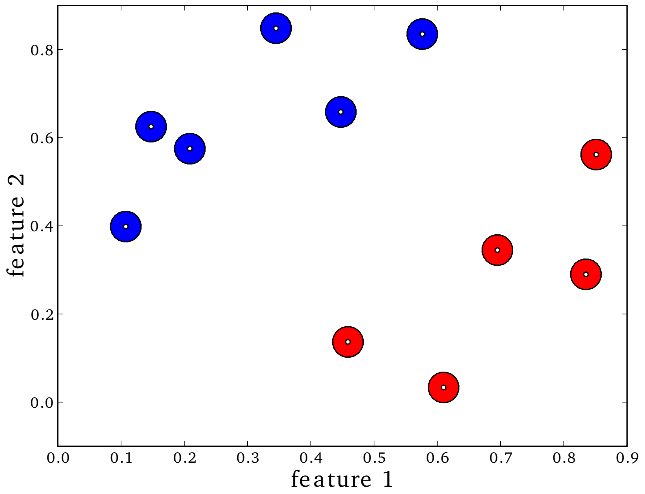
## AdaBoost – Example

Task: $\mathcal{X} = \mathbb{R}^2$, weak classifiers look at each dimension separately



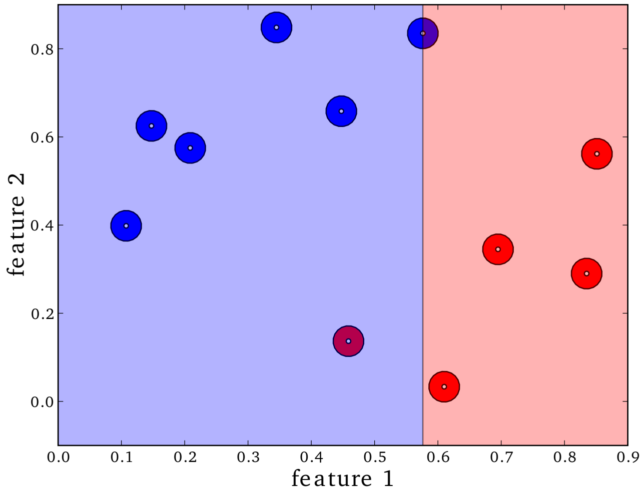
## AdaBoost – Example

Iteration $t = 1, \quad d_1, \ldots, d_n = (\frac{1}{11}, \frac{1}{11}, \frac{1}{11}, \frac{1}{11}, \frac{1}{11}, \frac{1}{11}, \frac{1}{11}, \frac{1}{11}, \frac{1}{11}, \frac{1}{11}, \frac{1}{11}, \frac{1}{11})$



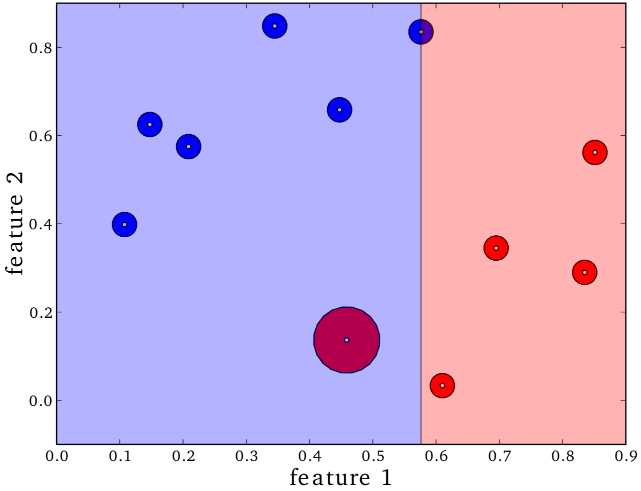
## AdaBoost – Example

Iteration $t = 1$, best weak classifier, $e_1(h_1) = \frac{1}{11}$, $\alpha_1 = 1.15$



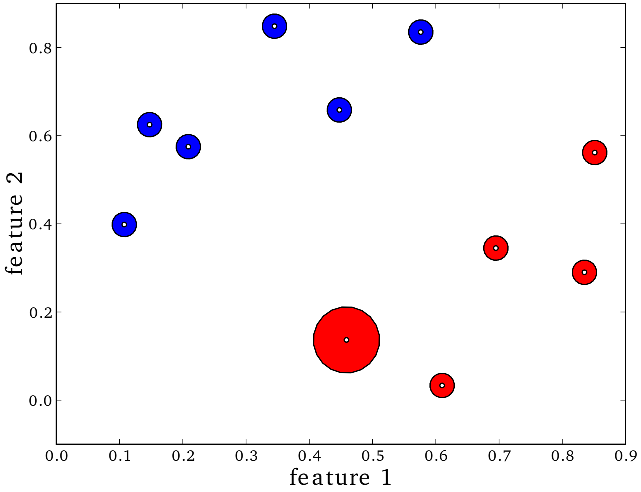
## AdaBoost – Example

Iteration $t = 1$, best weak classifier, $e_1(h_1) = \frac{1}{11}$, $\alpha_1 = 1.15$



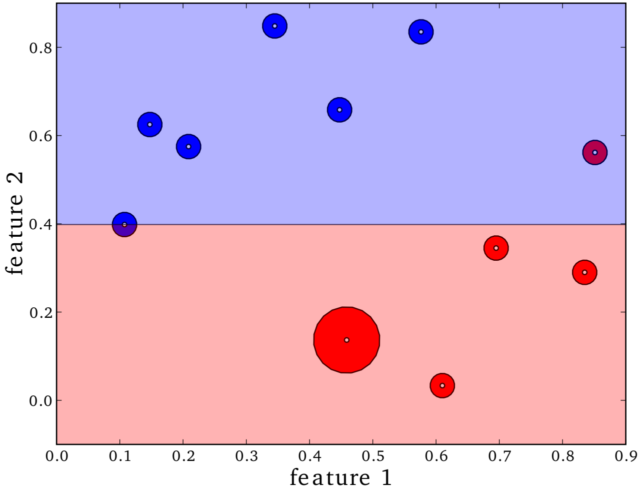
## AdaBoost – Example

Iteration $t = 2$, $\quad d_1, \ldots, d_n \approx (\frac{1}{20}, \frac{1}{20}, \frac{1}{20}, \frac{1}{20}, \frac{1}{20}, \frac{1}{20}, \frac{1}{20}, \ \frac{1}{2} \ , \frac{1}{20}, \frac{1}{20}, \frac{1}{20}, \frac{1}{20})$



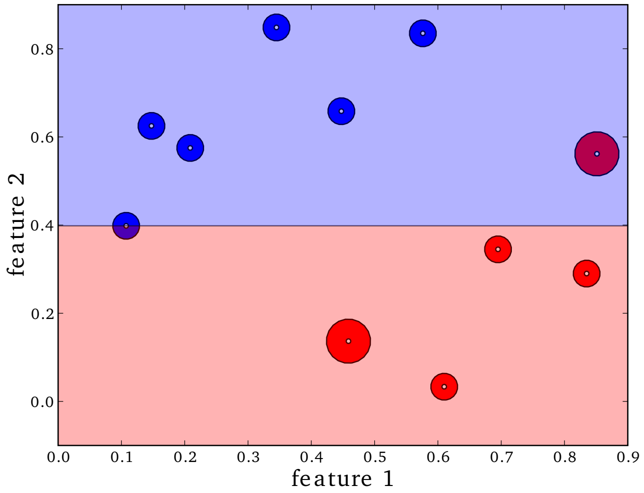
## AdaBoost – Example

Iteration $t = 2$, best weak classifier, $e_2(h_2) = \frac{1}{20}$, $\alpha_2 = 1.47$



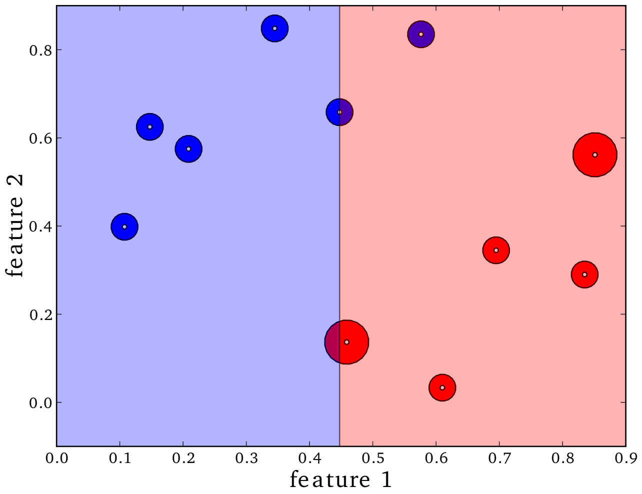
## AdaBoost – Example

Iteration $t = 2$, best weak classifier, $e_2(h_2) = \frac{1}{20}$, $\alpha_2 = 1.47$



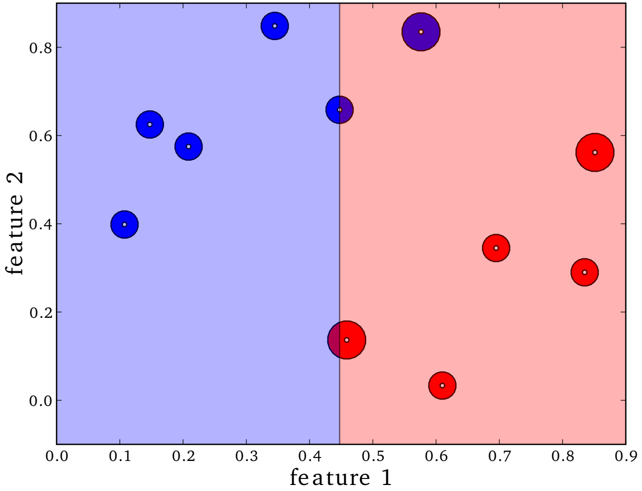
## AdaBoost – Example

Iteration $t = 3$



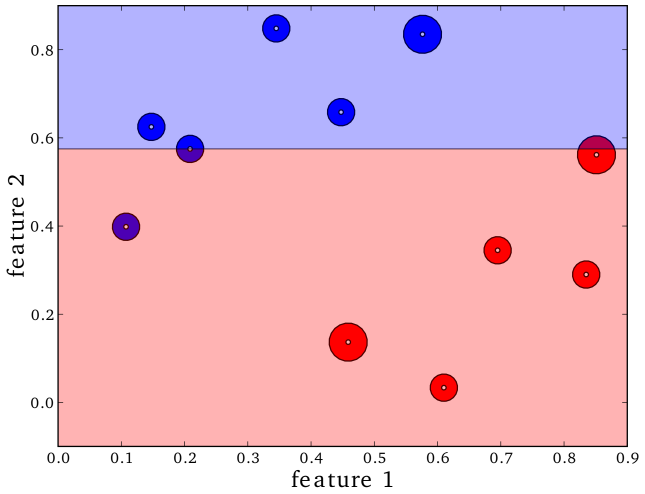
## AdaBoost – Example

Iteration $t = 3$



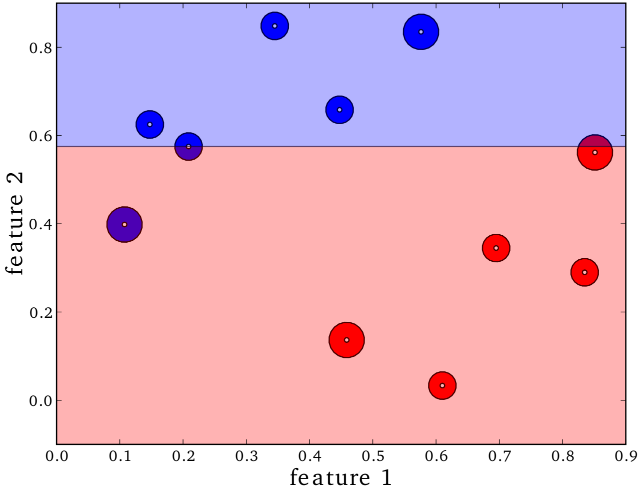
## AdaBoost – Example

Iteration $t = 4$



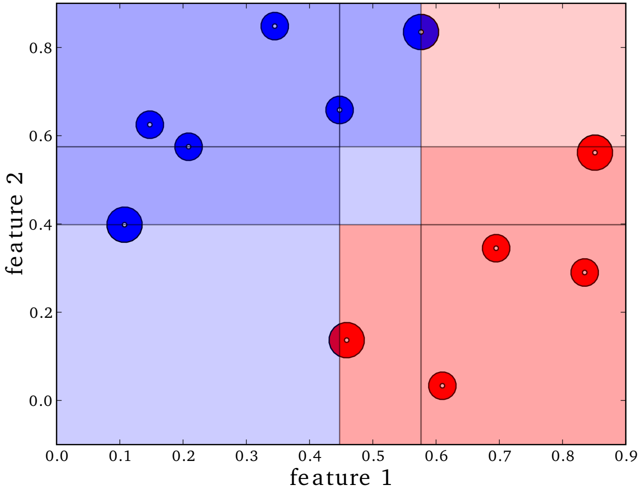
## AdaBoost – Example

Iteration $t = 5$



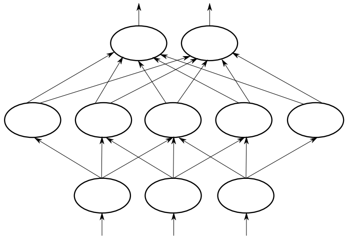
## AdaBoost – Example

Final classifier: $f(x) = \mathrm{sign}\left(\ 1.15h_1(x) + 1.5h_2(x) + \cdots + 0.9h_5(x)\ \right)$



## Artificial Neural Network [since the 1950s]

**Artificial Neural Networks** are predictive models inspired by (early) Neuroscience.



**Main idea:**

- stack layers of simple elements ("neurons")
- one layer's outputs are next layer's input.

**Network parametrizes a function:**

- each neuron $N_i$ computes a linear/affine function

$$a_i = \langle w_i, x_{\text{input}} \rangle + b_i \qquad \text{"activation"}$$

followed by a componentwise non-linear transformation, $\sigma : \mathbb{R} \to \mathbb{R}$,

$$o_i = \sigma(a_i) \qquad \text{e.g.} \quad \sigma(t) = \mathbf{max}\{0, t\}$$

Since approx. 2012, re-popularized under the name of **Deep Learning** $\to$ lectures 7 and 8.

## Understanding Machine Learning Methods

## Decision Theory (for Supervised Learning Problems)

Goal:

- Understand existing algorithms
- Develop new algorithms with specific (optimal?) properties

For this, we'll rely on mathematics. Forget about implementation, finite data etc... (for now)

### Notation

We treat all quantities of interest as random variables:

- input: random variable, $X$, taking values $x \in \mathcal{X}$
  (we think of $\mathcal{X}$ as continuous, but use discrete notation for simplicity)
- output: random variable, $Y$, taking values $y \in \mathcal{Y}$.
- joint probability distribution $p(X=x, Y=y) \quad = p(Y=y|X=x)p(X=x)$
  - $p(X=x)$: how likely is it that any $x \in \mathcal{X}$ will occur?
  - $p(Y=y|X=x)$: what's the probability that $y \in \mathcal{Y}$ is the correct answer for $x \in \mathcal{X}$?
- we write $p(x,y)$ for of $p(X=x, Y=y)$, $\quad p(y|x)$ instead of $p(Y=y|X=x)$, etc.

## Classification

First first look at classification, $\mathcal{Y} = \{1, \dots, M\}$, or $\mathcal{Y} = \{-1, +1\}$.

$\boxed{\textbf{Question: What's the best classifier for a fully known problem?}}$

## Classification

First first look at classification, $\mathcal{Y} = \{1, \dots, M\}$, or $\mathcal{Y} = \{-1, +1\}$.

**Question: What's the best classifier for a fully known problem?**

**Definition (Generalization error)**

Let $c : \mathcal{X} \to \mathcal{Y}$ be a decision rule. The *generalization error*, $\mathcal{R}$, of $c$ is the probability of $c$ making a wrong prediction, i.e.

$$\mathcal{R}(c) := \Pr_{(x,y) \sim p(x,y)} \{c(x) \neq y\}.$$

## Classification

First first look at classification, $\mathcal{Y} = \{1, \dots, M\}$, or $\mathcal{Y} = \{-1, +1\}$.

**Question: What's the best classifier for a fully known problem?**

**Definition (Generalization error)**

Let $c : \mathcal{X} \to \mathcal{Y}$ be a decision rule. The *generalization error*, $\mathcal{R}$, of $c$ is the probability of $c$ making a wrong prediction, i.e.

$$\mathcal{R}(c) := \Pr_{(x,y)\sim p(x,y)} \{c(x) \neq y\}.$$

**Definition (Bayes Classifier, Bayes Risk)**

The smallest achievable generalization error,

$$\mathcal{R}_{\text{Bayes}} = \min_{c:\mathcal{X}\to\mathcal{Y}} \mathcal{R}(c)$$

is called **Bayes** error. A classifier, $c^*$, that achieves the base error is called **Bayes classifier**.

**Lemma**

*For any $x \in \mathcal{X}$ with $p(x) > 0$, a Bayes classifier has the decision rule*

$$\hat{c}(x) \in \operatorname*{\mathbf{argmax}}_{y \in \mathcal{Y}} p(y|x) \qquad \textit{or (equivalently)} \qquad \hat{c}(x) \in \operatorname*{\mathbf{argmax}}_{y \in \mathcal{Y}} p(x,y) \qquad (*)$$

Proof. First: both rules are equivalent, because

$$\mathbf{argmax}_y\, p(x,y) = \mathbf{argmax}_y\, p(y|x)p(x) = \mathbf{argmax}_y\, p(y|x).$$

- 1) We rewrite the risk in terms of per-point contributions. For any $c : \mathcal{X} \to \mathcal{Y}$

$$\begin{aligned}
\mathcal{R}(c) &= \Pr_{(x,y)\sim p(x,y)} \{c(x) \neq y\} \\
&= \mathop{\mathbb{E}}_{(x,y)\sim p} [\![ c(x) \neq y ]\!] \\
&= \mathop{\mathbb{E}}_{x\sim p(x)} \mathop{\mathbb{E}}_{y\sim p(y|x)} [\![ c(x) \neq y ]\!] \\
&= \sum_{x \in \mathcal{X}} p(x) \underbrace{\mathop{\mathbb{E}}_{y\sim p(y|x)} [\![ c(x) \neq y ]\!]}_{=:\mathcal{R}_x(c)}
\end{aligned}$$

**Lemma**

*For any $x \in \mathcal{X}$ with $p(x) > 0$, a Bayes classifier has the decision rule*

$$\hat{c}(x) \in \operatorname*{\mathbf{argmax}}_{y \in \mathcal{Y}} p(y|x) \qquad \textit{or (equivalently)} \qquad \hat{c}(x) \in \operatorname*{\mathbf{argmax}}_{y \in \mathcal{Y}} p(x,y) \qquad (*)$$

- 2) For any $x \in \mathcal{X}$:

$$\begin{aligned}
\mathcal{R}_x(\hat{c}) = \mathop{\mathbb{E}}_{y \sim p(y|x)} [\![\hat{c}(x) \neq y]\!] &= \mathop{\mathbb{E}}_{y \sim p(y|x)} [\![\hat{c}(x) \neq y]\!] \\
&= 1 - \mathop{\mathbb{E}}_{y \sim p(y|x)} [\![\hat{c}(x) = y]\!] \\
&= 1 - \underbrace{p(Y = \hat{c}(x)|x)}_{\geq p(y|x) \text{for all } y \in \mathcal{Y}} \\
&\leq 1 - p(Y = c^*(x)|x) \\
&= \mathop{\mathbb{E}}_{y \sim p(y|x)} [\![c^*(x) \neq y]\!] = \mathcal{R}_x(c^*)
\end{aligned}$$

- 3) Let $A$ be the set of points where $c^*$ does not fulfill $(*)$. Then, for all $x \in A$:

$$p(Y = \hat{c}(x)|x) > p(Y = c^*(x)|x) \qquad \text{and} \qquad \mathcal{R}_x(\hat{c}) < \mathcal{R}_x(c^*)$$

**Lemma**

*For any $x \in \mathcal{X}$ with $p(x) > 0$, a Bayes classifier has the decision rule*

$$\hat{c}(x) \in \operatorname*{\mathbf{argmax}}_{y \in \mathcal{Y}} p(y|x) \qquad \textit{or (equivalently)} \qquad \hat{c}(x) \in \operatorname*{\mathbf{argmax}}_{y \in \mathcal{Y}} p(x,y) \qquad (*)$$

- 1) $\mathcal{R}(c) = \sum_{x \in \mathcal{X}} p(x)\mathcal{R}_x(c)$
- 2) for every $x \in \mathcal{X}$: $\mathcal{R}_x(\hat{c}) \leq \mathcal{R}_x(c^*)$
- 3) for every $x \in A$: $\mathcal{R}_x(\hat{c}) < \mathcal{R}_x(c^*) = \mathcal{R}_{\text{Bayes}}$
- 4) Consequently, if there's at least one point in $x \in A$ with $p(x) > 0$, then

  $$\mathcal{R}(\hat{c}) < \mathcal{R}(c^*)$$

  But that inequality is impossible by the definition of $\mathcal{R}_{\text{Bayes}}$. Therefore, no such point exists.

  In summary: in all points $x$ with $p(x) > 0$, the Bayes classifier fulfills $(*)$.

In binary classification we can write $c^*$ in closed form:

**Lemma**

*For $\mathcal{Y} = \{-1, +1\}$, the Bayes classifier is given by*

$$c^*(x) = \operatorname{sign}\Big[\log \frac{p(x,+1)}{p(x,-1)}\Big],$$

*as well as*

$$c^*(x) = \operatorname{sign}\Big[\log \frac{p(+1|x)}{p(-1|x)}\Big].$$

Proof: Exercise.

## Should we use $c^*$ to decide for every problem?

- $c^*$ is optimal when trying to *minimize the number of wrong decision*.
- That's often a good strategy, but not always.

**Reminder**

To evaluate a learning task, we use a *loss function* $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$.
$\ell(y, \bar{y})$ is the loss incurred when predicting $\bar{y}$ if the correct answer is $y$.

## Should we use $c^*$ to decide for every problem?

- $c^*$ is optimal when trying to *minimize the number of wrong decision*.
- That's often a good strategy, but not always.

**Reminder**

To evaluate a learning task, we use a *loss function* $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$.
$\ell(y, \bar{y})$ is the loss incurred when predicting $\bar{y}$ if the correct answer is $y$.

**Example: Doctor's dilemma**

*A patient has a cough but no fever. Should you make her a COVID19 suspect?*

$x$: symptoms. $y \in \{\texttt{yes}, \texttt{no}\}$: COVID19

- $\ell(\texttt{yes}, \texttt{yes}) = 0$ (you did your job well)
- $\ell(\texttt{no}, \texttt{no}) = 0$ (you did your job well)
- $\ell(\texttt{yes}, \texttt{no}) = 50$ (the patent goes home and might infect many others)
- $\ell(\texttt{no}, \texttt{yes}) = 1$ (the patient has to take an unpleasant unnecessary test)

Common: one outcome is rare, but has bad consequences if mispredicted

Instead of minimizing the error probability, minimize the *expected loss*!

## Definition

The classifier of minimal expected $\ell$-risk is given by

$$c^*_\ell(x) := \mathbf{argmin}_{y\in\mathcal{Y}} \mathop{\mathbb{E}}_{\bar{y}\sim p(\bar{y}|x)} \ell(\,\bar{y}, y\,).$$

## Lemma

*For $\mathcal{Y} = \{-1, +1\}$, and $\ell(y, \bar{y})$ given by the table*

| $y \setminus \bar{y}$ | $-1$ | $+1$ |
|---|---|---|
| $-1$ | $a$ | $b$ |
| $+1$ | $c$ | $d$ |

,

*the risk w.r.t. $\ell$ is minimized by the decision rule*

$$c^*_\ell(x) = \operatorname{sign}\Big[\quad \log\frac{p(x,+1)}{p(x,-1)} + \log\frac{c-d}{b-a} \quad\Big],$$

*or equivalently* $$c^*_\ell(x) = \operatorname{sign}\Big[\quad \log\frac{p(+1|x)}{p(-1|x)} + \log\frac{c-d}{b-a} \quad\Big].$$

Proof: Exercise...

**Observation**

The *generalization error* is the *risk* for 0/1-loss, i.e. $\ell(y, y') = [\![ y \neq y' ]\!]$.

Question: What's the best classifier for a fully known problem?

**Question answered. We have identified the optimal classifiers!**

## Learning from Data

In the real world, $p(x,y)$ is unknown, but we have a training set $\mathcal{D}$. What to do?

**Definition**

Given a training set $\mathcal{D}$, we call it

- a **generative probabilistic approach**:
  if we use $\mathcal{D}$ to build a model $\hat{p}(x,y)$ of $p(x,y)$, and then define

$$c(x) := \operatorname*{\mathbf{argmax}}_{y\in\mathcal{Y}} \hat{p}(x,y) \quad \text{or} \quad c_\ell(x) := \operatorname*{\mathbf{argmin}}_{y\in\mathcal{Y}} \mathop{\mathbb{E}}_{\bar{y}\sim\hat{p}(x,\bar{y})} \ell(\,\bar{y},y\,).$$

- a **discriminative probabilistic approach**:
  if we use $\mathcal{D}$ to build a model $\hat{p}(y|x)$ of $p(y|x)$ and define

$$c(x) := \operatorname*{\mathbf{argmax}}_{y\in\mathcal{Y}} \hat{p}(y|x) \quad \text{or} \quad c_\ell(x) := \operatorname*{\mathbf{argmin}}_{y\in\mathcal{Y}} \mathop{\mathbb{E}}_{\bar{y}\sim\hat{p}(\bar{y}|x)} \ell(\,\bar{y},y\,).$$

- a **decision theoretic approach**: if we use $\mathcal{D}$ to directly seach for a classifier $c$.

## Generative Probabilistic Models

### Setting

We are given

- a **training set** of examples $\mathcal{D} = \{(x^1, y^1), \dots, (x^n, y^n)\}$,
  (note: technically rather a multi-set, elements can occur more than once)

Assumption:

- $\mathcal{D}$ are *independent and identically distributed (i.i.d.)* samples from the unknown probability distribution $p(x, y)$.

Shorthand notation,

- $\mathcal{D}^X := \{x^1, \dots, x^n\}$, input part of $\mathcal{D}$ ,
- $\mathcal{D}^Y := \{y^1, \dots, y^n\}$, output part of $\mathcal{D}$,
- $\mathcal{D}_y := \{(x^i, y^i) \in \mathcal{D} : y^i = y\}$, all examples of label $y$.

## Generative Probabilistic Models

**Let's use $\mathcal{D}$ to form an estimate of $p(x, y)$.**

**Definition**

There's (at least) three approaches:

- **parametric estimate**:
  - fix a model class $\hat{p}(x, y; \theta)$,
  - estimate parameters $\hat{\theta}$ such that $\hat{p}(x, y; \hat{\theta}) \approx p(x, y)$.
  - the size of $\theta$ is independent of how large $\mathcal{D}$ is
- **non-parametric estimate**:
  - estimate any $\hat{p}(x, y) \approx p(x, y)$
  - the number of parameters/complexity of $\hat{p}(x, y)$ can grow with $|\mathcal{D}|$
- **hybrids of the two**

## Generative Probabilistic Models: Multinomial

If $\mathcal{X}$ and $\mathcal{Y}$ are *finite*, we can represent any $p(x, y)$ as a table of values.

To simplify notation, we look at generic $z \in \mathcal{Z}$ (think: $z = (x, y)$):

**Definition (Empirical estimate)**

Let $z^1, \dots, z^n$ be samples from $p(z)$, then we call

$$\hat{p}_n(z) := \frac{1}{n} \sum_{i=1}^{n} [\![ z^i = z ]\!]$$

the empirical estimate of $p(z)$ from $n$ samples.

Example: flipping a coin, $\mathcal{Z} = \{\texttt{heads}, \texttt{tails}\}$

- observed outcomes ($n = 6$): `heads`, `heads`, `heads`, `tails`, `heads`, `tails`
- $\hat{p}_6(\texttt{heads}) = \frac{1}{6}(1 + 1 + 1 + 0 + 1 + 0) = \frac{2}{3}$
- $\hat{p}_6(\texttt{tails}) = \frac{1}{6}(0 + 0 + 0 + 1 + 0 + 1) = \frac{1}{3}$

## Generative Probabilistic Models: Multinomial

**Theorem (Convergence of the empirical estimate)**

*Let* $z^1, z^2, \dots$ *be i.i.d. samples from* $p(z)$. *For every possible value* $z \in \mathcal{Z}$

$$\Pr\left\{ \lim_{n\to\infty} \hat{p}_n(z) \;=\; p(z) \right\} = 1.$$

**Proof.**

Every textbook on statistics: *law of large numbers* (strong version). □

## The curse of dimensionality

**Setting:**
Let $\mathcal{Z} = \mathcal{Z}_1 \times \cdots \times \mathcal{Z}_d$, i.e. data decomposes into $d$ non-trivial "features", "attributes", or "dimensions". Let $m_j := |\mathcal{Z}_j| \geq 2$ for $j = 1, \ldots, d$.

**Lemma**

*The number of samples needed to estimate $\hat{p}(z)$* ***grows exponentially in*** $d$ *(unless we made additional assumptions).*

**Proof.**

$\hat{p}(z)$ has $|\mathcal{Z}| = \prod_{j=1}^{d} m_j \geq 2^d$ entries. Without further assumptions, each entry can be set arbitrarily, independently, except for the one constraint that they must sum to $1$. Each sample influences only one bin, so we need at least $2^d - 1$ samples (in practice, many times that, of course). □

## Example (Dating agency table)

| **TRAINING** | eyes | height | handsome | sex | soccer | date? |
|---|---|---|---|---|---|---|
| Apu | blue | tall | yes | male | no | yes |
| Bernice | brown | short | yes | female | no | no |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| Itchy | brown | short | no | male | yes | yes |

Can we estimate $p(x, y)$ here?

- $|\mathcal{X} \times \mathcal{Y}| = (3 \times 2 \times 2 \times 2 \times 2) \times 2 = 96, \quad \rightarrow \quad p(x, y)$ has $95$ free parameters
- We have $9$ samples.
- **Most possible combinations we have never seen!**

## Example (Dating agency table)

| **TRAINING** | `eyes` | `height` | `handsome` | `sex` | `soccer` | `date?` |
|---|---|---|---|---|---|---|
| Apu | blue | tall | yes | male | no | yes |
| Bernice | brown | short | yes | female | no | no |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| Itchy | brown | short | no | male | yes | yes |

Can we estimate $p(x, y)$ here?

- $|\mathcal{X} \times \mathcal{Y}| = (3 \times 2 \times 2 \times 2 \times 2) \times 2 = 96, \quad \rightarrow \quad p(x, y)$ has $95$ free parameters
- We have $9$ samples.
- **Most possible combinations we have never seen!**

Bayes classifier from $\hat{p}(x, y)$: $\quad c(x) := \mathbf{argmax}_{y \in \mathcal{Y}} \hat{p}(x, y)$

- $\hat{p}(\text{Apu}, \texttt{yes}) = \frac{1}{9}, \quad \hat{p}(\text{Apu}, \texttt{no}) = 0, \quad \rightarrow \quad c(\text{Apu}) = \texttt{yes},$
- $\hat{p}(\text{Jimbo}, \texttt{yes}) = 0, \quad \hat{p}(\text{Jimbo}, \texttt{no}) = 0, \quad \rightarrow \quad c(\text{Jimbo}) = ???,$

No clue about previously unseen patterns $\rightarrow$ very little generalization ability

## Naive Bayes Model

**Definition**

Let $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_d$. The *Naive Bayes (NB)* estimate of $p(x, y)$ is

$$\hat{p}_{\mathsf{NB}}(x, y) := \hat{p}(y) \prod_{j=1}^{d} \hat{p}_j(x_j|y),$$

where

- $\hat{p}(y)$ is an estimate of $p(y)$,
- $\hat{p}_j(x_j|y)$ are estimates of $p(x_j|y)$ for every $j = 1, \ldots, d$.

## Naive Bayes Model

**Definition**

Let $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_d$. The *Naive Bayes (NB)* estimate of $p(x, y)$ is

$$\hat{p}_{\mathsf{NB}}(x, y) := \hat{p}(y) \prod_{j=1}^{d} \hat{p}_j(x_j | y),$$

where

- $\hat{p}(y)$ is an estimate of $p(y)$,
- $\hat{p}_j(x_j|y)$ are estimates of $p(x_j|y)$ for every $j = 1, \ldots, d$.

**Lemma**

*The number of free parameters in $p_{\mathsf{NB}}(x, y)$ grows* ***linearly*** *with $d$ (instead of exponentially).*

**Proof.**

$p_{\mathsf{NB}}(x, y)$ has $|\mathcal{Y}|[1 + \sum_{j=1}^{d}(m_j - 1)] - 1$ degrees of freedom. □

## Naive Bayes Classifier

**Definition**

The *Naive Bayes* classifier is given by

$$c(x) := \mathop{\mathbf{argmax}}_{y \in \mathcal{Y}} \hat{p}_{NB}(x, y)$$

A Naive Bayes classifier needs much fewer examples for 'training' than one based on a full probability table.

## Naive Bayes Classifier

**Definition**

The *Naive Bayes* classifier is given by

$$c(x) := \mathop{\mathbf{argmax}}_{y\in\mathcal{Y}} \hat{p}_{NB}(x,y)$$

A Naive Bayes classifier needs much fewer examples for 'training' than one based on a full probability table.

**Remark**

Even for $n \to \infty$, we likely won't have $\hat{p}_{\text{NB}}(x,y) \not\to p(x,y)$!

So, most likely, **the NB model is wrong** as a density estimate.
But that doesn't mean it doesn't work for making decisions!
In fact, NB is *very successful*, e.g. in Spam filtering (text classification).

## Naive Bayes Classifier

**Definition**

The *Naive Bayes* classifier is given by

$$c(x) := \operatorname*{\mathbf{argmax}}_{y \in \mathcal{Y}} \hat{p}_{NB}(x, y)$$

A Naive Bayes classifier needs much fewer examples for 'training' than one based on a full probability table.

**Remark**

Even for $n \to \infty$, we likely won't have $\hat{p}_{\text{NB}}(x, y) \not\to p(x, y)$!

So, most likely, **the NB model is wrong** as a density estimate.
But that doesn't mean it doesn't work for making decisions!
In fact, NB is *very successful*, e.g. in Spam filtering (text classification).

*"All models are wrong, but some are useful." (George E. P. Box, 1979)*

## Parametric models for finite domains

Both models we saw so far are *parametric*:

For finite $z \in \mathcal{Z}$, $p(z)$ is *multinomial* distribution:

- $|\mathcal{Z}|$ parameters: $\theta_z$ for $z \in \mathcal{Z}$ with $p(Z = z) = \theta_z$
- parameters fulfill
  - $\theta_z \geq 0$
  - $\sum_z \theta_z = 1$

Similar for Naive Bayes model:

- $\hat{p}(y)$ is multinomial for $y \in \mathcal{Y}$, parameter $\theta_y \in \mathbb{R}^{|\mathcal{Y}|}$,
  - $\hat{p}(y) = \theta_y$ with $\theta_y \geq 0$, $\sum_{y \in \mathcal{Y}} \theta_y = 1$,
- $\hat{p}(x_j|y)$ is multinomial for $x_j \in \mathcal{X}_j$, parameters $\theta^j_{x_j}$
  - $\hat{p}(x_j|y) = \theta^y_{x_j}$ with $\theta^y_{x_j} \geq 0$, $\sum_{x_j \in \mathcal{X}_j} \theta^y_{x_j} = 1$, for all $y \in \mathcal{Y}$

We set parameters as $\theta_z = \frac{1}{n} \sum_{i=1}^{n} [\![ z^i = z ]\!]$? Why?

Let $\hat{p}(z;\theta)$ be a parametric model with parameter $\theta \in \Theta$.
Let $\mathcal{D} = \{z^1, \dots, z^n\}$ be i.i.d. samples from $p(z)$.

## Definition (Parameter estimation)

There's (at least) two main approaches to set $\theta$:

**Maximum Likelihood (ML) Estimation:**
Which parameter value makes it most likely that we observed $\mathcal{D}$?

$$\theta_{\text{ML}} = \operatorname*{\mathbf{argmax}}_{\theta\in\Theta} \; \hat{p}(z^1, \dots, z^n; \theta) \; = \; \operatorname*{\mathbf{argmax}}_{\theta\in\Theta} \; \prod_i \hat{p}(z^i;\theta)$$

**Maximum-A-Posteriori (MAP) Parameter Estimation:**
Treat $\theta$ as a random variable itself. What's its most likely value given $\mathcal{D}$?

$$\begin{aligned}\theta_{\text{MAP}} &= \operatorname*{\mathbf{argmax}}_{\theta\in\Theta} \; p(\theta \mid z^1, \dots, z^n) \\ &= \operatorname*{\mathbf{argmax}}_{\theta\in\Theta} \; p(\theta)p(z^1, \dots, z^n|\,\theta) \; = \; \operatorname*{\mathbf{argmax}}_{\theta\in\Theta} \; p(\theta)\prod_i \hat{p}(z^i;\,\theta)\end{aligned}$$

where $p(\theta)$ is a *prior* distribution over the possible parameter values.

## Parameter Estimation in Practice

**Remark**

In practice, one almost always uses the log-likelihood, which gives the same $\theta$ (because $\log$ is a monotonous function):

$$\theta_{\mathsf{ML}} = \mathop{\mathbf{argmax}}_{\theta\in\Theta} \log \prod_{i=1}^{n} \hat{p}(x^i;\theta) = \mathop{\mathbf{argmax}}_{\theta\in\Theta} \sum_{i=1}^{n} \log \hat{p}(x^i;\theta)$$

and

$$\begin{aligned} \theta_{\mathsf{MAP}} &= \mathop{\mathbf{argmax}}_{\theta\in\Theta} \quad \log \left[\hat{p}(\theta) \prod_{i} \hat{p}(z^i;\,\theta)\right] \\ &= \mathop{\mathbf{argmax}}_{\theta\in\Theta} \quad \log \hat{p}(\theta) + \sum_{i} \log \hat{p}(z^i;\,\theta) \end{aligned}$$

Example in exercises: $z \in \{0,1\}$, $\hat{p}(z=1;\theta)=\theta$, $\hat{p}(z=0;\theta)=1-\theta$.