### Overview (tentative)

<table>
<thead>
<tr>
<th>Date</th>
<th>Day</th>
<th>no.</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oct 08</td>
<td>Mon</td>
<td>1</td>
<td>A Hands-On Introduction</td>
</tr>
<tr>
<td>Oct 10</td>
<td>Wed</td>
<td>–</td>
<td>self-study (Christoph traveling)</td>
</tr>
<tr>
<td>Oct 15</td>
<td>Mon</td>
<td>2</td>
<td>Bayesian Decision Theory</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Generative Probabilistic Models</td>
</tr>
<tr>
<td>Oct 17</td>
<td>Wed</td>
<td>3</td>
<td>Discriminative Probabilistic Models</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Maximum Margin Classifiers</td>
</tr>
<tr>
<td>Oct 22</td>
<td>Mon</td>
<td>4</td>
<td>Generalized Linear Classifiers, Optimization</td>
</tr>
<tr>
<td>Oct 24</td>
<td>Wed</td>
<td>5</td>
<td>Evaluating Predictors; Model Selection</td>
</tr>
<tr>
<td>Oct 29</td>
<td>Mon</td>
<td>–</td>
<td>self-study (Christoph traveling)</td>
</tr>
<tr>
<td>Oct 31</td>
<td>Wed</td>
<td>6</td>
<td>Overfitting/Underfitting, Regularization</td>
</tr>
<tr>
<td>Nov 05</td>
<td>Mon</td>
<td>7</td>
<td>Learning Theory I: classical/Rademacher bounds</td>
</tr>
<tr>
<td>Nov 07</td>
<td>Wed</td>
<td>8</td>
<td>Learning Theory II: miscellaneous</td>
</tr>
<tr>
<td>Nov 12</td>
<td>Mon</td>
<td>9</td>
<td>Probabilistic Graphical Models I</td>
</tr>
<tr>
<td>Nov 14</td>
<td>Wed</td>
<td>10</td>
<td>Probabilistic Graphical Models II</td>
</tr>
<tr>
<td>Nov 19</td>
<td>Mon</td>
<td>11</td>
<td>Probabilistic Graphical Models III</td>
</tr>
<tr>
<td>Nov 21</td>
<td>Wed</td>
<td>12</td>
<td>Probabilistic Graphical Models IV</td>
</tr>
<tr>
<td>until Nov 25</td>
<td></td>
<td></td>
<td>final project</td>
</tr>
</tbody>
</table>
Learning from Data

In the real world, $p(x, y)$ is unknown, but we have a training set $D$. There’s at least 3 approaches:

**Definition**

Given a training set $D$, we call it

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

  $$c(x) := \arg\max_{y \in Y} \hat{p}(x, y) \quad \text{or} \quad c_\ell(x) := \arg\min_{y \in Y} \mathbb{E}_{\bar{y} \sim \hat{p}(x, \bar{y})} \ell(\bar{y}, y).$$

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

  $$c(x) := \arg\max_{y \in Y} \hat{p}(y|x) \quad \text{or} \quad c_\ell(x) := \arg\min_{y \in Y} \mathbb{E}_{\bar{y} \sim \hat{p}(\bar{y}|x)} \ell(\bar{y}, y).$$

- a **decision theoretic approach**: if we use $D$ to directly search for a classifier $c$ in a hypothesis class $\mathcal{H}$. 

Observation

Task: spam classification, $\mathcal{X} = \{\text{all possible emails}\}$, $\mathcal{Y} = \{\text{spam, ham}\}$. What’s, e.g., $p(x|\text{ham})$?

For every possible email, a value how likely it is to see that email, including:

- all possible languages,
- all possible topics,
- an arbitrary length,
- all possible spelling mistakes, etc.

This is much more general (and much harder) than just deciding if an email is spam or not!

"When solving a problem, do not solve a more general problem as an intermediate step."

(Vladimir Vapnik, 1998)
Observation

Instead of \( p(x, y) = p(x|y)p(y) \), we can also use \( p(x, y) = p(y|x)p(x) \). Since \( \text{argmax}_y p(x, y) = \text{argmax}_y p(y|x) \), we don’t need to model \( p(x) \), only \( p(y|x) \).

Let’s use \( D \) to estimate \( p(y|x) \).
**Observation**

Instead of $p(x, y) = p(x|y)p(y)$, we can also use $p(x, y) = p(y|x)p(x)$. Since $\arg\max_y p(x, y) = \arg\max_y p(y|x)$, we don’t need to model $p(x)$, only $p(y|x)$.

Let’s use $\mathcal{D}$ to estimate $p(y|x)$.

Visual intuition:

- **Class conditional densities**
  
  $p(x|a)$, $p(x|b)$

  $p(x|a)P(a)$, $p(x|b)P(b)$

  = likelihood $p(x|y)$

- **Joint density**

  likelihood*prior: $p(x|y)p(y)$

- **Class posteriors**

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

Instead of \( p(x, y) = p(x \mid y)p(y) \), we can also use \( p(x, y) = p(y \mid x)p(x) \). Since \( \text{argmax}_y p(x, y) = \text{argmax}_y p(y \mid x) \), we don’t need to model \( p(x) \), only \( p(y \mid x) \).

Let’s use \( \mathcal{D} \) to estimate \( p(y \mid x) \).

Example (Spam Classification)

Is \( p(y \mid x) \) really easier than, e.g., \( p(x \mid y) \)?

- \( p("v1agra" \mid \text{spam}) \) is some positive value (not every spam is viagra)
- \( p(\text{spam} \mid "v1agra") \) is almost surely 1.

For \( p(y \mid x) \) we treat \( x \) as \textit{given}, we don’t need to know its probability.
Nonparametric Discriminative Model

Idea: split $\mathcal{X}$ into regions, for each region store an estimate $\hat{p}(y|x)$.

Note: prediction rule $c(x) = \text{argmax}_y \hat{p}(y|x)$ is predicts the most frequent label in each leaf (same as in first lecture).

| Region | $p(1|x)$ | $p(2|x)$ | $p(3|x)$ |
|--------|----------|----------|----------|
| Region 1 | 0.7      | 0.2      | 0.1      |
| Region 2 | 0.1      | 0.8      | 0.1      |
| Region 3 | 0.01     | 0.98     | 0.01     |
Idea: split $\mathcal{X}$ into regions, for each region store an estimate $\hat{p}(y|x)$.

For example, using a decision tree:

- training: build a tree
- prediction: for new example $x$, find its leaf
- output $\hat{p}(y|x) = \frac{n_y}{n}$, where
  - $n$ is the number of examples in the leaf,
  - $n_y$ is the number of example of label $y$ in the leaf.
Nonparametric Discriminative Model

Idea: split $\mathcal{X}$ into regions, for each region store an estimate $\hat{p}(y|x)$.

For example, using a decision tree:

- training: build a tree
- prediction: for new example $x$, find its leaf
- output $\hat{p}(y|x) = \frac{n_y}{n}$, where
  - $n$ is the number of examples in the leaf,
  - $n_y$ is the number of example of label $y$ in the leaf.

Note: prediction rule

$$c(x) = \arg\max_y \hat{p}(y|x)$$

is predicts the most frequent label in each leaf (same as in first lecture).
**Setting.** We assume $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{Y} = \{-1, +1\}$.

**Definition (Logistic Regression (LogReg) Model)**

Modeling

\[
\hat{p}(y|x; w) = \frac{1}{1 + \exp(-y\langle w, x \rangle)},
\]

with parameter vector $w \in \mathbb{R}^d$ is called a *logistic regression* model.
Setting. We assume $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{Y} = \{-1, +1\}$.

Definition (Logistic Regression (LogReg) Model)

Modeling

$$\hat{p}(y|x; w) = \frac{1}{1 + \exp(-y\langle w, x \rangle)},$$

with parameter vector $w \in \mathbb{R}^d$ is called a logistic regression model.

Lemma

$\hat{p}(y|x; w)$ is a well defined probability density w.r.t. $y$ for any $w \in \mathbb{R}^d$.

Proof. elementary.
How to set the weight vector $w$ (based on $\mathcal{D}$)

**Logistic Regression Training**

Given a training set $\mathcal{D} = \{(x^1, y^1), \ldots, (x^n, y^n)\}$, logistic regression training sets the free parameter vector as

$$w_{LR} = \arg\min_{w \in \mathbb{R}^d} \sum_{i=1}^{n} \log \left(1 + \exp(-y^i \langle w, x^i \rangle)\right)$$

**Lemma (Conditional Likelihood Maximization)**

$w_{LR}$ from Logistic Regression training maximizes the conditional data likelihood w.r.t. the LogReg model,

$$w_{LR} = \arg\max_{w \in \mathbb{R}^d} \hat{p}(y^1, \ldots, y^n | x^1, \ldots, x^n, w)$$
Proof.

Maximizing

\[ \hat{p}(\mathcal{D}^Y | \mathcal{D}^X, w) \overset{i.i.d.}{=} \prod_{i=1}^{n} \hat{p}(y^i | x^i, w) \]

is equivalent to minimizing its negative logarithm

\[ -\log \hat{p}(\mathcal{D}^Y | \mathcal{D}^X, w) = -\log \prod_{i=1}^{n} \hat{p}(y^i | x^i, w) = -\sum_{i=1}^{n} \log \hat{p}(y^i | x^i, w) \]

\[ = -\sum_{i=1}^{n} \log \left( \frac{1}{1 + \exp(-y^i \langle w, x^i \rangle)} \right), \]

\[ = -\sum_{i=1}^{n} [\log 1 - \log(1 + \exp(-y^i \langle w, x^i \rangle))], \]

\[ = \sum_{i=1}^{n} \log(1 + \exp(-y^i \langle w, x^i \rangle)). \]
Definition (Kullback-Leibler (KL) divergence)

Let $p$ and $q$ be two probability distributions (for discrete $\mathcal{Z}$) or probability densities with respect to a measure $d\lambda$ (for continuous $\mathcal{Z}$). The **Kullbach-Leibler (KL)-divergence** between $p$ and $q$ is defined as

$$KL(p \parallel q) = \sum_{z \in \mathcal{Z}} p(z) \log \frac{p(z)}{q(z)}, \quad \text{or} \quad KL(p \parallel q) = \int_{z \in \mathcal{Z}} p(z) \log \frac{p(z)}{q(z)} \, d\lambda(z),$$

(with convention $0 \log 0 = 0$, and $a \log \frac{a}{0} = \infty$ for $a > 0$).
**Definition (Kullback-Leibler (KL) divergence)**

Let $p$ and $q$ be two probability distributions (for discrete $\mathcal{Z}$) or probability densities with respect to a measure $d\lambda$ (for continuous $\mathcal{Z}$). The **Kullbach-Leibler (KL)-divergence** between $p$ and $q$ is defined as

$$KL(p \parallel q) = \sum_{z \in \mathcal{Z}} p(z) \log \frac{p(z)}{q(z)}, \quad \text{or} \quad KL(p \parallel q) = \int_{\mathcal{Z}} p(z) \log \frac{p(z)}{q(z)} \, d\lambda(z),$$

(with convention $0 \log 0 = 0$, and $a \log \frac{a}{0} = \infty$ for $a > 0$).

$KL$ is a similarity measure between probability distributions. It fulfills

$$0 \leq KL(p \parallel q) \leq \infty, \quad \text{and} \quad KL(p \parallel q) = 0 \iff p = q.$$  

However, $KL$ is **not a metric**.

- it is in general not symmetric, $KL(q \parallel p) \neq KL(p \parallel q)$,
- it does not fulfill the triangle inequality.
**Definition (Expected Kullback-Leibler (KL) divergence)**

Let $p(x, y)$ be a probability distribution over $(x, y) \in \mathcal{X} \times \mathcal{Y}$ and let $\hat{p}(y|x)$ be an approximation of $p(y|x)$. We measure the approximation quality by the **expected KL-divergence between** $p$ **and** $q$ **over all** $x \in \mathcal{X}$:

$$
KL_{exp}(p \parallel q) = \mathbb{E}_{x \sim p(x)} \{ KL(p(\cdot|x) \parallel q(\cdot|x)) \}
$$

**Theorem**

*The parameter $w_{LR}$ obtained by logistic regression training approximately minimizes the KL divergence between $\hat{p}(y|x; w)$ and $p(y|x)$.*
Proof.

We show how maximizing the conditional likelihood relates to $\text{KL}_{\text{exp}}$:

$$
\text{KL}_{\text{exp}}(p||\hat{p}) = \mathbb{E}_{x \sim p(x)} \sum_{y \in \mathcal{Y}} p(y|x) \log \frac{p(y|x)}{\hat{p}(y|x, w)}
$$

$$
= \mathbb{E}_{(x,y) \sim p(x,y)} \log p(y|x) - \mathbb{E}_{(x,y) \sim p(x,y)} \log \hat{p}(y|x, w)
$$

We can’t maximize $\mathbb{E}_{(x,y) \sim p(x,y)} \log \hat{p}(y|x, w)$ directly, because $p(x, y)$ is unknown. But we can maximize its empirical estimate based on $\mathcal{D}$:

$$
\mathbb{E}_{(x,y) \sim p(x,y)} \log \hat{p}(y|x, w) \approx \sum_{(x^i, y^i) \in \mathcal{D}} \log \hat{p}(y^i|x^i, w)
$$

log of conditional data likelihood

The approximation will get better the more data we have.

\qed
Theorem

Logistic Regression training,

\[ w_{LR} = \underset{w \in \mathbb{R}^d}{\text{argmin}} \mathcal{L}(w) \quad \text{for} \quad \mathcal{L}(w) = \sum_{i=1}^{n} \log \left( 1 + \exp \left( -y^i \langle w, x^i \rangle \right) \right), \]

is a \( C^\infty \)-smooth, unconstrained, convex optimization problem.

Proof.

1. it’s an optimization problem,
2. it’s unconstrained,
3. it’s smooth (the objective function is \( C^\infty \) differentiable),
4. remains to show: the objective function is a convex function. Since \( \mathcal{L} \) is smooth, it’s enough to show that its Hessian matrix (the matrix of 2nd partial derivatives) is everywhere positive definite.
We compute first the gradient and then the Hessian of

\[
\mathcal{L}(w) = \sum_{i=1}^{n} \log(1 + \exp(-y^i \langle w, x^i \rangle)).
\]

\[
\nabla_w \mathcal{L}(w) = \sum_{i=1}^{n} \nabla \log(1 + \exp(-y^i \langle w, x^i \rangle)).
\]

use the chain rule, \( \nabla f(g(w)) = \frac{df}{dt}(g(w)) \nabla g(w) \), and \( \frac{d \log(t)}{dt} = \frac{1}{t} \)

\[
= \sum_{i=1}^{n} \frac{\nabla[1 + \exp(-y^i \langle w, x^i \rangle)]}{1 + \exp(-y^i \langle w, x^i \rangle)}
\]

\[
= \sum_{i=1}^{n} \frac{\exp(-y^i \langle w, x^i \rangle)}{1 + \exp(-y^i \langle w, x^i \rangle)} \nabla(-y^i \langle w, x^i \rangle)
\]

\[
= \hat{p}(-y^i | x^i, w)
\]

use the chain rule again, \( \frac{d}{dt} \exp(t) = \exp(t) \), and \( \nabla_w \langle w, x^i \rangle = x^i \)

\[
= - \sum_{i=1}^{n} [\hat{p}(-y^i | x^i, w)] y^i x^i
\]
\[ H_w \mathcal{L}(w) = \nabla \nabla^\top \mathcal{L}(w) = - \sum_{i=1}^{n} [\nabla \hat{p}(-y^i|x^i, w)] y^i x^i \]

\[ \nabla \hat{p}(-y^i|x^i, w) = \nabla \frac{1}{1 + \exp(y^i \langle w, x^i \rangle)} = - \frac{\nabla [1 + \exp(y^i \langle w, x^i \rangle)]}{[1 + \exp(y^i \langle w, x^i \rangle)]^2} \]

use quotient rule, \( \nabla \frac{1}{f(w)} = - \frac{\nabla f(w)}{f^2(w)} \), and chain rule,

\[ = - \frac{\exp(y^i \langle w, x^i \rangle)}{[1 + \exp(y^i \langle w, x^i \rangle)]^2} \nabla y^i \langle w, x^i \rangle \]

\[ = -(\hat{p}(-y^i|x^i)) \hat{p}(y^i|x^i, w) y^i x^i \]

insert into above expression for \( H_w \mathcal{L}(w) \)

\[ H = \sum_{i=1}^{n} (\hat{p}(-y^i|x^i) \hat{p}(y^i|x^i, w) x^i x^i\top) \]

A positively weighted linear combination of pos.def. matrices is pos.def.
Example plot: LogReg objective for three examples in $\mathbb{R}^2$
Convex optimization is a well understood field. We can use, e.g., gradient descent will converge to the globally optimal solution!

**Steepest Descent Minimization with Line Search**

input \( \epsilon > 0 \) tolerance (for stopping criterion)

1: \( w \leftarrow 0 \)

2: repeat

3: \( v \leftarrow -\nabla_w \mathcal{L}(w) \) \{descent direction\}

4: \( \eta \leftarrow \arg\min_{\eta > 0} \mathcal{L}(w + \eta v) \) \{1D line search\}

5: \( w \leftarrow w + \eta v \)

6: until \( \|v\| < \epsilon \)

output \( w \in \mathbb{R}^d \) learned weight vector

Faster conference from methods that use second-order information, e.g., conjugate gradients or (L-)BFGS → convex optimization lecture
A discriminative probability model, $\hat{p}(y|x)$, is enough to make decisions:

$$
c(x) = \mathop{\arg\max}_{y \in \mathcal{Y}} \hat{p}(y|x) \quad \text{or} \quad c(x) = \mathop{\arg\min}_{y \in \mathcal{Y}} \mathbb{E}_{\tilde{y} \sim \hat{p}(y|x)} \ell(\tilde{y}, y).
$$

For Logistic Regression, this is particularly simple:

**Lemma**

*The LogReg classification rule for 0/1-loss is*

$$
c(x) = \text{sign} \langle w, x \rangle.
$$

*For a loss function $\ell = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ the rule is*

$$
c_\ell(x) = \text{sign}[\langle w, x \rangle + \log \frac{c - d}{b - a}],
$$

*In particular, the decision boundaries is linear (or affine).*

**Proof.** Elementary, since $\log \frac{\hat{p}(+1|x;w)}{p(-1|x;w)} = \langle w, x \rangle$
For $\mathcal{Y} = \{1, \ldots, M\}$, we can do two things:

- **Parametrize $\hat{p}(y|x; \vec{w})$ using $M - 1$ vectors, $w_1, \ldots, w_{M-1} \in \mathbb{R}^d$, as**

  \[
  \hat{p}(y|x, w) = \frac{\exp(\langle w_y, x \rangle)}{1 + \sum_{j=1}^{M-1} \exp(\langle w_j, x \rangle)} \quad \text{for } y = 1, \ldots, M - 1,
  \]

  \[
  \hat{p}(M|x, w) = \frac{1}{1 + \sum_{j=1}^{M-1} \exp(\langle w_j, x \rangle)}.
  \]

- **Parametrize $\hat{p}(y|x; \vec{w})$ using $M$ vectors, $w_1, \ldots, w_M \in \mathbb{R}^d$, as**

  \[
  \hat{p}(y|x, w) = \frac{\exp(\langle w_y, x \rangle)}{\sum_{j=1}^{M} \exp(\langle w_j, x \rangle)} \quad \text{for } y = 1, \ldots, M,
  \]

Second is more popular, since it’s easier to implement and analyze.

Decision boundaries are still *piecewise linear*, $c(x) = \arg\max_y \langle w_y, x \rangle$. 
Discriminative models treat the input data, $x$, as fixed and only model the distribution of the output labels $p(y|x)$.

Discriminative models, in particular LogReg, are popular, because

- they often need less training data than generative models,
- they provide an estimate of the uncertainty of a decision $p(c(x)|x)$,
- training them is often efficient, e.g. big companies train LogReg models routinely from billions of examples.

But: they also have drawbacks

- often $\hat{p}_{LR}(y|x) \not\rightarrow p(y|x)$, even for $n \rightarrow \infty$,
- they usually are good for prediction, but they do not reflect the actual mechanism.

Note: there are much more complex discriminative models than LogReg, e.g. Conditional Random Fields (maybe later).
Observation

Even easier than estimating $p(y|x)$ (or $p(x,y)$) should be to just estimate the decision boundary between classes.
Let’s use $\mathcal{D}$ to estimate a classifier $c : \mathcal{X} \rightarrow \mathcal{Y}$ directly.
Let’s use $\mathcal{D}$ to estimate a classifier $c : \mathcal{X} \rightarrow \mathcal{Y}$ directly.

For a start, we fix

- $\mathcal{D} = \{(x^1, y^1), \ldots, (x^n, y^n)\}$,
- $\mathcal{Y} = \{+1, -1\}$,
- we look for classifiers with linear decision boundary.

Several of the classifiers we saw had linear decision boundaries:

- Perceptron
- Generative classifiers for Gaussian class-conditional densities with shared covariance matrix
- Logistic Regression

What’s the best linear classifier?
**Definition**

Let

$$\mathcal{F} = \{ f : \mathbb{R}^d \rightarrow \mathbb{R} \text{ with } f(x) = b + a_1 x_1 + \cdots + a_d x_d = b + \langle w, x \rangle \}$$

be the set of linear (affine) function from $\mathbb{R}^d \rightarrow \mathbb{R}$.

A classifier $g : \mathcal{X} \rightarrow \mathcal{Y}$ is called **linear**, if it can be written as

$$g(x) = \text{sign } f(x)$$

for some $f \in \mathcal{F}$.

We write $\mathcal{G}$ for the set of all linear classifiers.
A linear classifier, \( g(x) = \text{sign}(\langle w, x \rangle) \), with \( b = 0 \)
A linear classifier $g(x) = \text{sign}(\langle w, x \rangle + b)$, with $b > 0$
Feature augmentation

The bias term is good for intuition, but annoying in analysis:

Useful trick: feature augmentation

Adding a constant feature allows us to avoid models with explicit bias term:

- instead of $x = (x^1, \ldots, x^d) \in \mathbb{R}^d$, use $\tilde{x} = (x^1, \ldots, x^d, 1) \in \mathbb{R}^{d+1}$
- for any $\tilde{w} \in \mathbb{R}^{d+1}$, think $\tilde{w} = (w, b)$ with $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$

Linear function in $\mathbb{R}^{d+1}$:

$$f(\tilde{x}) = \langle \tilde{w}, \tilde{x} \rangle = \sum_{i=1}^{d+1} \tilde{w}_i \tilde{x}_i = \sum_{i=1}^{d} \tilde{w}_i \tilde{x}_i + \tilde{w}_{d+1} \tilde{x}_{d+1} = \langle w, x \rangle + b$$

Linear classifier with bias in $\mathbb{R}^d \equiv$ linear classifier with no bias in $\mathbb{R}^{d+1}$

Augmenting with other (larger) values than 1 can make sense, see later...
**Linear classifiers**

**Definition (Ad hoc)**

We call a classifier, \( g \), **correct** (for a training set \( D \)), if it predicts the correct labels for all training examples:

\[
g(x^i) = y^i \quad \text{for} \quad i = 1, \ldots, n.
\]

**Example (Perceptron)**

- if the *Perceptron* converges, the result is an **correct** classifier.
- any classifier with zero training error is **correct**.
Linear classifiers

**Definition (Ad hoc)**

We call a classifier, $g$, **correct** (for a training set $\mathcal{D}$), if it predicts the correct labels for all training examples:

$$g(x^i) = y^i \quad \text{for } i = 1, \ldots, n.$$ 

**Example (Perceptron)**

- if the *Perceptron* converges, the result is an **correct** classifier.
- any classifier with zero training error is **correct**.

**Definition (Linear Separability)**

A training set $\mathcal{D}$ is called **linearly separable**, if it allows a correct linear classifier (i.e. the classes can be separated by a hyperplane).
A linearly separable dataset and a correct classifier
A linearly separable dataset and a correct classifier
A linearly separable dataset and a correct classifier
An incorrect classifier
**Definition (Ad hoc)**

The **robustness** of a classifier $g$ (with respect to $\mathcal{D}$) is the largest amount, $\rho$, by which we can perturb the training samples without changing the predictions of $g$.

$$g(x^i + \epsilon) = g(x^i), \quad \text{for all } i = 1, \ldots, n.$$  
for any $\epsilon \in \mathbb{R}^d$ with $\|\epsilon\| < \rho$.

**Example**

- constant classifier, e.g. $c(x) \equiv 1$: very robust ($\rho = \infty$), (but it is not correct, in the sense of the previous definition)
- robustness of the *Perceptron*: can be arbitrarily small (see Exercise...
Robustness, $\rho$, of a linear classifier.
Definition (Margin)

Let \( f(x) = \langle w, x \rangle + b \) define a correct linear classifier. Then the smallest (Euclidean) distance of any training example from the decision hyperplane is called the margin of \( f \) (with respect to \( D \)).

Lemma

We can compute the margin of a linear classifier \( f = \langle w, x \rangle + b \) as

\[
\rho = \min_{i=1,\ldots,n} \left| \frac{w}{\|w\|}, x^i \right| + b.
\]

Proof.

High school maths: distance between a points and a hyperplane in Hessian normal form.
Margin, $\rho$, of a linear classifier
Theorem

The robustness of a linear classifier function \( g(x) = \text{sign} \, f(x) \) with 
\[ f(x) = \langle w, x \rangle \] 
is identical to the margin of \( f \).
Theorem

The robustness of a linear classifier function $g(x) = \text{sign} \ f(x)$ with $f(x) = \langle w, x \rangle$ is identical to the margin of $f$.

Proof by Picture
Proof (blackboard). For any \( i = 1, \ldots, n \) and any \( \epsilon \in \mathbb{R}^d \)

\[
f(x^i + \epsilon) = \langle w, x^i + \epsilon \rangle = \langle w, x^i \rangle + \langle w, \epsilon \rangle = f(x^i) + \langle w, \epsilon \rangle,
\]

so it follows (Cauchy-Schwarz inequality) that

\[
f(x^i) - \|w\|\|\epsilon\| \leq f(x^i + \epsilon) \leq f(x^i) + \|w\|\|\epsilon\|.
\]

Checking the cases \( \epsilon = \pm \|\epsilon\| \frac{\epsilon}{\|w\|} w \), we see that these inequalities are sharp.

To ensure \( g(x^i + \epsilon) = g(x^i) \) for all training samples, \( f(x^i) \) and \( f(x^i + \epsilon) \) have the same sign for all \( \epsilon \), i.e. \( |f(x^i)| \geq \|w\|\|\epsilon\| \) for \( i = 1, \ldots, n \).

This inequality holds for all samples, so in particular it holds for the one of minimal score, and \( \min_i |f(x^i)| = \min_i |\langle w, x^i \rangle| = \rho \).  

\( \square \)
Theorem

Let \( \mathcal{D} \) be a linearly separable training set. Then the most robust, correct linear classifier (without bias term) is given by

\[
g(x) = \text{sign}\langle w^*, x \rangle \text{ where } w^* \text{ are the solution to}
\]

\[
\min_{w \in \mathbb{R}^d} \frac{1}{2} \|w\|^2
\]

subject to

\[
y^i(\langle w, x^i \rangle) \geq 1, \quad \text{for } i = 1, \ldots, n.
\]

Remark

- The classifier defined above is called **Maximum (Hard) Margin Classifier**, or **Hard-Margin Support Vector Machine (SVM)**.
- It is unique (follows from strictly convex optimization problem).
Proof.

1. All $w$ that fulfill the inequalities yield \textit{correct} classifiers.

2. Since $\mathcal{D}$ is linearly separable, there exists some $v$ with

$$\text{sign}\langle v, x^i \rangle = y_i,$$

i.e.

$$y_i\langle v, x^i \rangle \geq \gamma > 0.$$  

for $\gamma = \min_i y_i\langle v, x^i \rangle$. So $\tilde{v} = v/\gamma$, fulfills the inequalities and we see that the constraint set is at least not empty.
Proof.

1. All $w$ that fulfill the inequalities yield correct classifiers.

2. Since $\mathcal{D}$ is linearly separable, there exists some $v$ with

$$\text{sign} \langle v, x^i \rangle = y_i, \quad \text{i.e.} \quad y_i \langle v, x^i \rangle \geq \gamma > 0.$$ 

for $\gamma = \min_i y_i \langle v, x^i \rangle$. So $\tilde{v} = v / \gamma$, fulfills the inequalities and we see that the constraint set is at least not empty.

3. Now we check (with $i = 1, \ldots, n$):

$$\min_{w \in \mathbb{R}^d} \frac{1}{2} \|w\|^2 \quad \text{s.t.} \quad y^i \langle w, x^i \rangle \geq 1$$

$$\Leftrightarrow \max_{w \in \mathbb{R}^d} \frac{1}{\|w\|} \quad \text{s.t.} \quad y^i \langle w, x^i \rangle \geq 1$$

$$\Leftrightarrow \max_{\{w': \|w'\|=1\}, \rho \in \mathbb{R}} \rho \quad \text{s.t.} \quad y^i \langle \frac{w'}{\rho}, x^i \rangle \geq 1$$

$$\Leftrightarrow \max_{\{w': \|w'\|=1\}, \rho \in \mathbb{R}} \rho \quad \text{s.t.} \quad y^i \langle w', x^i \rangle \geq \rho$$

$$\Leftrightarrow \max_{\{w': \|w'\|=1\}, \rho \in \mathbb{R}} \rho \quad \text{s.t.} \quad |\langle w', x^i \rangle| \geq \rho \quad \text{and} \quad \text{sign} \langle w', x^i \rangle = y_i
Proof.

1. All $w$ that fulfill the inequalities yield correct classifiers.

2. Since $D$ is linearly separable, there exists some $v$ with

\[ \text{sign} \langle v, x^i \rangle = y_i, \quad \text{i.e.} \quad y_i \langle v, x^i \rangle \geq \gamma > 0. \]

for $\gamma = \min_i y_i \langle v, x^i \rangle$. So $\tilde{v} = v / \gamma$, fulfills the inequalities and we see that the constraint set is at least not empty.

3. Now we check (with $i = 1, \ldots, n$):

\[ \min_{w \in \mathbb{R}^d} \frac{1}{2} \|w\|^2 \quad \text{sb.t.} \quad y^i \langle w, x^i \rangle \geq 1 \]

\[ \iff \max_{w \in \mathbb{R}^d} \frac{1}{\|w\|} \quad \text{sb.t.} \quad y^i \langle w, x^i \rangle \geq 1 \]

\[ \iff \max_{\{w': \|w'\| = 1\}, \rho \in \mathbb{R}} \rho \quad \text{sb.t.} \quad y^i \langle \frac{w'}{\rho}, x^i \rangle \geq 1 \]

\[ \iff \max_{\{w': \|w'\| = 1\}, \rho \in \mathbb{R}} \rho \quad \text{sb.t.} \quad y^i \langle w', x^i \rangle \geq \rho \]

\[ \iff \max_{\{w': \|w'\| = 1\}, \rho \in \mathbb{R}} \rho \quad \text{sb.t.} \quad |\langle w', x^i \rangle| \geq \rho \quad \text{and} \quad \text{sign} \langle w', x^i \rangle = y_i \]
Observation (Not all training sets are linearly separable.)

- Non-Separable Training Sets
Definition (Maximum Soft-Margin Classifier)

Let $D$ be a training set, not necessarily linearly separable. Let $C > 0$. Then the classifier $g(x) = \text{sign}\langle w^*, x \rangle + b$ where $(w^*, b^*)$ are the solution to

$$\min_{w \in \mathbb{R}^d, \xi \in \mathbb{R}^n} \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{n} \xi_i$$

subject to

$$y^i(\langle w, x^i \rangle + b) \geq 1 - \xi^i, \quad \text{for } i = 1, \ldots, n.$$  
\[\xi^i \geq 0, \quad \text{for } i = 1, \ldots, n.\]

is called Maximum (Soft-)Margin Classifier or Linear Support Vector Machine.
Theorem

The maximum soft-margin classifier exists and is unique for any $C > 0$.

Proof. optimization problem is strictly convex

Remark

The constant $C > 0$ is called regularization parameter.

It balances the wishes for robustness and for correctness

- $C \to 0$: mistakes don’t matter much, emphasis on short $w$
- $C \to \infty$: as few errors as possible, might not be robust

We’ll see more about this in the next lecture.
Remark

Sometimes, a soft margin is better even for linearly separable datasets!

Left: small margin, no errors) Right: large margin, but 1 error