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

- $x \in \mathcal{X}$: inputs, $y \in \mathcal{Y}$: outputs, $\mathcal{D} \subset \mathcal{X} \times \mathcal{Y}$: training set, $\theta \in \Theta$: model parameters, ...
- $p(x, y)$: underlying data distribution, $p(\theta)$: prior knowledge about parameters, ...
Refresher from Previous Lecture

We treat all quantities of interest as random variables.

- $x \in \mathcal{X}$: inputs, $y \in \mathcal{Y}$: outputs, $\mathcal{D} \subset \mathcal{X} \times \mathcal{Y}$: training set, $\theta \in \Theta$: model parameters, ...
- $p(x, y)$: underlying data distribution, $p(\theta)$: prior knowledge about parameters, ...

If $p(x, y)$ is known, the optimal classifier is easy:

- $c^*(x) = \arg \max_{y \in \mathcal{Y}} p(x, y) = \arg \max_{y \in \mathcal{Y}} p(y | x)$
- $c^*_\ell(x) = \arg \max_{\bar{y} \in \mathcal{Y}} \mathbb{E}(y | x) \ell(y, \bar{y})$
Refresher from Previous Lecture

We treat all quantities of interest as random variables.

- $x \in \mathcal{X}$: inputs, $y \in \mathcal{Y}$: outputs, $\mathcal{D} \subset \mathcal{X} \times \mathcal{Y}$: training set, $\theta \in \Theta$: model parameters, . . .

- $p(x, y)$: underlying data distribution, $p(\theta)$: prior knowledge about parameters, . . .

If $p(x, y)$ is known, the optimal classifier is easy:

- $c^*(x) = \arg\max_{y \in \mathcal{Y}} p(x, y) = \arg\max_{y \in \mathcal{Y}} p(y|x)$
- $c^*_\ell(x) = \arg\max_{\bar{y} \in \mathcal{Y}} \mathbb{E}(y|x) \ell(y, \bar{y})$

If $p(x, y)$ is unknown, we can use generative probabilistic modeling:

- estimate $\hat{p}(x, y)$ from a dataset $\mathcal{D}$, then use as plug-in for true $p(x, y)$
We treat all quantities of interest as random variables.

- \( x \in \mathcal{X} \): inputs, \( y \in \mathcal{Y} \): outputs, \( \mathcal{D} \subset \mathcal{X} \times \mathcal{Y} \): training set, \( \theta \in \Theta \): model parameters, \ldots
- \( p(x, y) \): underlying data distribution, \( p(\theta) \): prior knowledge about parameters, \ldots

If \( p(x, y) \) is known, the optimal classifier is easy:

- \( c^*(x) = \arg\max_{y \in \mathcal{Y}} p(x, y) = \arg\max_{y \in \mathcal{Y}} p(y|x) \)
- \( c^*_{\ell}(x) = \arg\max_{\bar{y} \in \mathcal{Y}} \mathbb{E}(y|x) \ell(y, \bar{y}) \)

If \( p(x, y) \) is unknown, we can use generative probabilistic modeling:

- estimate \( \hat{p}(x, y) \) from a dataset \( \mathcal{D} \), then use as plug-in for true \( p(x, y) \)

For parametric models \( p(x, y; \theta) \), find parameters \( \theta \) by

- maximum likelihood method: \( \theta = \arg\max_{\theta} p(\mathcal{D}; \theta) \)
- maximum-a-posteriori method: \( \theta = \arg\max_{\theta} p(\theta|\mathcal{D}) \)
Definition (Laplace smoothing)

Let \( z^1, \ldots, z^n \) be i.i.d. samples from \( p(z) \). For \( \alpha \geq 0 \) we call

\[
\hat{p}_n(z) := \frac{1}{n + |Z| \alpha} (\alpha + \sum_{i=1}^{n} [z^i = z])
\]

the \textit{smoothed empirical estimate} of \( p(z) \) (with smoothing parameter \( \alpha \)).

Bayesian interpretation:

- Maximum-a-posteriori estimate of parameters \( \theta_z \) of a multinomial distribution
- Prior on \( \theta \): symmetric Dirichlet distribution with parameter \( \alpha \)

\[
p(\theta) = \frac{1}{B(\alpha)} \prod_{z=1}^{|Z|} (\theta_z)^{\alpha - 1} \quad \text{with} \quad B(\alpha) = \frac{\Gamma(\alpha)^{|Z|}}{\Gamma(\alpha |Z|)}
\]

Laplace’s "rule of succession": \( \alpha = 1 \). More common: \( \alpha < 1 \), e.g. \( \alpha = \frac{1}{2} \) or \( \alpha = \frac{1}{|Z|} \).
If $\mathcal{X}$ is continuous, $p(x, y)$ is a strange object, mixing continuous and discrete. Instead of modeling $p(x, y)$, we decompose it:

**Definition**

Let $p(x, y) = p(x|y)p(y)$.

- $p(y)$ are called **class priors**,
- $p(x|y)$, for $y \in \mathcal{Y}$, are called **class conditional densities**.

**Remark**

$p(y)$ is a discrete probability distribution over $|\mathcal{Y}|$ possible values, i.e.

- $p(y) \geq 0$ for all $y \in \mathcal{Y}$, and $\sum_y p(y) = 1$.

For any fixed $y \in \mathcal{Y}$, $p(x|y)$ is a probability density, i.e.

- $p(x|y) \geq 0$ for all $x \in \mathcal{X}$, and $\int_x p(x|y) \, dx = 1$. 
Most popular parametric model for continuous data is **Gaussian**:

**Definition (Gaussian Density Parameter Estimation)**

For \( x \in \mathbb{R}^d \), let \( \hat{p}(x|y; \mu, \Sigma) = G(x, \mu_y, \Sigma_y) \) with

\[
G(x, \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^d \det \Sigma_y}} \exp\left(-\frac{1}{2}(x - \mu_y)\top \Sigma_y^{-1}(x - \mu_y)\right).
\]

Given a set \( D = \{(x^1, y^1), \ldots, (x^n, y^n)\} \), we estimate all \( \mu_y \) and \( \Sigma_y \) for \( y \in Y \) using the classical formulas:

\[
\mu_y = \frac{1}{n_y} \sum_{i:y^i = y} x^i \quad \Sigma_y = \frac{1}{n_y} \sum_{i:y^i = y} (x^i - \mu_y)(x^i - \mu_y)\top
\]

(1)

Remark: Alternatively, we can assume a fixed \( \Sigma_y \) and estimate only \( \mu_y \), or estimate a single \( \Sigma \) for all classes, or set \( \Sigma_y = \sigma_y I_d \) and estimate \( \sigma \), etc.
Example (Gaussian Model of Height Distribution)

We observe the following situation:

- $X$: height of a person in cm, $Y = \{(\text{male}, \text{female})\}$.
- $D = \{(181, \text{m}), (165, \text{f}), (161, \text{f}), (172, \text{m}), (175, \text{m}), (178, \text{f})\}$. 

\[
\hat{p}(x | y) = \frac{1}{\sqrt{2\pi\sigma^2_y}} \exp\left(-\frac{1}{2\sigma^2_y}(x - \mu_y)^2\right).
\]

\[
\mu_m = \frac{1}{3}(181 + 172 + 175) = 176
\]

\[
\sigma^2_m = \frac{1}{3}(5^2 + 4^2 + 1^2) = 14
\]

\[
\mu_f = \frac{1}{3}(161 + 165 + 178) = 168
\]

\[
\sigma^2_f = \frac{1}{3}(7^2 + 3^2 + 10^2) \approx 52.7
\]
Example (Gaussian Model of Height Distribution)

We observe the following situation:

- $X$: height of a person in cm, $Y = \{(\text{male}, \text{female})\}$.
- $D = \{(181, \text{m}), (165, \text{f}), (161, \text{f}), (172, \text{m}), (175, \text{m}), (178, \text{f})\}$.

$\mathcal{X} = \mathbb{R}^1$, so $\hat{p}(x|y) = \frac{1}{\sqrt{2\pi \sigma^2_y}} \exp\left(-\frac{1}{2\sigma^2_y}(x - \mu_y)^2\right)$.

$$
\mu_m = \frac{1}{3}(181 + 172 + 175) = 176 \quad \sigma^2_m = \frac{1}{3}(5^2 + 4^2 + 1^2) = 14
$$

$$
\mu_f = \frac{1}{3}(161 + 165 + 178) = 168 \quad \sigma^2_f = \frac{1}{3}(7^2 + 3^2 + 10^2) \approx 52.7
$$
Example (Gaussian Model of Height Distribution)

We observe the following situation:

- $X$: height of a person in cm, $Y = \{(\text{male}, \text{female})\}$.
- $\mathcal{D} = \{(181, \text{m}), (165, \text{f}), (161, \text{f}), (172, \text{m}), (175, \text{m}), (178, \text{f})\}$.

$X = \mathbb{R}^1$, so $\hat{p}(x|y) = \frac{1}{\sqrt{2\pi \sigma_y^2}} \exp\left(-\frac{1}{2\sigma_y^2} (x - \mu_y)^2\right)$. 

![Graph showing Gaussian distributions for male and female heights with data points indicated.](image-url)
Example: 2D Gaussian
Example: 2D Gaussian
Lemma

The classical expressions for estimating $\mu_y$ and $\Sigma_y$ for a Gaussian are the maximum likelihood estimates for the parameters of $\hat{p}(x|y; \mu, \sigma)$. 

Proof. With $G(x; \mu, \Sigma) = \frac{1}{(2\pi)^{d/2}(\text{det} \Sigma)^{1/2}} \exp\left\{ -\frac{1}{2} (x - \mu)^\top \Sigma^{-1} (x - \mu) \right\}$, solve $\mu_{ML} = \arg\max \mu L(\mu)$ for $L(\mu) = \log \sum_{i=1}^{n} \log G(x_i; \mu, \Sigma)$.

$L(\mu) = \frac{1}{2} n \sum_{i=1}^{n} (x_i - \mu)^\top \Sigma^{-1} (x_i - \mu) - \frac{d}{2} \log 2\pi - \frac{d}{2} \log \text{det} \Sigma$.

$\nabla_{\mu} L(\mu, \Sigma) = n \sum_{i=1}^{n} \Sigma^{-1} (x_i - \mu) = \Sigma^{-1} \sum_{i=1}^{n} (x_i - \mu)$.

$H_{\mu} L(\mu, \Sigma) = -\Sigma^{-1} \preceq 0$\implies $\mu_{ML} = \frac{1}{n} \sum_{i=1}^{n} x_i$\implies $\nabla_{\mu} L(\mu_{ML}, \Sigma) = 0$\implies maximum of $L$.

$\Sigma_{ML}$ analogously, but requires some matrix derivatives.
Lemma

The classical expressions for estimating $\mu_y$ and $\Sigma_y$ for a Gaussian are the maximum likelihood estimates for the parameters of $\hat{p}(x|y; \mu, \sigma)$.

**Proof.** With $\mathcal{G}(x; \mu, \Sigma) = \frac{1}{(2\pi \det \Sigma)^{d/2}} \exp\{-\frac{1}{2} (x - \mu)^\top \Sigma^{-1} (x - \mu)\}$, solve

$\mu_{\text{ML}} = \text{argmax}_\mu \mathcal{L}(\mu)$ for \( \mathcal{L}(\mu) = \log \sum_{i=1}^n \log \mathcal{G}(x^i; \mu, \Sigma) \).

\[
\mathcal{L}(\mu) = \frac{1}{2} \sum_{i=1}^n (x^i - \mu)^\top \Sigma^{-1} (x^i - \mu) - \frac{d}{2} \log 2\pi - \frac{d}{2} \log \det \Sigma
\]

\[
\nabla_\mu \mathcal{L}(\mu, \Sigma) = \sum_{i=1}^n \Sigma^{-1} (x^i - \mu) = \Sigma^{-1} \sum_{i=1}^n (x^i - \mu)
\]

\[
H_\mu \mathcal{L}(\mu, \Sigma) = -\Sigma^{-1} \preceq 0
\]

$\mu_{\text{ML}} = \frac{1}{n} \sum_{i=1}^n x^i \Rightarrow \nabla_\mu \mathcal{L}(\mu_{\text{ML}}, \Sigma) = 0 \Rightarrow$ maximum of $\mathcal{L}$

$\Sigma_{\text{ML}}$ analogously, but requires some matrix derivatives.
Classification based on Gaussian models

Let \( \hat{p}(x|y; \mu_y, \Sigma_y) = \frac{1}{\sqrt{(2\pi)^d \det \Sigma_y}} \exp\left(-\frac{1}{2} (x - \mu_y)^\top \Sigma_y^{-1} (x - \mu_y)\right) \). How to make decisions?

General Bayes classifier:

\[
c(x) = \arg\max_{y \in \mathcal{Y}} \frac{\hat{p}(y)}{\sqrt{(2\pi)^d \det \Sigma_y}} \exp\left(-\frac{1}{2} (x - \mu_y)^\top \Sigma_y^{-1} (x - \mu_y)\right)
\]

For two classes, \( \mathcal{Y} = \{+1, -1\} \):

\[
c(x) = \text{sign} \left[ \log \frac{p(x, +1)}{p(x, -1)} \right]
= \text{sign} \left[ (x - \mu_{-1})^\top (\Sigma_{-1})^{-1} (x - \mu_{-1}) \right.
- (x - \mu_{+1})^\top (\Sigma_{+1})^{-1} (x - \mu_{+1}) - \log \frac{\det \Sigma_{+1}}{\det \Sigma_{-1}} \]
\]
Gaussian Mixture Models (GMMs)

More flexibility by modeling each class as a **Mixture of Gaussians**

\[
\hat{p}(x|y; \pi, \vec{\mu}, \vec{\Sigma}) = \sum_{k=1}^{K} \pi_k \ G(x; \mu_k, \Sigma_k) \quad \text{with } \pi_k \geq 0 \text{ and } \sum_{k=1}^{K} \pi_k = 1.
\]
Gaussian Mixture Models (GMMs)

More flexibility by modeling each class as a **Mixture of Gaussians**

\[
\hat{p}(x|y; \pi, \vec{\mu}, \vec{\Sigma}) = \sum_{k=1}^{K} \pi_k G(x; \mu_k, \Sigma_k) \quad \text{with } \pi_k \geq 0 \text{ and } \sum_{k=1}^{K} \pi_k = 1.
\]

No closed form for maximum likelihood parameters, but popular iterative algorithm:

**Expectation-Maximization (EM) algorithm for GMMs**

**input** \(x^1, \ldots, x^n, K\)

init \(\pi, \vec{\mu}, \vec{\Sigma}\)

repeat

\[
\hat{\gamma}_{ik} = \pi_k G(x^i; \mu_k, \Sigma_k), \quad \gamma_{ik} = \hat{\gamma}_{ik}/(\sum_j \hat{\gamma}_{ij})
\]

E-step

\[
\pi_k = \frac{1}{n} \sum_{i=1}^{n} \gamma_{ik} \\
\mu_k = \frac{1}{n\pi_k} \sum_i \gamma_{ik} x^i \\
\Sigma_k = \frac{1}{n\pi_k} \sum_i \gamma_{ik} (x^i - \mu_k)(x^i - \mu_k)^\top
\]

M-step(s)

until convergence

**output** \(\pi, \vec{\mu}, \vec{\Sigma}\)
Example: Mixture of Gaussians in $\mathbb{R}^2$
Example: Mixture of Gaussians in $\mathbb{R}^2$

Single Gaussian model does not fit well.
Example: Mixture of Gaussians in $\mathbb{R}^2$
Example: Mixture of Gaussians in $\mathbb{R}^2$

Mixture of Gaussian model.
Example: Mixture of Gaussians in $\mathbb{R}^2$

Individual Gaussians in the model.
Non-parametric density estimation

Definition

Let \( K_h(x) : \mathcal{X} \to \mathbb{R} \) be a (fixed) kernel function, where \( h \) is a \textit{bandwidth} parameter. Then

\[
\hat{p}(x|y) := \frac{1}{|\{y_i = y\}|} \sum_{\{i:y_i=y\}} K_h(x - x^i)
\]

is called a \textit{kernel density estimate (KDE)} of \( p(x|y) \).

Alternative name: \textit{Parzen windows estimate}.

Kernel density estimates are \textit{non-parametric}. The number of terms grows with the number of examples.
Example: Kernel density estimate

Example

- \( X \): height of a person in cm, \( Y = \{\text{male}, \text{female}\} \).
- \( D = \{(181, \text{m}), (165, \text{f}), (161, \text{f}), (172, \text{m}), (175, \text{m}), (178, \text{f})\} \).

For \( K_h(x) = \frac{1}{\sqrt{2\pi h^2}} \exp\left(-\frac{1}{h^2} \|x\|^2\right) \) (Gaussian with bandwidth \( h \)):
Example: Kernel density estimate

**Example**

- $X$: height of a person in cm, $Y = \{(\text{male}, \text{female})\}$.
- $\mathcal{D} = \{(181, \text{m}), (165, \text{f}), (161, \text{f}), (172, \text{m}), (175, \text{m}), (178, \text{f})\}$.

For $K_h(x) = \frac{1}{2h} \left[ |x| < h \right]$ (Box kernel):
For generative models, one uses the available data to estimate \( p(x, y) \)
- either directly, or
- through the decomposition \( p(x, y) = p(x|y)p(y) \)

Generative models are popular in the natural sciences and engineering because they
- model all information in the data
- often reflect the actual data generation process

Recently, generative models made a come-back in machine learning
- autoregressive/Markov models, variational autoencoders, ...

But: generative models suffer from **curse of dimensionality**!
- one either needs a *lot* of data,
- or, one must resort to a simple (usually wrong) model,
- or, one must have strong additional assumptions, e.g. known independence relations.
In the real world, $p(x, y)$ is unknown, but we have a training set $\mathcal{D}$. At least 3 approaches:

**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

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

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

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

- a **decision theoretic approach**: if we use $\mathcal{D}$ to directly seach for a classifier $c$. 
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 \( p(x, y) = p(y|x)p(x) \).

Because \( \text{argmax}_y p(x, y) = \text{argmax}_y p(y|x) \), 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 \( p(x, y) = p(y|x)p(x) \).
Because \( \text{argmax}_y p(x, y) = \text{argmax}_y p(y|x) \), we don’t need to model \( p(x) \), only \( p(y|x) \).

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

Visual intuition:
Observation

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

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

Example (Spam Classification)

Is \( p(y|x) \) really easier than, e.g., \( p(x|y) \)?
- \( p(\text{"v1agra"}|\text{spam}) \) is some positive value (what fraction of spam words are "v1agra"?)
- \( p(\text{spam}|\text{"v1agra"}) \) is almost surely 1.

For \( p(y|x) \) we can treat \( x \) as given/known, we don’t need to know its probability.
Idea: split $\mathcal{X}$ into regions, for each region store an estimate $\hat{p}(y|x)$.
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 Model, "LogReg", "LR")

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 Model, "LogReg", "LR")**

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

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

**Lemma (Conditional Likelihood Maximization)**

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

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

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

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

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

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

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

(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 \( Z \)) or probability densities with respect to a measure \( d\lambda \) (for continuous \( Z \)).

The **Kullbach-Leibler (KL)-divergence** between \( p \) and \( q \) is defined as

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

(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 $x \in \mathcal{X}$:

$$
\text{KL}_{\text{exp}}(p || q) = \mathbb{E}_{x \sim p(x)} \left\{ \text{KL}(p(\cdot|x) || q(\cdot|x)) \right\}
$$

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 $\text{KL}_{\text{exp}}$:

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

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

indep. of $w$

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 more data we have, the better the approximation will get.
Theorem

Logistic Regression training,

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

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

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

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

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

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

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 \( H_w \mathcal{L}(w) \)

\[
H = \sum_{i=1}^{n} \hat{p}(-y^i | x^i) \hat{p}(y^i | x^i, w) \underbrace{x^i x^i\top}_{>0} \quad \text{sym.pos.def.}
\]
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**, which will converge to a globally optimal solution!

### Steepest Descent Minimization with Line Search

**input** $\epsilon > 0$, tolerance (for stopping criterion)

1: $w \leftarrow 0$

2: **repeat**

3: $v \leftarrow -\nabla_w \mathcal{L}(w)$ \hspace{1cm} \{descent direction\}

4: $\eta \leftarrow \text{argmin}_{\eta>0} \mathcal{L}(w + \eta v)$ \hspace{1cm} \{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.
A discriminative probability model, $\hat{p}(y|x)$, is enough to make decisions:

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

For Logistic Regression, this is particularly simple:

**Lemma**

The LogReg classification rule for 0/1-loss is

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

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

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

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

**Proof.** Elementary, since $\log \frac{\hat{p}(+1|x;w)}{\hat{p}(-1|x;w)} = \langle w, x \rangle$. 
Multiclass Logistic Regression

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

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

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

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

Decision boundaries are still piecewise linear, $c(x) = \text{argmax}_y \langle w_y, x \rangle$. 
**Summary: Discriminative Models**

**Discriminative models** treats the input data, $x$, as fixed and only model the distribution of the outputs $p(y|x)$.

Discriminative models, in particular logistic regression, 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

- usually $\hat{p}_{LR}(y|x) \not\rightarrow p(y|x)$, 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" (→ course on probabilistic graphical models).
In the real world, \( p(x, y) \) is unknown, but we have a training set \( D \). 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

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

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

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

- **a decision theoretic approach:** if we use \( D \) to directly search for a classifier \( c \).
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?