Statistical Machine Learning

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

Christoph Lampert

IST AUSTRIA

Institute of Science and Technology

Fall Semester 2020/2021

Lecture 3

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

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, ...

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:

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

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

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

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

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 $p(x, y)$

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:

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

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 θ by

- maximum likelihood method: $\theta = \mathbf{argmax}_\theta p(\mathcal{D}; \theta)$
- maximum-a-posteriori method: $\theta = \mathbf{argmax}_\theta p(\theta|\mathcal{D})$

Definition (Laplace smoothing)

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

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

the *smoothed empirical estimate* of $p(z)$ (with smoothing parameter α).

Bayesian interpretation:

- Maximum-a-posteriori estimate of parameters θ_z of a multinomial distribution
- Prior on θ: symmetric Dirichlet distribution with parameter α

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

Laplace's "rule of succession": $\alpha = 1$. More common: $\alpha < 1$, e.g. $\alpha = \frac{1}{2}$ or $\alpha = \frac{1}{|\mathcal{Z}|}$.

Continuous Data

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 $\int_x p(x|y)\ \mathrm{dx} = 1$.

Most popular parametric model for continuous data is **Gaussian**:

Definition (Gaussian Density Parameter Estimation)

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

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

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

$$\mu_y = \frac{1}{n_y} \sum_{\{i:y^i=y\}} x^i \qquad \Sigma_y = \frac{1}{n_y} \sum_{\{i:y^i=y\}} (x^i-\mu_y)(x^i-\mu_y)^\top \tag{1}$$

Remark: Alternatively, we can assume a fixed Σ_y and estimate only μ_y, or estimate a single Σ for all classes, or set $\Sigma_y = \sigma_y Id$ and estimate σ, etc.

Example (Gaussian Model of Height Distribution)

We observe the following situation:

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

Example (Gaussian Model of Height Distribution)

We observe the following situation:

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

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

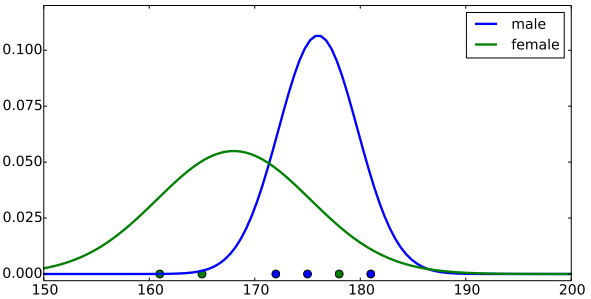
$$\mu_{\texttt{m}} = \frac{1}{3}(181 + 172 + 175) = 176 \qquad \sigma^2_{\texttt{m}} = \frac{1}{3}(5^2 + 4^2 + 1^2) = 14$$
$$\mu_{\texttt{f}} = \frac{1}{3}(161 + 165 + 178) = 168 \qquad \sigma^2_{\texttt{f}} = \frac{1}{3}(7^2 + 3^2 + 10^2) \approx 52.7$$

Example (Gaussian Model of Height Distribution)

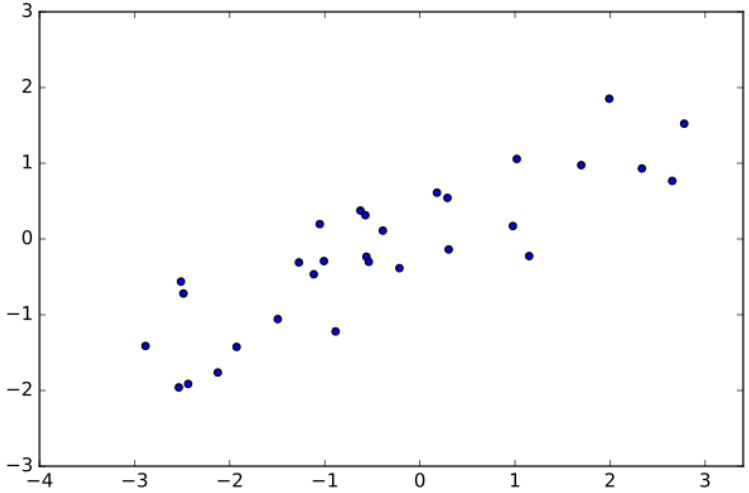
We observe the following situation:

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

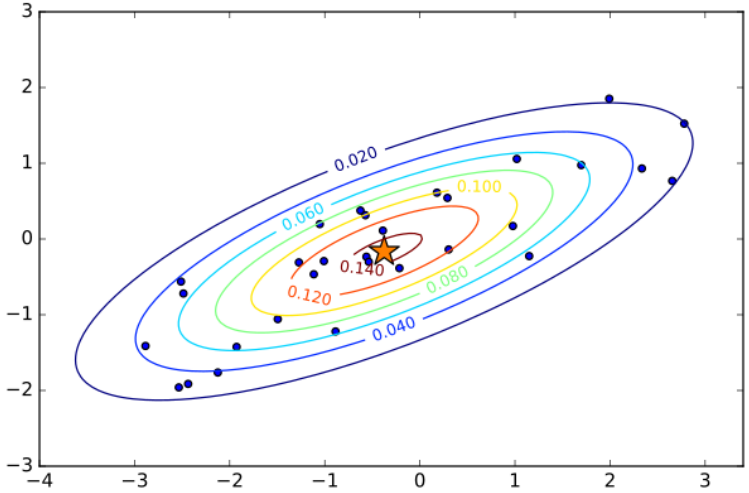
$\mathcal{X} = \mathbb{R}^1$, so $\hat{p}(x|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp(-\frac{1}{2\sigma_y^2}(x - \mu_y)^2)$.



Example: 2D Gaussian



Example: 2D Gaussian



Lemma

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

Lemma

The classical expressions for estimating μ_y and Σ_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 $\mu_{\mathsf{ML}} = \mathbf{argmax}_\mu \mathcal{L}(\mu)$ for $\mathcal{L}(\mu) = \log \sum_{i=1}^n \log \mathcal{G}(x^i;\mu,\Sigma)$.

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

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

$$H_\mu L(\mu,\Sigma) = -\Sigma^{-1} \preccurlyeq 0$$

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

Σ_{ML} analogously, but requires some matrix derivatives.

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

General Bayes classifier:

$$c(x) = \underset{y\in\mathcal{Y}}{\mathbf{argmax}} \{ \frac{\hat{p}(y)}{\sqrt{(2\pi)^d \det \Sigma_y}} \exp(-\frac{1}{2}(x-\mu_y)^\top \Sigma_y^{-1}(x-\mu_y)) \}$$

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

$$\begin{aligned} c(x) &= \text{sign}\left[\log \frac{p(x,+1)}{p(x,-1)}\right] \\ &= \text{sign}\Big[(x-\mu_{-1})^\top(\Sigma_{-1})^{-1}(x-\mu_{-1}) \\ &\qquad - (x-\mu_{+1})^\top(\Sigma_{+1})^{-1}(x-\mu_{+1}) - \log\frac{\det\Sigma_{+1}}{\det\Sigma_{-1}}\Big] \end{aligned}$$

Gaussian Mixture Models (GMMs)

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

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

Gaussian Mixture Models (GMMs)

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

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

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

$\hat{\gamma}_{ik} = \pi_k \mathcal{G}(x^i;\mu_k,\Sigma_k), \quad \gamma_{ik} = \hat{\gamma}_{ik}/(\sum_j \hat{\gamma}_{ij})$ E-step

$\pi_k = \frac{1}{n}\sum_{i=1}^{n} \gamma_{ik}$

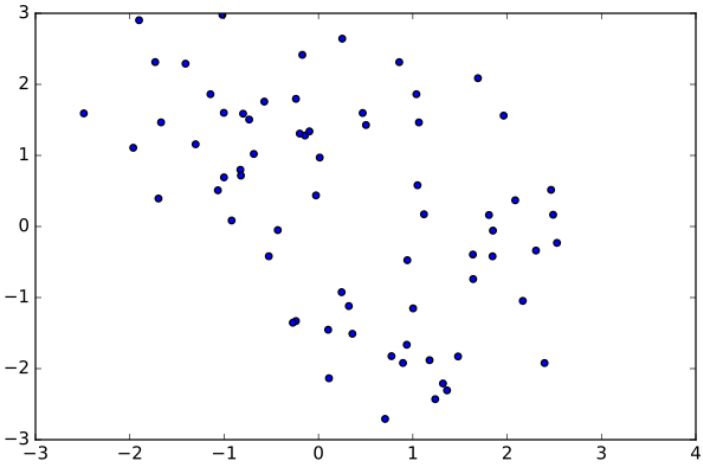
$\mu_k = \frac{1}{n\pi_k}\sum_i \gamma_{ik} x^i$ M-step(s)

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

until convergence

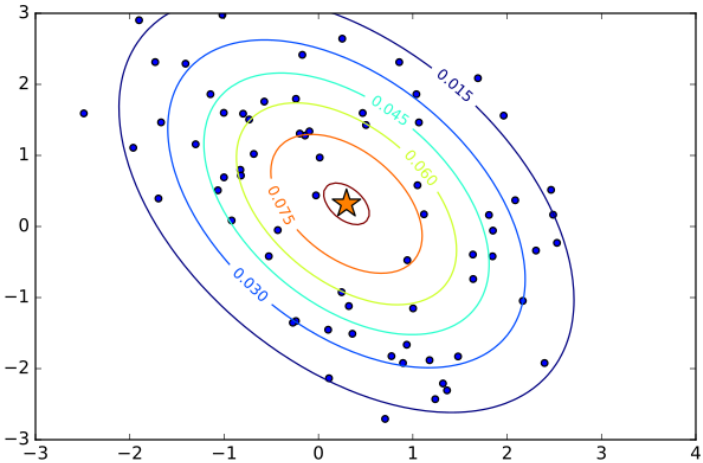
output $\pi, \vec{\mu}, \vec{\Sigma}$

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



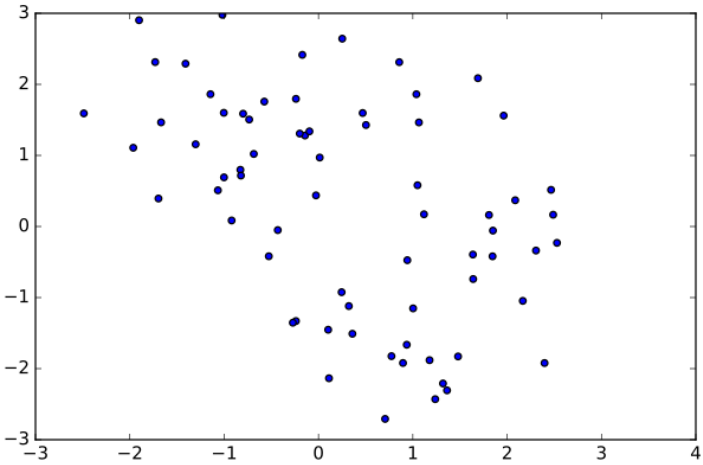
data

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



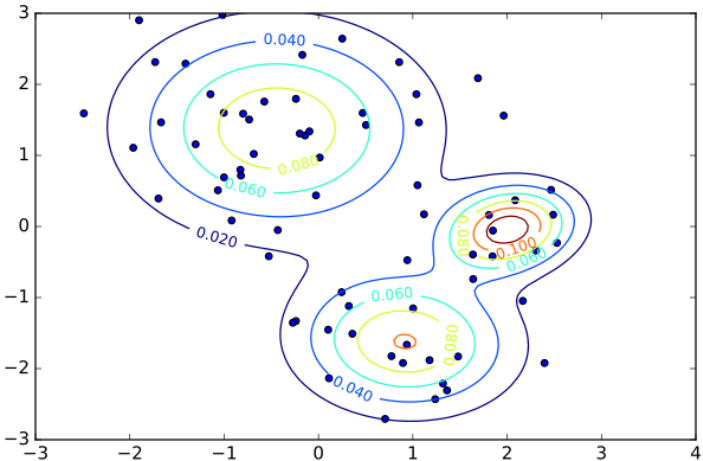
Single Gaussian model does not fit well.

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



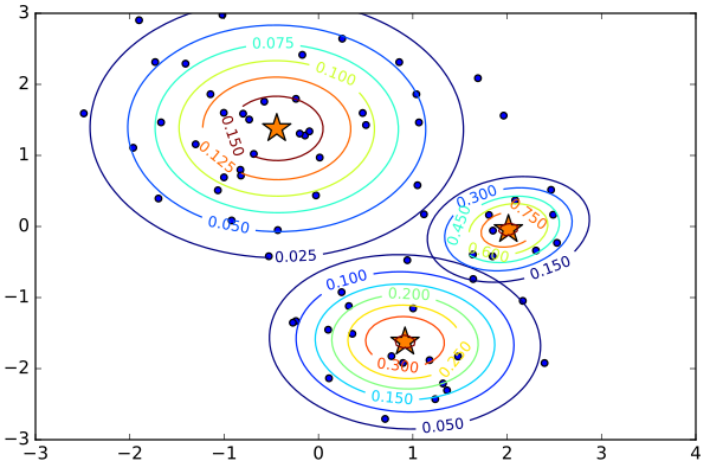
data

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.

Definition

Let $K_h(x) : \mathcal{X} \to \mathbb{R}$ be a (fixed) kernel function, where h is a *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 *kernel density estimate (KDE)* of $p(x|y)$.

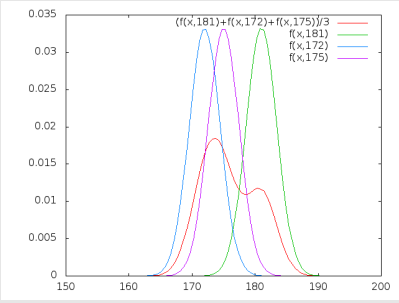
Alternative name: *Parzen windows estimate*.

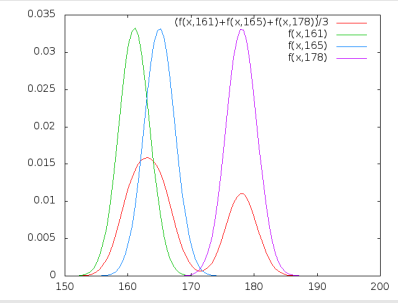
Kernel density estimates are *non-parametric*. The number of terms grows with the number of examples.

Example

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

For $K_h(x) = \frac{1}{\sqrt{2\pi h^2}} \exp(-\frac{1}{h^2}\|x\|^2)$ (Gaussian with bandwidth h):

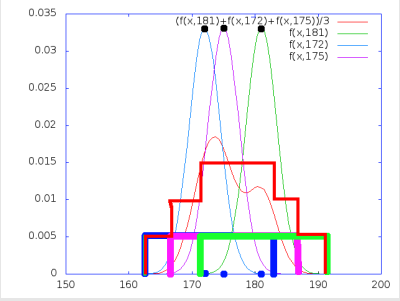


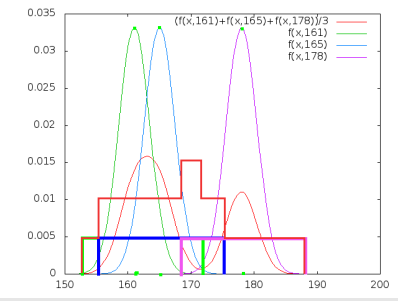


Example

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

For $K_h(x) = \frac{1}{2h}[\![|x| < h]\!]$ (Box kernel):





Summary: Generative Models

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

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

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

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

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

Observation

Task: spam classification, $\mathcal{X} = \{\text{all possible emails}\}, \mathcal{Y} = \{\texttt{spam}, \texttt{ham}\}$.

What's, e.g., $p(x|\texttt{ham})$? For every possible email, a value how likely it is to see that email, including:

- all possible languages,
- all possbile 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 $\mathbf{argmax}_y\, p(x, y) = \mathbf{argmax}_y\, p(y|x)$, we don't need to model $p(x)$, only $p(y|x)$.

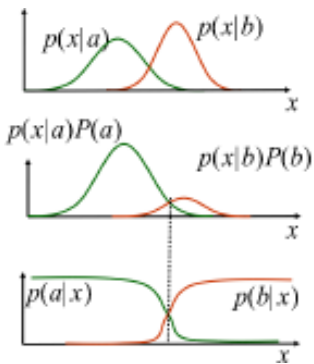
Let's use $\mathcal{D}$ to estimate $p(y|x)$.

Observation

Instead of $p(x, y) = p(x|y)p(y)$, we can also use $p(x, y) = p(y|x)p(x)$.
Because $\mathbf{argmax}_y\, p(x, y) = \mathbf{argmax}_y\, p(y|x)$, 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
= likelihood p(x|y)

joint density
likelihood*prior: p(x|y)p(y)

class posteriors
p(y|x)=p(x|y)p(y)/p(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 $\mathbf{argmax}_y\, p(x, y) = \mathbf{argmax}_y\, p(y|x)$, we don't need to model $p(x)$, only $p(y|x)$.

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

Example (Spam Classification)

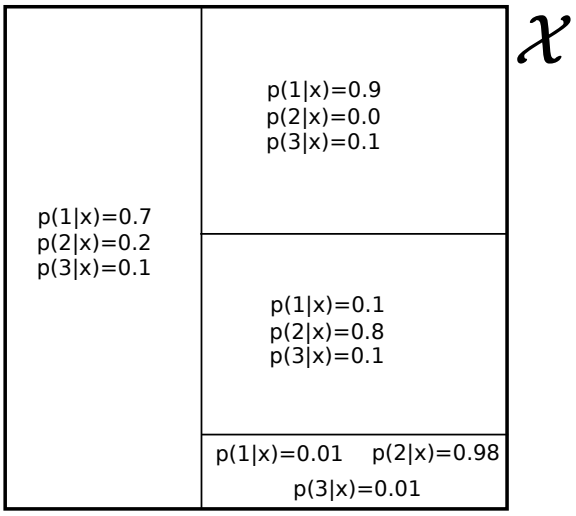
Is $p(y|x)$ really easier than, e.g., $p(x|y)$?

- $p(\textit{"v1agra"}|\texttt{spam})$ is some positive value (what fraction of spam words are "v1agra"?)
- $p(\texttt{spam}|\textit{"v1agra"})$ is almost surely 1.

For $p(y|x)$ we can treat x as *given/known*, we don't need to know its probability.

Nonparametric Discriminative Model

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



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.

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

$$c(x) = \operatorname*{\mathbf{argmax}}_y \hat{p}(y|x)$$

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.

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

$$w_{\mathsf{LR}} \;=\; \operatorname*{\mathbf{argmin}}_{w \in \mathbb{R}^d} \sum_{i=1}^{n} \log\left(1 + \exp(-y^i \langle w, x^i \rangle)\right)$$

Lemma (Conditional Likelihood Maximization)

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

$$w_{LR} \;=\; \operatorname*{\mathbf{argmax}}_{w \in \mathbb{R}^d} \hat{p}(y^1, \ldots, y^n | x^1, \ldots, x^n, w)$$

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

$$\begin{aligned}
-\log\ \hat{p}(\mathcal{D}^Y|\mathcal{D}^X, w) &= -\log \prod_{i=1}^{n} \hat{p}(y^i|x^i, w) = -\sum_{i=1}^{n} \log \hat{p}(y^i|x^i, w) \\
&= -\sum_{i=1}^{n} \log \frac{1}{1+\exp(-y^i\langle w, x^i\rangle)}, \\
&= -\sum_{i=1}^{n} [\log 1 - \log(1+\exp(-y^i\langle w, x^i\rangle)], \\
&= \sum_{i=1}^{n} \log(1+\exp(-y^i\langle w, x^i\rangle).
\end{aligned}$$

Definition (Kullback-Leibler (KL) divergence)

Let p and q be two probability distributions (for discrete $\mathcal{Z}$) or probabilitiy densities with respect to a measure $\mathrm{d}\lambda$ (for continuous $\mathcal{Z}$).

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

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

(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 probabilitiy densities with respect to a measure $\mathrm{d}\lambda$ (for continuous $\mathcal{Z}$).

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

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

(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

$$0 \le KL(p\,\|q) \le \infty, \qquad \text{and} \qquad KL(p\,\|q) = 0 \;\Leftrightarrow\; p = q.$$

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

- it is in general not symmetric, $KL(q\,\|p) \neq KL(p\,\|q)$,
- 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 $x \in \mathcal{X}$:

$$\mathrm{KL}_{exp}(p \,\|q) = \mathop{\mathbb{E}}_{x \sim p(x)} \{\ \mathrm{KL}(p(\cdot|x) \| q(\cdot|x))\ \}$$

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 KL_{exp}:

$$\begin{aligned}\mathrm{KL}_{exp}(p\|\hat{p}) &= \mathop{\mathbb{E}}_{x\sim p(x)} \sum_{y\in\mathcal{Y}} p(y|x) \log \frac{p(y|x)}{\hat{p}(y|x,w)} \\ &= \underbrace{\mathop{\mathbb{E}}_{(x,y)\sim p(x,y)} \log p(y|x)}_{\text{indep. of } w} - \mathop{\mathbb{E}}_{(x,y)\sim p(x,y)} \log \hat{p}(y|x,w)\end{aligned}$$

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 maximimize its empirical estimate based on $\mathcal{D}$:

$$\mathop{\mathbb{E}}_{(x,y)\sim p(x,y)} \log \hat{p}(y|x,w) \;\approx\; \underbrace{\sum_{(x^i,y^i)\in\mathcal{D}} \log \hat{p}(y^i|x^i,w)}_{\log \text{ of conditional data likelihood}} \quad .$$

The more data we have, the better the approximation will get. □

Theorem

Logistic Regression training,

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

is a C^∞-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^∞ 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 $\frac{d\log(t)}{dt} = \frac{1}{t}$

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

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

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

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

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

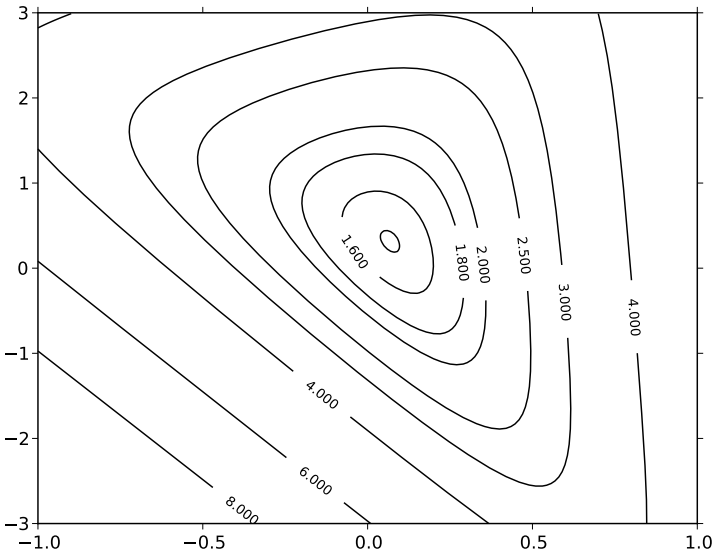
use quotient rule, $\nabla\frac{1}{f(w)} = -\frac{\nabla f(w)}{f^2(w)}$, and chain rule,

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

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

$$H = \sum_{i=1}^{n}\underbrace{\hat{p}(-y^i|x^i)\hat{p}(y^i|x^i,w)}_{>0}\ \underbrace{x^i x^{i\top}}_{\text{sym.pos.def.}}$$

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: $\quad v \leftarrow -\nabla_w \mathcal{L}(w)$ {descent direction}
4: $\quad \eta \leftarrow \mathbf{argmin}_{\eta>0} \mathcal{L}(w + \eta v)$ {1D line search}
5: $\quad w \leftarrow w + \eta v$
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*.

Binary classification with a LogReg Models

A discriminative probability model, $\hat{p}(y|x)$, is enough to make decisions:

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

For Logistic Regression, this is particularly simple:

Lemma

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

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

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

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

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

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

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

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

- 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*, $c(x) = \mathbf{argmax}_y \langle w_y, x \rangle$.

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 $\hat{p}_{\mathsf{LR}}(y|x) \not\to p(y|x)$, even for $n \to \infty$,
- 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" ($\to$ course on probabilistic graphical models).

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

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

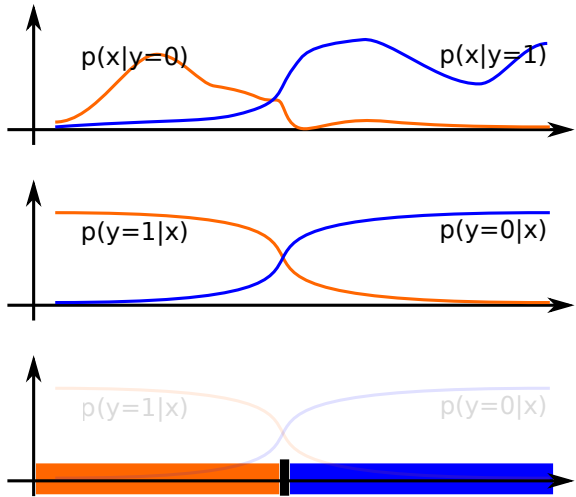
- a discriminative probabilistic approach:
  if we use $\mathcal{D}$ to build a model $\hat{p}(y|x)$ of $p(y|x)$ and define

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

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

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 $c : \mathcal{X} \to \mathcal{Y}$ directly.

Let's use $\mathcal{D}$ to estimate a classifier $c : \mathcal{X} \to \mathcal{Y}$ 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**?