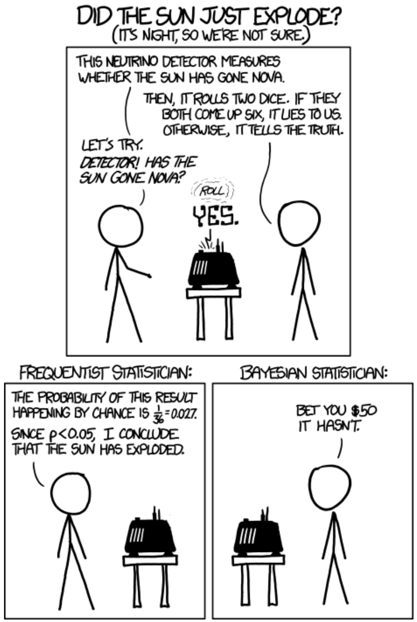
# Advanced course in machine learning
# 582744
# Lecture 3

Arto Klami


DID THE SUN JUST EXPLODE?
(IT'S NIGHT, SO WE'RE NOT SURE.)
THIS NEUTRINO DETECTOR MEASURES WHETHER THE SUN HAS GONE NOVA.
THEN, IT ROLLS TWO DICE. IF THEY BOTH COME UP SIX, IT LIES TO US. OTHERWISE, IT TELLS THE TRUTH.
LET'S TRY.
DETECTOR! HAS THE SUN GONE NOVA?
ROLL
YES.
FREQUENTIST STATISTICIAN:
THE PROBABILITY OF THIS RESULT HAPPENING BY CHANCE IS $\frac{1}{36}$ = 0.027.
SINCE $p < 0.05$, I CONCLUDE THAT THE SUN HAS EXPLODED.
BAYESIAN STATISTICIAN:
BET YOU $50 IT HASN'T.

## Updates

- I will present model solutions on Thursday lectures (which means the hard deadline for returning the solutions is before the lecture)
- Exercise sessions will start with brief overview of the exercise problems
- Would you want a IRC channel for the course?

## Outline

## Reminder: Risk

$$R(\delta) = E_{p(y,\mathbf{x})}[L(y, \delta(\mathbf{x}))] = \int_{\mathbf{x},y} L(y, \delta(\mathbf{x}))p(y, \mathbf{x})d\mathbf{x}dy$$

**Statistical learning theory:**
Treat the data generating process as truly unknown, and try to somehow bound the risk, for example by considering the worst-case scenario

**Bayesian approach:**
Assume we know the correct model but are just unsure of the parameters. Then we can average over the parameters conditional on the data:
$R(\delta) = E_{p(\boldsymbol{\theta}|D)}[L(y, \delta(\mathbf{x}))] = \int_{\boldsymbol{\theta}} L(y, \delta(\mathbf{x}))p(y, \mathbf{x}|\boldsymbol{\theta})p(\boldsymbol{\theta}|D)d\boldsymbol{\theta}$

## Reminder: Risk

$$R(\delta) = E_{p(y,\mathbf{x})}[L(y, \delta(\mathbf{x}))] = \int_{\mathbf{x},y} L(y, \delta(\mathbf{x}))p(y, \mathbf{x})d\mathbf{x}dy$$

**Statistical learning theory:**
Treat the data generating process as truly unknown, and try to somehow bound the risk, for example by considering the worst-case scenario

**Bayesian approach:**
Assume we know the correct model but are just unsure of the parameters. Then we can average over the parameters conditional on the data:
$$R(\delta) = E_{p(\boldsymbol{\theta}|D)}[L(y, \delta(\mathbf{x}))] = \int_{\boldsymbol{\theta}} L(y, \delta(\mathbf{x}))p(y, \mathbf{x}|\boldsymbol{\theta})p(\boldsymbol{\theta}|D)d\boldsymbol{\theta}$$

$$E_{p(\boldsymbol{\theta},y,\mathbf{x}|D)}[L(y, \delta(\mathbf{x}))] = \int_{\boldsymbol{\theta},y,\mathbf{x}} L(y, \delta(\mathbf{x}))p(y, \mathbf{x}, \boldsymbol{\theta})p(\boldsymbol{\theta}|D)d\boldsymbol{\theta}dyd\mathbf{x}$$

## Generative models

- Any probabilistic description of data
- Tells a "story" of how the data was (supposedly) generated
- ...and has some unknown parameters we should ground based on some observed data
- If we believe in that story then averaging over those parameters is the correct solution for the modeling problem

## Generative models

- Any probabilistic description of data
- Tells a "story" of how the data was (supposedly) generated
- ...and has some unknown parameters we should ground based on some observed data
- If we believe in that story then averaging over those parameters is the correct solution for the modeling problem

George Box: "All models are wrong but some are useful"

# Probabilistic modeling

A *generative model* is a model that is written as a collection of probability distributions, describing a process generating the data

$$\begin{aligned} p(y|\mathbf{x},\boldsymbol{\theta}) &= N(\boldsymbol{\theta}^T\mathbf{x},\sigma^2) \\ p(\mathbf{x}) &= \text{Uniform}(-1,1) \\ p(\boldsymbol{\theta}) &= N(0,\mathbf{I}) \\ p(\sigma^2) &= \text{Ga}(0.1,0.1) \end{aligned}$$

Characterized by the joint probability
$p(\mathbf{y},\mathbf{X},\boldsymbol{\theta},\sigma^2) = \prod_n \left[p(y_n|\mathbf{x_n},\boldsymbol{\theta})p(\mathbf{x_n})\right] p(\boldsymbol{\theta})p(\sigma^2)$

## Probabilistic modeling

Now consider the logarithmic loss $L = \log p(\cdot)$, which factorizes into $\log p(y|\mathbf{x}, \boldsymbol{\theta}) + \log p(\mathbf{x}) + \log p(\boldsymbol{\theta}) + \log p(\sigma^2)$

- $p(y|\mathbf{x}, \boldsymbol{\theta})p(\mathbf{x})$ is the *likelihood*, quantifying how well the model fits the data
- $p(\boldsymbol{\theta})p(\sigma^2)$ is the *prior distribution*, which controls the complexity (in some way)

The book presents most methods from this perspective, even though some of them were not originally developed as probabilistic models

## Probabilistic terminology

- Empirical risk minimization is called *maximum likelihood* (ML) estimation since we only care about the likelihood part
- If we include also the prior, corresponding to regularized risk minimization, then we are searching for *maximum a posteriori* (MAP) estimate

## Probabilistic terminology

- Empirical risk minimization is called *maximum likelihood* (ML) estimation since we only care about the likelihood part
- If we include also the prior, corresponding to regularized risk minimization, then we are searching for *maximum a posteriori* (MAP) estimate

- *Evidence / marginal likelihood*: $p(y, \mathbf{x}) = \int_{\boldsymbol{\theta},\sigma^2} p(y|\boldsymbol{\theta}, \sigma^2, \mathbf{x}) p(\mathbf{x}) p(\boldsymbol{\theta}, \sigma^2) d\boldsymbol{\theta} d\sigma^2$
- *Posterior distribution*: $p(\boldsymbol{\theta}, \sigma^2|\mathbf{x}, y) = \frac{p(y|\boldsymbol{\theta},\sigma^2,\mathbf{x})p(\mathbf{x})p(\boldsymbol{\theta},\sigma^2)}{p(y,\mathbf{x})}$ summarizes everything we know about the parameters

## On Bayesian inference vs optimization

Bayesian inference is about averaging predictions over the posterior distribution $p(\boldsymbol{\theta}|D)$

- Finding $p(\boldsymbol{\theta}|D)$ is actually not an optimization problem; it is given directly by the Bayes' rule and all we need is algebraic manipulation and integration
- It is typically not tractable, so in practice we still end up with a learning problem
- Markov chain Monte Carlo: Design a stochastic process that draws samples from $p(\boldsymbol{\theta}|D)$
- Variational inference: Find an approximation $q(\boldsymbol{\theta}|\boldsymbol{\psi}) \approx p(\boldsymbol{\theta}|D)$, optimize $\boldsymbol{\psi}$ to minimize the distance – this is again an optimization problem!

Not covered on this course: See e.g. Advanced Statistical Inference

## Unified view

- Empirical risk minimization $\approx$ maximum likelihood estimation
- Regularized risk minimization $\approx$ maximum a posteriori estimation
- (Bootstrap $\approx$ full Bayesian inference)
- (Leave-one-out cross-validation $\approx$ marginal likelihood)

## Unified view

- Empirical risk minimization ≈ maximum likelihood estimation
- Regularized risk minimization ≈ maximum a posteriori estimation
- (Bootstrap ≈ full Bayesian inference)
- (Leave-one-out cross-validation ≈ marginal likelihood)

Most (all?) regularized loss functions correspond to MAP estimation for some probabilistic model

## Unified view

- Empirical risk minimization $\approx$ maximum likelihood estimation
- Regularized risk minimization $\approx$ maximum a posteriori estimation
- (Bootstrap $\approx$ full Bayesian inference)
- (Leave-one-out cross-validation $\approx$ marginal likelihood)

Most (all?) regularized loss functions correspond to MAP estimation for some probabilistic model

ML should be about full posterior inference, but in practice we often do RRM/MAP, trusting that we regularize well enough

## Reporting distributions

Averaging all predictions over $p(\boldsymbol{\theta}|\mathbf{x})$ would be optimal

Presenting the whole distribution is also optimal if the task is to report it (e.g. weather forecast)

- But in practice we often summarize it with *point estimates*
- ...or by approximating it with a simpler distribution
- ...or by drawing random samples from it

## Point estimates

- Maximum likelihood (ML): Find the *mode* of the likelihood $p(\mathbf{x}|\boldsymbol{\theta})$
- Maximum a posteriori (MAP): Find the mode of the joint likelihood $p(\mathbf{x}, \boldsymbol{\theta})$
- Note that these do not in general match the mean of the posterior distribution
- Mode is optimal for summarizing the posterior for loss $L(\theta, \hat{\theta}) = I(\theta \neq \hat{\theta})$
- Mean would be optimal for $L(\theta, \hat{\theta}) = (\theta - \hat{\theta})^2$
- Median would correspond to $L(\theta, \hat{\theta}) = |\theta - \hat{\theta}|$

## Point estimates

How useful are these?

- For a given model we can compute also the variance of the posterior distribution
- It typically behaves $\propto \frac{1}{N}$, and hence for $N \to \infty$ the posterior becomes a delta distribution
- ...and the posterior mean and mode perfectly characterize the whole distribution

...but in general the point estimates overfit just like all other kinds of models

## Point estimates

How useful are these?

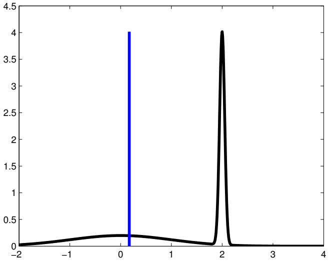
- For a given model we can compute also the variance of the posterior distribution
- It typically behaves $\propto \frac{1}{N}$, and hence for $N \to \infty$ the posterior becomes a delta distribution
- ...and the posterior mean and mode perfectly characterize the whole distribution

...but in general the point estimates overfit just like all other kinds of models

*Data overrides the prior:* For independent data points the logarithmic likelihood is a sum over the data samples, and hence the more data we have the smaller the effect of the prior

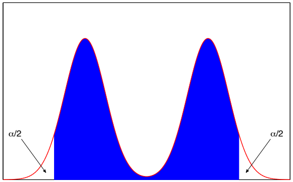
## What's wrong with point estimates?

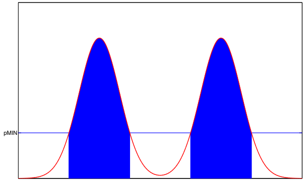
- Mode can be very untypical
- Not invariant to re-parameterization
- ...and naturally they overfit and do not produce confidence intervals of any kind



# Summarizing the full posterior

- Central credible interval vs highest posterior density
- Often it is enough to average the predictions over the posterior; after all, it is simply a tool for solving the risk minimization problem





## Independence day

What kind of models are easy?

## Independence day

What kind of models are easy?

(Conditional) independence simplifies the joint probability, making hard models easier: Remember that $p(x_1, x_2) = p(x_1)p(x_2)$ if the variables are independent

## Independence day

What kind of models are easy?

(Conditional) independence simplifies the joint probability, making hard models easier: Remember that $p(x_1, x_2) = p(x_1)p(x_2)$ if the variables are independent

(Prior distributions with suitable form – conjugate priors – also make the derivations considerably easier)

## Discrete Bayes networks

$p(x_1, x_2, x_3, x_4, x_5)$, a joint distribution over 5 binary variables, has $2^5 - 1 = 31$ parameters

If we assume it factorizes as $p(x_1)p(x_2)p(x_3|x_2)p(x_4|x_1)p(x_5|x_3, x_4)$ then we have only $1 + 1 + 2 + 2 + 4 = 10$ parameters

## Discrete Bayes networks

$p(x_1, x_2, x_3, x_4, x_5)$, a joint distribution over 5 binary variables, has $2^5 - 1 = 31$ parameters

If we assume it factorizes as $p(x_1)p(x_2)p(x_3|x_2)p(x_4|x_1)p(x_5|x_3, x_4)$ then we have only $1 + 1 + 2 + 2 + 4 = 10$ parameters

...but more importantly estimating them is easier: If $N$ samples is enough to estimate $p(x_1)$ well then how much is needed for estimating $p(x_1, x_2)$? How about $p(x_1, x_2, x_3)$? Why

## Discrete Bayes networks

$p(x_1, x_2, x_3, x_4, x_5)$, a joint distribution over 5 binary variables, has $2^5 - 1 = 31$ parameters

If we assume it factorizes as $p(x_1)p(x_2)p(x_3|x_2)p(x_4|x_1)p(x_5|x_3, x_4)$ then we have only $1 + 1 + 2 + 2 + 4 = 10$ parameters

...but more importantly estimating them is easier: If $N$ samples is enough to estimate $p(x_1)$ well then how much is needed for estimating $p(x_1, x_2)$? How about $p(x_1, x_2, x_3)$? Why

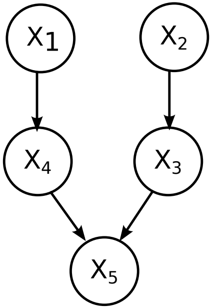
Roughly $2N$ and $4N$; we need to estimate $p(x_1)$ for both choices of $x_2$ and hence have less effective data samples

## Discrete Bayes networks

We went from $p(x_1, x_2, x_3, x_4, x_5)$ to $p(x_1)p(x_2)p(x_3|x_2)p(x_4|x_1)p(x_5|x_3, x_4)$

This is handy to present in graphical form, as a Bayes network

We will not discuss (binary) Bayes networks further on this course, but will still use some of the same tools



## Graphical models

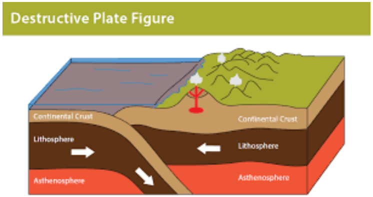
A *graphical model* refers to any probabilistic model with some independence assumptions presented visually – often in form of a *plate diagram* or a *factor graph*

We focus on the former, corresponding to directed models

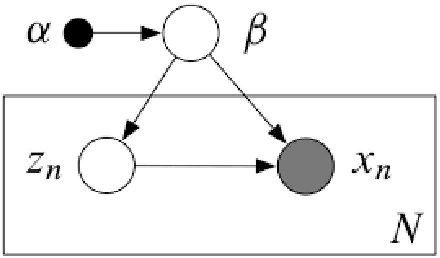
Besides the graph we naturally need to describe the probabilities as well!

## Plate diagrams

The eatwell plate
Use the eatwell plate to help you get the balance right. It shows how much of what you eat should come from each food group.
Fruit and vegetables
Bread, rice, potatoes, pasta and other starchy foods
Meat, fish, eggs, beans and other non-dairy sources of protein
Foods and drinks high in fat and/or sugar
Milk and dairy foods

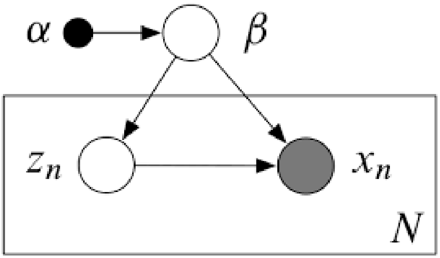


## Plate diagrams



- Arrow: Conditional dependency
- Shaded node: Observed data
- Hollow node: Parameter or latent variable
- (Dot: Fixed prior parameter)
- Plate: Replication

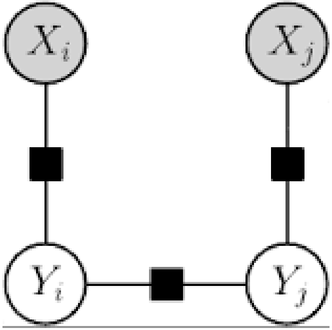
## Plate diagrams



- Arrow: Conditional dependency
- Shaded node: Observed data
- Hollow node: Parameter or latent variable
- (Dot: Fixed prior parameter)
- Plate: Replication

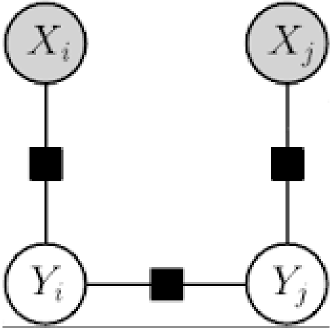
$$p(X, Z, \beta|\alpha) = p(\beta|\alpha) \prod_n p(x_n|z_n, \beta)p(z_n|\beta)$$

## Factor graphs



- Nodes and plates as before
- Square: Factor tying two or more variables together

## Factor graphs



- Nodes and plates as before
- Square: Factor tying two or more variables together

$$p(x_i, x_j, y_i, y_j) = \frac{1}{Z} q(x_i, y_i) q(y_i, y_j) q(x_j, y_j)$$

## Parameters and latent variables

- Both are random variables
- Parameters are “global”, latent variables are “local”
- Cluster centroids vs cluster assignments
- Many non-probabilistic models relax the assumption of parameters being random variables (for computational reasons, or for philosophical)

## Hierarchical models

Three ways of modeling a data set with samples from three different sources

- Model each of them independently
- Pool all of the data together and learn a single model
- Model them together but assume joint priors for the parameters, pulling them towards each other

The last one should, in principle, be always preferred – especially as the model family can include the other two extremes as special cases

This kind of models are trivial to formulate in the probabilistic framework

Non-probabilistic formulations of this called *parameter tying* or *parameter sharing*, motivated by a regularizing effect

## Gaussian models (Section 4)

A lot revolves around the multivariate normal distribution

$$p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \exp(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}))$$

...so let's look at inference for that alone. Imagine we have observed $N$ data points $\mathbf{x}_n \in \mathbb{R}^D$ and want to fit a simple model $\prod_n p(\mathbf{x}_n|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with uniform priors

## Gaussian models (Section 4)

Start by taking the log:

$$\log \prod_n p(\mathbf{x}_n|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = -\frac{ND}{2}\log(2\pi) - \frac{N}{2}\log|\boldsymbol{\Sigma}| - \frac{1}{2}\sum_n (\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})$$

...and notice that it is a quadratic function with respect to $\boldsymbol{\mu}$. The gradient w.r.t. $\boldsymbol{\mu}$ is

$$\boldsymbol{\Sigma}^{-1}\sum_n (\mathbf{x}_n - \boldsymbol{\mu}),$$

which is zero when $\boldsymbol{\mu} = \frac{1}{N}\mathbf{x}_n$

## Gaussian models (Section 4)

The derivative w.r.t to $\boldsymbol{\Delta} = \boldsymbol{\Sigma}^{-1}$ is

$$\frac{N}{2}\boldsymbol{\Sigma} - \frac{1}{2}\mathbf{S}^T,$$

where $\mathbf{S} = \sum_n (\mathbf{x}_n - \boldsymbol{\mu})(\mathbf{x}_n - \boldsymbol{\mu})^T$

...and hence $\boldsymbol{\Sigma} = \frac{1}{N}\mathbf{S}$

So: $\boldsymbol{\mu}$ is the empirical mean and $\boldsymbol{\Sigma}$ is the empirical covariance matrix – what a surprise!

## Gaussian models (Section 4)

Now that we know how to solve the MLE estimate, we can easily solve similar problems with multivariate normal distributions

In particular, we can add prior distributions and solve the MAP estimate

Another important result in Section 4.3:
The conditional distribution of a multivariate normal can be expressed in closed-form, and is still a normal distribution

## Supervised learning

The thing we all know:

- Input: $\mathbf{x}_n, y_n$
- Learning task: $f(\mathbf{x}_n) \approx y_n$
- Use: $f(\mathbf{x})$

## Regression

If the output is some continuous space, we call the problem *regression*

Linear regression is the canonical example, but also non-linear regression models exist

...and the output space need not be $\mathbb{R}$:

- Multiple regression: $\mathbf{y} \in \mathbb{R}^L$
- Ordinal regression: only the relative order matters but the values are not on a real scale
- Logistic regression: $y \in [0, 1]$ – but often this is used for classification instead, interpreting the output as probability of a class

The key element: The order of $y$ means something

## Classification

Any supervised learning task for which which the order of $y$ is not meaningful, but instad we simply have some collection of unordered alternatives

- Binary classification: $y \in [0, 1]$ or $y \in [-1, 1]$
- Multi-class classification: $y \in [1, ..., K]$
- Multi-label classification: $y \in [0, 1]^D$ (each data point can belong to multiple "classes")

Multi-class often solved using binary classifiers

- *One-vs-all*: Solve $K$ binary tasks for $[y_k, y_{-k}]$, use $\arg\max f_k$ to pick the class
- *One-vs-one*: Solve $K(k-1)/2$ binary tasks between all pairs of classes, use voting the make the decision

...but direct approaches exist as well

## Unsupervised learning

No distinction between inputs and outputs; we have some data $\mathbf{x}_n$ and want to understand the process generating them

Often solved with latent-variable models: Some unknown quantity for each data point becomes the new representation for that data

- Clustering: Data points represented by cluster index, similar points grouped together
- Dimensionality reduction: Data points represented by lower-dimensional vectors (linear or non-linear mapping)
- Missing value imputation: Estimate data elements that are unknown or censored

Some people call clustering "unsupervised classification", or otherwise mix clusters with classes. Don't do that

## Unsupervised learning

Most (all?) UL techniques somehow based on *reconstruction error*: We find a simpler representations that is enough to represent the original data sufficiently well

- Reconstruct sample as its cluster mean
- Reconstruct sample in a linear subspace
- Reconstruct sample with a neural network that has a bottleneck

## Reinforcement learning

Supervised learning with delayed and possibly indirect feedback

We do not have clear output value for every sample, but instead receive some corrective signal or loss at a later stage

AlphaGo, robots, old Atari games, ...

Not covered further on this course, but you should remember that classical supervised learning tasks are not enough for everything

## Transfer learning

The above modeling tasks consider scenarios with $N$ typically i.i.d. samples. What if we have $K$ groups of samples that might or might not be fromt he same distribution?

The umbrella term *transfer learning* studies techniques for modeling such setups

## Transfer learning: Multi-task learning

- Learning $K$ supervised taks, each with their own inputs and outputs, together, often so that the data space for the inputs is the same
- Directly matches the hierarchical modeling scenario, but non-probabilistic alternatives also exist
- Often useful in rather narrow set of conditions: With too little data pooling is good, with too much data the independent models are good
- *Negative transfer*: If the tasks are not related closely enough, modeling them together can hurt instead of helping

## Transfer learning: Domain adaptation

What if the training data and test data do not come from the same distribution? Perhaps one was collected last year or with different sensors?

- Domain adaptation: We can attempt to model the change between the *source domain* and *target domain*
- Corresponds to assuming a model for the change itself, for example linear rotation and scaling
- *Covariate shift*: $p(y|\mathbf{x})$ is the same but $p(\mathbf{x})$ changes
- *Class imbalance*: $p(\mathbf{x}|y)$ is the same but $p(y)$ changes

## Semi-supervised learning

Setups where some samples have labels and some do not

We hope that knowing more about $p(\mathbf{x})$ (in form of additional samples) helps in solving the supervised learning task

*Transductive learning*: We only need to classify the unlabaled samples
*Inductive learning*: We want to solve the classification problem for future samples as well

Helps if the output causes (some of) the input features (`http://icml.cc/2012/papers/625.pdf`)

## Multi-view learning

Often the data samples are represented by multiple complementary views: image and its caption, web page and its visitor counts, the same text in multiple languages, ...

The phrase *multi-view learning* covers various methods that model such data sets

Unsupervised: Search for dependencies between the views
Supervised: Maximally utilize the complementary information

The naive solution of just ignoring the split into multiple views is often a reasonable approach

## Structured outputs

Above the output $y$ was always a scalar or a simple vector

What if we need to predict a three-dimensional structure of a protein, a translation of a given sentence into another language, or a phylogenetic tree?

This is called *structured (output) prediction*