USI/SUPSI
Istituto
Dalle Molle
di studi
sull'intelligenza
artificiale
IDSIA

Probabilistic Models and Methods

Machine Learning

Michael Wand, TA: Mikhail Andronov

{michael.wand, mikhail.andronov}@idsia.ch

Dalle Molle Institute for Artificial Intelligence Studies (IDSIA) USI - SUPSI

Fall semester 2024

Introduction

- So far, we have considered the machine learning system as a computer with adjustable parameters.
- *Training*, i.e. searching for the best parameters, was performed by minimizing an error function on the training data. In order to encourage generalization, we have introduced *regularization*.
- All this was done with limited theoretical justification.

We now consider machine learning based on probabilistic methods. In particular, we model our data by probability distributions, which

$\Rightarrow$ enables novel methods

$\Rightarrow$ naturally gives probabilistic outputs

$\Rightarrow$ helps combatting overfitting

$\Rightarrow$ yields theoretical guarantees

$\Rightarrow$ allows model interpretation.

Contents

- Introduction to Probabilistic and Bayesian Modelling
- Learning The Gaussian Distribution
- Linear Regression Revisited
- Logistic Regression Revisited
- The Bias-Variance Decomposition
- Gaussian Mixture Models, Latent Variables, and EM Training
- Variational Inference

Further reading:

- Bishop, *Pattern Recognition and Machine Learning*
- Hastie, *The Elements of Statistical Learning*
- Wasserman: *All of Statistics*
- Probabilistic ML course from UCL:
  `http://www.gatsby.ucl.ac.uk/teaching/courses/ml1-2017`

USI/SUPSI
Istituto
Dalle Molle
di studi
sull'intelligenza
artificiale
IDSIA

Introduction to Probabilistic and Bayesian Modelling

Probabilistic Modeling

- Probabilistic modeling is based on the premise that data (either in a supervised setup with targets, or in an unsupervised setup) is generated by some unknown random process.
- Therefore, we postulate a *probabilistic model* of data production:

$$\mathcal{X} \sim p(x).$$

- We assume that (training/test) data has been generated by *sampling* from this distribution. In this lecture, we always assume that the samples have been sampled independently from a fixed p, one then says that the data $\mathcal{X}$ is independent and identically distributed (iid).
- p is the generative model. It can be used to
  - ➔ make predictions
  - ➔ make inferences about missing inputs
  - ➔ generate predictions/fantasies/imagery.
- In supervised tasks, data *and* targets are comprised in the vector x.

This section (in particular the coin toss example) is heavily based on the UCL slides, lecture 1.

Probabilistic Modeling

- All of probabilistic machine learning deals with estimating the distribution p from the training data.
- If we accurately estimate p, we will obtain optimal generalization in the underlying task: the (inference/prediction/generative) system will also work well on new samples.
- As we know already, this is not always possible, in particular, our model can *overfit* or *underfit*.

We will see that the boundaries between probabilistic modeling and other methods of Machine Learning are fluent. The power of probabilistic modeling is based on:

- a powerful way to reason about uncertainties, both in the prediction and in the data itself
- an extensive set of mathematical tools.

Probabilistic Modeling

- For estimating $p(x)$, we need to a) define a suitable criterion, b) describe the space of probability distributions in a way that allows optimization.
- In practice, we usually assume that p can be written as a mathematical formula which depends on some parameter vector θ:

$$p(x) = p(x|\theta).$$

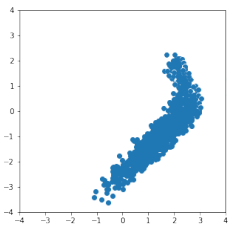
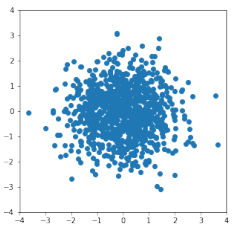
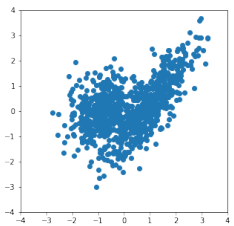
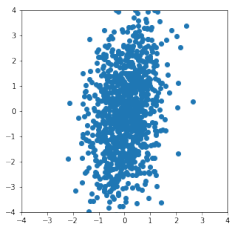
- Optimization now amounts to finding the "best" parameter vector θ.
- However, this means that out of all possible probability distributions, we only consider the subset whose elements match the function template $p(x|\theta)$ for some θ.

Probabilistic Modeling

- As an example, consider the Gaussian distribution

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

- The parameter vector consists of (the coefficients of) $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$.
- However, many realistic data sets cannot be well described with a single Gaussian distribution. Which of the following data sets are well described by a Gaussian distribution?

- Two possible approaches:
  - ➔ use a template for a distribution family $p(x|\theta)$ which is versatile enough to approximate any true distribution
  - ➔ try to determine the true distribution of the data.

The Maximum Likelihood Criterion

- Assume that we have a (finite) set of training data $\mathcal{X} = \{x_n\}_{n=1,\ldots,N}$.
- We also assume that this data is iid. Then we have

$$\mathcal{L}(\theta) := p(\mathcal{X}|\theta) = \prod_{n=1}^{N} p(x_n|\theta).$$

  This is the likelihood of the training data: The joint probability as a function of the parameter vector θ.
- A straightforward criterion for choosing θ: The Maximum Likelihood (ML) criterion

$$\hat{\theta}_{\mathrm{ML}} = \arg\max_{\theta} \mathcal{L}(\theta) = \arg\max_{\theta} p(\mathcal{X}|\theta).$$

Coin Toss example

- We do a simple example for estimating a parameter with the ML criterion!
- Assume we throw a single coin, which may come up heads or tails. Let $q \in [0, 1]$ be the probability of heads, then $1 - q$ is the probability of tails.
  - ➔ Thus we have a single parameter q which we need to optimize.
  - ➔ (If we assign heads=1, tails=0, this is a Bernoulli distribution).
- Assume that we throw the coin 6 times, and we obtain 4 heads and 2 tails.
- Goal: estimate the value of q!

Coin Toss example

- We compute the likelihood of the entire dataset:

$$\mathcal{L}(q) = p(\mathrm{HHHHTT}|q) = q^4(1-q)^2$$

- The maximum of this function (in the range between 0 and 1) can be found by computing the derivative and setting it to zero.
- It is often easier to compute the maximum of the *logarithm* of the likelihood[1]. So we define

$$\ell(q) = \log \mathcal{L}(q) = 4 \log q + 2 \log(1-q).$$

- The derivative is $\frac{\mathrm{d}\ell}{\mathrm{d}q} = \frac{4}{q} - \frac{2}{1-q}$, and it is easy to see that the only zero of this function between 0 and 1 is $q = \frac{2}{3}$.
- Thus the Maximum Likelihood estimate of q is

$$\hat{q}_{\mathrm{ML}} = \frac{2}{3}.$$

[1] it may also be better when the calculation is performed numerically, since underflows are avoided

Maximum Likelihood Problems

- However, there are some issues with this approach:

Maximum Likelihood Problems

- However, there are some issues with this approach:
- Assume you throw the coin three times and get heads three times.
- What is the ML estimate for q? Do you think it makes sense?

Maximum Likelihood Problems

- However, there are some issues with this approach:
- Assume you throw the coin three times and get heads three times.
- What is the ML estimate for q? Do you think it makes sense?
- $q = 1$, but intuitively this is a rather drastic estimate...

Maximum Likelihood Problems

- However, there are some issues with this approach:
- Assume you throw the coin three times and get heads three times.
- What is the ML estimate for q? Do you think it makes sense?
- $q = 1$, but intuitively this is a rather drastic estimate...
- And if you throw the coin 100 times and always get heads?

Maximum Likelihood Problems

- However, there are some issues with this approach:
- Assume you throw the coin three times and get heads three times.
- What is the ML estimate for q? Do you think it makes sense?
- $q = 1$, but intuitively this is a rather drastic estimate...
- And if you throw the coin 100 times and always get heads?
- We start to believe that the coin is actually very heavily biased.

Maximum Likelihood Problems

- However, there are some issues with this approach:
- Assume you throw the coin three times and get heads three times.
- What is the ML estimate for q? Do you think it makes sense?
- $q = 1$, but intuitively this is a rather drastic estimate...
- And if you throw the coin 100 times and always get heads?
- We start to believe that the coin is actually very heavily biased.
- And if you *know* the coin is from the joke shop?

Maximum Likelihood Problems

- However, there are some issues with this approach:
- Assume you throw the coin three times and get heads three times.
- What is the ML estimate for q? Do you think it makes sense?
- $q = 1$, but intuitively this is a rather drastic estimate...
- And if you throw the coin 100 times and always get heads?
- We start to believe that the coin is actually very heavily biased.
- And if you *know* the coin is from the joke shop?
- We will consider extreme values for q more likely.

Maximum Likelihood Problems

We see:

- Maximum likelihood is *very* unstable when the estimate is made from a small amount of data.
- There is no way to model how certain we are about our estimate.
- There is no way to incorporate *prior knowledge*, e.g. the fact that a normal coin can at most be slightly imbalanced.

Additionally:

- Maximum likelihood is problematic if the model is not good.
- Maximum likelihood inverts cause and consequence: We have the data and look for parameters, and we do it by choosing parameters from which we can derive the data with highest probability.

Maximum Likelihood Problems

We see:

- Maximum likelihood is *very* unstable when the estimate is made from a small amount of data.
- There is no way to model how certain we are about our estimate.
- There is no way to incorporate *prior knowledge*, e.g. the fact that a normal coin can at most be slightly imbalanced.

Additionally:

- Maximum likelihood is problematic if the model is not good.
- Maximum likelihood inverts cause and consequence: We have the data and look for parameters, and we do it by choosing parameters from which we can derive the data with highest probability.
- What does this remind you of?

Maximum Likelihood Problems

We see:

- Maximum likelihood is *very* unstable when the estimate is made from a small amount of data.
- There is no way to model how certain we are about our estimate.
- There is no way to incorporate *prior knowledge*, e.g. the fact that a normal coin can at most be slightly imbalanced.

Additionally:

- Maximum likelihood is problematic if the model is not good.
- Maximum likelihood inverts cause and consequence: We have the data and look for parameters, and we do it by choosing parameters from which we can derive the data with highest probability.
- What does this remind you of?
- Answer: *Bayes' Theorem, which can be used to invert cause and consequence!*

Towards a Bayesian Approach

- Bayes Theorem allows us to invert the logical direction of probabilities: $P(B|A) \rightsquigarrow P(A|B)$.
- Also remember that incorporating *prior* probabilities and updating them with new evidence can be modeled based on Bayes' Theorem.
- This is the core of Bayesian learning: All parameters which we estimate have prior probability distributions which get updated by evidence to the posterior distribution.
- Naturally allows to model a) prior knowledge, b) uncertainty about our parameters, c) how more data increases our knowledge.

Towards a Bayesian Approach - Definitions

- We make the following (generic) definitions:
  - ➔ the parameters θ (can be seen as a vector, or a set)
  - ➔ the Likelihood model or data model $P(\mathcal{X}|\theta)$, where P is a family of probability distributions parametrized by θ
  - ➔ the prior probability of the parameters $P(\theta)$.
- We obtain a *probabilistic* estimate of the parameters θ using Bayes' Theorem:

$$P(\theta|\mathcal{X}) = \frac{P(\mathcal{X}|\theta)P(\theta)}{P(\mathcal{X})}$$

with the evidence

$$P(\mathcal{X}) = \int P(\mathcal{X}, \theta)\, \mathrm{d}\theta = \int P(\mathcal{X}|\theta)P(\theta)\, \mathrm{d}\theta.$$

- $P(\theta|\mathcal{X})$ is the posterior probability of θ, given the data.

Bayesian Coin Tossing

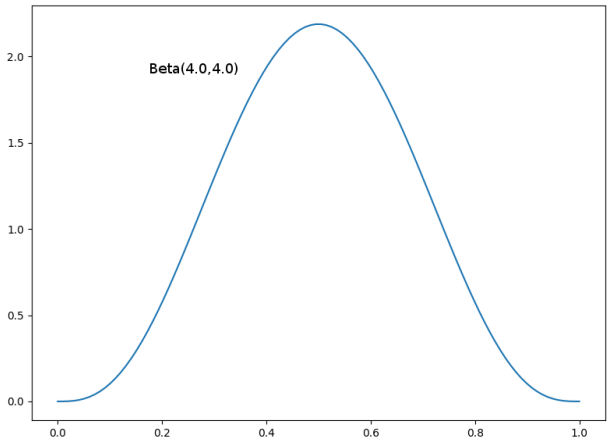
- We apply Bayesian reasoning to coin tossing!
- Our goal is the same as before: We aim to estimate the probability q that the coin comes up heads.
- Great difference to the ML approach: q is now itself *probabilistic*!
- Thus we estimate not a single value for q, but a *probability distribution* which describes possible values for q.
- This allows to model how uncertain we are about the true value of q.

Bayesian Coin Tossing

- We start this analysis by imposing a prior for q.
- This prior should reflect our belief about q. Intuitively, we might assume that the coin is not extremely biased, i.e. q is close to 0.5.
- We describe this belief by a *Beta distribution*, whose density function is given by

$$\text{Beta}(x|\alpha,\beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)}$$

- $B(\alpha,\beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$ is the normalization factor, where $\Gamma(z) = \int_0^\infty t^{z-1}e^{-t}\mathrm{d}t$ is the gamma function.
- The figure shows our prior probability, the beta distribution with parameters $\alpha = \beta = 4.0$.



Bayesian Coin Tossing

- Our prior distribution for q: $p(q) = \text{Beta}(q|4,4) = \frac{q^3(1-q)^3}{B(4,4)}$.
- Now we toss the coin! We have the two possible outcomes H and T with respective probabilities

$$p(H|q) = q \qquad P(T|q) = 1 - q.$$

- Assume the coin shows heads.

Bayesian Coin Tossing

- Our prior distribution for q: $p(q) = \text{Beta}(q|4,4) = \frac{q^3(1-q)^3}{B(4,4)}$.
- Now we toss the coin! We have the two possible outcomes H and T with respective probabilities

$$p(H|q) = q \qquad P(T|q) = 1 - q.$$

- Assume the coin shows heads.
- We derive the posterior probability distribution for q by applying Bayes' rule:

$$p(q|H) = \frac{p(H|q)p(q)}{p(H)} =$$

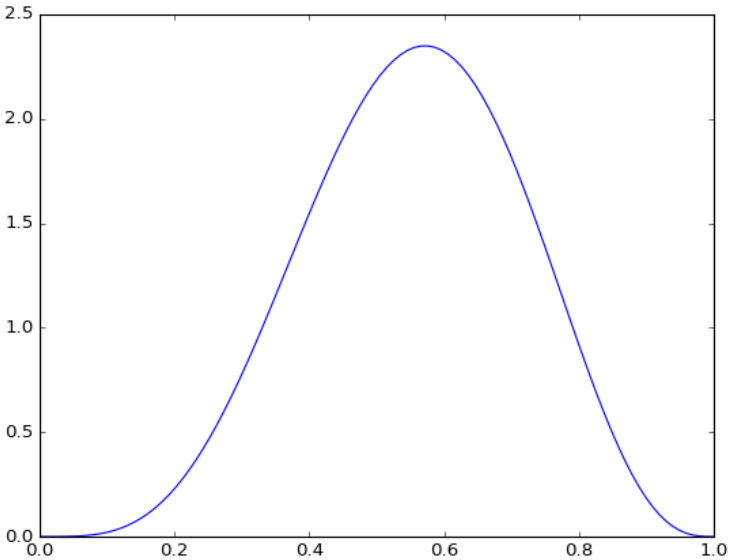
$$= c \cdot q \cdot q^3(1-q)^3 = c \cdot q^4(1-q)^3 \qquad (*)$$

for a constant $c = (p(H) \cdot B(4,4))^{-1}$ which does not depend on q.

Bayesian Coin Tossing

- c can be derived by observing that $p(q|H)$ is a probability distribution (i.e. it must be normalized).
- From $(*)$, we see that $p(q|H)$ has the form of a Beta distribution with $\alpha = 5$ and $\beta = 4$!
- Therefore the normalization coefficient must be $c = (B(5,4))^{-1}$.
- The posterior distribution for q is

$$p(q|H) = \text{Beta}(5,4).$$

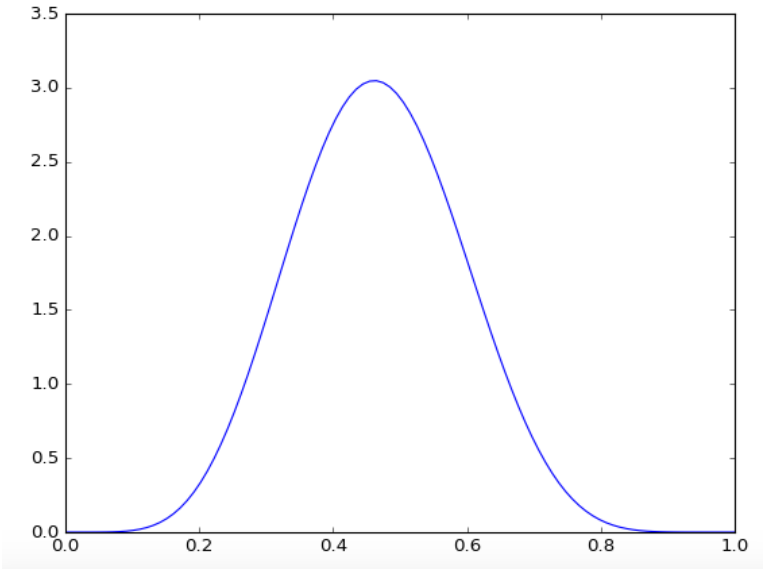


Bayesian Coin Tossing

- What about observing a series of coin throws, e.g. *HTHHTTT*?
- We can apply the same reasoning as before to derive

$$p(q|HTHHTTT) = \text{Beta}(7, 8).$$

- Each Heads increases the parameter α by 1, each Tails increases the parameter β by 1.

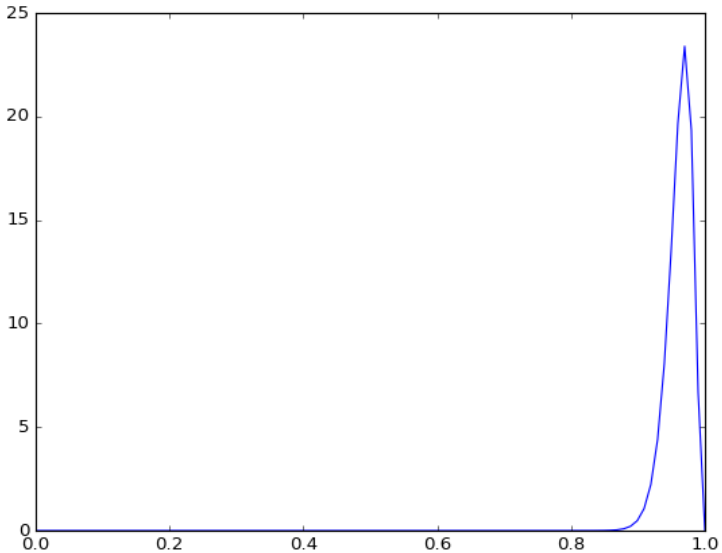


Bayesian Coin Tossing

- Q: What happens in Bayesian Coin Tossing if we throw the coin 100 times and obtain 100 Heads?

Bayesian Coin Tossing

- Q: What happens in Bayesian Coin Tossing if we throw the coin 100 times and obtain 100 Heads?



- The posterior Beta$(100, 4)$ will be very shifted towards $q = 1$, but will still be probabilistic.
- Note that due to the choice of prior, the probability of $q = 1$ is always zero.

Bayesian Coin Tossing

- In this example, we have the important property that the prior and posterior distributions belong to the same family.
- In this case, prior and posterior are called conjugate distributions, and the prior is a conjugate prior for the likelihood function $p(\mathcal{X}|\theta)$.
- In the coin toss example, the likelihood is the binomial distribution

$$p(h) = \binom{n}{h} q^h (1-q)^{n-h}$$

  where h is the number of Heads in a total of n tosses. The Beta distribution is a conjugate prior for the binomial distribution.
- Choosing a conjugate prior simplifies many calculations!
- It implies, however, that we choose the prior distribution for mathematical convenience, not (only) due to our prior belief.
- Also note that we have chosen the initial parameters $\alpha = \beta = 4.0$ somewhat randomly.

Bayesian Coin Tossing: Conclusion

- The lack of a principled way to choose the prior is often considered a weakness of the Bayesian method!
- In many realistic cases, the involved distributions are more complicated, and exactly computing the posterior distribution is not possible ("intractable").
- Solution 1: Make point estimates! For example, compute the parameter vector which maximizes the posterior distribution, instead of deriving the full posterior – the Maximum a posteriori (MAP) estimate:

$$\hat{q}_{\text{MAP}} = \arg\max_{\theta} P(\theta|\mathcal{X})$$

- Solution 2: Approximate the posterior distribution, e.g. by Variational inference.
- A large number of methods has been developed for specific tasks and situations!

USI/SUPSI
Istituto
Dalle Molle
di studi
sull'intelligenza
artificiale
IDSIA

Learning The Gaussian Distribution

The Gaussian Distribution

- We now perform calculations for the most important probability distribution – the Gaussian distribution.
- The density of the Gaussian is given by

$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right)$$

in the univariate case, and

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

in the multivariate case with D dimensions.

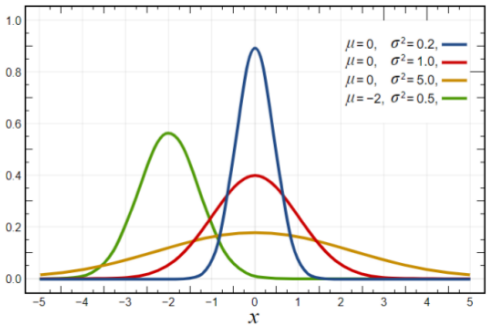


Image source: Wikipedia

The Gaussian Distribution: ML Estimate

- Consider the very common task of estimating a single Gaussian distribution from a set of iid data $\mathcal{D} = \{\mathbf{x}_n\}_{n=1,\ldots,N}$.
- The parameters to be estimated are the mean vector $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Sigma}$.
- We first derive the ML solution! The Likelihood function is

$$\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \prod_{n=1}^{N} \left[\frac{1}{(2\pi)^{D/2} |\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2} (\mathbf{x}_n - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}) \right) \right].$$

- For simplicity, we maximize the log-likelihood

$$\ell(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \left[-\log(2\pi)^{D/2} - \log(|\boldsymbol{\Sigma}|^{1/2}) - \frac{1}{2} (\mathbf{x}_n - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}) \right].$$

The Gaussian Distribution: ML Estimate

- As usual, we will compute the gradient of the log-likelihood and set it to zero.
- We use the simple rule $\frac{\mathrm{d}}{\mathrm{d}\mathbf{v}}\mathbf{v}^T A\mathbf{v} = (A^T + A)\mathbf{v}$, which reduces to $\frac{\mathrm{d}}{\mathrm{d}\mathbf{v}}\mathbf{v}^T A\mathbf{v} = 2A\mathbf{v}$ if A is symmetric. (Note that the covariance matrix and its inverse are symmetric.)
- The gradient of the log-likelihood w.r.t. $\boldsymbol{\mu}$ is

$$\frac{\mathrm{d}\ell}{\mathrm{d}\boldsymbol{\mu}} = -\sum_{n=1}^{N}\left[\boldsymbol{\Sigma}^{-1}(\mathbf{x}_n - \boldsymbol{\mu})\right].$$

  Solving $\frac{\mathrm{d}\ell}{\mathrm{d}\boldsymbol{\mu}} = 0$ yields, by canceling $\boldsymbol{\Sigma}^{-1}$:

$$\sum_{n=1}^{N}\mathbf{x}_n = N\cdot\boldsymbol{\mu} \Rightarrow \hat{\boldsymbol{\mu}}_{\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n.$$

- Thus the ML estimate for $\boldsymbol{\mu}$ is the sample average.

The Gaussian Distribution: ML Estimate

- The solution for $\mathbf{\Sigma}$ is a bit more involved. We limit ourselves to the one-dimensional case, for the full derivation see e.g. `https://stats.stackexchange.com/questions/351549/maximum-likelihood-estimators-multivariate-gaussian`.
- Assume μ is fixed. We have

$$\frac{\mathrm{d}\ell(\mu,\sigma)}{\mathrm{d}\sigma} = \frac{\mathrm{d}}{\mathrm{d}\sigma}\sum_{n=1}^{N}\left[-\log(\sqrt{2\pi}) - \log(\sigma) - \frac{1}{2}\frac{(x_n-\mu)^2}{\sigma^2}\right] = -\frac{N}{\sigma} + \sum_{n=1}^{N}\frac{(x_n-\mu)^2}{\sigma^3}.$$

- From the condition $0 = -\frac{N}{\sigma} + \sum_{n=1}^{N}\frac{(x_n-\mu)^2}{\sigma^3}$, we get by multiplying with σ^3:

$$N\sigma^2 = \sum_{n=1}^{N}(x_n-\mu)^2.$$

The Gaussian Distribution: ML Estimate

- Thus we get the ML estimate

$$\hat{\sigma}_{\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^{N}(x_n - \mu)^2$$

  where we can plug in $\mu = \hat{\mu}_{\mathrm{ML}}$ if μ is not known beforehand.
- In the D-dimensional case, the estimate is

$$\hat{\boldsymbol{\Sigma}}_{\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^{N}(x_n - \boldsymbol{\mu})(x_n - \boldsymbol{\mu})^T$$

  where we can likewise set $\boldsymbol{\mu} = \hat{\boldsymbol{\mu}}_{\mathrm{ML}}$.
- Note the similarity of the ML estimates of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ with the definitions of the mean and the variance!

The Gaussian Distribution: Bayesian Estimate

- We can also perform Bayesian reasoning on the Gaussian. In this lecture, we only consider the simplest case: Bayesian estimation of the mean vector in the one-dimensional case.
- So assume that σ is fixed. The likelihood function for a dataset $\mathcal{X} = \{x_1, \ldots, x_N\}$ is

$$\mathcal{L}(\mathcal{X}|\mu) = \prod_{n=1}^{N} \mathcal{N}(x|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{N/2}} \exp\left(-\frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n - \mu)^2\right).$$

- We assume a Gaussian prior on the mean value:

$$p(\mu) = \mathcal{N}(\mu|\mu_0, \sigma_0^2).$$

The Gaussian Distribution: Bayesian Estimate

- It turns out that the posterior distribution for μ is also Gaussian.
- Thus, the Gaussian is conjugate prior for the mean value of the Gaussian, provided that the variance is fixed (!).
- After observing N data points, the posterior distribution for μ is given by

$$p(\mu) = \mathcal{N}(\mu|\mu_N, \sigma_N^2)$$

with

$$\mu_N = \frac{\sigma^2\mu_0 + \sigma_0^2 N\mu_{\mathrm{ML}}}{\sigma^2 + N\sigma_0^2} \qquad \text{and} \qquad \frac{1}{\sigma_N^2} = \frac{1}{\sigma_0^2} + \frac{N}{\sigma^2}.$$

- Note that both Bayesian mean and variance interpolate between the prior and the ML solution!

The Gaussian Distribution: Conclusion

- We see that the Gaussian distribution allows exact Bayesian inference.
- Unfortunately, the Gaussian distribution is not very versatile when it comes to modeling real data sets...
- ...and more complex distributions do not usually allow this analytical treatment.
- In many cases, approximations involve modeling certain distributions as Gaussians!

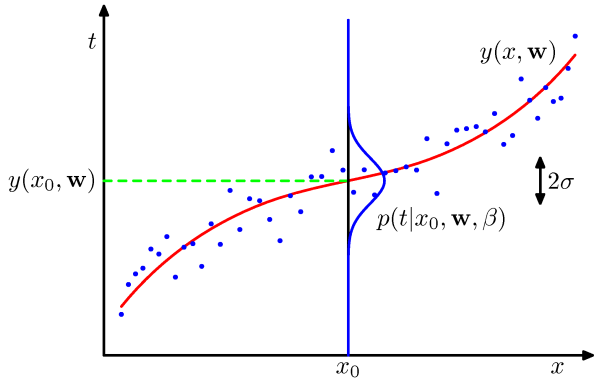
USI/SUPSI
Istituto
Dalle Molle
di studi
sull'intelligenza
artificiale
IDSIA

Linear Regression Revisited

Linear Regression: Setup

- Remember the *curve fitting* task with which we started this course?
- We will now reconsider it from a probabilistic perspective.
- Assume we have a set of N samples $\mathcal{X}$ and corresponding (scalar) targets $\mathcal{T}$. Our features are given as $\phi(\mathbf{x})$.
- We aim at performing linear regression: $t \approx \mathbf{w}^T\phi =: y(\phi, \mathbf{w})$.
- We express our uncertainty about the true target values probabilistically, specifically, we assume that t follows a Gaussian distribution with mean $y(\phi, \mathbf{w})$ and precision (inverse variance) β:

$$p(t|\phi, \mathbf{w}, \beta) = \mathcal{N}(t|y(\phi, \mathbf{w}), \beta^{-1}) \quad \text{with} \quad \beta^{-1} = \sigma^2.$$



Img src: Bishop, figure 1.16 (special case of polynomial fitting)

Linear Regression: Setup

- Our data model is

$$p(t|\phi, \mathbf{w}, \beta) = \mathcal{N}(t|y(\phi, \mathbf{w}), \beta^{-1})$$

  with $y(\phi, \mathbf{w}) = \mathbf{w}^T\phi$.
- Note that in this example we are interested in the conditional distribution $p(t|\phi)$.
- In practice, we do not actually know whether the noise is necessarily Gaussian, and neither do we know whether the relationship between features and targets is linear.
- But clearly, the better the model fits the data, the better will our results be.
- In the case of unknown noise, there is some theoretical justification (central limit theorem) to assume that the noise is approximately Gaussian.

Linear Regression: Maximum Likelihood Solution

- We compute the likelihood function:

$$L(\mathbf{w}, \beta) = p(\mathcal{T}|\mathcal{X}, \mathbf{w}, \beta) = \prod_{n=1}^{N} p(t_n|\phi_n, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|y(\phi_n, \mathbf{w}), \beta^{-1}).$$

- We assume the samples are iid (so we can multiply probabilities).
- We need to maximize the likelihood function. As in many cases, it is easier to maximize the log-likelihood instead:

$$\ell(\mathbf{w}, \beta) = \ln p(\mathcal{T}|\mathcal{X}, \mathbf{w}, \beta) = -\frac{\beta}{2}\sum_{n=1}^{N}(y(\phi_n, \mathbf{w}) - t_n)^2 + \frac{N}{2}\ln\beta - \frac{N}{2}\ln 2\pi.$$

Linear Regression: Maximum Likelihood Solution

- We compute the likelihood function:

$$L(\mathbf{w}, \beta) = p(\mathcal{T}|\mathcal{X}, \mathbf{w}, \beta) = \prod_{n=1}^{N} p(t_n|\phi_n, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|y(\phi_n, \mathbf{w}), \beta^{-1}).$$

- We assume the samples are iid (so we can multiply probabilities).
- We need to maximize the likelihood function. As in many cases, it is easier to maximize the log-likelihood instead:

$$\ell(\mathbf{w}, \beta) = \ln p(\mathcal{T}|\mathcal{X}, \mathbf{w}, \beta) = -\frac{\beta}{2}\sum_{n=1}^{N}(y(\phi_n, \mathbf{w}) - t_n)^2 + \frac{N}{2}\ln\beta - \frac{N}{2}\ln 2\pi.$$

- Assuming β is fixed, the last two terms are constant, and we arrive at the Squared Error which we know already. The solution is given by

$$\mathbf{w}_{\mathrm{ML}} = (\Phi^T\Phi)^{-1}\Phi^T\mathcal{T}$$

  with the *design matrix* Φ.
- The (Mean) Squared Error arises naturally when maximizing likelihood under the assumption of Gaussian noise.

Linear Regression: Maximum Likelihood Solution

- You can also determine β_{ML} by maximizing the log-likelihood function. The result is

$$\beta_{\text{ML}} = \left(\frac{1}{N} \sum_{n=1}^{N} (t_n - y(\phi_n, \mathbf{w}))^2 \right)^{-1}.$$

- This is just the ML estimate of the variance of t!
- Using the solution $\mathbf{w}_{\text{ML}}$, we can describe possible values of t probabilistically. The predictive distribution of t is found by plugging the ML solution into the likelihood function:

$$p(t|\phi, \mathbf{w}_{\text{ML}}, \beta_{\text{ML}}) = \mathcal{N}(t|y(\phi, \mathbf{w}_{\text{ML}}), \beta_{\text{ML}}^{-1}).$$

- But the reasoning is not foolproof. We remember that (in particular for small amounts of data) we may get drastic overfitting of the solution to the inherent noise in the data.
- Can one use Bayesian methods to avoid overfitting?

Bayesian Linear Regression

- We now create a Bayesian setup for curve fitting.
- The only parameter is the vector $\mathbf{w}$, we assume it follows a prior distribution

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|0, \alpha^{-1}I) = \left(\frac{\alpha}{2\pi}\right)^{(M+1)/2} e^{-(\alpha/2)\mathbf{w}^T\mathbf{w}}$$

  where I is the identity matrix, and M is the order of the fitting polynomial (then $\mathbf{w}$ has $M+1$ coefficients).
- α is a hyperparameter of the model.
- Why do we choose the prior distribution this way?
  - ➔ Mean 0 makes sense since the weights can have either sign.
  - ➔ Since all weights have equal probabilities, they should have the same variance.
  - ➔ Clearly, the weights should be independent from each other.
  - ➔ When you are not really sure, choose a Gaussian distribution...

Bayesian Linear Regression

- The data consists of samples $\mathcal{X}$ and targets $\mathcal{T}$.
- The posterior distribution is

$$
\begin{aligned}
p(\mathbf{w}|\mathcal{X},\mathcal{T},\alpha,\beta) &= \frac{p(\mathcal{X},\mathcal{T}|\mathbf{w},\alpha,\beta)p(\mathbf{w}|\alpha,\beta)}{p(\mathcal{X},\mathcal{T}|\alpha,\beta)} \\
&= \frac{p(\mathcal{T}|\mathcal{X},\mathbf{w},\alpha,\beta)p(\mathcal{X}|\mathbf{w},\alpha,\beta)p(\mathbf{w}|\alpha,\beta)}{p(\mathcal{X},\mathcal{T}|\alpha,\beta)} \\
&= \frac{p(\mathcal{T}|\mathcal{X},\mathbf{w},\beta)p(\mathcal{X})p(\mathbf{w}|\alpha)}{p(\mathcal{X},\mathcal{T}|\beta)}
\end{aligned}
$$

where in the last line conditional dependencies were removed where the conditioning variable and the conditioned variable are actually independent.

Bayesian Linear Regression

- The posterior distribution is

$$p(\mathbf{w}|\mathcal{X}, \mathcal{T}, \alpha, \beta) = \frac{p(\mathcal{T}|\mathcal{X}, \mathbf{w}, \beta)p(\mathcal{X})p(\mathbf{w}|\alpha)}{p(\mathcal{X}, \mathcal{T}|\beta)}$$

- We start by maximizing the posterior for $\mathbf{w}$, obtaining the MAP estimate. We do this by maximizing

$$\tilde{\mathcal{L}}(\mathbf{w}) = p(\mathcal{T}|\mathcal{X}, \mathbf{w}, \beta)p(\mathbf{w}|\alpha)$$

  (removing terms which do not depend on $\mathbf{w}$).
- The result can easily be computed and is given by the minimum of

$$\frac{\beta}{2}\sum_{n=1}^{N}(y(\phi_n, \mathbf{w}) - t_n)^2 + \frac{\alpha}{2}\mathbf{w}^T\mathbf{w}.$$

- Thus maximizing the posterior distribution is equivalent to minimizing the MSE with a *regularization term*, the influence of the regularization is given by the value of α.

Bayesian Linear Regression

- We have derived the predictive distribution for t in the ML case:

$$p(t|\phi, \mathbf{w}_{\text{ML}}, \beta_{\text{ML}}) = \mathcal{N}(t|y(\phi_n, \mathbf{w}_{\text{ML}}), \beta_{\text{ML}}^{-1})$$

- We can plug in the MAP estimates $\mathbf{w}_{\text{MAP}}$ and β_{MAP} just as for ML:

$$p(t|\phi, \mathbf{w}_{\text{MAP}}, \beta_{\text{MAP}}) = \mathcal{N}(t|y(\phi_n, \mathbf{w}_{\text{MAP}}), \beta_{\text{MAP}}^{-1})$$

- However in the Bayesian approach, we should integrate over *all* possible parameter values, weighted with their respective probabilities.
- Thus, the Bayesian predictive distribution has the form

$$p(t|\phi, \mathcal{X}, \mathcal{T}) = \int p(t|\phi, \mathbf{w}) p(\mathbf{w}|\mathcal{X}, \mathcal{T}) \, \mathrm{d}\mathbf{w}.$$

- In the special (Gaussian) case which we have considered so far, this integral can be solved analytically.

Gaussian Manipulations

- We need the following formulas for marginal and conditional Gaussian distributions, which can be derived by direct manipulation of the formula for the Gaussian distribution (Bishop, p.93).
- Assume

$$p(\mathbf{u}) = \mathcal{N}(\mathbf{u}|\mu, \Lambda^{-1})$$
$$p(\mathbf{v}|\mathbf{u}) = \mathcal{N}(\mathbf{v}|A\mu + b, L^{-1})$$

- Then

$$p(\mathbf{v}) = \mathcal{N}(\mathbf{v}|A\mu + b, L^{-1} + A\Lambda^{-1}A^T)$$
$$p(\mathbf{u}|\mathbf{v}) = \mathcal{N}(\mathbf{u}|\Sigma(A^T L(\mathbf{v} - b) + \Lambda\mu, \Sigma)$$

with $\Sigma = (\Lambda + A^T LA)^{-1}$.

Bayesian Linear Regression

- We use the formulas from the previous slide as follows:
  - $p(\mathbf{w})$ is Gaussian distributed (with mean 0 and precision $\alpha^{-1}I$)
  - $p(t|\mathbf{w}, \phi)$ is Gaussian distributed with mean $\phi^T\mathbf{w}$ and precision β^{-1}.
  - $p(\mathcal{T}|\mathbf{w}, \mathcal{X})$ is multivariate Gaussian distributed with mean $\Phi\mathbf{w}$ and precision $\beta^{-1}I$, where Φ is the design matrix.
- Thus, we can directly use the formulas from the previous slide to write

$$p(\mathbf{w}|\mathcal{T}, \mathcal{X}) = \mathcal{N}(\mathbf{w}|m_N, S_N)$$
$$S_N = \alpha I + \beta\Phi^T\Phi$$
$$m_N = \beta S_N \Phi^T \mathcal{T}.$$

- This is the formula for the Bayesian posterior distribution for $\mathbf{w}$, given all training data.
- (Note that we have refrained from estimating β.)

Bayesian Linear Regression

- The predictive distribution

$$p(t|\boldsymbol{\phi}, \mathcal{X}, \mathcal{T}, \alpha, \beta) = \int p(t|\boldsymbol{\phi}, \mathbf{w}, \beta) p(\mathbf{w}|\mathcal{X}, \mathcal{T}, \alpha, \beta) \mathrm{d}\mathbf{w}$$

  can be evaluated similarly. The result is

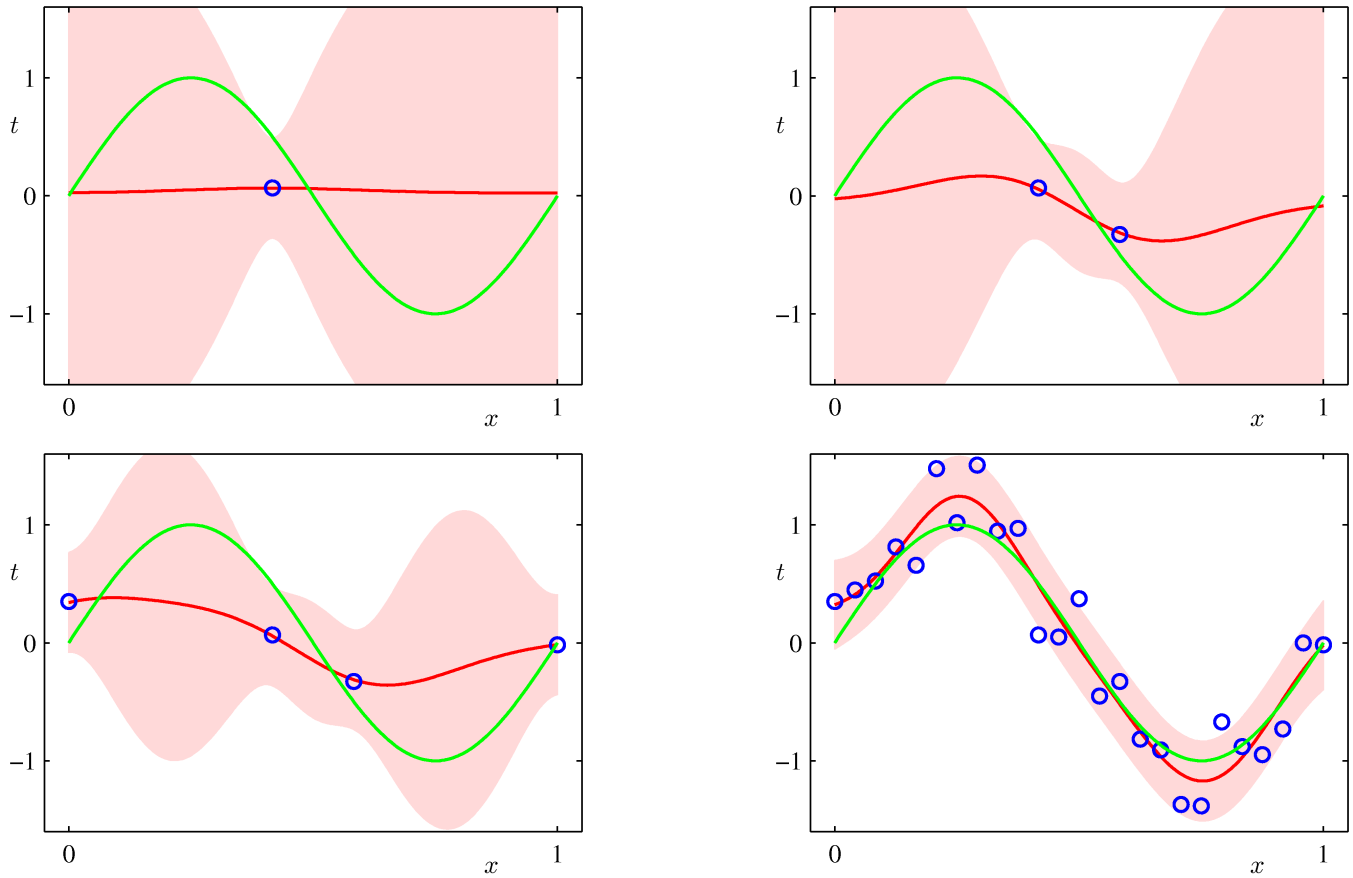
$$p(t|\boldsymbol{\phi}, \mathcal{X}, \mathcal{T}, \alpha, \beta) = \mathcal{N}(t|m_N \boldsymbol{\phi}, \sigma_N^2(\boldsymbol{\phi}))$$

  where the variance $\sigma_N^2(\boldsymbol{\phi})$ is given by

$$\sigma_N^2(\boldsymbol{\phi}) = \beta^{-1} + \boldsymbol{\phi}^T S_N \boldsymbol{\phi}.$$

- In particular, note that the variance of the prediction is given by two components: The inherent noise of the target (β^{-1}), and a term derived from the actual training data.

Bayesian Linear Regression



Examples for the predictive distribution, where the features are polynomial and the data is sampled from a sine wave. Shaded area indicates variance.

Image source: Bishop, fig. 3.8

USI/SUPSI
Istituto
Dalle Molle
di studi
sull'intelligenza
artificiale
IDSIA

Logistic Regression Revisited

Logistic Regression

- We have gotten to know logistic regression as a simple probabilistic model for *classification* (despite its name).
- Q: Under which conditions might such a simple model (remember that the decision boundary is linear) actually work well?

Logistic Regression

- We have gotten to know logistic regression as a simple probabilistic model for *classification* (despite its name).
- Q: Under which conditions might such a simple model (remember that the decision boundary is linear) actually work well?
- Assume there are two classes c_0, c_1, with targets $t = 0, 1$, respectively. We have annotated data points in form of features, i.e. $\phi_1, \ldots, \phi_N$.
- Using Bayes' formula, we get, for any data point ϕ:

$$p(c_0|\phi) = \frac{p(\phi|c_0)p(c_0)}{p(\phi|c_0)p(c_0) + p(\phi|c_1)p(c_1)} = \frac{1}{1 + e^{-a}} = \sigma(a)$$

with the logistic sigmoid σ and

$$a = \ln \frac{p(\phi|c_0)p(c_0)}{p(\phi|c_1)p(c_1)}.$$

(It is not difficult – try this at home.)

Logistic Regression

- Assume that the data for classes c_0 and c_1 is Gaussian distributed with the *same* covariance matrix Σ, and means $\boldsymbol{\mu}_0, \boldsymbol{\mu}_1$.
- The data model is

$$p(\boldsymbol{\phi}|c_k) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{\phi} - \boldsymbol{\mu}_k)^T \Sigma^{-1} (\boldsymbol{\phi} - \boldsymbol{\mu}_k)\right).$$

- Defining

$$\mathbf{w} = \Sigma^{-1}(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)$$
$$w_0 = -\frac{1}{2}\boldsymbol{\mu}_0^T \Sigma^{-1} \boldsymbol{\mu}_0 + \frac{1}{2}\boldsymbol{\mu}_1^T \Sigma^{-1} \boldsymbol{\mu}_1 + \ln \frac{p(c_0)}{p(c_1)}$$

  one obtains the surprising result

$$p(c_0|\boldsymbol{\phi}) = \sigma\left(\ln \frac{p(\boldsymbol{\phi}|c_0)p(c_0)}{p(\boldsymbol{\phi}|c_1)p(c_1)}\right) = \sigma(\mathbf{w}^T \boldsymbol{\phi} + w_0).$$

Logistic Regression

- Thus we have recovered the original definition of Logistic Regression by directly computing the probabilities for classes c_0 and c_1, under the assumption that the data for those classes is Gaussian distributed with identical covariance matrices.
- In this case, the decision boundary is linear.
- Note how the prior probabilities of the classes affect the bias w_0!
- The method does not work if the class covariances are different, or the distributions are not Gaussian (the decision boundaries might not be linear any more).
- How to compute the parameters for Logistic Regression? There is no closed-form solution (not even for the Maximum Likelihood case), one could for example use gradient descent to find an approximate solution, just as we have done for neural networks.

Logistic Regression - Conclusion

- Just as other linear models, the quality of logistic regression depends on the choice of features.
- Furthermore, it would be hard to verify that the class distributions have (roughly) the same covariance.
- The model is prone to *overfitting* if the classes are linearly separable (the sigmoid gets extremely steep, yielding degenerate probabilities). This can be avoided by regularization, or by fitting a prior to the weights. However, a full Bayesian treatment of Logistic Regression turns out to be intractable.
- Despite its simplicity, this model lies at the heart of the NN setup we have gotten to know for classification!

Advanced Topic: The Bias-Variance Decomposition

The Bias-Variance Decomposition

Probabilistic modeling can shed some light on the problem of overfitting.

- Assume that we have data $\mathcal{D} = \{(\mathbf{x}_n, t_n)\}_n$ which follow a joint probability distribution $p(\mathbf{x}, t)$.
- Here we study only the squared loss $L(t, y(\mathbf{x})) = (y(\mathbf{x}) - t)^2$ (for simplicity, we leave out the factor $\frac{1}{2}$).
- So far, we have considered specific algorithms and methods to minimize L. But what can we say *theoretically*, independent from any specific technique?

The Bias-Variance Decomposition

- We want to have a small loss on *unknown* (test) data which follows the distribution $p(\mathbf{x}, t)$, thus we aim at minimizing the expected value of the loss $E[L]$:

$$E[L] = \int (y(\mathbf{x}) - t)^2 p(\mathbf{x}, t)\, \mathrm{d}(\mathbf{x}, t).$$

- Using the *Calculus of Variations*, this can be formally differentiated and minimized. The optimal y is

$$y(\mathbf{x}) = E[t|\mathbf{x}].$$

- This is the *conditional expectation* of t, given $\mathbf{x}$. (This intuitive interpretation is one reason why we chose the MSE loss.)

The Bias-Variance Decomposition

- Then we still have to estimate $p(\mathbf{x}, t)$.
- If we had an unlimited supply of training data, we could estimate this distribution with arbitrary small error.
- Would this mean that we could exactly predict the target t for new data points $\mathbf{x}$?

The Bias-Variance Decomposition

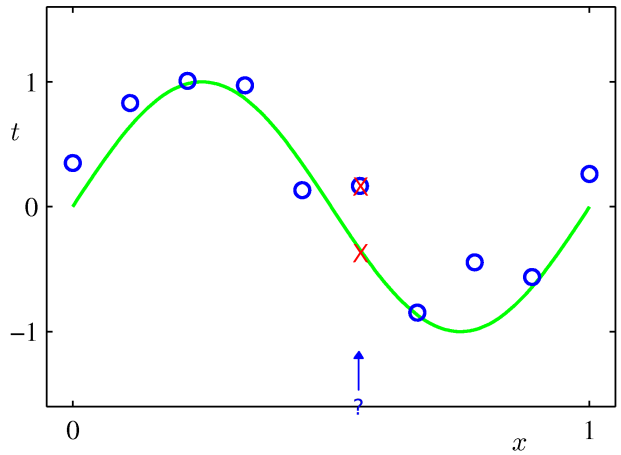
- Then we still have to estimate $p(\mathbf{x}, t)$.
- If we had an unlimited supply of training data, we could estimate this distribution with arbitrary small error.
- Would this mean that we could exactly predict the target t for new data points $\mathbf{x}$?
- **No**, since the data contains random noise!

The Bias-Variance Decomposition

- Then we still have to estimate $p(\mathbf{x}, t)$.
- If we had an unlimited supply of training data, we could estimate this distribution with arbitrary small error.
- Would this mean that we could exactly predict the target t for new data points $\mathbf{x}$?
- **No**, since the data contains random noise!
- This random noise represents an *irreducible error* which is a lower bound of the expected loss:

$$E[L] \geq \text{noise} \quad \text{with}$$

$$\text{noise} = \int (E[t|\mathbf{x}] - t)^2 p(\mathbf{x}, t)\, \mathrm{d}(\mathbf{x}, t).$$



Img src: Bishop, figure 1.2 (modified)

The Bias-Variance Decomposition

- Also, one cannot exactly estimate $E[t|\mathbf{x}]$ or $p(t|\mathbf{x})$ or $p(\mathbf{x}, t)$ from a finite amount of data.
- We saw that in practice, the optimal regression function is approximated with a *parametric* model.

The Bias-Variance Decomposition

- Also, one cannot exactly estimate $E[t|\mathbf{x}]$ or $p(t|\mathbf{x})$ or $p(\mathbf{x}, t)$ from a finite amount of data.
- We saw that in practice, the optimal regression function is approximated with a *parametric* model.
- Assume a concrete estimator y which computes an approximation for $E[t|\mathbf{x}]$, based on a dataset $\mathcal{D}$:

$$y = y(\mathbf{x}; \mathcal{D}).$$

- Clearly, different datasets $\mathcal{D}$ lead to different estimates.

The Bias-Variance Decomposition

- Also, one cannot exactly estimate $E[t|\mathbf{x}]$ or $p(t|\mathbf{x})$ or $p(\mathbf{x}, t)$ from a finite amount of data.
- We saw that in practice, the optimal regression function is approximated with a *parametric* model.
- Assume a concrete estimator y which computes an approximation for $E[t|\mathbf{x}]$, based on a dataset $\mathcal{D}$:

$$y = y(\mathbf{x}; \mathcal{D}).$$

- Clearly, different datasets $\mathcal{D}$ lead to different estimates.
- You can imagine that the dataset $\mathcal{D}$ is itself random - a random sample from $p(\mathbf{x}, t)$.
- Then it makes sense to talk about the *expected value* of the function y:

$$E_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})].$$

The Bias-Variance Decomposition

- The expected loss can be decomposed in the following way:

$$E[L] = (\text{bias})^2 + \text{variance} + \text{noise}$$

with

$$(\text{bias})^2 = \int \left(E_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] - E[t|\mathbf{x}]\right)^2 p(\mathbf{x})\,\mathrm{d}\mathbf{x}$$

$$\text{variance} = \int E_{\mathcal{D}}\left[\left(y(\mathbf{x};\mathcal{D}) - E_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})]\right)^2\right] p(\mathbf{x})\,\mathrm{d}\mathbf{x}$$

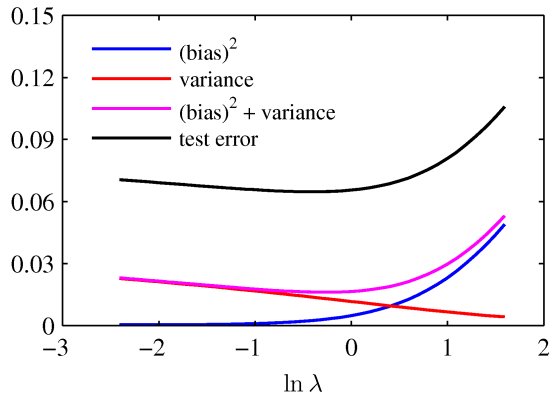
$$\text{noise} = \int \left(E[t|\mathbf{x}] - t\right)^2 p(\mathbf{x}, t)\,\mathrm{d}(\mathbf{x}, t).$$

- The bias indicates how much the estimate for $E[t|\mathbf{x}]$ will differ from the actual $E[t|\mathbf{x}]$, the variance indicates how much the estimate for $E[t|\mathbf{x}]$ depends on the dataset $\mathcal{D}$.
- (For details on the derivation, we refer to Bishop's book.)

The Bias-Variance Decomposition

- Since we cannot do anything about the noise, we would like to minimize the bias and the variance.
- Unfortunately, there is often a tradeoff: Reducing one of them increases the other.
- In certain cases, the optimal tradeoff of the sum $(\text{bias})^2 + \text{variance}$ leads to an optimal estimate y.

- The plot shows bias, variance, and test error for estimates of the noisy regression task, with a regularization parameter λ in ridge regresssion.
- Stronger regularization leads to lower variance and higher bias, and vice versa.
- The sum $(\text{bias})^2 + \text{variance}$ reaches its minimum where the test error reaches its minimum.



Img src: Bishop, figure 3.6

The Bias-Variance Decomposition

- The bias-variance decomposition is, unfortunately, not practically useful.
- This is because we cannot estimate the expectations over many datasets $\mathcal{D}$, since there is only one dataset.
- (In case there is a large amount of data, we want to use *all* of it for training.)
- Still, it paves the way to a probabilistic (Bayesian) treatment of parameter estimation.

$\Rightarrow$ You should remember that probabilistic analysis can be a *powerful tool* to understand the quality of a model!

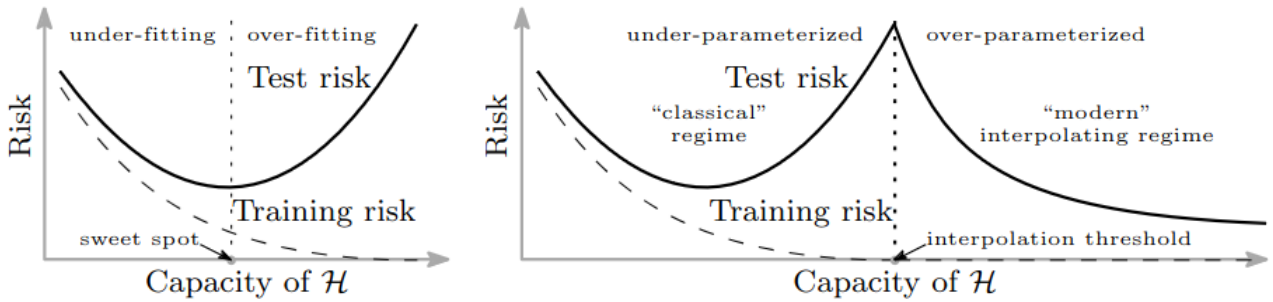
The Overparametrized Regime

- In recent years, another phenomenon has gained much attention: **For models with a very large number of parameters, the classical bias-variance tradeoff does not hold!**
- In "classical" Machine Learning, one usually assumes relatively small models (measured in number of parameters).
- Modern neural networks have a very large number of parameters (i.e. weights), often far surpassing the number of training data samples!
- These networks have very surprising properties:
  - ➔ They *interpolate* the training data (zero training loss).
  - ➔ They do so even if the labels are completely random, i.e. they are powerful enough to "memorize" large datasets.
  - ➔ Of course, a network trained on random labels cannot generalize, but on true labels, large networks generalize exceedingly well.
  - ➔ Even when zero training error is attained, adding more parameters or training longer often improves generalization!
- This situation is called the overparametrized regime.

The Overparametrized Regime

- A major result in analyzing the overparametrized regime: Double Descent.
- When the hypothesis class $\mathcal{H}$ is large enough to interpolate the data, increasing its size *improves* generalization.
- This is true for large neural networks, but also for other methods (even linear methods / kernel methods).
- The exact behavior depends not only on $\mathcal{H}$ itself, but also on the specific setup of the training process!
- Many training methods have an *implicit bias* towards well-generalizing solutions.

$\Rightarrow$ Many questions about the NN learning process are still open and work in progress!



Img src: Belkin, Reconciling modern machine learning practice and the bias-variance trade-off

USI/SUPSI
Istituto
Dalle Molle
di studi
sull'intelligenza
artificiale
IDSIA

Gaussian Mixture Models, Latent Variables, and EM Training

Overview

- In this section, we talk about data models based on Gaussian distributions.
- These can be used for
  - ➔ unsupervised learning: understand the structure of data, generate new samples
  - ➔ supervised learning: classification and regression
  - ➔ as part of a larger probabilistic framework.

Overview

- In this section, we talk about data models based on Gaussian distributions.
- These can be used for
  - ➔ unsupervised learning: understand the structure of data, generate new samples
  - ➔ supervised learning: classification and regression
  - ➔ as part of a larger probabilistic framework.
- Gaussian *Mixture* models are a versatile extension of the single Gaussian.
- They are probabilistically described using *hidden* or *latent* variables, which also play a role in other architectures.
- The EM algorithm is an important method for training hidden variable models.

Modeling Data with a Gaussian Distribution

- Assume we have a set of samples $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$.
- How do we model these samples with a Gaussian distribution?

Modeling Data with a Gaussian Distribution

- Assume we have a set of samples $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$.
- How do we model these samples with a Gaussian distribution?
- We here consider a *Maximum Likelihood* approach!
- The model is

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

and we need to find the values for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ which maximize the likelihood of the data:

$$\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \prod_{n=1}^{N} \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}, \boldsymbol{\Sigma}).$$

Modeling Data with a Gaussian Distribution

- We have already seen how to equivalently maximize the log-likelihood

$$\ell(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = -\frac{1}{2}\sum_{n=1}^{N}(\mathbf{x}_n - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}_n - \boldsymbol{\mu}) + \text{constants.}$$

- The maximum is given by

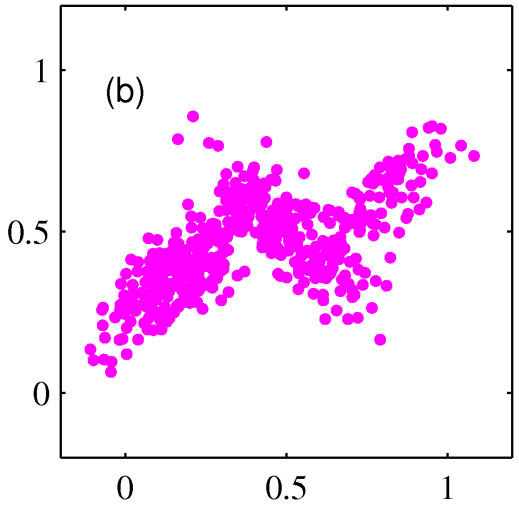
$$\boldsymbol{\mu}_{\text{ML}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$$

$$\boldsymbol{\Sigma}_{\text{ML}} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \boldsymbol{\mu}_{\text{ML}})(\mathbf{x}_n - \boldsymbol{\mu}_{\text{ML}})^T = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n\mathbf{x}_n^T - \left(\frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n\right)\left(\frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n\right)^T.$$

- These are the sample mean and covariance, respectively.

Gaussian Mixtures

- Clearly, there is a problem: Not all distributions can be reasonably modeled by a Gaussian!
- Consider the example to the right. Fitting a Gaussian to this data is obviously a bad model.
- Intuitively, the data given above could be modeled by multiple Gaussians.



Img src: Bishop, figure 9.5b

- Indeed, one can show[2] that *any* reasonable probability distribution can be approximated by a *weighted sum* of Gaussians.

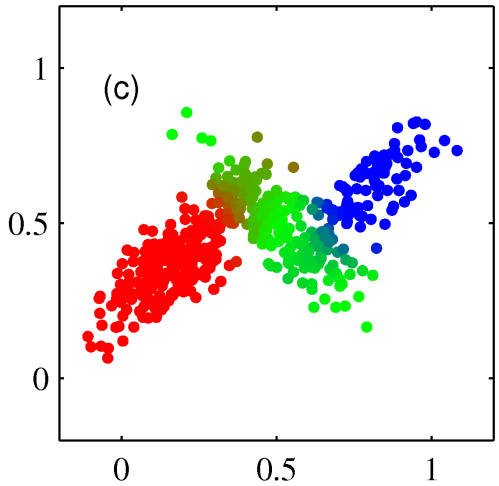
[2] e.g. Bacharoglou: *Approximation of Probability Distributions by Convex Mixtures of Gaussian Measures*, Proc. AMS 138(7), 2010

Gaussian Mixtures

- This gives rise to the Gaussian Mixture Model (GMM), a very important classical probabilistic model.
- We model our data as a weighted sum of (finitely many) Gaussians:

$$p(\mathbf{x}) = \sum_{m=1}^{M} w_m \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m) \quad \text{with} \quad \sum_{m=1}^{M} w_m = 1.$$

- Computing the probability of a data point (i.e. evaluating the model) is straightforward.



Img src: Bishop, figure 9.5c

Training Gaussian Mixture Models

- With N samples $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, the log-likelihood function of the model becomes

$$\ell(\mu, \Sigma) = \sum_{n=1}^{N} \ln \left[\sum_{m=1}^{M} w_m \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m) \right].$$

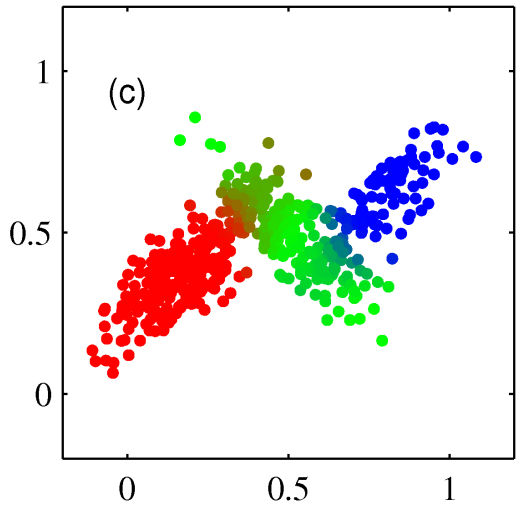
- Unfortunately, there is *no known way* to analytically maximize this function over the $w_m, \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m$ simultaneously.

Training Gaussian Mixture Models

Some thoughts about maximizing the likelihood of the Gaussian Mixture Model:

- Clearly, we wish to estimate each component Gaussian from a subset of the available samples (we know how that is done).
- Subsets might be formed by hard assignments, but we could also think of a weighted assignment of any sample to several component Gaussians.

- The assignment of samples to component Gaussians (which we also used during sampling) is a **fundamental model assumption** of the Gaussian Mixture Model.



Img src: Bishop, figure 9.5c

The EM Algorithm for Gaussians

- Idea: *Alternatingly* reestimate the assignment of samples to component Gaussians, and the parameters of the Gaussians!
- This is the Expectation Maximization algorithm for Gaussians (we will soon see where the name comes from).
- Definition: Let $\gamma_{n,m}$ the contribution of sample n to Gaussian m, where $\sum_m \gamma_{n,m} = 1$.
- We can calculate and interpret these contributions as probabilities, more exactly, $\gamma_{n,m}$ is the *posterior* probability that sample n was generated by component m.
- We call $\gamma_{n,m}$ assignment probabilities.

The EM Algorithm for Gaussians

1. Initialize randomly the assignment probabilities $\gamma_{n,m}$, the Gaussian parameters $\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m$, and the mixture weights w_m.
2. E-step – recompute all $\gamma_{n,m}$ by evaluating the Gaussian distributions and normalizing:

$$\gamma_{n,m} = \frac{w_m \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)}{\sum_{\ell=1}^{M} w_\ell \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_\ell, \boldsymbol{\Sigma}_\ell)}.$$

3. M-step – update the Gaussian and mixture parameters:

$$w_m = \frac{1}{N} \sum_{n=1}^{N} \gamma_{n,m} \qquad \boldsymbol{\mu}_m = \frac{1}{\sum_n \gamma_{n,m}} \sum_{n=1}^{N} \gamma_{n,m} \mathbf{x}_n$$

$$\boldsymbol{\Sigma}_m = \frac{1}{\sum_n \gamma_{n,m}} \sum_{n=1}^{N} \gamma_{n,m} (\mathbf{x}_n - \boldsymbol{\mu}_m)(\mathbf{x}_n - \boldsymbol{\mu}_m)^T.$$

4. Repeat steps 2 and 3 until satisfied (usually, until the likelihood of the data does not increase very much any more).

The EM Algorithm for Gaussians

- We have traded the simplicity of an analytic solution for a rather restrictive model for an *iterative* approach to train a far more powerful model.
- As with iterative training of neural networks, we need a criterion to determine when to stop the iteration.
- Usually, one considers the (log-)likelihood of the training or validation data and stops training when it appears to converge.
- The EM algorithm for Gaussian mixtures normally converges quite fast.
- In contrast to NN training, each step (E or M) *always increases* the likelihood of the data!

The EM Algorithm for Gaussians

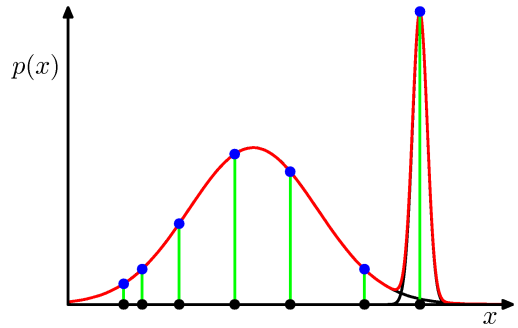
- In specific cases, training may fail; in particular, if the data is in a low-dimensional subspace, the variance of the data approaches zero in one more directions, leading to degenerate distributions.

$\Rightarrow$ Apply *dimensionality reduction* to the features.

- This can also occur if only few samples are assigned (with high γ) to a component: The component becomes "sharper" (small variance in many directions), which causes even less samples to be assigned to this component, which in turn increases the problem.

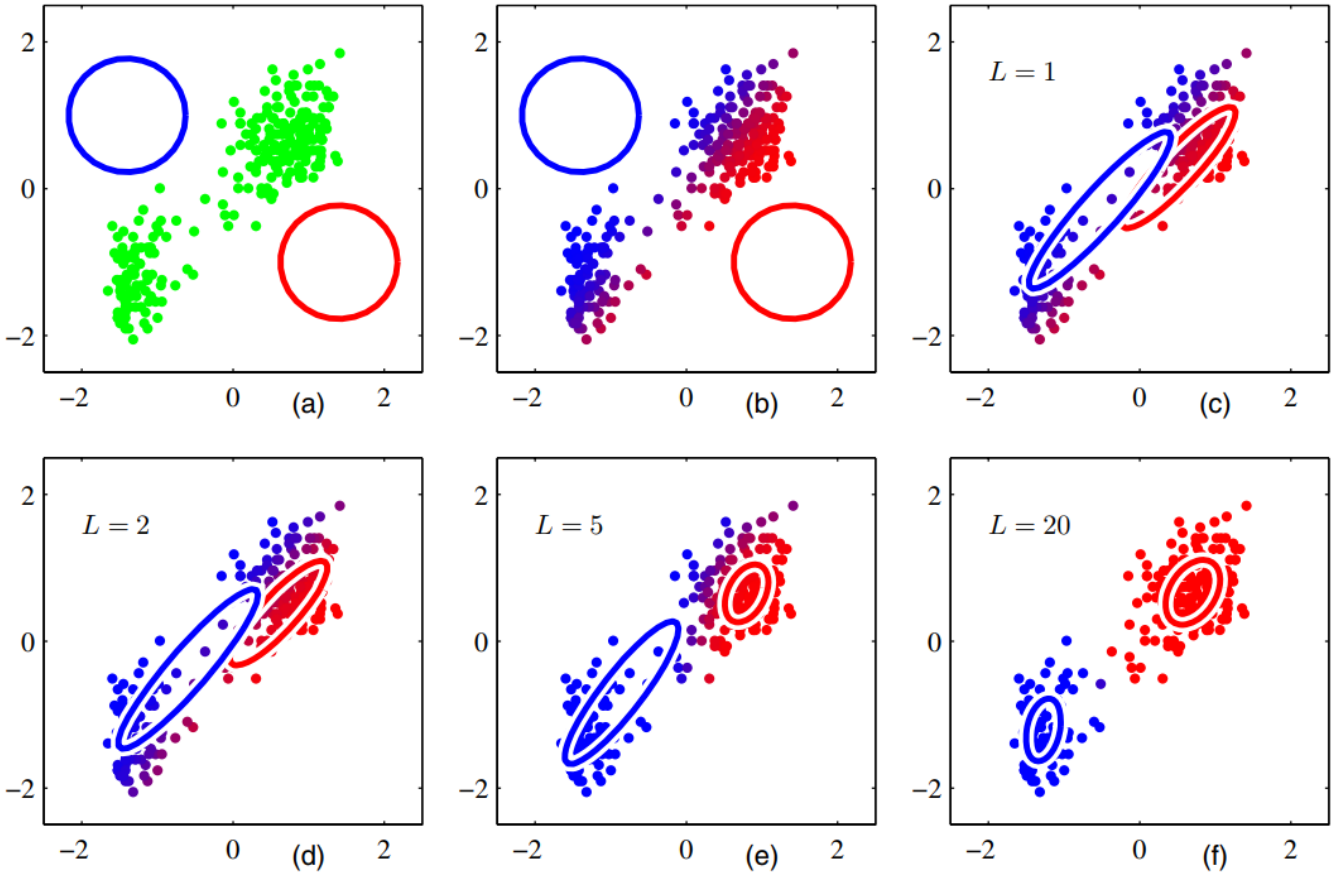
$\Rightarrow$ One usually deals with such situations heuristically (e.g. delete a Gaussian if its combined assignment probabilities become too small).

$\Rightarrow$ Alternatively, this kind of overfitting can be avoided with a Bayesian setup of the GMM training.



Img src: Bishop, figure 9.7

The EM Algorithm for Gaussians



EM for two Gaussians in two dimensions. Note that the Gaussians start to take a reasonable shape after just 5 iterations.

K-Means

- We introduce K-Means as a simplification of the EM algorithm for Gaussians.
- Assume that the covariance matrices Σ_k are fixed to the identity matrix - we only train the *means* of the Gaussians.
- We furthermore assume that assignments of data points are hard, i.e. $\gamma_{n,k} \in 0, 1$ (this does not mean that it cannot change!).
- This yields a straightforward *clustering* method, i.e. data points are assigned to K clusters.
- The objective measure is

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_{n,k} ||\mathbf{x}_n - \boldsymbol{\mu}_k||^2$$

with $\gamma_{n,k} \in 0, 1$.

K-Means

- This task can be solved in exactly the same way as for the EM algorithm, but with less computation (since we do not need to do all the matrix multiplications):
  - ➔ Initialize the cluster means randomly.
  - ➔ *Assignment step*: assign each data point to the cluster whose mean is nearest, i.e.

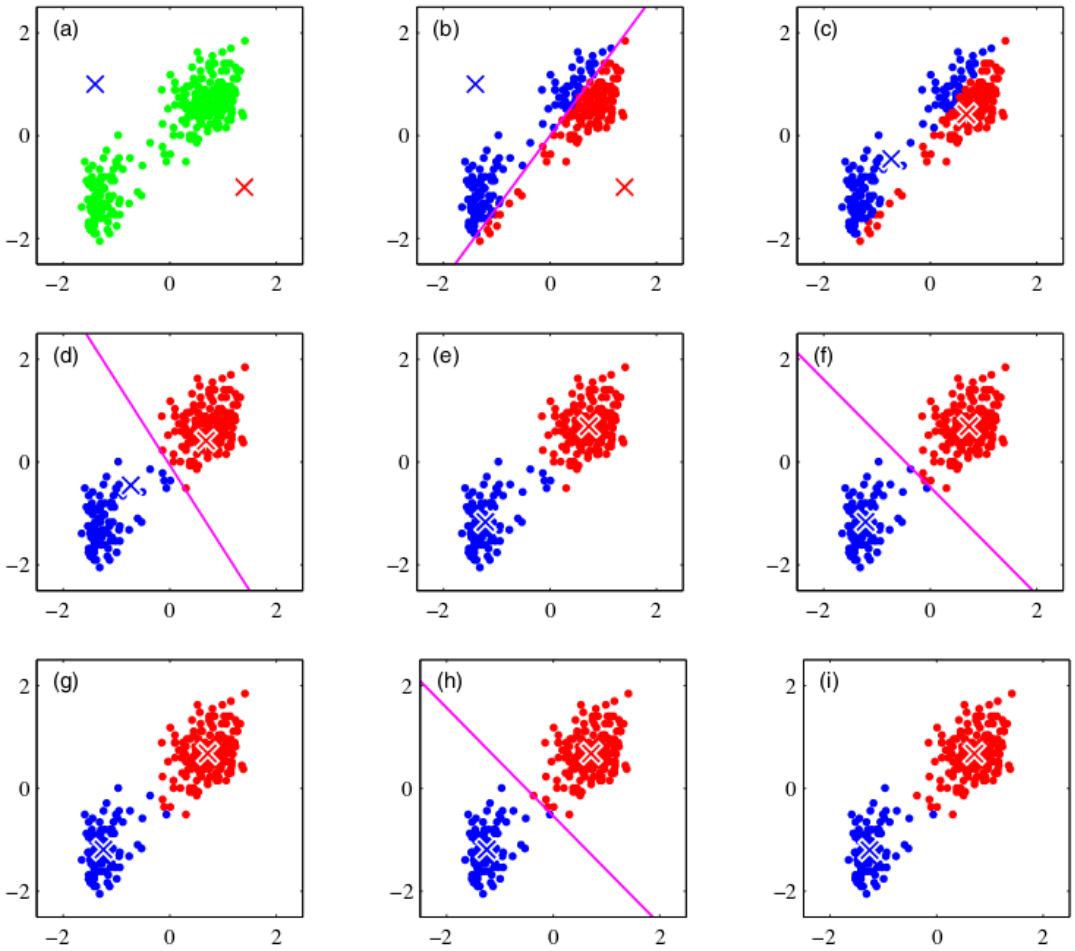
$$\gamma_{n,k} = \begin{cases} 1 & \text{if } k = \arg\min_j ||\mathbf{x}_n - \boldsymbol{\mu}_j||^2 \\ 0 & \text{otherwise.} \end{cases}$$

  - ➔ *Update step*: Recompute the means of each cluster by

$$\boldsymbol{\mu}_k = \frac{1}{S_k} \sum_{n=1}^{N} \gamma_{n,k} \mathbf{x}_n,$$

    where $S_k = \sum_{n=1}^{N} \gamma_{n,k}$ is the number of data points assigned to cluster k.
  - ➔ Run the algorithm until the assignments do not change any more, or until a maximum number of iterations is reached.

K-Means



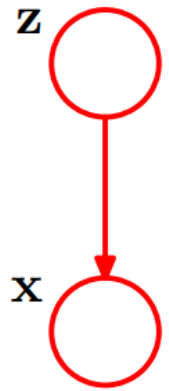
K-Means with two clusters. The pink line indicates the cluster boundary.

EM Algorithm: Remarks

- The EM algorithm can be used to train a variety of models, including linear regression models and sequential models (Hidden Markov Model, HMM).
- While it tends to require less computation than training a complex neural network, it makes still sense to optimize it.
- A very common optimization in the case of Gaussian mixtures is to *pretrain* a clustering using k-means, and then use the cluster means to initialize the Gaussian means for the EM algorithm.
- How to determine the optimal number of Gaussians? Some methods have been proposed, e.g. the *Split-and-Merge* algorithm (Ueda et al, 2000).

Latent Variables and the EM Algorithm

- In order to better understand the EM algorithm and its convergence properties, we describe the GMM in a new way: We formalize the concept of assigning samples to component Gaussians probabilistically.
- Assume that each sample $\mathbf{x}_n$ is created by exactly one component Gaussian m_n. We denote this assignment with an m-dimensional *one-hot* vector $\mathbf{z}_n = (z_{n,m})_m$, which has a 1 at position m_n, and zeros otherwise.



Img src: Bishop, figure 9.4

- We do **not** know the "correct" values of the $z_{n,m}$. We call such variables latent (or hidden) variables.
- Clearly, if we *knew* the values of the latent variables, we could easily compute the ML solution for all the Gaussian and mixture parameters.
- In understanding the EM algorithm, latent variables play a central role!

- Assuming that the latent variables $\mathcal{Z} = \{z_{n,m}\}$ are known, we would have to maximize

$$\ln p(\mathcal{X}, \mathcal{Z}|\theta),$$

for θ, where θ is the set of all Gaussian and mixture parameters $(\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m, w_m)$.

- We assume that this maximization is straightforward (we have seen it is in the case of Gaussians).
- Since we do not know the correct values of $\mathcal{Z}$, we have to maximize instead over the sum of the marginals

$$\ln p(\mathcal{X}|\theta) = \ln \sum_{\mathcal{Z}} p(\mathcal{X}, \mathcal{Z}|\theta),$$

which is usually not tractable.

- We *replace* the likelihood of the complete dataset $\ln p(\mathcal{X}, \mathcal{Z}|\theta)$ by its expected value under the posterior distribution $p(\mathcal{Z}|\mathcal{X}, \theta)$.

- This yields the abstract form of the EM algorithm.
- In the Expectation (E) step, we compute the posterior distribution of the latent variables:

$$p(\mathcal{Z}|\mathcal{X}, \theta^{\text{old}}),$$

thus defining an auxiliary value

$$\mathcal{Q}(\theta, \theta^{\text{old}}) = \sum_{\mathcal{Z}} p(\mathcal{Z}|\mathcal{X}, \theta^{\text{old}}) \ln p(\mathcal{X}, \mathcal{Z}|\theta).$$

Remember that in the case of the Gaussian, the latent variables are the component assignments $z_{n,m}$.

- In the Maximization (M) step, we maximize the function $\mathcal{Q}$ - the expected log-likelihood of the full data set:

$$\arg\max_{\theta} \mathcal{Q}(\theta, \theta^{\text{old}}).$$

Latent Variables and the EM Algorithm

- It can be shown (see e.g. Bishop, chapter 9.4) that these steps a) always increase the data likelihood, b) converge to an optimum.
- We see that in the case of estimating the ML solution for the GMM model,
  - ➔ $\mathcal{Z}$ contains the sample assignments, which are latent
  - ➔ the E step corresponds to computing the assignment probabilities - the expected value of $\mathcal{Z}$
  - ➔ the M step corresponds to recomputing the Gaussian parameters, which are subsumed in the parameter vector θ.

Gaussian Mixtures: Applications

- We have seen that the GMM can be used to describe the structure of the data with a small set of parameters ($\boldsymbol{\mu}_m$, $\boldsymbol{\Sigma}_m$, w_m).
- This falls into the category of Unsupervised Learning, which we do not cover here.
- We now consider how to use GMMs in supervised applications, namely classification and regression.

Gaussian Mixtures: Regression

- Goal: supervised regression from multidimensional inputs $\mathbf{x} \in \mathbb{R}^N$ to multidimensional targets $\mathbf{t} \in \mathbb{R}^K$.
- Train a *joint* GMM on the combined observations

$$\begin{pmatrix} \mathbf{x} \\ \mathbf{t} \end{pmatrix},$$

which yields a joint model with component means and covariances structured like this:

$$\mu = \left(\begin{array}{c} \boldsymbol{\mu}^{\mathbf{x}} \\ \hline \boldsymbol{\mu}^{\mathbf{t}} \end{array}\right) \qquad \boldsymbol{\Sigma} = \left(\begin{array}{c|c} \boldsymbol{\Sigma}^{\mathbf{xx}} & \boldsymbol{\Sigma}^{\mathbf{tx}} \\ \hline \boldsymbol{\Sigma}^{\mathbf{xt}} & \boldsymbol{\Sigma}^{\mathbf{tt}} \end{array}\right)$$

(note that $\boldsymbol{\Sigma}^{\mathbf{xt}} = (\boldsymbol{\Sigma}^{\mathbf{tx}})^T$).

- You can easily reduce this model to a model for $\mathbf{x}$ only by considering the relevant parts of the mean vectors and covariance matrices ($\boldsymbol{\mu}^{\mathbf{x}}$, $\boldsymbol{\Sigma}^{\mathbf{xx}}$), for each component Gaussian, respectively.

Gaussian Mixtures: Regression

- Regression for a new value $\mathbf{x}_0$ is performed by
  - ➔ computing weights w_m, using *only* the model for $\mathbf{x}$, as

$$w_m = \frac{\mathcal{N}_m^{\mathbf{x}}(\mathbf{x}_0)}{\sum_{\ell=1}^{M} \mathcal{N}_\ell^{\mathbf{x}}(\mathbf{x}_0)}$$

  - ➔ computing the w_m-weighted average of the expected values of the *conditional* distributions of $\mathbf{t}$ given $\mathbf{x}$:

$$\hat{\mathbf{t}} = \sum_{m=1}^{M} w_m E[\mathbf{t}|\mathbf{x} = \mathbf{x}_0, \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m]$$

    where the conditional probabilities of the components take the form of Gaussians with expected values

$$E[\mathbf{t}|\mathbf{x} = \mathbf{x}_0, \boldsymbol{\mu}, \boldsymbol{\Sigma}] = \boldsymbol{\mu}^{\mathbf{t}} + \boldsymbol{\Sigma}^{\mathbf{xt}}(\boldsymbol{\Sigma}^{\mathbf{tt}})^{-1}(\mathbf{x}_0 - \boldsymbol{\mu}^{\mathbf{x}}).$$

Gaussian Mixtures: Classification

- For *classification*, one trains separate GMMs for each class c:

$$p^c(\mathbf{x}) = \sum_{m=1}^{M} w_m^c \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_m^c, \boldsymbol{\Sigma}_m^c).$$

- Classification for a new data point $\mathbf{x}_0$ is performed by computing its probabilities for all classes, and then taking the class with maximal probability:

$$\hat{c} = \arg\max_c p^c(\mathbf{x}_0).$$

- You can also obtain a probability distribution over classes by taking the p^c and normalizing ($P(c) = \frac{p^c(\mathbf{x}_0)}{\sum_\gamma p^\gamma(\mathbf{x}_0)}$), or (much better) with a Bayesian approach:

$$P(c|\mathbf{x}_0) = \frac{P(\mathbf{x}_0|c) \cdot P(c)}{P(\mathbf{x}_0)} = \frac{p^c(\mathbf{x}_0) \cdot P(c)}{P(\mathbf{x}_0)},$$

where $P(c)$ is the prior probability of class c (may be uniform), and $P(x)$ (the prior probability of the data) amounts to a normalization factor which ensures that the probabilities sum up to 1.

Gaussian Mixtures: Classification

- If you use the GMM for classification, note that there is *no guarantee* that the class boundaries are modeled very well.
- This is due to the fact that the classwise models are trained without considering the distributions of the other classes.
- Many methods to remedy this problem have been proposed; unfortunately they are much less elegant than the basic method.
- This is a fundamental drawback of GMM-style classification, and a fundamental advantage of the more flexible neural network (which learns the actual task, e.g. classification, not a surrogate).

Final Remarks

- The idea of mixing probabilistic models can also be generalized to other distributions (example: discrete *Bernoulli* distributions).
- The concept of latent variables goes far beyond mixtures of distributions. In particular, one can model the alignment of sequential data by assuming that the *alignment* is a latent variable (Hidden Markov model).
- This yields elegant solutions to the problem of training and inference when the alignment of data and classes is not known.
- In this (and other) cases, latent variables represent hidden *causes* of observations.
- Note that the basic EM algorithm (for Gaussians, and for other models) computes maximum likelihood solutions, with all their problems (e.g. overfitting).

USI/SUPSI
Istituto
Dalle Molle
di studi
sull'intelligenza
artificiale
IDSIA

Advanced Topic: Variational Inference

Variational Inference

- In this last section, we will present an advanced method to approximate Bayesian integrals when they cannot be computed analytically.
- Idea: A complex distribution is approximated with a simpler distribution which allows to analytically compute, or to efficiently approximate, the desired result.
- This sounds straightforward, but it means that we have to find a way to formally approximate one probability distribution with another.

Definition: Kullback-Leibler Divergence

- For continuous random variables P and Q with distributions p and q, define the Kullback-Leibler Divergence (KLD)

$$D_{\mathrm{KL}}(P||Q) = \int p(x) \log \left(\frac{p(x)}{q(x)} \right) \mathrm{d}x = E_{x \sim p} \log \left(\frac{p(x)}{q(x)} \right).$$

- The definition also works for discrete random variables, replacing the density with the distribution function and the integral by the sum:

$$D_{\mathrm{KL}}(P||Q) = \sum_x p(x) \log \left(\frac{p(x)}{q(x)} \right) = E_{x \sim p} \log \left(\frac{p(x)}{q(x)} \right).$$

- The distributions can be one-dimensional or multivariate.
- The KLD is *not* a metric in the classical sense (it is not symmetric). It is, however, always nonnegative, and $D_{\mathrm{KL}}(P||Q) = 0$ when the distributions are equal (in a probabilistic sense).

S. Kullback and R. A. Leibler: *On Information and Sufficiency*. Annals of Mathematical Statistics 22(1): 79 - 86, 1951

Definition: Kullback-Leibler Divergence

- In the context of machine learning, the KLD can be interpreted as the information gain when transiting from a prior distribution P to a posterior distribution Q.
- We will mostly use the KLD as a measure for the (nonsymmetric) difference between two probability distributions.
- We follow the notation of Bishop, not strictly distinguishing random variables and their densities:

$$D_{\mathrm{KL}}(p||q) = \int p(\mathbf{X}) \log \left(\frac{p(\mathbf{X})}{q(\mathbf{X})} \right) \mathrm{d}\mathbf{X} = E_{\mathbf{X}\sim p} \log \left(\frac{p(\mathbf{X})}{q(\mathbf{X})} \right).$$

Approximating the Evidence

- Consider again the generic Bayesian setup $p(\mathbf{Z}|\mathbf{X}) = \frac{p(\mathbf{X}|\mathbf{Z})p(\mathbf{Z})}{p(\mathbf{X})}$ and remember that typically, $p(\mathbf{X})$ and $p(\mathbf{Z}|\mathbf{X})$ are intractable.
- Goal: approximate $p(\mathbf{Z}|\mathbf{X})$ with a tractable distribution $q(\mathbf{Z})$.
- From $p(\mathbf{Z}|\mathbf{X}) = \frac{p(\mathbf{Z},\mathbf{X})}{p(\mathbf{X})}$, we obtain

$$
\begin{aligned}
D_{\mathrm{KL}}(q||p) &= \int q(\mathbf{Z}) \log \left(\frac{q(\mathbf{Z})}{p(\mathbf{Z}|\mathbf{X})} \right) \mathrm{d}\mathbf{Z} = \int q(\mathbf{Z}) \log \left(\frac{q(\mathbf{Z})p(\mathbf{X})}{p(\mathbf{Z},\mathbf{X})} \right) \mathrm{d}\mathbf{Z} \\
&= \int q(\mathbf{Z}) \left[\log q(\mathbf{Z}) - \log p(\mathbf{Z},\mathbf{X}) + \log p(\mathbf{X}) \right] \mathrm{d}\mathbf{Z} \\
&= \int q(\mathbf{Z}) \left[\log q(\mathbf{Z}) - \log p(\mathbf{Z},\mathbf{X}) \right] \mathrm{d}\mathbf{Z} + \int q(\mathbf{Z}) \log p(\mathbf{X}) \, \mathrm{d}\mathbf{Z} \\
&= -\mathcal{L}(q) + \log p(\mathbf{X})
\end{aligned}
$$

with

$$
\mathcal{L}(q) = \int q(\mathbf{Z}) \left[\log p(\mathbf{Z},\mathbf{X}) - \log q(\mathbf{Z}) \right] \mathrm{d}\mathbf{Z} = E_{\mathbf{Z}\sim q} \left[\log p(\mathbf{Z},\mathbf{X}) - \log q(\mathbf{Z}) \right].
$$

Approximating the Evidence

- What have we gained by decomposing $D_{\mathrm{KL}}(q||p) = -\mathcal{L}(q) + \log p(\mathbf{X})$?
- Writing

$$\log p(\mathbf{X}) = \underbrace{D_{\mathrm{KL}}(q||p)}_{\geq 0} + \mathcal{L}(q),$$

  we have derived that $\mathcal{L}(q)$ is a *lower bound* for the logarithm of the evidence $p(\mathbf{X})$!
- Consequently, $\mathcal{L}(q)$ is called the Evidence Lower Bound (ELBO).
- Furthermore, the left-hand side of the equation does not depend on q. Thus, minimizing $D_{\mathrm{KL}}(q||p) = D_{\mathrm{KL}}(q(\mathbf{Z})||p(\mathbf{Z}|\mathbf{X}))$ over a family of distributions q is equivalent to maximizing $\mathcal{L}(q)$.

$\Rightarrow$ If we manage to maximize $\mathcal{L}(q)$, we have found an analytical approximation $q(\mathbf{Z})$ for the intractable distribution $p(\mathbf{Z}|\mathbf{X})$!

Approximating the Evidence

- From the decomposition

$$\log p(\mathbf{X}) = \underbrace{D_{\mathrm{KL}}(q||p)}_{\geq 0} + \mathcal{L}(q),$$

  we have derived a way to choose a distribution $q(\mathbf{Z})$ to be similar (in terms of KLD) to the intractable posterior $p(\mathbf{Z}|\mathbf{X})$.
- Requirement: The *full* distribution $p(\mathbf{X}, \mathbf{Z})$ (which appears in $\mathcal{L}(q)$) must be tractable.
- This assumption is often met. Remember the similar situation for the EM algorithm, where optimization was easy having the full data distribution $p(\mathbf{X}, \mathbf{Z})$ (including the cluster assignments $\mathbf{Z}$), and we approximated this distribution using an estimate for $\mathbf{Z}$.
- Remaining task: choose a (parametrized) family of distributions q which is tractable, but still expressive enough to capture the true posterior p sufficiently well!

The Mean-Field Approximation

- One way to choose $q(\mathbf{Z})$ is to assume that is *factorizes* over the latent variables:

$$q(\mathbf{Z}) = \prod_{i=1}^{M} q_i(\mathbf{Z}_i) = \prod_i q_i .$$

- Let us compute the form with $\mathcal{L}(q)$ takes! For *arbitrary* j,

$$\begin{aligned}
\mathcal{L}(q) &= \int q(\mathbf{Z}) \left[\log p(\mathbf{Z}, \mathbf{X}) - \log q(\mathbf{Z})\right] \mathrm{d}\mathbf{Z} \\
&= \int \prod_i q_i \left[\log p(\mathbf{X}, \mathbf{Z}) - \log \prod_k q_k\right] \mathrm{d}\mathbf{Z} \\
&= \int \prod_i q_i \log p(\mathbf{X}, \mathbf{Z}) \, \mathrm{d}\mathbf{Z} - \int \prod_i q_i \log \prod_k q_k \, \mathrm{d}\mathbf{Z} \\
&= \int q_j \left[\int_{i \neq j} \log p(\mathbf{X}, \mathbf{Z}) \prod_{i \neq j} q_i \, \mathrm{d}\mathbf{Z}_i\right] \mathrm{d}\mathbf{Z}_j - \int \prod_i q_i \left[\sum_k \log q_k\right] \mathrm{d}\mathbf{Z}
\end{aligned}$$

The Mean-Field Approximation

- The second term of the equation simplifies to:

$$\int \prod_i q_i \left[\sum_k \log q_k \right] \mathrm{d}\mathbf{Z} = \sum_k \int \prod_i q_i \log q_k \, \mathrm{d}\mathbf{Z} = \sum_k \int q_k \log q_k \, \mathrm{d}\mathbf{Z}_k .$$

- For the first term, let us define a new distribution $\tilde{p}(\mathbf{X}, \mathbf{Z}_j)$ by

$$\log \tilde{p}(\mathbf{X}, \mathbf{Z}_j) = E_{i \neq j}[\log p(\mathbf{X}, \mathbf{Z})]$$

where

$$E_{i \neq j}[\log p(\mathbf{X}, \mathbf{Z})] = \int_{i \neq j} \log p(\mathbf{X}, \mathbf{Z}) \prod_{i \neq j} q_i \, \mathrm{d}\mathbf{Z}_i .$$

The Mean-Field Approximation

- Then we can estimate $\mathcal{L}(q)$ as

$$\begin{aligned}\mathcal{L}(q) &= \int q_j \left[\int_{i \neq j} \log p(\mathbf{X}, \mathbf{Z}) \prod_{i \neq j} q_i \, \mathrm{d}\mathbf{Z}_i \right] \mathrm{d}\mathbf{Z}_j - \int \prod_i q_i \left[\sum_j \log q_j \right] \mathrm{d}\mathbf{Z} \\ &= \int q_j \log \tilde{p}(\mathbf{X}, \mathbf{Z}_j) \, \mathrm{d}\mathbf{Z}_j - \sum_i \int q_i \log q_i \, \mathrm{d}\mathbf{Z}_i .\end{aligned}$$

- If we keep q_i, $i \neq j$ fixed, the formula further simplifies to

$$\begin{aligned}\mathcal{L}(q) &= \int q_j \log \tilde{p}(\mathbf{X}, \mathbf{Z}_j) \, \mathrm{d}\mathbf{Z}_j - \int q_j \log q_j \, \mathrm{d}\mathbf{Z}_j + \text{const} \\ &= -D_{\mathrm{KL}}(q_j || \tilde{p}(\mathbf{X}, \mathbf{Z}_j)) + \text{const}.\end{aligned}$$

The Mean-Field Approximation

- Maximizing $\mathcal{L}(q)$ can now be performed by minimizing $D_{\mathrm{KL}}(q_j||\tilde{p}(\mathbf{X}, \mathbf{Z}_j))$. The minimum is found for q_j^* with

$$\log q_j^*(\mathbf{Z}_j) = \log \tilde{p}(\mathbf{X}, \mathbf{Z}_j) + \text{const} = E_{i \neq j}[\log p(\mathbf{X}, \mathbf{Z})] + \text{const},$$

  where the constant normalizes the distribution.
- This is, however, not a closed form solution, since the solution for each q_j depends on the solution for the other q_i.
- An approximation to the solution may be found iteratively by optimizing each q_j in turn until the improvement of $\mathcal{L}(q)$ for each step becomes sufficiently small.
- This is a generalization of the EM algorithm!
  - ➔ Just like for the EM algorithm, it can be proved that the solution converges.

The Mean-Field Approximation

- The fully probabilistic treatment avoids singularities like in the EM case (where point estimates for the means and covariances of the Gaussians are made).
- Consider the fitting of a mixture of six Gaussians in a fully Bayesian setting.
- The underlying data is best described by only 2 Gaussians!
- Indeed: After 120 steps, only two Gaussians remain with a positive weight, all other Gaussians have vanishing mixture weights (not drawn).

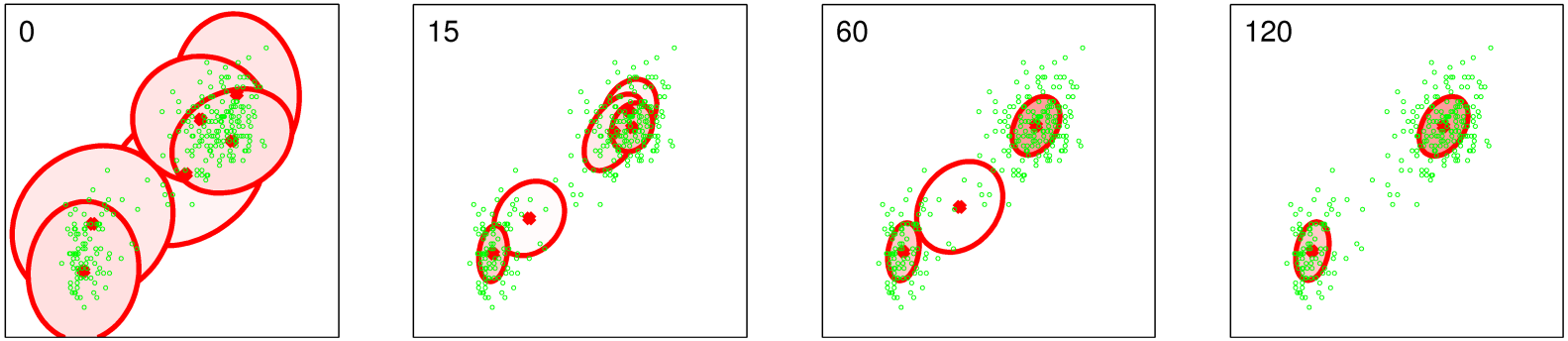


Image: Bishop, figure 10.6

Variational Inference: Conclusion / Outlook

- We have introduced the concept of Variational Inference to approximate intractable probability distributions.
- We have derived a solution for a specific form of the approximation of the posterior, namely under the assumption that the posterior *factorizes* over the latent variables.
- Bishop, chapter 10, is recommended reading for more details and concrete examples.
- We will revisit Variational Inference in the context of the Variational Autoencoder, a generative model which approximates the probability distribution of data using a neural network.

Conclusion

We have talked about

- probabilistic curve fitting
- the Bayesian framework
- Bias-Variance decomposition
- Bayes classification and logistic regression (where we saw the origin of the sigmoiod and softmax functions, and of the cross-entropy loss)

as rather simple models.
The GMM is very versatile and has been used a lot, in particular before the current neural network era. You should remember a) the model itself, and its applications b) the concept of *latent variables*.