5MB20 Adaptive Information Processing

Prof.dr.ir. Bert de Vries[1,2]   Dr.ir. Tjalling J. Tjalkens[1]

2014-2015

Course Outline and Administrative Issues

5MB20: Adaptive Information Processing

Title Adaptive Information Processing (5MB20)

When 2nd quartile, at alternating frequencies of 4 hours and 2 hours per week.

Exams Written exams in January and TBA

Load Total workload is 3 ECTS $\Rightarrow 3 \times 28 = 84$ hours or $84/24 = 3.5$ study hours per lecture.

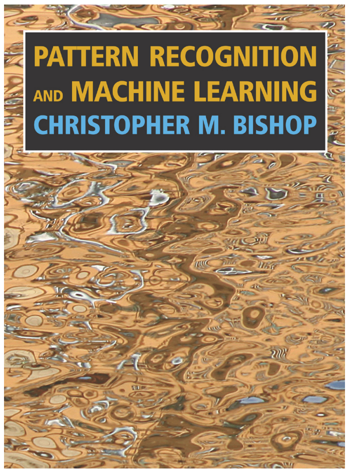
Web `www.sps.ele.tue.nl/members/b.vries/teaching/5mb20/index.html` (or go through `www.bertdv.nl`).

Course in Two Parts

- Part I: *Linear Gaussian Models and the EM Algorithm*
  - $6 \times 2 = 12$ lecture hours, mostly in November
  - Instructor: Bert de Vries, rm PT-3.15 (on Wednesdays)
  - email `bdevries@gnresound.com`

- Part II: *Model Complexity Control and the MDL Principle*
  - $6 \times 2 = 12$ lecture hours, mostly in December/January
  - instructor: Tjalling Tjalkens, rm PT-3.01
  - email `t.j.tjalkens@tue.nl`, tel. $\times$3690

Book



- find at `http://research.microsoft.com/~cmbishop/PRML/index.htm`.
- Background reading; covers about the same stuff as (mandatory) slides.
- Contains much more material; great for future study and reference.

Exam Guide Part-1 (LGM + EM)

- Tested material consists of these lecture notes, reading assignments (see website) and exercises.
- Slides that are not required for the exam are indicated by the word (OPTIONAL) in the header.
- Advice: download (and make free use of) Sam Roweis' cheat sheets for Matrix and Gaussian formulas.
- *Very strong advice: Make old exams!*
- Further exam instructions for part-2 from Tjalling.

Outline Part-1: Linear Gaussian Models and EM Algorithm

Outline

1 Introduction–What is Machine Learning?

What is Machine Learning?

- Machine Learning relates to *building models from data*
- It is known in various scientific communities under different names such as statistical inference, system identification, data mining, source coding, data compression, etc.

Machine Learning is Difficult

- Modeling (Learning) Problems
  - Is there any regularity in the data anyway?
  - What is our prior knowledge and how to express it mathematically?
  - How to pick the model library?
  - How to tune the models to the data?
  - How to measure the generalization performance?

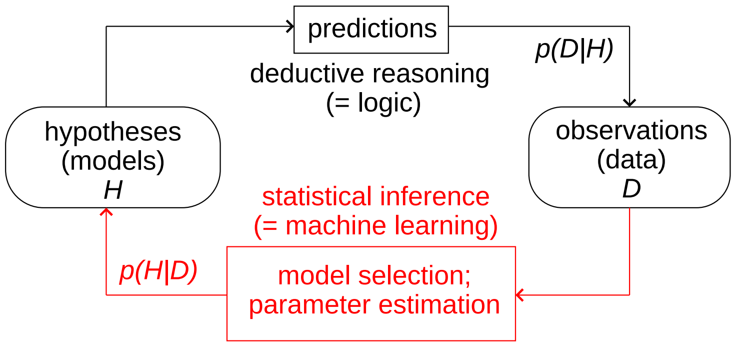


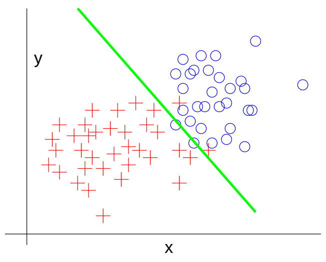
Figure 1: Machine learning and its place in the Scientific Method

- Quality of Observed Data
  - Not enough data
  - Too much data?
  - Available data may be messy (measurement noise, missing data points, outliers)

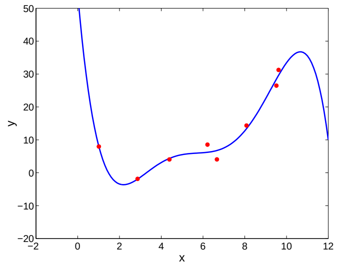
A Machine Learning Taxonomy

- *(Supervised Learning).* Given examples of inputs and corresponding desired outputs, predict outputs on future inputs. Ex: classification, regression, time series prediction
- *(Unsupervised Learning)* (a.k.a. density estimation). Given only inputs, automatically discover representations, features, structure, etc. Ex: clustering, outlier detection, compression
- *(Reinforcement Learning).* Given sequences of inputs, actions from a fixed set, and scalar rewards/punishments, learn to select action sequences in a way that maximizes expected reward, e.g. chess and robotics. (Not covered in this course.)

Supervised Learning
Given observations $\mathcal{D} = \{(\mathbf{x}_1, t_1), \ldots, (\mathbf{x}_N, t_N)\}$, the goal is to estimate the conditional distribution $p(t|\mathbf{x})$



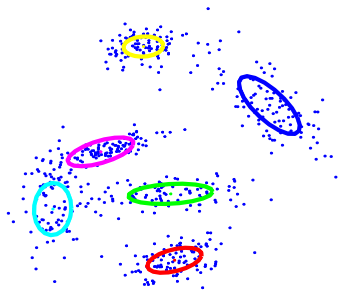
Classification. The target variable t is a discrete-valued vector representing class labels.



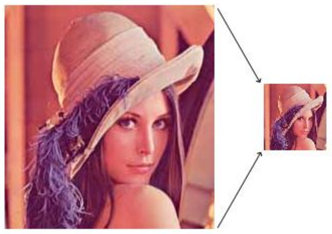
Regression. Same problem statement as classification but now the target variable is a real-valued vector.

Unsupervised Learning

Given data $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, model the (unconditional) probability distribution $p(\mathbf{x})$ (aka *density estimation*).



Clustering. Group data into clusters such that all data points in a cluster have similar properties.



Compression. Output from coder is much smaller in size than original, but if coded signal if further processed by decoder, then the result is very close (or exactly equal) to the original.

Some Machine Learning Applications

- computer speech recognition, speaker recognition
- face recognition, iris identification
- printed and handwritten text parsing
- financial prediction, outlier detection (credit-card fraud)
- user preference modeling (amazon); modeling of human perception
- machine translation
- medical expert systems for disease diagnosis (e.g., mammogram)
- strategic games (chess, go, backgammon)
- *any 'knowledge-poor' but 'data-rich' problem*

2 Probability Theory Review

Why Probability Theory?

- Probability theory (PT) is the *theory of optimal processing of incomplete information*, and as such provides a quantitative framework for drawing conclusions from a finite (read: incomplete) data set.
- In general, nearly all interesting questions in machine learning can be stated in the following form (a conditional probability),

$$p(\text{whatever-we-want-to-know} \,|\, \text{whatever-we-do-know})$$

- For example
  - Generate data predictions, $p(x_{\text{future}}|x_{\text{past}})$

  - Classify a received data point, $p(\mathcal{C}_k|x)$

- Information theory ('theory of log-probability') provides a source coding view on machine learning that is consistent with probability theory (more in part-2).

Probability Theory Notation

- An *event* A is a statement, whose truth is contemplated by a person, e.g.,

$$A = \text{'it will rain tomorrow'}$$

- For any event A, with background knowledge $\mathcal{I}$, the *probability of* A*, given* $\mathcal{I}$, $p(A|\mathcal{I})$, is a number between 0 and 1. If, given $\mathcal{I}$, you know that A is true, than $p(A|\mathcal{I}) = 1$.

- The probability $p(A|\mathcal{I})$ should be *interpreted* as your *degree of belief* that event A is true, given that $\mathcal{I}$ is true.

- (Conjunction). $p(A, B|\mathcal{I})$ is the *joint* probability that both A and B are true, given $\mathcal{I}$.

- (Disjunction). $p(A + B|\mathcal{I})$ is the probability that either A or B, or both A and B, are true, given $\mathcal{I}$.

Probability Theory Calculus

- Under some mild conditions relating to 'rational reasoning', the following calculation rules can be derived, (e.g. Cox, 1946):

- (Product rule). For 2 events A, B with given background $\mathcal{I}$,

$$p(A, B|\mathcal{I}) = p(A|\, B, \mathcal{I})\, p(B|\mathcal{I})\,.$$

- (Sum rule). For 2 events A, B with given background $\mathcal{I}$,

$$p(A + B|\mathcal{I}) = p(A|\mathcal{I}) + p(B|\mathcal{I}) - p(A, B|\mathcal{I})\,.$$

- Note that the background information may not change, e.g.

$$p(A, B|\mathcal{I}) \neq p(A|\, B, \mathcal{I})\, p(B|\mathcal{I}')$$

  if $\mathcal{I}' \neq \mathcal{I}$.

- *!! All legitimate relations between probabilities can be derived from the sum and product rules!!*

Some Notation and useful facts

- Iff $p(A, B|\mathcal{I}) = p(A|\mathcal{I})p(B|\mathcal{I})$, then A and B are said to be *independent*, given $\mathcal{I}$. Note that this is equivalent to the statement $p(A|B, \mathcal{I}) = p(A|\mathcal{I})$.
- All probabilities are in principle conditional probabilities of the type $p(A|\mathcal{I})$, since there is always some background knowledge.
- (Shorthand notation 1). Still, we often write $p(A)$ rather than $p(A|\mathcal{I})$ if the background knowledge $\mathcal{I}$ is assumed to be obviously present. E.g., $p(X = 5)$ rather than $p(X = 5|\,\text{the sun comes up tomorrow})$.
- (Shorthand notation 2). If X is a random variable and $X = x$ is an event, then we often write $p(x)$ rather than $p(X = x)$ (hoping again that the reader understands the context ;-)
- (Shorthand notation 3). $p(X)$ denotes the distribution over random variable X.

The Sum Rule and Marginalization

- If X and Y are random variables, than it follows from the sum rule that

$$p(X) = \sum_Y p(X, Y) \left(= \sum_Y p(X|Y)p(Y) \right) \tag{1}$$

  - Note that Bishop (p.14) calls eqn 1 the sum rule.
  - Of course, in the continuous domain, $p(x) = \int p(x, y)\, \mathrm{d}y$
- Integrating Y out according to eqn 1 is called *marginalization* and the result $p(X)$ is sometimes referred to as the *marginal* probability.

The Product Rule and Bayes Rule

- Consider 2 variables $\mathcal{D}$ and $\boldsymbol{\theta}$; it follows from $p(\mathcal{D}, \boldsymbol{\theta}) = p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta}) = p(\boldsymbol{\theta}|\mathcal{D})p(\mathcal{D})$ that

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})} \qquad \text{(Bayes rule)}$$

- While Bayes rule is always true, a particularly useful application occurs when $\mathcal{D}$ refers to an observed data set and $\boldsymbol{\theta}$ is set of model parameters that relates to the data. In that case,
  - $p(\boldsymbol{\theta})$, the prior probability for $\boldsymbol{\theta}$ represents our *degree-of-belief* about proper values for $\boldsymbol{\theta}$, before seeing the data $\mathcal{D}$.
  - $p(\boldsymbol{\theta}|\mathcal{D})$, the posterior probability for $\boldsymbol{\theta}$ after we have seen the data.

$\Rightarrow$ Bayes rule tells us how to update our knowledge about model parameters (or other hypotheses) when facing new data. Hence, *Bayes rule is the fundamental rule for machine learning.*

Bayes Rule Nomenclature

- Some nomenclature associated with Bayes rule

$$\underbrace{p(\boldsymbol{\theta}|\mathcal{D})}_{\text{posterior}} = \frac{\overbrace{p(\mathcal{D}|\boldsymbol{\theta})}^{\text{likelihood}} \times \overbrace{p(\boldsymbol{\theta})}^{\text{prior}}}{\underbrace{p(\mathcal{D})}_{\text{evidence}}}$$

- Note that the evidence can be computed through marginalization since

$$p(\mathcal{D}) = \int p(\mathcal{D},\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta} = \int p(\mathcal{D}|\boldsymbol{\theta})\,p(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta}$$

- For given $\mathcal{D}$, the posterior probabilities of the parameters scale relatively against each other as

$$p(\boldsymbol{\theta}|\mathcal{D}) \propto p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})$$

$\Longrightarrow$ All that we can learn from the observed data is contained in the likelihood function $p(\mathcal{D}|\boldsymbol{\theta})$. (This is called the *likelihood principle*).

The Likelihood Function

Consider a probabilistic model $p(\mathcal{D}|\boldsymbol{\theta})$, where $\mathcal{D}$ relates to a data set and $\boldsymbol{\theta}$ are model parameters.

- In general, $p(\mathcal{D}|\boldsymbol{\theta})$ is a function of both $\mathcal{D}$ and $\boldsymbol{\theta}$.
- The *sampling distribution* $p(\mathcal{D}|\boldsymbol{\theta}=\boldsymbol{\theta}_0)$ describes the probability that data $\mathcal{D}$ is observed, assuming that it is generated by the given model with parameter values set to $\boldsymbol{\theta}=\boldsymbol{\theta}_0$.
- In a machine learning context, often $\mathcal{D}=\mathcal{D}_0$ is given (observed), and $\boldsymbol{\theta}$ is the free variable.
- When viewed as a function of the free variable $\boldsymbol{\theta}$ with given $\mathcal{D}=\mathcal{D}_0$, $p(\mathcal{D}=\mathcal{D}_0|\boldsymbol{\theta})$ is called the *likelihood function*

$$L(\boldsymbol{\theta}) \equiv p(\mathcal{D}_0|\boldsymbol{\theta})$$

- Note that $L(\boldsymbol{\theta})$ is not a probability distribution for $\boldsymbol{\theta}$. (Is $\sum_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = 1$ always true?)

Probabilistic Inference

- *Probabilistic inference* is computing

$$p(\text{whatever-we-want-to-know} \,|\, \text{whatever-we-do-know})$$

- E.g., p('Mr.S. killed Mrs.S.'|'evidence'), p('transmitted codeword'|'received codeword'), p('articulatory movements'|'speech signal'), p('fetal HR signal'|'mother signal')
- This can be accomplished by repeated application of sum and product rules, (of course).
- In practice, Bayes Rule and marginalization are very useful tools.
- The resulting expressions often contain a bunch of (hard) integrals and/or sums (with many terms).

Inference Exercise: Disease Diagnosis

- [Q.] Given a disease D with prevalence of 1% and a test procedure T with sensitivity ('true positive' rate) of 95% and specificity ('true negative' rate) of 85%, what is the chance that somebody who tests positive actually has the disease?
- [A.] The given data are $p(D=1)=0.01$, $p(T=1|D=1)=0.95$ and $p(T=0|D=0)=0.85$. Then according to Bayes rule,

$$\begin{aligned}
&p(D=1|T=1)\\
&\quad\overset{Bayes}{=} \frac{p(T=1|D=1)p(D=1)}{p(T=1)}\\
&\quad\overset{Marg.}{=} \frac{p(T=1|D=1)p(D=1)}{p(T=1|D=1)p(D=1)+p(T=1|D=0)p(D=0)}\\
&\quad= \frac{0.95\times 0.01}{0.95\times 0.01+0.15\times 0.99}=0.0601
\end{aligned}$$

Inference Exercise: Bag Counter

- [Q.] A bag contains one ball, known to be either white or black. A white ball is put in, the bag is shaken, and a ball is drawn out, which proves to be white. What is now the chance of drawing a white ball?
- [A.] Again, use Bayes and marginalization to arrive at $p(\text{white}|\text{data})=2/3$, see homework exercise

⇒ Note that probabilities describe *a person's state of knowledge* rather than a 'property of nature'.

- [Q.] Is a speech signal a 'probabilistic' (random) variable? (homework)

Inference Exercise: Causality?

- [Q.] A dark bag contains five red balls and seven green ones. (a) What is the probability of drawing a red ball on the first draw? Balls are not returned to the bag after each draw. (b) If you know that on the second draw the ball was a green one, what is now the probability of drawing a red ball on the first draw?
- [A.] (a) 5/12. (b) 5/11, see homework.

⇒ Again, we conclude that conditional probabilities reflect *implications for a state of knowledge* rather than temporal causality.

PDF for the Sum of Two Variables

- [Q.]: Given two random *independent* variables X and Y, with PDF's $p_x(x)$ and $p_y(y)$. What is the PDF of
$$Z = X + Y\,?$$
- [A.]: Let $p_z(z)$ be the probability that Z has value z. This occurs if X has some value x and at the same time $Y = z - x$, with joint probability $p_x(x)p_y(z-x)$. Since x can be any value, we sum over all possible values for x to get
$$p_z(z) = \int_{-\infty}^{\infty} p_x(x) p_y(z-x)\,\mathrm{d}x$$
i.e., the *convolution* of p_x and p_y.

⇒ In linear stochastic systems theory, the Fourier Transform of a PDF (i.e., the characteristic function) plays an important computational role.

Expectation and Variance

- $\mathbb{E}[f] \equiv \int f(x)\,p(x)\,\mathrm{d}x$, *expected value* or *mean*
- $\mathrm{var}[f] \equiv \mathbb{E}\left[(f(x) - \mathbb{E}[f(x)])^2\right]$, *variance*
- *covariance* between vectors $\mathbf{x}$ and $\mathbf{y}$,
$$\begin{aligned}\mathrm{cov}[\mathbf{x},\mathbf{y}] &= \mathbb{E}\left[(\mathbf{x} - \mathbb{E}[\mathbf{x}])(\mathbf{y}^T - \mathbb{E}[\mathbf{y}^T])\right]\\ &= \mathbb{E}[\mathbf{x}\mathbf{y}^T] - \mathbb{E}[\mathbf{x}]\mathbb{E}[\mathbf{y}^T]\end{aligned}$$
- (Linear Transformations). No matter how $\mathbf{x}$ is distributed, we can easily derive that *(do as exercise)*
$$\mathbb{E}[A\mathbf{x} + b] = A\mathbb{E}[\mathbf{x}] + b \tag{SRG-3a}$$
$$\mathrm{cov}[A\mathbf{x} + b] = A\,\mathrm{cov}[\mathbf{x}]\,A^T \tag{SRG-3b}$$
  - (The tag (SRG-3a) refers to the corresponding eqn number in Sam Roweis' Gaussian Identities notes.)

Ex.: Mean and Variance for the Sum of Two Variables

- For any distribution of x and y and $z = x + y$,

$$\begin{aligned}\mathbb{E}[z] &= \int_z z \left[\int_x p_x(x) p_y(z-x)\, \mathrm{d}x \right] \mathrm{d}z \\ &= \int_x p_x(x) \left[\int_z z p_y(z-x)\, \mathrm{d}z \right] \mathrm{d}x \\ &= \int_x p_x(x) \left[\int_{y'} (y'+x) p_y(y')\, \mathrm{d}y' \right] \mathrm{d}x \\ &= \int_x p_x(x) \left(\mathbb{E}[y] + x \right) \mathrm{d}x \\ &= \mathbb{E}[x] + \mathbb{E}[y] \qquad \text{(always; follows from SRG-3a)}\end{aligned}$$

- Derive as an exercise that

$$\begin{aligned}\mathrm{var}[z] &= \mathrm{var}[x] + \mathrm{var}[y] + 2\mathrm{cov}[x,y] \qquad \text{(always, see SRG-3b)} \\ &= \mathrm{var}[x] + \mathrm{var}[y] \qquad \text{(if X and Y are independent)}\end{aligned}$$

Review Probability Theory

- Interpretation as a degree of belief, i.e. a state-of-knowledge, not as a property of nature.
- We can do everything with only the *sum rule* and the *product rule*. In practice, Bayes rule and marginalization are often very useful for computing

$$p(\text{what-we-want} | \text{what-we-know})\,.$$

- Bayes rule $p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})}$ is the fundamental rule for learning.
- That's really about all you need to know about probability theory, but you need to "really" know it, so do the exercises.

3 Bayesian Machine Learning

Bayesian Machine Learning

- Suppose that your task is to predict a 'new' datum $\mathbf{x}$, based on N observations $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$.
- The Bayesian approach for this task involves three stages: (1) model specification, (2) inference and (3) prediction.
- (*1. Model specification.*) Your first task is to propose a model with tuning parameters $\boldsymbol{\theta}$ for generating the data $\mathcal{D}$. This involves specification of $p(\mathcal{D}|\boldsymbol{\theta})$ and a prior for the parameters $p(\boldsymbol{\theta})$.

- *You* choose the distribution $p(\mathbf{x}|\boldsymbol{\theta})$ based on your physical understanding of the data generating process.
- Note that, for independent observations $\mathbf{x}_n$,

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{n=1}^{N} p(\mathbf{x}_n|\boldsymbol{\theta})$$

- *You* choose the prior $p(\boldsymbol{\theta})$ to reflect what you know about the parameter values before you see the data $\mathcal{D}$.

Bayesian Machine Learning cont'd: (2.) inference

- After model specification, use Bayes rule to find the posterior distribution for the parameters,

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})} \propto p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})$$

- This is the *parameter estimation* stage. There's no need to design a 'smart learning algorithm'. The only complexity lies in the computational issues.
- (*Inference cont'd: model comparison*). What if I have more candidate models, say $\mathcal{H} = \{h_1, \ldots, h_H\}$?
- Answer: specify a prior $p(h)$ for the models and use Bayes again to absorb what we can learn from the data,

$$p(h|\mathcal{D}) = \frac{p(\mathcal{D}|h)p(h)}{p(\mathcal{D})} \propto p(\mathcal{D}|h)p(h)$$

$\Rightarrow$ *Machine learning is easy,* *apart from computational details*

Bayesian Machine Learning cont'd: (3.) Prediction

- Given the data $\mathcal{D}$, our knowledge about the yet unobserved datum $\mathbf{x}$ is captured by

$$p(\mathbf{x}|\mathcal{D}) = \int p(\mathbf{x}, \boldsymbol{\theta}|\mathcal{D})\, \mathrm{d}\boldsymbol{\theta} = \int p(\mathbf{x}|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})\, \mathrm{d}\boldsymbol{\theta}$$

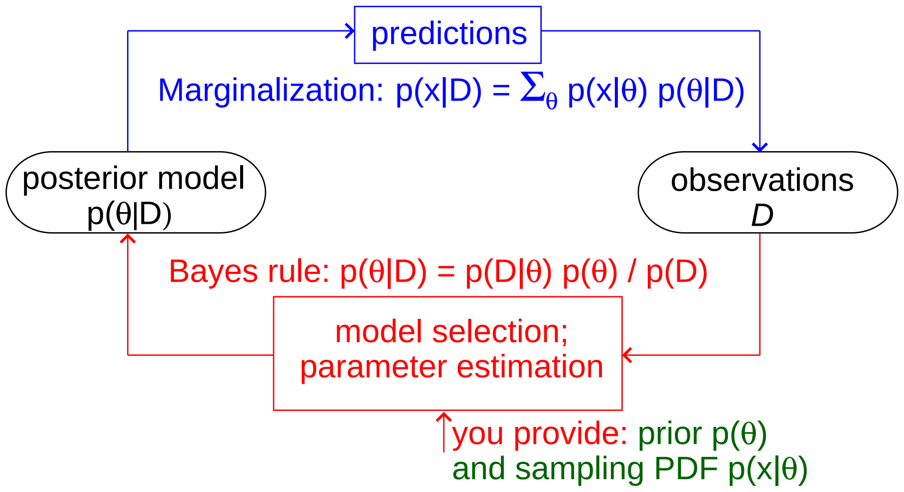
- Again, no need to invent a prediction algorithm. Probability theory takes care of all that. The complexity of prediction is just computational: how to carry out the marginalization over $\boldsymbol{\theta}$.
- What did we learn from $\mathcal{D}$? Without access to $\mathcal{D}$, we would predict new observations through

$$p(\mathbf{x}) = \int p(\mathbf{x}, \boldsymbol{\theta})\, \mathrm{d}\boldsymbol{\theta} = \int p(\mathbf{x}|\boldsymbol{\theta})p(\boldsymbol{\theta})\, \mathrm{d}\boldsymbol{\theta}$$

- Remaining problem: How good really were our model assumptions $p(\mathbf{x}|\boldsymbol{\theta})$ and $p(\boldsymbol{\theta})$? More on this in part 2 (Tjalkens).

Machine Learning and the Scientific Method Revisited

- Bayesian probability theory provides a unified framework for information processing (and even the Scientific Method).


predictions
Marginalization: p(x|D) = Σ_θ p(x|θ) p(θ|D)
posterior model p(θ|D)
observations D
Bayes rule: p(θ|D) = p(D|θ) p(θ) / p(D)
model selection; parameter estimation
you provide: prior p(θ) and sampling PDF p(x|θ)

(OPTIONAL) Coin Toss Example

- Observe a sequence of N coin tosses with n heads. What's the probability that heads (h) comes up next?
- [I- Model Specification] Assume a Bernoulli variable $p(x_n = h|\mu) = \mu$, leading to

$$p(\mathcal{D}|\mu) \overset{\text{IID}}{=} \prod_{n=1}^{N} p(x_n|\mathcal{D}) \overset{\text{Bernoulli}}{=} \mu^n (1-\mu)^{N-n}$$

- Assume prior belief is governed by a *beta distribution*

$$p(\mu) = \mathcal{B}(\mu|\alpha,\beta) = \frac{\gamma(\alpha+\beta)}{\gamma(\alpha)\gamma(\beta)} \mu^{\alpha-1}(1-\mu)^{\beta-1}$$

  - The beta-distribution is a 'binomial' over the continuous range $[0,1]$.
  - α and β are called *hyperparameters*, since they parameterize the distribution for another parameter (μ). E.g., $\alpha=\beta=1$ (uniform), $\alpha=\beta=0.5$ (Jeffreys prior), $\alpha=\beta=0$ (improper Laplace prior)

(OPTIONAL) Coin Toss Example (2): Learning

- [II- Learning] Infer posterior PDF over μ through Bayes rule

$$\begin{aligned} p(\mu|\mathcal{D}) &= \frac{p(\mathcal{D}|\mu)p(\mu|\alpha,\beta)}{\int_0^1 p(\mathcal{D}|\mu)p(\mu|\alpha,\beta)\,\mathrm{d}\mu} \\ &= \frac{\mu^n(1-\mu)^{N-n} \times \mu^{\alpha-1}(1-\mu)^{\beta-1}}{\int_0^1 \mu^n(1-\mu)^{N-n}\mu^{\alpha-1}(1-\mu)^{\beta-1}\,\mathrm{d}\mu} \\ &= \frac{(N+\alpha+\beta-1)!}{(n+\alpha-1)!(N-n+\beta-1)!}\mu^{n+\alpha-1}(1-\mu)^{N-n+\beta-1} \end{aligned}$$

where we used the formula for the *beta integral*

$$\int_0^1 x^p(1-x)^q\,\mathrm{d}x = \frac{p!q!}{(p+q+1)!}$$

  - Essentially, *here ends the machine learning activity*
  - Note: $p(\mu|\mathcal{D}) \sim \mathcal{B}(\mu|\,n+\alpha, N-n+\beta)$ is again beta, or

$$\text{beta} \propto \text{binomial} \times \text{beta}$$

  - The Beta dist. is a *conjugate prior* for the Binomial dist.

(OPTIONAL) Coin Toss Example (3): Prediction

- [III- Prediction] Marginalize over the parameter posterior to get the predictive PDF, given the data $\mathcal{D}$,

$$\begin{aligned} p(h|\mathcal{D}) &= \int_0^1 p(h|\mu)p(\mu|\mathcal{D})\,\mathrm{d}\mu \\ &= \frac{(N+\alpha+\beta-1)!}{(n+\alpha-1)!(N-n+\beta-1)!}\int_0^1 \mu \times \mu^{n+\alpha-1}(1-\mu)^{N-n+\beta-1}\,\mathrm{d}\mu \\ &= \frac{n+\alpha}{N+\alpha+\beta} \qquad \text{(a.k.a. Laplace rule)} \end{aligned}$$

  - For large N, $p(h|\mathcal{D}) = (n+\alpha)/(N+\alpha+\beta)$ goes to relative frequency n/N.
  - Example: for uniform prior ($\alpha=\beta=1$) and $\mathcal{D}=\{hthhtth\}$, we get

$$p(h|\mathcal{D}) = \frac{n+1}{N+2} = \frac{4+1}{7+2} = \frac{5}{9}$$

(OPTIONAL) Coin Toss Example: What did we learn?

- What did we learn from the data? Before seeing any data, we think that $p(h) = p(h|\mathcal{D})|_{n=N=0} = \alpha/(\alpha+\beta)$.
- After the N coin tosses, we think that

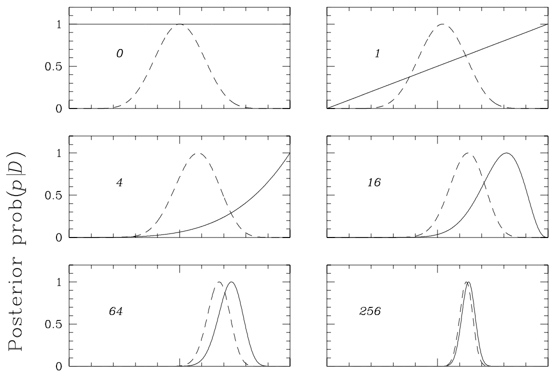
$$p(h|\mathcal{D}) = (n+\alpha)/(N+\alpha+\beta)$$

- Note the following decomposition

$$\begin{aligned} p(h|\,\mathcal{D},\alpha,\beta) &= (n+\alpha)/(N+\alpha+\beta) \\ &= \frac{n}{N+\alpha+\beta} + \frac{\alpha}{N+\alpha+\beta} \\ &= \frac{N}{N+\alpha+\beta}\cdot\frac{n}{N} + \frac{\alpha+\beta}{N+\alpha+\beta}\cdot\frac{\alpha}{\alpha+\beta} \\ &= \underbrace{\frac{\alpha}{\alpha+\beta}}_{prior} + \underbrace{\frac{N}{N+\alpha+\beta}}_{gain}\cdot(\underbrace{\frac{n}{N}}_{MLE} - \underbrace{\frac{\alpha}{\alpha+\beta}}_{prior}) \end{aligned}$$

- Note that estimate lies between prior and MLE.

Bayesian Evolution of $p(\mu|\mathcal{D})$ for the Coin Toss



- Evolution of posterior prob $p(\mu|\mathcal{D})$ after increasing number of coin tosses. Left-upper solid curve for uniform prior; dashed curve for Gaussian prior $\Longrightarrow$ *with more data, the relevance of the prior diminishes.*

From Posterior to Point-Estimate

Sometimes we want just one ‘best’ parameter (vector), rather than a posterior distribution over parameters. Why?

- Recall Bayesian prediction

$$p(\mathbf{x}|\mathcal{D}) = \int p(\mathbf{x}|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})\,\mathrm{d}\boldsymbol{\theta} \tag{2}$$

- If we approximate posterior $p(\boldsymbol{\theta}|\mathcal{D})$ by a delta function for one 'best' value $\hat{\boldsymbol{\theta}}$, then the predictive distribution collapses to

$$p(\mathbf{x}|\mathcal{D}) = \int p(\mathbf{x}|\boldsymbol{\theta})\delta(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})\,\mathrm{d}\boldsymbol{\theta} = p(\mathbf{x}|\hat{\boldsymbol{\theta}})$$

  This is the model $p(\mathbf{x}|\boldsymbol{\theta})$ evaluated at $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}$.

- Note that $p(\mathbf{x}|\hat{\boldsymbol{\theta}})$ is much easier to evaluate than the integral in Eq. 2.

Some Well-known Point-Estimates

- *Bayes* estimate $\hat{\boldsymbol{\theta}}_{bayes} = \int \boldsymbol{\theta}\, p\left(\boldsymbol{\theta}|\mathcal{D}\right)\,\mathrm{d}\boldsymbol{\theta}$
  - (homework). Proof that the Bayes estimate minimizes the expected mean-square error, i.e., proof that

$$\hat{\boldsymbol{\theta}}_{bayes} = \arg\min_{\hat{\boldsymbol{\theta}}} \int_{\boldsymbol{\theta}} (\hat{\boldsymbol{\theta}} - \boldsymbol{\theta})^2 p\left(\boldsymbol{\theta}|\mathcal{D}\right)\,\mathrm{d}\boldsymbol{\theta}$$

- *Maximum A Posteriori* (MAP) estimate

$$\hat{\boldsymbol{\theta}}_{map} = \arg\max_{\boldsymbol{\theta}} p\left(\boldsymbol{\theta}|\mathcal{D}\right) = \arg\max_{\boldsymbol{\theta}} p\left(\mathcal{D}|\boldsymbol{\theta}\right)\, p\left(\boldsymbol{\theta}\right)$$

- *Maximum Likelihood* (ML) estimate

$$\hat{\boldsymbol{\theta}}_{ml} = \arg\max_{\boldsymbol{\theta}} p\left(\mathcal{D}|\boldsymbol{\theta}\right)$$

  - ML is MAP with uniform prior

Learning with Maximum Likelihood Estimation

- Same task: predict $\mathbf{x}$ from observed data set $\mathcal{D}$.
- (1. model specification). Choose a model with parameters $\boldsymbol{\theta} \in \Theta$ and the data generating distribution $p(\mathbf{x}|\boldsymbol{\theta})$. (No need for priors).
- (2. learning). By Maximum Likelihood (ML) optimization,

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} p(\mathcal{D}|\boldsymbol{\theta}, h)$$

- (3. prediction). Easy through

$$p(\mathbf{x}|\mathcal{D}) = p(\mathbf{x}|\hat{\boldsymbol{\theta}})$$

- Note that this is a computationally easy approximation to the Bayesian approach. What is the price?

Report Card on Maximum Likelihood Estimation

- ML is MAP with uniform prior, or MAP is 'penalized' ML

$$\hat{\boldsymbol{\theta}}_{map} = \arg\max_{\boldsymbol{\theta}} \{ \overbrace{\log p(\mathcal{D}|\boldsymbol{\theta})}^{\text{log-likelihood}} + \overbrace{\log p(\boldsymbol{\theta})}^{\text{penalty}} \}$$

- (good!). Works rather well if we have a lot of data because the influence of the prior diminishes with more data.
- (bad). Cannot be used for model comparison. E.g. best model does generally not correspond to largest likelihood (see part-2, Tjalkens).
- (good). Computationally often do-able. Useful fact (since log is monotonously increasing):

$$\arg\max_{\boldsymbol{\theta}} \log p(\mathcal{D}|\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} p(\mathcal{D}|\boldsymbol{\theta})$$

$\Longrightarrow$ *ML estimation is an approximation to Bayesian learning*, but in the face of lots of available data for good reason a very popular learning method.

4 Working with Gaussians

Sums and Transformations of Gaussian Variables

- The Gaussian distribution

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

for variable $\mathbf{x}$ is completely specified by its mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\Sigma}$. ($\Lambda = \boldsymbol{\Sigma}^{-1}$ is called the *precision matrix*).

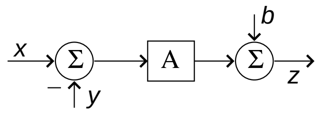
- *A linear transformation* $\mathbf{z} = A\mathbf{x} + b$ *of a Gaussian variable* $\mathcal{N}(\mathbf{x}|\,\boldsymbol{\mu}, \boldsymbol{\Sigma})$ *is Gaussian distributed*

$$p(\mathbf{z}) = \mathcal{N}\left(\mathbf{z}|\,A\boldsymbol{\mu} + b, A\boldsymbol{\Sigma}A^T\right) \qquad \text{(SRG-4a)}$$

- *The sum of two independent Gaussian variables is also Gaussian distributed.* If $\mathbf{x} \sim \mathcal{N}(\mathbf{x}|\,\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x)$ and $\mathbf{y} \sim \mathcal{N}(\mathbf{y}|\,\boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y)$, then the PDF for $\mathbf{z} = \mathbf{x} + \mathbf{y}$ is given by

$$\begin{aligned} p(\mathbf{z}) &= \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x) \ast \mathcal{N}(\mathbf{y}|\boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y) \\ &= \mathcal{N}(\mathbf{z}|\,\boldsymbol{\mu}_x + \boldsymbol{\mu}_y, \boldsymbol{\Sigma}_x + \boldsymbol{\Sigma}_y) \qquad \text{(SRG-8)} \end{aligned}$$

  - (homework): Work out the formula for $\mathbf{z}$ when $\mathbf{x}$ and $\mathbf{y}$ are *not* independent.



Example: Gaussian Signals in a Linear System

- [Q.]: Given independent variables $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_y)$ and $\mathbf{y} \sim \mathcal{N}(\boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y)$, what is the PDF for $\mathbf{z} = A(\mathbf{x} - \mathbf{y}) + b$?
- [A.]: $\mathbf{z}$ is also Gaussian with

$$p_z(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\, A(\boldsymbol{\mu}_x - \boldsymbol{\mu}_y) + b,\; A(\boldsymbol{\Sigma}_x + \boldsymbol{\Sigma}_y)A^T)$$

- Think about the role of the Gaussian distribution for stochastic linear systems in relation to what sinusoidals mean for deterministic linear system analysis.

Example: Bayesian Estimation of a Constant

- [Q.] Estimate a constant θ from a 'noisy' measurement x. Assume the following model specification,

$$x = \theta + \epsilon, \text{ where } \theta \sim \mathcal{N}(\theta|\mu_\theta, \sigma_\theta^2), \quad \epsilon \sim \mathcal{N}(\epsilon|0, \sigma_\epsilon^2)$$

- [A.] Note that you can rewrite these specifications as an equivalent set of PDF's:

$$\begin{aligned} p(x|\theta) &= \mathcal{N}(x|\theta, \sigma_\epsilon^2) && \text{(likelihood)} \\ p(\theta) &= \mathcal{N}(\theta|\mu_\theta, \sigma_\theta^2) && \text{(prior)} \end{aligned}$$

- NB: (just making sure that you understand the context): we use notation $p(x|\theta)$ as a shorthand for $p(x|\theta, h)$, where h refers to the assumed model. These model assumptions are *always* there, but usually not written if we focus merely on parameter estimation.

Bayesian Estimation of a Constant (2): Inference

- Inference for the posterior PDF $p(\theta|x)$

$$\begin{aligned}
p(\theta|x) &= \frac{p(x|\theta)p(\theta)}{p(x)} = \frac{p(x|\theta)p(\theta)}{\int p(x|\theta)p(\theta)\,\mathrm{d}\theta} \\
&= \frac{1}{C}\,\mathcal{N}(x|\theta,\sigma_\epsilon^2)\,\mathcal{N}(\theta|\mu_\theta,\sigma_\theta^2) \\
&= \frac{1}{C_1}\exp\left\{-\frac{(x-\theta)^2}{2\sigma_\epsilon^2} - \frac{(\theta-\mu_\theta)^2}{2\sigma_\theta^2}\right\} \\
&= \frac{1}{C_1}\exp\left\{\theta^2\left(-\frac{1}{2\sigma_\epsilon^2} - \frac{1}{2\sigma_\theta^2}\right) + \theta\left(\frac{x}{\sigma_\epsilon^2} + \frac{\mu_\theta}{\sigma_\theta^2}\right) + C_2\right\} \\
&= \frac{1}{C_1}\exp\left\{-\frac{\sigma_\theta^2+\sigma_\epsilon^2}{2\sigma_\theta^2\sigma_\epsilon^2}\left(\theta - \frac{x\sigma_\theta^2+\mu_s\sigma_\epsilon^2}{\sigma_\theta^2+\sigma_\epsilon^2}\right)^2 + C_3\right\}
\end{aligned}$$

- This computational 'trick' for multiplying two Gaussians is called *completing the square*. Compare the procedure to

$$ax^2 + bx + c_1 = a\left(x + \frac{b}{2a}\right)^2 + c_2$$

Bayesian Estimation of a Constant (3): Inference Cont'd

- Hence, it follows that the posterior for θ is

$$p(\theta|x) = \mathcal{N}(\theta|\,\mu_{\theta|x},\sigma_{\theta|x}^2)$$

where

$$\begin{aligned}
\sigma_{\theta|x}^2 &= \frac{\sigma_\epsilon^2\sigma_\theta^2}{\sigma_\epsilon^2+\sigma_\theta^2} = \left(\frac{1}{\sigma_\theta^2}+\frac{1}{\sigma_\epsilon^2}\right)^{-1} && \text{(precisions add)} \\
\mu_{\theta|x} &= \sigma_{\theta|x}^2\left(\frac{1}{\sigma_\epsilon^2}x + \frac{1}{\sigma_\theta^2}\mu_\theta\right) && \text{(weighted means add)}
\end{aligned}$$

- So, multiplication of two Gaussians yields another (unnormalized) Gaussian.

Multivariate Gaussian Multiplication

- In general, the multiplication of two multi-variate Gaussians yields an (unnormalized) Gaussian, see [SRG-6]:

$$\mathcal{N}(\mathbf{x}|\,\boldsymbol{\mu}_a,\boldsymbol{\Sigma}_a)\cdot\mathcal{N}(\mathbf{x}|\,\boldsymbol{\mu}_b,\boldsymbol{\Sigma}_b) = \alpha\cdot\mathcal{N}(\mathbf{x}|\,\boldsymbol{\mu}_c,\boldsymbol{\Sigma}_c)$$

$$\text{where}\quad \begin{aligned}\boldsymbol{\Sigma}_c &= \left(\boldsymbol{\Sigma}_a^{-1}+\boldsymbol{\Sigma}_b^{-1}\right)^{-1} \\ \boldsymbol{\mu}_c &= \boldsymbol{\Sigma}_c\left(\boldsymbol{\Sigma}_a^{-1}\boldsymbol{\mu}_a+\boldsymbol{\Sigma}_b^{-1}\boldsymbol{\mu}_b\right)\end{aligned}$$

and normalization constant $\alpha = \mathcal{N}(\boldsymbol{\mu}_a|\,\boldsymbol{\mu}_b,\boldsymbol{\Sigma}_a+\boldsymbol{\Sigma}_b)$.

- If we define the *precision* as $\mathbf{\Lambda} \equiv \mathbf{\Sigma}^{-1}$, then we see that *precisions add* and *precision-weighted means add* too.
- As we just saw, great application to Bayesian inference!

$$\underbrace{\text{Gaussian}}_{\text{posterior}} \propto \underbrace{\text{Gaussian}}_{\text{likelihood}} \times \underbrace{\text{Gaussian}}_{\text{prior}}$$

Conditioning and Marginalization of a Gaussian

- Let $\mathbf{z} = \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}$ be jointly normal distributed as

$$p(\mathbf{z}|\,\boldsymbol{\mu}, \mathbf{\Sigma}) = \mathcal{N}\left(\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \middle| \begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{bmatrix}, \begin{bmatrix} \mathbf{\Sigma}_x & \mathbf{\Sigma}_{xy} \\ \mathbf{\Sigma}_{yx} & \mathbf{\Sigma}_y \end{bmatrix} \right)$$

- Note that the symmetry $\mathbf{\Sigma} = \mathbf{\Sigma}^T$ implies that $\mathbf{\Sigma}_x$ and $\mathbf{\Sigma}_y$ are symmetric and $\mathbf{\Sigma}_{xy} = \mathbf{\Sigma}_{yx}^T$.
- Now let's factorize $p(\mathbf{x}, \mathbf{y})$ into $p(\mathbf{y}|\mathbf{x})\, p(\mathbf{x})$ through conditioning and marginalization (for applications to Bayesian inference in jointly Gaussian systems).
- *Marginalization*

$$p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{y})\, \mathrm{d}\mathbf{y} = \mathcal{N}\left(\mathbf{x}|\,\boldsymbol{\mu}_x, \mathbf{\Sigma}_x\right), \qquad p(\mathbf{y}) = \mathcal{N}\left(\mathbf{y}|\,\boldsymbol{\mu}_y, \mathbf{\Sigma}_y\right)$$

- *Conditioning*

$$\begin{aligned} p(\mathbf{y}|\mathbf{x}) &= p(\mathbf{x}, \mathbf{y})/p(\mathbf{x}) \\ &= \mathcal{N}\left(\mathbf{y}|\,\boldsymbol{\mu}_y + \mathbf{\Sigma}_{yx}\mathbf{\Sigma}_x^{-1}(\mathbf{x} - \boldsymbol{\mu}_x),\, \mathbf{\Sigma}_y - \mathbf{\Sigma}_{yx}\mathbf{\Sigma}_x^{-1}\mathbf{\Sigma}_{xy}\right) \end{aligned}$$

Example: Conditioning of Gaussian

- Consider (again) the system $x = \theta + \epsilon$, where

$$\theta \sim \mathcal{N}(\theta|\,\mu_\theta, \sigma_\theta^2) \quad \text{and} \quad \epsilon \sim \mathcal{N}(\epsilon|\,0, \sigma_\epsilon^2)$$

- This system is equivalent to (derive this!)

$$p(\theta, x|\,\boldsymbol{\mu}, \mathbf{\Sigma}) = \mathcal{N}\left(\begin{bmatrix} \theta \\ x \end{bmatrix} \middle| \begin{bmatrix} \mu_\theta \\ \mu_\theta \end{bmatrix}, \begin{bmatrix} \sigma_\theta^2 & \sigma_\theta^2 \\ \sigma_\theta^2 & \sigma_\theta^2 + \sigma_\epsilon^2 \end{bmatrix} \right)$$

- Direct substitution of the rule for Gaussian conditioning:

$$\begin{aligned} k &= \frac{\sigma_\theta^2}{\sigma_\theta^2 + \sigma_\epsilon^2} \qquad \text{(Kalman gain)} \\ p(\theta|x) &= \mathcal{N}\left(\theta|\,\mu_\theta + k \cdot (x - \mu_\theta),\, \sigma_\theta^2\,(1 - k)\right) \end{aligned}$$

- Exercises: (1) Actually derive this; (2) show that the result is equivalent to the previous slide on 'estimation of a constant'; and (3) Try to interpret the resulting formula's

- Moral: For jointly Gaussian systems, we do inference simply in one step by using the formulas for conditioning and marginalization.

Review Gaussians

The success of Gaussian distributions in probabilistic modeling is large due to the following properties:

- The product of two Gaussian functions is another Gaussian function. (ex.: Bayes rule)

- The convolution of two Gaussian functions is another Gaussian function. (It follows that the Fourier transform of a Gaussian is another Gaussian). (ex.: sum of 2 variables)

- A linear transformation of a Gaussian variable $\mathcal{N}(\mathbf{x}|\,\boldsymbol{\mu},\boldsymbol{\Sigma})$ is Gaussian distributed

- Conditioning and marginalization of multivariate Gaussian distributions produce Gaussians again. (ex.: observations and predictions)

- The Gaussian PDF has higher entropy than any other with the same variance. (Not discussed in this course).

- Any smooth function with single rounded maximum, if raised to higher and higher powers, goes into a Gaussian function. (Not discussed).

(OPTIONAL) Recursive Bayesian Estimation

- Now consider the signal $x(t) = \theta + \epsilon(t)$, where $\mathcal{D}_t = \{x(1), \ldots, x(t)\}$ is observed sequentially (over time).

- [Q] We want a sequential algorithm for $p(\theta|\mathcal{D}_t)$.

- [A] Again we assume $\theta \sim \mathcal{N}(\mu, \sigma^2)$ and define

$$\hat{\mu}(t) = \text{the estimate of (the mean of) } \theta \textit{ after} \text{ observing } x(t)$$

  (and similarly for $\sigma^2(t)$).

- We will solve this by using the estimate after $t-1$ as the prior distribution in conjunction with the likelihood for observation $x(t)$,

$$p(\mu(t)|D_t) \propto p(x(t)|\mu(t-1), \sigma^2(t-1)) \times p(\mu_t|\mathcal{D}_{t-1})$$

- Use the "batch processing" posteriors for μ and σ^2 to get

$$\begin{aligned}
\hat{\mu}(t) &= \sigma_\mu^2(t)\left(\frac{1}{\sigma_\epsilon^2(t)}x(t) + \frac{1}{\sigma_\mu^2(t-1)}\hat{\mu}(t-1)\right) \\
&= \frac{\sigma_\epsilon^2(t)}{\sigma_\epsilon^2(t)+\sigma_\mu^2(t-1)}x(t) + \frac{\sigma_\mu^2(t-1)}{\sigma_\epsilon^2(t)+\sigma_\mu^2(t-1)}\hat{\mu}(t-1) \\
&= \hat{\mu}(t-1) + K(t)\left[x(t) - \hat{\mu}(t-1)\right] \\
\sigma_\mu^2(t) &= \sigma_\mu^2(t-1)\frac{\sigma_\epsilon^2(t)}{\sigma_\epsilon^2(t)+\sigma_\mu^2(t-1)} \\
&= \sigma_\mu^2(t-1)\left(1-K(t)\right)
\end{aligned}$$

  where we defined the *Kalman gain*

$$K(t) = \frac{\sigma_\mu^2(t-1)}{\sigma_\epsilon^2(t)+\sigma_\mu^2(t-1)}$$

- This linear sequential estimator of mean and variance in Gaussian observations is a *Kalman Filter*.
- The new observation $x(t)$ 'corrects' the old estimate $\hat{\mu}(t-1)$ by a quantity that is proportional to the *innovation* (or *residual*) $[x(t) - \hat{\mu}(t-1)]$.
- Note that the uncertainty about μ decreases over time
- Recursive Bayesian estimation is the basis for *adaptive signal processing* algorithms such as Least Mean Squares (LMS) and Recursive Least Squares (RLS).

Some Useful Matrix Calculus (Final Math Preparations)

- Define the *gradient* of a scalar function $f(A)$ w.r.t. $n \times k$ matrix A

$$\nabla_A f \equiv \begin{bmatrix} \frac{\partial f}{\partial a_{11}} & \frac{\partial f}{\partial a_{12}} & \cdots & \frac{\partial f}{\partial a_{1k}} \\ \frac{\partial f}{\partial a_{21}} & \frac{\partial f}{\partial a_{22}} & \cdots & \frac{\partial f}{\partial a_{2k}} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial f}{\partial a_{n1}} & \frac{\partial f}{\partial a_{n2}} & \cdots & \frac{\partial f}{\partial a_{nk}} \end{bmatrix}$$

- The following formulas are useful (see Bishop App.-C)

$$\begin{aligned}
|A^{-1}| &= |A|^{-1} && \text{(B-C.4)} \\
\nabla_A \log|A| &= (A^T)^{-1} = (A^{-1})^T && \text{(B-C.28)} \\
\mathrm{Tr}[ABC] &= \mathrm{Tr}[CAB] = \mathrm{Tr}[BCA] && \text{(B-C.9)} \\
\nabla_A \mathrm{Tr}[AB] &= \nabla_A \mathrm{Tr}[BA] = B^T && \text{(B-C.25)} \\
\nabla_A \mathrm{Tr}[ABA^T] &= A(B+B^T) && \text{(B-C.27)} \\
\nabla_x x^T A x &= (A+A^T)x && \text{(from B-C.27)} \\
\nabla_X a^T X b &= \nabla_X \mathrm{Tr}[ba^T X] = ab^T
\end{aligned}$$

What's Next?

- We discussed how Bayesian probability theory provides an integrated framework for making predictions based on observed data.
- The process involves model specification (your main task!), inference and actual model-based prediction.
- The latter two tasks are only difficult because of computational issues.
- Maximum likelihood was introduced as a computationally simpler approximation to the Bayesian approach.
- In particular under some linear Gaussian assumptions, a few interesting models can be designed.
- The rest of this class (part-1) concerns introduction to these Linear Gaussian models.

5 Density estimation

Why Density Estimation?

- Why are we interested to build a density model $p(\mathbf{x}|\boldsymbol{\theta})$ from data observations $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$? Some examples:
- *(Outlier detection).* Suppose $\{\mathbf{x}_n\}$ are benign mammogram images. Build $p(\mathbf{x}|\boldsymbol{\theta})$ from $\mathcal{D}$. Then low value for $p(\mathbf{x}_{new}|\boldsymbol{\theta})$ indicates a risky mammogram.
- *(Compression).* Code a new data item based on *entropy*

$$H[p] = -\sum_x p(\mathbf{x}|\boldsymbol{\theta}) \log p(\mathbf{x}|\boldsymbol{\theta})$$

- *(Classification).* Let $p(\mathbf{x}|\boldsymbol{\theta}_1)$ is a model of attributes $\mathbf{x}$ for credit-card holders that paid on time; $p(\mathbf{x}|\boldsymbol{\theta}_2)$ for clients that defaulted on payments. Then, assign a potential new client to either class based on the relative probs $p(\mathbf{x}_{new}|\boldsymbol{\theta}_1)$ vs. $p(\mathbf{x}_{new}|\boldsymbol{\theta}_2)$.

Log-Likelihood for a Multivariate Gaussian

- Assume we are given a set of IID data points $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, where $\mathbf{x}_n \in \mathbb{R}^D$. We want to build a model for these data.
- (Model specification). Let's assume a MVG model $\mathbf{x}_n = \boldsymbol{\mu} + \boldsymbol{\epsilon}_n$ with $\boldsymbol{\epsilon}_n \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$, or equivalently,

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

- Since the data are IID, $p(\mathcal{D}|\boldsymbol{\theta})$ factorizes

$$p(\mathcal{D}|\boldsymbol{\theta}) = p(\mathbf{x}_1, \ldots, \mathbf{x}_N|\boldsymbol{\theta}) \overset{\text{IID}}{=} \prod_n p(\mathbf{x}_n|\boldsymbol{\theta})$$

- This choice of model yields the following log-likelihood (use (B-C.9) and (B-C.4)),

$$\begin{aligned}\log p(\mathcal{D}|\boldsymbol{\theta}) &= \log \prod_n p(\mathbf{x}_n|\boldsymbol{\theta}) = \sum_n \log p(\mathbf{x}_n|\boldsymbol{\theta}) \\ &= N \cdot \log|2\pi\boldsymbol{\Sigma}|^{-1/2} - \frac{1}{2}\textstyle\sum_n (\mathbf{x}_n - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu})\end{aligned}$$

Maximum Likelihood estimation of mean of MVG

- We want to maximize $\log p(\mathcal{D}|\boldsymbol{\theta})$ wrt the parameters $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$. Let's take derivatives; first to mean $\boldsymbol{\mu}$, (making use of (B-C.25) and (B-C.27)),

$$\begin{aligned}\nabla_\mu \log p(\mathcal{D}|\boldsymbol{\theta}) &= -\frac{1}{2}\sum_n \nabla_\mu \left[(\mathbf{x}_n - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu})\right] \\ &= -\frac{1}{2}\sum_n \nabla_\mu \mathrm{Tr}\left[-2\boldsymbol{\mu}^T \boldsymbol{\Sigma}^{-1}\mathbf{x}_n + \boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right] \\ &= -\frac{1}{2}\sum_n \left(-2\boldsymbol{\Sigma}^{-1}\mathbf{x}_n + 2\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right) \\ &= \boldsymbol{\Sigma}^{-1}\sum_n (\mathbf{x}_n - \boldsymbol{\mu})\end{aligned}$$

- Set to zero yields the *sample mean*

$$\boxed{\hat{\boldsymbol{\mu}} = \frac{1}{N}\sum_n \mathbf{x}_n}$$

Maximum Likelihood estimation of variance of MVG

- Now we take the log-likelihood *gradient wrt precision matrix* $\boldsymbol{\Sigma}^{-1}$ (making use of B-C.28 and B-C.24)

$$\begin{aligned}\nabla_{\boldsymbol{\Sigma}^{-1}} L &= \nabla_{\boldsymbol{\Sigma}^{-1}} \left[\frac{N}{2}\log|2\pi\boldsymbol{\Sigma}|^{-1} - \frac{1}{2}\sum_{n=1}^N (\mathbf{x}_n - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}_n - \boldsymbol{\mu})\right] \\ &= \nabla_{\boldsymbol{\Sigma}^{-1}} \left[\frac{N}{2}\log|\boldsymbol{\Sigma}^{-1}| - \frac{1}{2}\sum_{n=1}^N \mathrm{Tr}\left[(\mathbf{x}_n - \boldsymbol{\mu})(\mathbf{x}_n - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}\right]\right] \\ &= \frac{N}{2}\boldsymbol{\Sigma} - \frac{1}{2}\sum_n (\mathbf{x}_n - \boldsymbol{\mu})(\mathbf{x}_n - \boldsymbol{\mu})^T\end{aligned}$$

- Get optimum by setting the gradient to zero,

$$\boxed{\hat{\boldsymbol{\Sigma}} = \frac{1}{N}\sum_n (\mathbf{x}_n - \hat{\boldsymbol{\mu}})(\mathbf{x}_n - \hat{\boldsymbol{\mu}})^T}$$

  which is the *sample variance*.

Discrete Data: the 1-of-K Coding Scheme

- Consider a coin-tossing experiment with outcomes $x \in \{0, 1\}$ (corresponding to tail and head, resp.) and $0 \leq \mu \leq 1$ the probability of heads. This model can written as a *Bernoulli distribution*,

$$p(x|\mu) = \mu^x (1-\mu)^{1-x}$$

- Note that in expression $\mu^x(1-\mu)^{1-x}$, the variable x acts as a (binary) selector for the tail or head probabilities.
- (1-of-K scheme). Now consider a K-sided coin, a.k.a. a *die* (pl.: dice). It will be very convenient to code the outcomes by a vector $\mathbf{x} = (x_1, \ldots, x_K)^T$ with *binary selection variables*

$$x_k = \begin{cases} 1 & \text{if landed on } k\text{th face} \\ 0 & \text{else} \end{cases}$$

Discrete Data: the 1-of-K Coding Scheme, cont'd

- E.g., For $K = 6$, outcome $x_3 = 1 \iff \mathbf{x} = (0, 0, 1, 0, 0, 0)^T$.
- Assume the probabilities $p(x_k = 1) = \mu_k$ with $\sum_k \mu_k = 1$. The data generating distribution is then (note the similarity to the Bernoulli distribution),

$$p(\mathbf{x}|\boldsymbol{\mu}) = \mu_1^{x_1}\mu_2^{x_2}\cdots\mu_k^{x_k} = \prod_k \mu_k^{x_k}$$

- Note that $\sum_k x_k = 1$ and verify for yourself that $\mathbb{E}[\mathbf{x}|\boldsymbol{\mu}] = \boldsymbol{\mu}$.

The Multinoulli (and Multinomial) Distribution

- Observe a data set $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ of N IID rolls of a K-sided die, with generating PDF

$$p(\mathcal{D}|\boldsymbol{\mu}) = \prod_n \prod_k \mu_k^{x_{nk}} = \prod_k \mu_k^{\sum_n x_{nk}} = \prod_k \mu_k^{m_k}$$

  where $m_k = \sum_n x_{nk}$ is the total number of occurrences that we 'threw' k eyes. $p(\mathcal{D}|\boldsymbol{\mu})$ is sometimes called the *'multinoulli'* distribution.

- (side note). Note that this distribution depends on the observations *only* through the quantities $\{m_k\}$, with generally $K \ll N$. The *distribution* over $\mathbf{m} = (m_1, \ldots, m_K)^T$ is called the *multinomial distribution*,

$$p(\mathbf{m}|\boldsymbol{\mu}) = \frac{N!}{m_1! m_2! \ldots m_K!} \prod_k \mu_k^{m_k} \,.$$

  - Be aware that the literature is sloppy in distinguishing between the names multinoulli and multinomial.

- Note that $\mathcal{D}$ and $\mathbf{m}$ have N and K components, respectively.

Maximum Likelihood Estimation for Multinoulli

- Now let's find the ML estimate for $\boldsymbol{\mu}$, based on N throws of a K-sided die. The log-likelihood with Lagrange multiplier to include the constraint is

$$\begin{aligned}\tilde{L}(\theta) &= \log \prod_k \mu_k^{m_k} + \lambda \cdot (1 - \sum_k \mu_k) \\ &= \sum_k m_k \log \mu_k + \lambda \cdot (1 - \sum_k \mu_k)\,.\end{aligned}$$

- Set derivative to zero yields the *sample proportion* for μ_k

$$\nabla_{\mu_k} \tilde{L} = \frac{m_k}{\hat{\mu}_k} - \lambda = 0 \;\Rightarrow\; \boxed{\hat{\mu}_k = \frac{m_k}{\lambda} = \frac{m_k}{N}}$$

  where we get λ from the constraint

$$\sum_k \hat{\mu}_k = \sum_k \frac{m_k}{\lambda} = \frac{N}{\lambda} = 1$$

Recap ML for Density Estimation

- Given N IID observations $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$
- For MVG model, $p(\mathbf{x}_n|\boldsymbol{\theta}) = \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}, \boldsymbol{\Sigma})$, we obtain ML estimates

$$\hat{\boldsymbol{\mu}} = \frac{1}{N} \sum_n \mathbf{x}_n \qquad \text{(sample mean)}$$

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{N} \sum_n (\mathbf{x}_n - \hat{\boldsymbol{\mu}})(\mathbf{x}_n - \hat{\boldsymbol{\mu}})^T \qquad \text{(sample variance)}$$

- For discrete outcomes modeled by a multinoulli distribution $p(\mathbf{x}_n|\boldsymbol{\mu}) = \prod_k \mu_k^{x_{nk}}$ (with $\sum_k \mu_k = 1$), we find

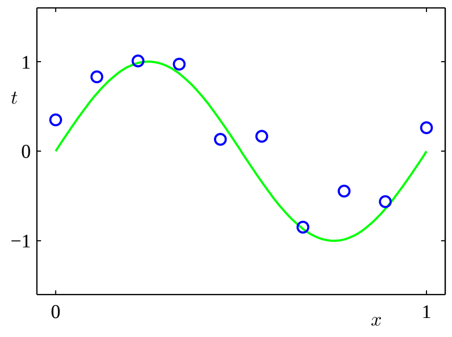
$$\hat{\mu}_k = \frac{m_k}{N} \qquad \text{(sample proportion)}$$

  where $m_k = \sum_n x_{nk}$.

- (exercise). Note the similarity for the means between discrete and continuous data. But we didn't use a co-variance matrix for discrete data. Why?

6 Linear Regression

Linear Regression–Illustration



- Given a set of (noisy) data measurements, find the 'best' (linear) relation between an input variables $\mathbf{x}$ and input-dependent outcomes t.

The Regression Model

- Observe N IID data *pairs* $\mathcal{D} = \{(\mathbf{x}_1, t_1), \ldots, (\mathbf{x}_N, t_N)\}$, $\mathbf{x}_n \in \mathbb{R}^D$ and $t_n \in \mathbb{R}$.
- [Q.] We could try to build a model for the data by density estimation, $p(\mathbf{x}, t)$, but what if we are interested only in (a model for) the responses t_n for *given inputs* $\mathbf{x}_n$?
- [A.] We will build models for the conditional distribution $p(t|\mathbf{x})$. (Note that, since $p(\mathbf{x}, t) = p(t|\mathbf{x})\, p(\mathbf{x})$, this is a building block for the joint data density.)
- In a *regression model*, we try to 'explain the data' by a purely deterministic term $f(\mathbf{x}, \mathbf{w})$, plus a purely random term ϵ_n for 'unexplained noise',

$$t_n = f(\mathbf{x}_n, \mathbf{w}) + \epsilon_n$$

Model Specification for Linear Regression

- In *linear* regression, we assume that $f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \mathbf{x}$.

- In *ordinary linear regression*, the noise process ϵ_n is zero-mean Gaussian with constant variance σ^2, i.e.

$$t_n = \mathbf{w}^T \mathbf{x}_n + \mathcal{N}(0, \sigma^2)$$

or equivalently the likelihood model, (from $\mathbb{E}[t_n|\mathbf{x}_n, \mathbf{w}] = \mathbb{E}[\mathbf{w}^T\mathbf{x}_n + \epsilon_n|\,\mathbf{x}_n, \mathbf{w}] = \mathbf{w}^T\mathbf{x}_n$),

$$p(t_n|\,\mathbf{x}_n, \mathbf{w}) = \mathcal{N}(t_n|\,\mathbf{w}^T\mathbf{x}_n, \sigma^2)$$

- Remember that for full Bayesian learning we should also choose a prior $p(\mathbf{w})$; In ML estimation, the prior is uniformly distributed (so it can be ignored).

ML Estimation for Linear Regression Model

- Inference by ML; work out the log-likelihood for $\mathbf{w}$,

$$\begin{aligned}\log p(\mathcal{D}|\mathbf{w}) &\overset{IID}{=} \sum_n \log \mathcal{N}(t_n|\,\mathbf{w}^T\mathbf{x}_n, \sigma^2) \propto -\frac{1}{2\sigma^2}\sum_n (t_n - \mathbf{w}^T\mathbf{x}_n)^2 \\ &= -\frac{1}{2\sigma^2}\left(\mathbf{t} - \mathbf{X}\mathbf{w}\right)^T\left(\mathbf{t} - \mathbf{X}\mathbf{w}\right)\end{aligned}$$

where we defined $N \times 1$ vector $\mathbf{t} = \left(t_1, t_2, \dots, t_N\right)^T$ and $(N \times D)$-dim matrix $\mathbf{X} = \left(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\right)^T$.

- Set the derivative $\nabla_w \log p(\mathcal{D}|\mathbf{w}) = \frac{1}{\sigma^2}\mathbf{X}^T(\mathbf{t} - \mathbf{X}\mathbf{w})$ to zero for the maximum likelihood estimate

$$\boxed{\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{t}}$$

- *Prediction* of new data points by

$$\hat{t}_{\text{new}} = \hat{\mathbf{w}}^T\mathbf{x}_{\text{new}} = \mathbf{t}^T\mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{x}_{\text{new}}$$

- N.B. The matrix $\mathbf{X}^\dagger \equiv (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$ is also known as the *Moore-Penrose pseudo-inverse*.

Deterministic Least-Squares Regression

- (Some may say that) we don't need to work with probabilistic models. E.g., there's also the deterministic *least-squares* solution: minimize sum of squared errors,

$$\hat{\mathbf{w}}_{LS} = \arg\min_{\mathbf{w}} \sum_n \left(t_n - \mathbf{w}^T\mathbf{x}_n\right)^2$$

- Derivative $\partial\left(\mathbf{t} - \mathbf{X}\mathbf{w}\right)^T\left(\mathbf{t} - \mathbf{X}\mathbf{w}\right)/\partial\mathbf{w} = -2\mathbf{X}^T\left(\mathbf{t} - \mathbf{X}\mathbf{w}\right)$

- Setting derivative to zero yields *normal equations* $\mathbf{X}^T\mathbf{X}\hat{\mathbf{w}}_{LS} = \mathbf{X}^T\mathbf{t}$ and

$$\hat{\mathbf{w}}_{LS} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{t}$$

$\Longrightarrow$ Least-squares regression corresponds to (probabilistic) maximum likelihood if

1. *IID samples* (determines how errors are combined)
2. Noise $\epsilon_n \sim \mathcal{N}(0,\,\sigma^2)$ is *Gaussian* (determines error metric)

Probabilistic vs. Deterministic Approach

- The (deterministic) least-squares approach assumed IID Gaussian distributed data, but these assumptions are not obvious from looking at the least-squares (LS) criterion.
- If the data were better modeled by non-Gaussian assumptions (or not IID), then LS might not be appropriate.
- The probabilistic approach makes all these issues completely transparent by focusing on the *model specification* rather than the error criterion.
- Next, we will show this by two examples: (1) samples not identically distributed, and (2) few data points.

Not Identically Distributed Data

- What if we assume that the variance of the measurement error varies with the sampling index, $\epsilon_n \sim \mathcal{N}(0, \sigma_n^2)$?

$$L(\mathbf{w}) \sim -\frac{1}{2}\sum_n \frac{(t_n - \mathbf{w}^T\mathbf{x}_n)^2}{\sigma_n^2} = -\frac{1}{2}(\mathbf{t}-\mathbf{X}\mathbf{w})^T\mathbf{\Lambda}(\mathbf{t}-\mathbf{X}\mathbf{w})$$

  where $\mathbf{\Lambda} = \mathrm{diag}[1/\sigma_n^2]$.

- Set derivative $\partial L/\partial \mathbf{w} = -\mathbf{X}^T\mathbf{\Lambda}(\mathbf{t}-\mathbf{X}\mathbf{w})$ to zero to get *normal equations*

$$\boxed{\mathbf{X}^T\mathbf{\Lambda}\mathbf{X}\hat{\mathbf{w}} = \mathbf{X}^T\mathbf{\Lambda}\mathbf{t}}$$

- This is called the Weighted Least Squares (WLS) solution. (Note that we just stumbled upon it, the crucial aspect is appropriate model specification!)
- Note also that the dimension of $\mathbf{\Lambda}$ grows with the number of data points. In general, models with this property are called *non-parametric* models.

Too Few Training Samples

- Problem: If we have fewer training samples than input dimensions, $\mathbf{X}^T\mathbf{X}$ will not be invertible.
- Solution: In case of (expected) problems, *go back to full Bayesian!* Do proper model specification, Bayesian inference etc.
- Let's try a Gaussian prior for $\mathbf{w}$ (why is this reasonable?),

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0},\mathbf{\Sigma}) = \mathcal{N}(\mathbf{w}|\mathbf{0},\varepsilon I)$$

- Let's do Bayesian inference,

$$\begin{aligned}\log p(\mathbf{w}|\mathcal{D}) \propto \log p(\mathcal{D}|\mathbf{w})p(\mathbf{w}) &\overset{IID}{=} \log \sum_n p(t_n|\mathbf{x}_n,\mathbf{w}) + \log p(\mathbf{w}) \\ &= \log \sum_n \mathcal{N}(t_n|\,\mathbf{w}^T\mathbf{x}_n,\sigma^2) + \log \mathcal{N}(\mathbf{w}|\mathbf{0},\varepsilon I) \\ &\propto \frac{1}{2\sigma^2}(\mathbf{t}-\mathbf{X}\mathbf{w})^T(\mathbf{t}-\mathbf{X}\mathbf{w}) + \frac{1}{2\epsilon}\mathbf{w}^T\mathbf{w}\end{aligned}$$

- Done! $p(\mathbf{w}|\mathcal{D})$ specifies all we know about $\mathbf{w}$ after seeing the data $\mathcal{D}$.

Too Few Training Samples, cont'd

- As discussed, for practical purposes, you often want a point estimate for $\mathbf{w}$, rather than a posterior distribution.
- Let's take a MAP estimate, set derivative

$$\nabla_w \log p(\mathbf{w}|\mathcal{D}) = -\frac{1}{\sigma^2}\mathbf{X}^T(\mathbf{t}-\mathbf{X}\mathbf{w}) + \frac{1}{\varepsilon}\mathbf{w}$$

  to zero, yielding

$$\boxed{\hat{\mathbf{w}}_{MAP} = (\mathbf{X}^T\mathbf{X} + \frac{\sigma^2}{\varepsilon}I)^{-1}\mathbf{X}^T\mathbf{t}}$$

  The matrix $(\mathbf{X}^T\mathbf{X} + \frac{\sigma^2}{\varepsilon}I)$ is always invertible!

- Note that LS corresponds to $\varepsilon \to \infty$. Does that make sense?

Adaptive Linear Regression

- What if the data arrives one point at a time?
- Two standard adaptive linear regression approaches: RLS and LMS.

- Least Mean Squares (LMS) is gradient-descent on a 'local-in-time' approximation of the square-error cost function.
- Define *'cost of current sample'* $E_n(\mathbf{w}) = \frac{1}{2}(t_n - \mathbf{w}^T\mathbf{x}_n)^2$ and track the optimum by *gradient descent*

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \eta \left.\frac{\partial E_n}{\partial \mathbf{w}}\right|_{\mathbf{w}_n}$$

  leads to

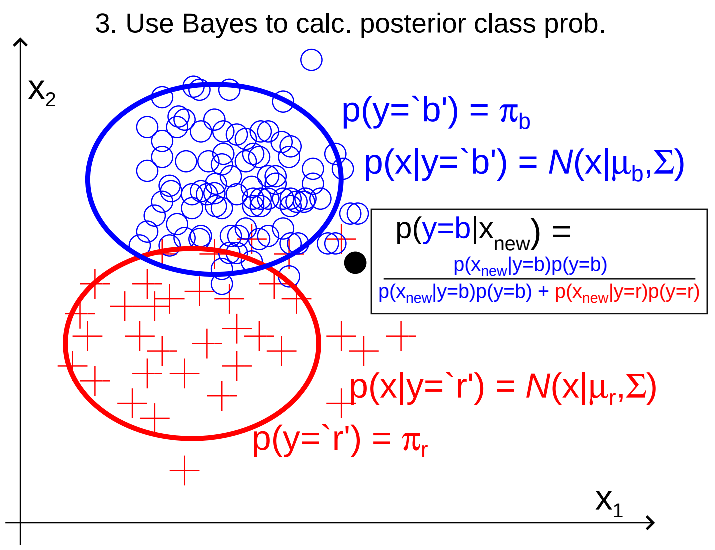
$$\mathbf{w}_{n+1} = \mathbf{w}_n + \eta\,(t_n - \mathbf{w}_n^T\mathbf{x}_n)\mathbf{x}_n \tag{LMS}$$

- (OPTIONAL) For a really cool Bayesian view on LMS, download G. Deng, 'A model-based approach for the development of LMS algorithms', *ISCAS-05 symposium*, 2005.

7 Linear Classification

7.1 Generative Classification

Probabilistic Classification


3. Use Bayes to calc. posterior class prob.
x_2
x_1
$p(y=\text{`b'}) = \pi_b$
$p(x|y=\text{`b'}) = N(x|\mu_b,\Sigma)$
$p(y=b|x_{new}) = \frac{p(x_{new}|y=b)p(y=b)}{p(x_{new}|y=b)p(y=b) + p(x_{new}|y=r)p(y=r)}$
$p(x|y=\text{`r'}) = N(x|\mu_r,\Sigma)$
$p(y=\text{`r'}) = \pi_r$

Classification Nomenclature/Notation

- Given data $\mathcal{D} = \{(\mathbf{x}_1, t_1), \ldots, (\mathbf{x}_N, t_N)\}$

- Inputs $\mathbf{x}_n \in \mathbb{R}^D$ are called *features*.
- The K *discrete* targets $\mathcal{C}_k$ are called *classes*.
- We will again use the 1-of-K notation for the discrete classes. Define the binary *class selection variable*

$$t_{nk} = \begin{cases} 1 & \text{if } t_n \text{ in class } \mathcal{C}_k \\ 0 & \text{else} \end{cases}$$

- Supervised learning goal: build a model for the joint

$$p(\mathbf{x}, t) = p(\mathbf{x}|t)p(t)$$

Model specification

- The plan for generative classification: model spec for the joint pdf $p(\mathbf{x}|t)p(t)$ and use Bayes to infer the posterior class probabilities

$$p(t|\mathbf{x}) = \frac{p(\mathbf{x}|t)p(t)}{\sum_t p(\mathbf{x}|t)p(t)}$$

.

- Assume Gaussian *class-conditional distributions* with *constant covariance matrix* across the classes,

$$p(\mathbf{x}|\mathcal{C}_k, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma})$$

  - with notational shorthand: $\mathcal{C}_k \equiv (t_k = 1)$
- Prior on class labels t_k is multinomial

$$p(\mathcal{C}_k|\boldsymbol{\pi}) = \pi_k$$

- As usual, the rest (inference for parameters and model prediction) through straight probability theory.

Parameter Inference for Classification

- Goal: ML estimation of $\{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}\}$ from data $\mathcal{D}$
- We will maximize the *joint* log-likelihood $\sum_n \log p(\mathbf{x}_n, t_n|\boldsymbol{\theta})$,

$$\begin{aligned}\log p(\mathcal{D}|\boldsymbol{\theta}) &= \sum_n \log \prod_k p(\mathbf{x}_n, t_{nk}|\boldsymbol{\theta})^{t_{nk}} = \sum_{n,k} t_{nk} \log p(\mathbf{x}_n, t_{nk}|\boldsymbol{\theta}) \\ &= \sum_{n,k} t_{nk} \log \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}) + \sum_{n,k} t_{nk} \log \pi_k \\ &= \boxed{\sum_{n,k} t_{nk} \underbrace{\log \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma})}_{\text{Gaussian dens. est.}} + \underbrace{\sum_k N_k \log \pi_k}_{\text{multinom. est.}}}\end{aligned}$$

- Problem breaks down into
  - *Multinomial density estimation for class priors* π_k
  - *Gaussian density estimation for parameters* $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}$

ML Estimation for Generative Classification

- Prior is ML for multinomial (done this before!)

$$\hat{\pi}_k = N_k/N \qquad \text{(class prior)}$$

- Now group the data into separate classes and do MVG ML for class-conditional parameters (done this as well).

$$\begin{aligned}
\hat{\boldsymbol{\mu}}_k &= \frac{\sum_n t_{nk}\mathbf{x}_n}{\sum_n t_{nk}} = \frac{1}{N_k}\sum_n t_{nk}\mathbf{x}_n && \text{(class-cond. mean)}\\
\hat{\boldsymbol{\Sigma}} &= \frac{1}{N}\sum_{n,k} t_{nk}(\mathbf{x}_n - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_n - \hat{\boldsymbol{\mu}}_k)^T && \text{(variance)}\\
&= \sum_k \hat{\pi}_k \cdot \underbrace{\left(\frac{1}{N_k}\sum_n t_{nk}(\mathbf{x}_n - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_n - \hat{\boldsymbol{\mu}}_k)^T\right)}_{\text{class-cond. variance}}
\end{aligned}$$

- Note that t_{nk} groups data from the same class.

Model Prediction for Generative Classification

- Given a 'new' input x, use Bayes rule to get posterior class probability

$$\begin{aligned}
p(\mathcal{C}_k|\mathbf{x},\boldsymbol{\theta}) &= \frac{p(\mathbf{x}|\mathcal{C}_k,\boldsymbol{\theta})p(\mathcal{C}_k|\pi)}{\sum_j p(\mathbf{x}|\mathcal{C}_j,\boldsymbol{\theta})p(\mathcal{C}_j|\pi)}\\
&= \frac{\pi_k \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right\}}{\sum_j \pi_j \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_j)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_j)\right\}}\\
&= \frac{\exp\left\{\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \log\pi_k\right\}}{\sum_j \exp\left\{\boldsymbol{\mu}_j^T\boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_j^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_j + \log\pi_j\right\}}\\
&= \exp\{\beta_k^T\mathbf{x} + \gamma_k\}/Z
\end{aligned}$$

  where $\beta_k = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k$, and $\gamma_k = -\frac{1}{2}\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \log\pi_k$ and $Z = \sum_j \exp\{\beta_j^T\mathbf{x} + \gamma_j\}$ is a normalizing factor.

- Note that the (softmax) function $\phi(a_k) = \frac{\exp\{a_k\}}{\sum_j \exp\{a_j\}}$ is by construction normalized, $\sum_k \phi(_k) = 1$.

Discrimination Boundaries

- The class posterior $\log p(\mathcal{C}_k|\mathbf{x}, \boldsymbol{\theta}) = \beta_k^T \mathbf{x} + \gamma_k - \log Z$ is a linear function of the input features.
- Thus, the contours of equal probability (*discriminant functions*) are lines (hyperplanes) in feature space

$$\log \frac{p(\mathcal{C}_k|\mathbf{x}, \boldsymbol{\theta})}{p(\mathcal{C}_j|\mathbf{x}, \boldsymbol{\theta})} = \beta_{kj}^T \mathbf{x} + \gamma_{kj} = 0$$

  where we defined $\beta_{kj} = \beta_k - \beta_j$ and similarly for γ_{kj}.
- (homework). What happens if we had not assumed class-independent variances $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}$? Are the discrimination functions still linear? quadratic?
- How to apply a trained classifier to a classification problem? E.g., choose class with maximum posterior class probability

$$k^* = \arg\max_k p(\mathcal{C}_k|\mathbf{x}_{new}, \boldsymbol{\theta}) = \arg\max_k \left(\beta_k^T \mathbf{x}_{new} + \gamma_k\right)$$

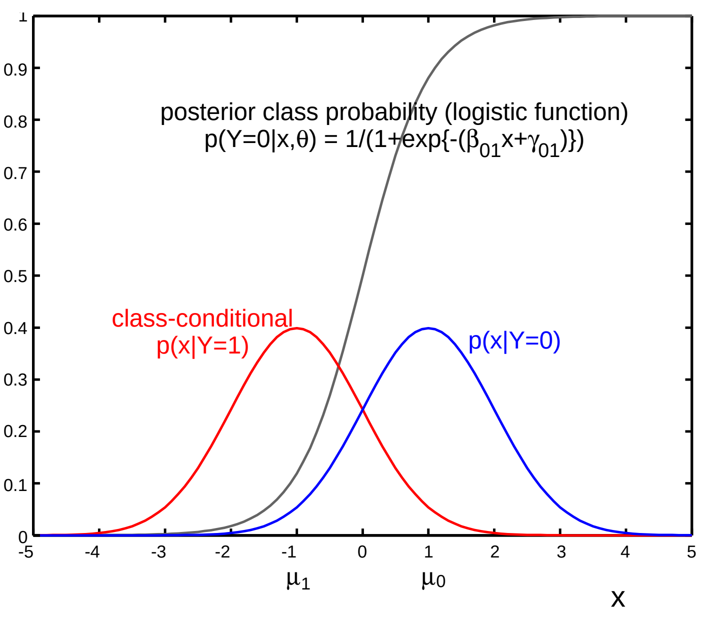
Example: Binary Classification

- Special case example: binary classification ($t \in \{0, 1\}$)

$$\begin{aligned} p(t = 1|\mathbf{x}, \boldsymbol{\theta}) &= \frac{\exp\{\beta_1^T \mathbf{x} + \gamma_1\}}{\exp\{\beta_1^T \mathbf{x} + \gamma_1\} + \exp\{\beta_0^T \mathbf{x} + \gamma_0\}} \\ &= \frac{1}{1 + \exp\{-\left(\beta_{10}^T \mathbf{x} + \gamma_{10}\right)\}} \\ p(t = 0|\mathbf{x}, \boldsymbol{\theta}) = 1 - p(t = 1|\mathbf{x}, \boldsymbol{\theta}) &= \frac{1}{1 + \exp\{-\left(\beta_{01}^T \mathbf{x} + \gamma_{01}\right)\}} \end{aligned}$$

- The function $\phi(a) \equiv 1/(1 + e^{-a})$ is called the *logistic* function, which is the binary case of the *softmax* function.

The Logistic Function as Posterior Probability



Recap Generative Classification

- Model spec. $p(\mathbf{x}, \mathcal{C}_k | \boldsymbol{\theta}) = \pi_k \cdot \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma})$
- If the class-conditional distributions are Gaussian with equal covariance matrices across classes ($\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}$), then the discriminant functions are hyperplanes in feature space.
- ML estimation for $\{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}\}$ breaks down to simple density estimation for Gaussian and multinomial.
- Posterior class probability is a softmax (or logistic) function

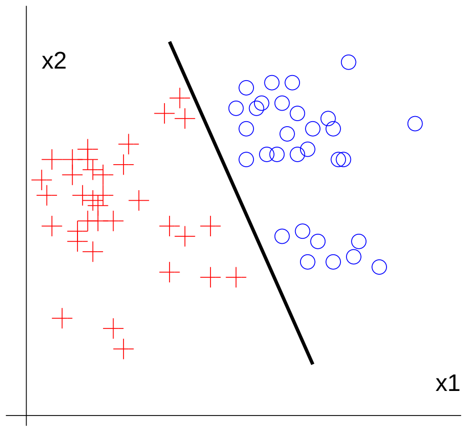
$$p(\mathcal{C}_k|\mathbf{x}, \boldsymbol{\theta}) = \exp\{\beta_k^T \mathbf{x} + \gamma_k\}/Z$$

where $\beta_k = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k$ and $\gamma_k = -\frac{1}{2}\boldsymbol{\mu}_k^T \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \log \pi_k$.

7.2 Discriminative Classification

Difficult Class-Conditional Densities

- Sometimes, the class-conditional densities are very non-Gaussian, but the linear discriminative boundary looks easy enough.



Discriminative Linear Classification

- Sometimes, the precise assumptions of the generative model
$$p(\mathbf{x}, \mathcal{C}_k|\boldsymbol{\theta}) = \pi_k \cdot \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \mathbf{\Sigma})$$
are not met.
- Alternative approach: model the posterior $p(\mathcal{C}_k|\mathbf{x})$ directly, without any assumptions on the class densities.
- What model should we use for $p(\mathcal{C}_k|\mathbf{x})$?
- Get inspiration from the generative approach: choose the familiar softmax structure for the posterior class probability
$$p(\mathcal{C}_k|\mathbf{x}, \boldsymbol{\theta}_k) = \frac{e^{\boldsymbol{\theta}_k^T \mathbf{x}}}{\sum_j e^{\boldsymbol{\theta}_j^T \mathbf{x}}}$$
but *do not impose a Gaussian structure on the classes.*

Discriminative vs. Generative Classification

- *Two key differences* for discriminative approach
  1. Parameter $\boldsymbol{\theta}_k$ is *not* structured into $\{\boldsymbol{\mu}_k, \mathbf{\Sigma}, \pi_k\}$. This provides discriminative approach with more flexibility.
  2. ML learning by optimization of *conditional* likelihood $\prod_n p(t_n|\mathbf{x}_n, \boldsymbol{\theta})$ rather than *joint* likelihood $\prod_n p(t_n, \mathbf{x}_n|\boldsymbol{\theta})$.
- As we will see, ML estimation for discriminative classification is more complex than for the linear Gaussian generative approach.

- Discriminative model-based prediction for a new input $\mathbf{x}$ is easy, namely substitute the ML estimate in the model to get

$$p(\mathcal{C}_k|\,\mathbf{x},\hat{\boldsymbol{\theta}}) = \frac{e^{\hat{\boldsymbol{\theta}}_k^T\mathbf{x}}}{\sum_j e^{\hat{\boldsymbol{\theta}}_j^T\mathbf{x}}} \propto e^{\hat{\boldsymbol{\theta}}_k^T\mathbf{x}}$$

ML Estimation for Discriminative Classification

- Work out the conditional log-likelihood (t_k is the target, while $y_k = p(\mathcal{C}_k|\mathbf{x},\boldsymbol{\theta})$ is the model prediction).

$$L(\boldsymbol{\theta}) = \log \prod_n \prod_k \underbrace{p(\mathcal{C}_k|\mathbf{x}_n,\boldsymbol{\theta})}_{y_{nk}}{}^{t_{nk}} = \sum_{n,k} t_{nk} \log y_{nk}$$

- Note that softmax $\phi_k = e^{z_k}/\sum_j e^{z_j}$ has analytical derivative,

$$\begin{aligned}\frac{\partial \phi_k}{\partial z_j} &= \frac{(\sum_j e^{z_j})e^{z_k}\delta_{kj} - e^{z_j}e^{z_k}}{(\sum_j e^{z_j})^2} = \frac{e^{z_k}}{\sum_j e^{z_j}}\delta_{kj} - \frac{e^{z_j}}{\sum_j e^{z_j}}\frac{e^{z_k}}{\sum_j e^{z_j}}\\ &= \phi_k \cdot (\delta_{kj} - \phi_j)\end{aligned}$$

- Taking the derivative (or: how to spend a hour ...)

$$\begin{aligned}\frac{\partial L}{\partial \boldsymbol{\theta}_j} &= \sum_{n,k} \frac{\partial L_{nk}}{\partial y_{nk}} \frac{\partial y_{nk}}{\partial z_{nj}} \frac{\partial z_{nj}}{\partial \boldsymbol{\theta}_j} = \sum_{n,k} \frac{t_{nk}}{y_{nk}} y_{nk}(\delta_{kj} - y_{nj})\,\mathbf{x}_n\\ &= \sum_n \left(t_{nj}(1-y_{nj}) - \sum_{k\neq j} t_{nk} y_{nj} \right) \mathbf{x}_n = -\sum_n (t_{nj} - y_{nj})\,\mathbf{x}_n\end{aligned}$$

Linear vs. Logistic Regression

- Compare the gradients for linear and logistic regression,

$$\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = -\sum_n (t_n - \boldsymbol{\theta}^T \mathbf{x}_n)\,\mathbf{x}_n \qquad \text{(linear regression)}$$

$$\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = -\sum_n (t_n - \frac{1}{1+e^{-\boldsymbol{\theta}^T\mathbf{x}_n}})\,\mathbf{x}_n \qquad \text{(logistic regression)}$$

- The parameter vector $\boldsymbol{\theta}$ for both linear regression and logistic regression can be estimated through gradient-based adaptation, e.g.,

$$\begin{aligned}\boldsymbol{\theta}_{n+1} &= \boldsymbol{\theta}_n - \eta \cdot \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta})\\ &= \boldsymbol{\theta}_n + \sum_n (t_n - \frac{1}{1+e^{-\boldsymbol{\theta}^T\mathbf{x}_n}})\,\mathbf{x}_n\end{aligned}$$

- Run `demo_classification.m` in matlab

Recap Classification

Generative

- Like *density estimation*, model joint prob.

$$p(\mathcal{C}_k)p(\mathbf{x}|\mathcal{C}_k) = \pi_k \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma})$$

- Leads to *softmax* posterior class probability

$$p(\mathcal{C}_k|\mathbf{x}, \boldsymbol{\theta}) = e^{\boldsymbol{\theta}_k^T \mathbf{x}}/Z$$

  with *structured* $\boldsymbol{\theta}$

- For Gaussian $p(\mathbf{x}|\mathcal{C}_k)$ and multinomial priors,

$$\boldsymbol{\theta}_k = \begin{bmatrix} -\frac{1}{2}\boldsymbol{\mu}_k^T \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \log \pi_k \\ \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k \end{bmatrix}$$

Discriminative

- Like (linear) *regression*, model conditional

$$p(\mathcal{C}_k|\mathbf{x}, \boldsymbol{\theta})$$

- *Choose* also softmax posterior class probability

$$p(\mathcal{C}_k|\mathbf{x}, \boldsymbol{\theta}) = e^{\boldsymbol{\theta}_k^T \mathbf{x}}/Z$$

  with 'free' $\boldsymbol{\theta}$

- Find $\hat{\boldsymbol{\theta}}$ through gradient-based adaptation

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = -\sum_n (t_n - \frac{1}{1 + e^{-\boldsymbol{\theta}^T \mathbf{x}_n}})\mathbf{x}_n$$

8 Clustering with Gaussian Mixture Models

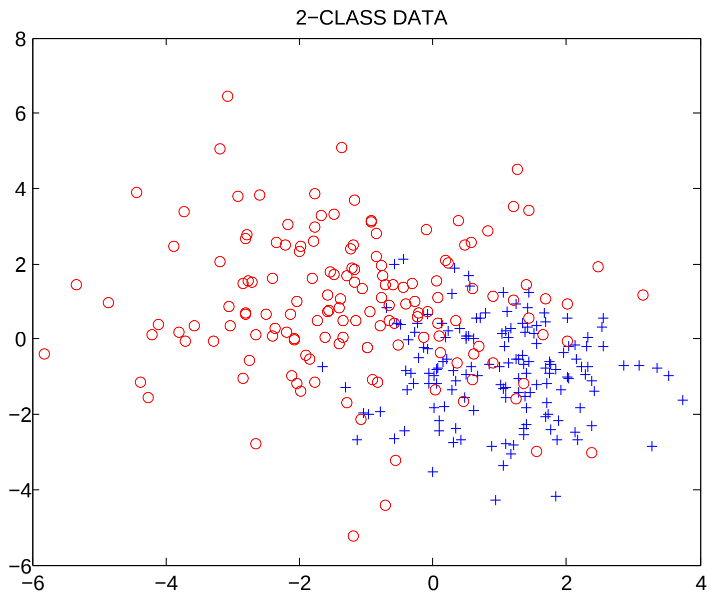
Limitations of Simple IID Gaussian Models

- Sofar, model inference was solved analytically, but we used strong assumptions
  - IID sampling, $p(\mathcal{D}) = \prod_n p(\mathbf{x}_n)$
  - Simple Gaussian (or multinomial) PDFs, $p(\mathbf{x}_n) \sim \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}, \boldsymbol{\Sigma})$
- Some limitations of Simple Gaussian Models with IID Sampling
  - What if the PDF is multi-modal (or is just not Gaussian in any other way)?
  - Covariance matrix $\boldsymbol{\Sigma}$ has $D(D+1)/2$ parameters. This quickly becomes a very large number for increasing dimension D.
  - Temporal signals are often not IID.

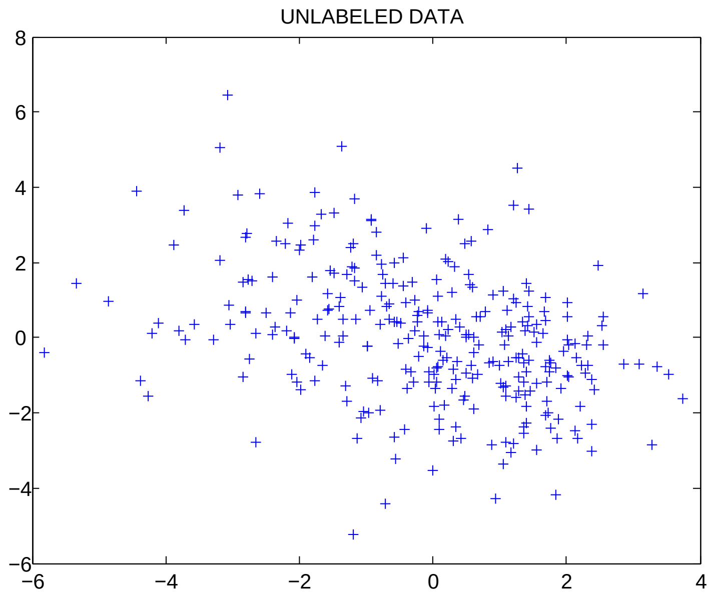
Towards More Flexible Models

- What if the PDF is multi-modal (or is just not Gaussian in any other way)?
  - Discrete *latent* variable models. (mixture models)
- Covariance matrix $\mathbf{\Sigma}$ has $D(D+1)/2$ parameters. This quickly becomes very large for increasing dimension D.
  - continuous *latent* variable models. ('dimensionality reduction' models)
- Temporal signals are often not IID.
  - introduce *Markov dependencies* and *latent* state variable models.

What if the Data are Not like This ...



... but like This

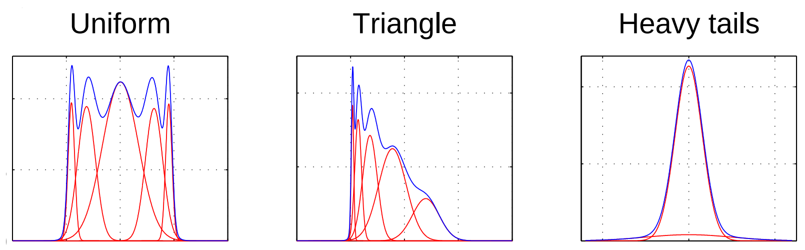


Unobserved Classes

- Consider again a set of observed data $\mathcal{D} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$
- This time we suspect that there are unobserved class labels that would help explain (or predict) the data, e.g.,
  - the observed data are the color of living things; the unobserved classes are animals and plants.
  - observed are wheel sizes; unobserved categories are trucks and personal cars.
  - observed is an audio signal; unobserved classes include speech, music, traffic noise, etc.
- Classification problems with unobserved classes are called *Clustering* problems. The learning algorithm needs to discover the classes from the observed data.

Latent Variable Model Specification

- If the categories were observed as well, these data could be nicely modeled by the previously discussed generative classification framework.
- Introduce the 1-of-K variable $\mathbf{z} = (z_1, \dots, z_K)^T$ to represent the unobserved classes.



The red curves show the (weighted) Gaussians; the blue curve the resulting density.

- Use completely *equivalent model assumptions to linear generative classification*, (except now the class labels z_k are not observed),

$$\begin{aligned} p(\mathbf{x}_n) &= \sum_{k=1}^{K} p(z_{nk})\, p(\mathbf{x}_n|z_{nk}) \\ &= \sum_k \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \end{aligned}$$

- This model is called *Gaussian Mixture Model*.

Gaussian Mixture Models

- GMMs are *universal approximators of densities* (as long as there are enough Gaussians of course)

Inference: Log-Likelihood for GMM

- The log-likelihood for observed data $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$,

$$\begin{aligned} \log p(\mathcal{D}|\boldsymbol{\theta}) &\overset{\text{IID}}{=} \sum_n \log p(\mathbf{x}_n|\boldsymbol{\theta}) = \sum_n \log \sum_{\mathbf{z}_n} p(\mathbf{x}_n, \mathbf{z}_n|\boldsymbol{\theta}) \\ &= \sum_n \log \sum_{\mathbf{z}_n} p(\mathbf{z}_n|\boldsymbol{\theta}) p(\mathbf{x}_n|\mathbf{z}_n, \boldsymbol{\theta}) \\ &= \sum_n \log \sum_k p(z_{nk} = 1|\boldsymbol{\theta}) p(\mathbf{x}_n|z_{nk} = 1, \boldsymbol{\theta}) \\ &= \boxed{\sum_n \log \sum_k \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)} \end{aligned}$$

... and now the log-of-sum cannot be further simplified.

- Compare to classification:

$$\sum_k N_k \log \pi_k + \sum_{n,k} t_{nk} \log \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \mathbf{\Sigma})$$

Posterior Responsibility is a Soft Class Indicator

- Consider the (posterior) expectation for the (hidden) class labels

$$\begin{aligned}\gamma_{nk} &\equiv \mathbb{E}[z_{nk}|\mathbf{x}_n, \boldsymbol{\theta}] = 0 \times p(z_{nk}=0|\mathbf{x}_n, \boldsymbol{\theta}) + 1 \times p(z_{nk}=1|\mathbf{x}_n, \boldsymbol{\theta}) \\ &= p(z_{nk}=1|\mathbf{x}_n, \boldsymbol{\theta}) = \frac{p(\mathbf{x}_n|z_{nk}=1)p(z_{nk}=1)}{\sum_j p(\mathbf{x}_n|z_{nj}=1)p(z_{nj}=1)} \\ &= \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \mathbf{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \mathbf{\Sigma}_j)}\end{aligned}$$

- Note that $0 \leq \gamma_{nk} \leq 1$ and is available (i.e., can be evaluated).
- γ_{nk} are (soft) *reponsibilities.*
- PLAN: Let's use the reponsibilities γ_{nk} (rather than the binary class indicators t_{nk}) and apply the classification formulas.

ML estimation for Clustering

- Try parameter updates (like conditional Gaussian classification):

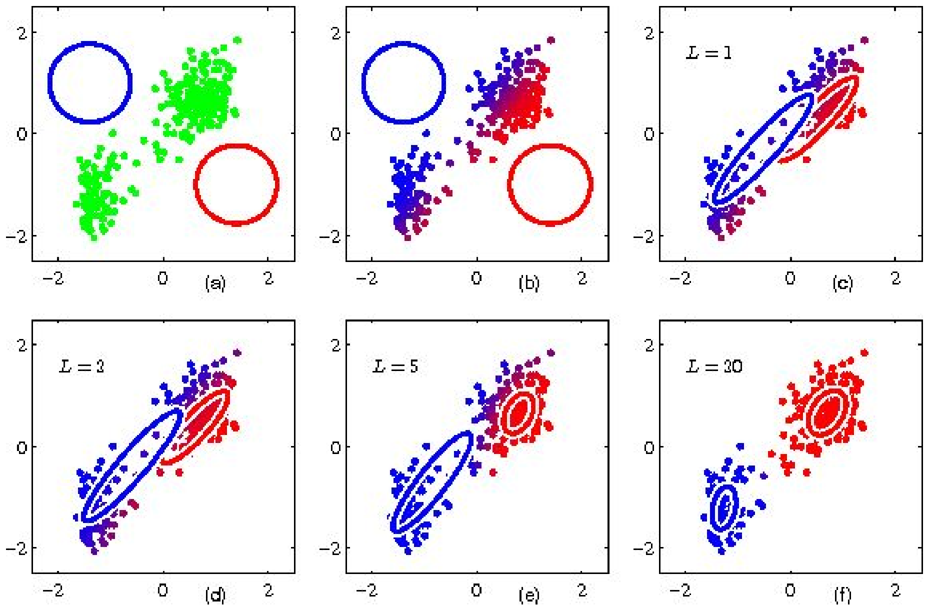
$$\hat{\pi}_k = \frac{N_k}{N};\ \hat{\boldsymbol{\mu}}_k = \frac{1}{N_k}\sum_n \gamma_{nk}\mathbf{x}_n;\ \hat{\mathbf{\Sigma}}_k = \frac{1}{N_k}\sum_n \gamma_{nk}(\mathbf{x}_n - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_n - \hat{\boldsymbol{\mu}}_k)^T$$

  where $N_k = \sum_n \gamma_{nk}$.

- But wait, the responsibilities $\gamma_{nk} = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k,\mathbf{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j,\mathbf{\Sigma}_j)}$ are a function of the model parameters $\{\boldsymbol{\pi}, \boldsymbol{\mu}, \mathbf{\Sigma}\}$ and the parameter updates depend on the responsibilities ...
- Solution(?): iterate between updating the responsibilities γ_{nk} and the model parameters $\{\boldsymbol{\pi}, \boldsymbol{\mu}, \mathbf{\Sigma}\}$.
- This iteration works (!) and has a name:

$$\boxed{\textit{Expectation-Maximization (EM) Algorithm}}$$

EM for GMM Simulation



9 The (General) EM Algorithm

Intuition EM Algorithm

- Let's denote all observed data by $N \times D$ matrix $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^T$ and the set of all latent variables by $N \times K$ matrix $\mathbf{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_N)^T$.
- The ML optimization problem with observed data $\mathbf{X}$ is,

$$\hat{\boldsymbol{\theta}} = \arg\max_{\theta} \log p(\mathbf{X}|\boldsymbol{\theta})$$

  but this appears to be a very difficult optimization problem.
- *Plan:* Introduce (extra) latent variables $\mathbf{Z}$ such that the *complete-data log-likelihood* $\log p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$ is easily maximized.
- Alas, $\mathbf{Z}$ are latent variables, i.e. NOT observed, and we cannot evaluate $p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$ as a function of $\boldsymbol{\theta}$ only.
- *idea 1*: optimize instead the *expected* complete-data likelihood

$$\sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}) \log p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$$

  which is no longer a function of (the unobserved) $\mathbf{Z}$.

Intuition EM Algorithm, cont'd

- Note that the posterior $p(\mathbf{Z}|\mathbf{X},\boldsymbol{\theta})$ depends on $\boldsymbol{\theta}$.
- *idea 2*: Use *iterative optimization* so that we can use the estimate $\hat{\boldsymbol{\theta}}$ from the previous optimization step in order to compute the posteriors $p(\mathbf{Z}|\mathbf{X},\hat{\boldsymbol{\theta}})$.
- These ideas lead to the iterative *EM Algorithm*,

$$\begin{aligned}&\text{(E-step): evaluate } p(\mathbf{Z}|\mathbf{X},\boldsymbol{\theta}^{\text{old}}) \quad \text{('responsibility')}\\ &\text{(M-step): evaluate } \boldsymbol{\theta}^{\text{new}} := \arg\max_{\boldsymbol{\theta}} \mathcal{Q}(\boldsymbol{\theta},\boldsymbol{\theta}^{\text{old}})\end{aligned}$$

  where $\mathcal{Q}(\boldsymbol{\theta},\boldsymbol{\theta}^{\text{old}}) = \sum_z p(\mathbf{Z}|\mathbf{X},\boldsymbol{\theta}^{\text{old}}) \log p(\mathbf{X},\mathbf{Z}|\boldsymbol{\theta})$

- (In the next few slides), we will show that the EM algorithm is guaranteed to converge to a local maximum, or saddle-point, of the *observed-data* likelihood function $p(\mathbf{X}|\boldsymbol{\theta})$.

Concavity and Jensen's Inequality

- $f(x)$ is *concave*$\frown$ over (a,b) if, for all $x_1, x_2 \in (a,b)$ and $0 \le \lambda \le 1$,

$$f(\lambda x_1 + (1-\lambda)x_2) \ge \lambda f(x_1) + (1-\lambda) f(x_2)$$

- *Jensen's Inequality*: If f is concave$\frown$ and x is a RV then:

$$f(\mathbb{E}[x]) \ge \mathbb{E}[f(x)]$$

- Example

$$\log(\mathbb{E}[x]) \ge \mathbb{E}[\log(x)]$$

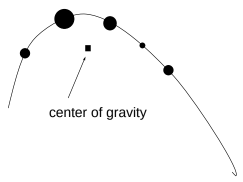
The 'EM Trick'

- The log-likelihood based on data set $\mathbf{X}$ is

$$L(\boldsymbol{\theta}) = \log p(\mathbf{X}|\boldsymbol{\theta}) = \log \sum_{\mathbf{Z}} p(\mathbf{X},\mathbf{Z}|\boldsymbol{\theta})$$

- For *any* distribution $q(\mathbf{Z})$, we can write

$$\begin{aligned}L(\boldsymbol{\theta}) &= \log \sum_{\mathbf{Z}} q(\mathbf{Z}) \frac{p(\mathbf{X},\mathbf{Z}|\boldsymbol{\theta})}{q(\mathbf{Z})}\\ &\overset{Jensen}{\ge} \sum_{\mathbf{Z}} q(\mathbf{Z}) \log \frac{p(\mathbf{X},\mathbf{Z}|\boldsymbol{\theta})}{q(\mathbf{Z})}\\ &\equiv \mathcal{L}(\boldsymbol{\theta},q) \qquad \text{('free energy')}\end{aligned}$$



(Physical interpretation). Put masses p_i at locations $(x_i, f(x_i))$. Then, *center of gravity* at $(\mathbb{E}[x], \mathbb{E}[f(x)])$ lies under the curve (at location $(\mathbb{E}[x], f(\mathbb{E}[x]))$)

The 'EM Trick', Cont'd

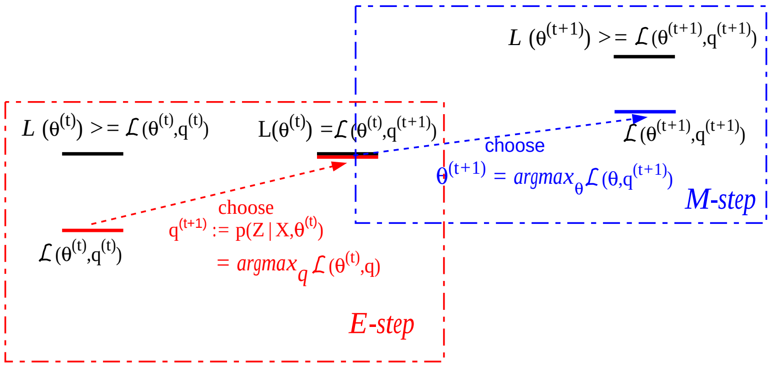
- Furthermore, if we choose $q(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta})$, then $\mathcal{L}(\boldsymbol{\theta}, q) = L(\boldsymbol{\theta})$ (is maximal), since

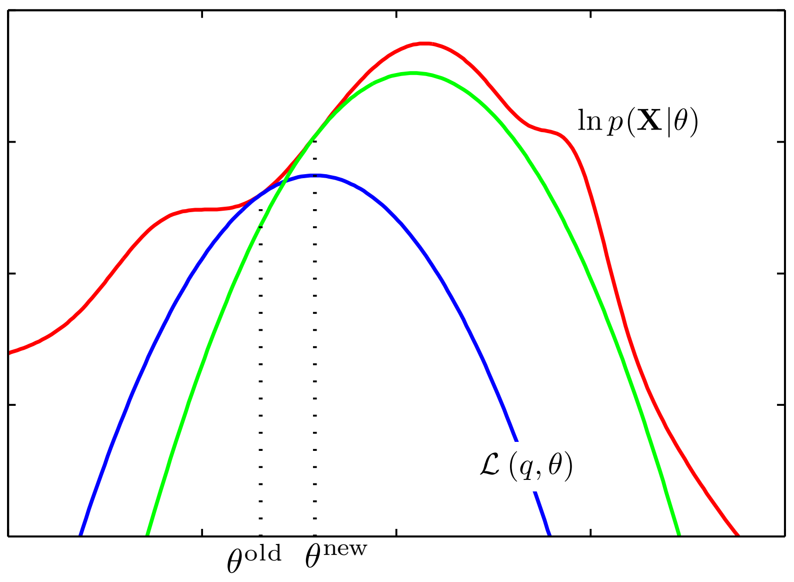
$$
\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}, q) &= \sum_{\mathbf{Z}} q(\mathbf{Z}) \log \frac{p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})}{q(\mathbf{Z})} = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}) \log \frac{p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})}{p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta})} \\
&= \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}) \log \frac{p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}) p(\mathbf{X}|\boldsymbol{\theta})}{p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta})} = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}) \log p(\mathbf{X}|\boldsymbol{\theta}) \\
&= \log p(\mathbf{X}|\boldsymbol{\theta}) \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}) = \log p(\mathbf{X}|\boldsymbol{\theta}) = L(\boldsymbol{\theta})
\end{aligned}
$$

- $\Longrightarrow$ We have just shown that the following procedure (coordinate ascent on $\mathcal{L}$) will increase the log-likelihood:

$$
\begin{aligned}
q^{\text{new}} &:= p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}}) = \arg\max_q \mathcal{L}(\boldsymbol{\theta}^{\text{old}}, q) && \text{(E-step)} \\
\boldsymbol{\theta}^{\text{new}} &:= \arg\max_{\theta} \mathcal{L}(\boldsymbol{\theta}, q^{\text{new}}) && \text{(M-step)}
\end{aligned}
$$

EM Geometry


$L(\theta^{(t+1)}) >= \mathcal{L}(\theta^{(t+1)}, q^{(t+1)})$
$L(\theta^{(t)}) >= \mathcal{L}(\theta^{(t)}, q^{(t)})$
$L(\theta^{(t)}) = \mathcal{L}(\theta^{(t)}, q^{(t+1)})$
$\mathcal{L}(\theta^{(t+1)}, q^{(t+1)})$
choose
$\theta^{(t+1)} = argmax_\theta \mathcal{L}(\theta, q^{(t+1)})$
M-step
$\mathcal{L}(\theta^{(t)}, q^{(t)})$
choose
$q^{(t+1)} := p(Z|X, \theta^{(t)})$
$= argmax_q \mathcal{L}(\theta^{(t)}, q)$
E-step



- In the E-step, the log-likelihood $L(\boldsymbol{\theta})$ remains the same, but $\mathcal{L}$ is pushed up to equal $L(\boldsymbol{\theta})$.
- In the M-step, $\mathcal{L}$ increases even more and $L(\boldsymbol{\theta})$ *must* follow (since always $L(\boldsymbol{\theta}) \geq \mathcal{L}(q, \boldsymbol{\theta})$)

EM involves optimizing lower bound

EM for GMM revisited

- (total) log-likelihood $L(\boldsymbol{\theta}) = \sum_n \log \sum_k \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$
- E-step: compute responsibilities for latent classes

$$\gamma_{nk} = p(z_{nk} = 1|\mathbf{x}_n, \boldsymbol{\theta}^{\text{old}}) = \frac{\pi_k^{\text{old}} \mathcal{N}(\mathbf{x}_n|\boldsymbol{\theta}_k^{\text{old}})}{\sum_j \pi_j^{\text{old}} \mathcal{N}(\mathbf{x}_n|\boldsymbol{\theta}_j^{\text{old}})}$$

- M-step: Maximize expected complete data log-likelihood

$$\begin{aligned}\mathbb{E}_{\mathbf{Z}}[\log p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi})] &= \sum_n \sum_k \gamma_{nk} \log p(\mathbf{x}_n, z_{nk} = 1|\boldsymbol{\theta}) \\ &= \sum_{nk} \gamma_{nk} \log \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) + \sum_{nk} \gamma_{nk} \log \pi_k\end{aligned}$$

  We've maximized this before, see section on Density estimation (Gaussian, multinomial)

EM Example–Three Coins

- You have three coins in your pocket
  - Coin 0: $p(\text{Head}) = \lambda$
  - Coin 1: $p(\text{Head}) = \rho$
  - Coin 2: $p(\text{Head}) = \theta$
- (Scenario). Toss coin 0. If Head comes up, toss three times with coin 1; otherwise, toss three times with coin 2.
- The observed sequences *after* each toss with coin 0 were $\langle$HHH$\rangle$, $\langle$HTH$\rangle$, $\langle$HHT$\rangle$, and $\langle$HTT$\rangle$
- [Q.] Estimate most likely values for λ, ρ and θ
- [A.] homework. Use EM.

Some Properties of EM

- EM is a general procedure for learning in the presence of unobserved variables.
- In a sense, it is a family of algorithms. The update rules you will derive depend on the probability model assumed.
- (Good!) *No tuning parameters* such a learning rate, unlike gradient descent-type algorithms
- (Bad). EM is an iterative procedure that is very sensitive to initial conditions! EM converges to a *local optimum.*
- Start from trash $\rightarrow$ end up with trash. Hence, we need a good and fast initialization procedure (often used: K-Means)
- Also used to train HMMs, etc.

(OPTIONAL) The Kullback-Leibler Divergence

- The *Kullback-Leibler Divergence* or *relative entropy* between two distributions q and p is defined as

$$\text{KL}(q\|p) \equiv \sum_z q(z) \log \frac{q(z)}{p(z)}$$

- In general $\text{KL}(q\|p) \neq \text{KL}(p\|q)$
- Theorem *(Gibbs Inequality)*:

$$\boxed{\text{KL}(q\|p) \geq 0 \quad \text{with equality only if} \quad p = q}$$

- Proof *(use Jensen Inequality)* Define $f(u) = \log 1/u$ (f is convex$^{\smile}$) and let $u = p(x)/q(x)$.

$$\begin{aligned}\mathrm{KL}(q\|p) &= \mathbb{E}_q[f(u)] \overset{\textit{Jensen}}{\geq} f\left(\mathbb{E}_q[u]\right) = f\left(\sum_x q(x)\frac{p(x)}{q(x)}\right)\\ &= \log(1/\sum_x p(x)) = 0\end{aligned}$$

with equality only if u is constant, i.e., $q(x) = p(x)$.

(OPTIONAL) The Free Energy Functional

- For *any* distribution $q(\mathbf{Z})$, the following holds

$$\begin{aligned}L(\boldsymbol{\theta}) = \log p(\mathbf{X}|\boldsymbol{\theta}) &= \sum_{\mathbf{Z}} q(\mathbf{Z}) \log p(\mathbf{X}|\boldsymbol{\theta})\\ &= \sum_{\mathbf{Z}} q(\mathbf{Z})\left[\log p(\mathbf{X}|\boldsymbol{\theta}) - \log\frac{q(\mathbf{Z})}{p(\mathbf{Z}|\mathbf{X})}\right] + \sum_{\mathbf{Z}} q(\mathbf{Z})\log\frac{q(\mathbf{Z})}{p(\mathbf{Z}|\mathbf{X})}\\ &= \sum_{\mathbf{Z}} q(\mathbf{Z})\log\frac{p(\mathbf{X},\mathbf{Z}|\boldsymbol{\theta})}{q(\mathbf{Z})} + \mathrm{KL}(q(\mathbf{Z})\|p(\mathbf{Z}|\mathbf{X}))\\ &\overset{\textit{Gibbs}}{\geq} \sum_{\mathbf{Z}} q(\mathbf{Z})\log\frac{p(\mathbf{X},\mathbf{Z}|\boldsymbol{\theta})}{q(\mathbf{Z})} \qquad \text{(`=' iff } q(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{X}))\\ &= \sum_{\mathbf{Z}} q(\mathbf{Z})\log p(\mathbf{X},\mathbf{Z}|\boldsymbol{\theta}) - \sum_{\mathbf{Z}} q(\mathbf{Z})\log q(\mathbf{Z})\\ &= \langle\log p(\mathbf{X},\mathbf{Z}|\boldsymbol{\theta})\rangle_q + \mathcal{H}(q)\\ &\equiv \mathcal{L}(\boldsymbol{\theta}, q) \qquad \text{(free energy)}\end{aligned}$$

10 Continuous Latent Variable Models–FA, PCA, ICA

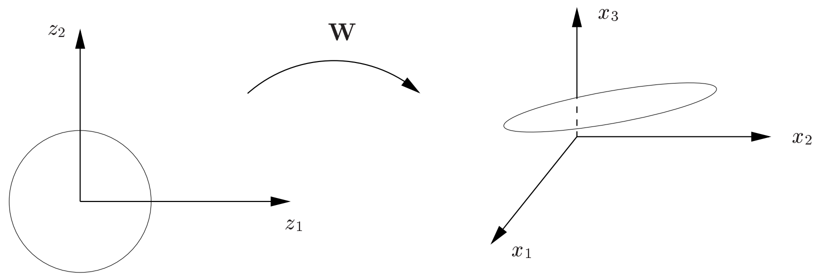
Continuous Latent Variables

- Mixture models use a discrete class variable.
- Sometimes, it is more appropriate to think in terms of *continuous* underlying causes (factors) that control the observed data.
- E.g., observe test results for subjects: English, Spanish and French

$$\underbrace{\begin{bmatrix} x_1(\textit{English})\\ x_2(\textit{Spanish})\\ x_3(\textit{French})\end{bmatrix}}_{\text{observed}} = \begin{bmatrix}\lambda_{11},\lambda_{12}\\ \lambda_{21},\lambda_{22}\\ \lambda_{31},\lambda_{32}\end{bmatrix}\cdot\underbrace{\begin{bmatrix} z_1(\textit{illiteracy})\\ z_2(\textit{intelligence})\end{bmatrix}}_{\text{causes}} + \underbrace{\begin{bmatrix} v_1\\ v_2\\ v_3\end{bmatrix}}_{\text{noise}}$$

- (Unsupervised Regression). This is like (linear) regression with unobserved inputs.

Dimensionality Reduction



- If the dimension for the hidden 'causes' ($\mathbf{z}$) is smaller than for the observed data ($\mathbf{x}$), then the model (tries to) achieve *dimensionality reduction.*
- Think of this as an observed data pancake in a 3-D space.

Why Dimensionality Reduction?

- Key applications include *compression* (store $\mathbf{z}$ rather than $\mathbf{x}$) and *visualization.*
- Compression through *real-valued* latent variables can be far more efficient than with discrete clusters.
- E.g., with two 8-bit hidden factors, one can describe $2^{16} \approx 10^5$ settings; this would take 2^{16} clusters!
- *Noise reduction* (e.g. in biomedical, financial or speech signals), *efficient representation and communication* and *feature extraction* (e.g. as a pre-processor for classification) can also be achieved through compression.

Factor Analysis Model specification

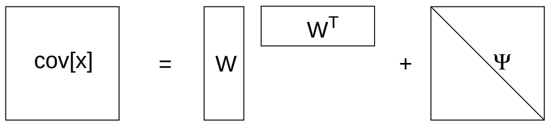
- Introduce observation vector $\mathbf{x} \in \mathbb{R}^D$ and M-dim ($M < D$) real-valued *latent factor* vector $\mathbf{z}$.

$$\mathbf{x} = \mathbf{W}\mathbf{z} + \boldsymbol{\mu} + \boldsymbol{\epsilon} \qquad \mathbf{z} \sim \mathcal{N}(0, I) \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \Psi)$$

or equivalently

$$p(\mathbf{x}|\mathbf{z}, \theta) = \mathcal{N}(\mathbf{x}|\, \mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \Psi) \qquad \text{(likelihood)}$$
$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\, 0, I) \qquad \text{(prior)}$$

where $\mathbf{W}$ is the $(D \times M)$-dim *factor loading matrix* and *observation noise covariance matrix* Ψ *is diagonal.*



- Compare to linear regression: $p(t|\mathbf{x}) = \mathcal{N}(t|\boldsymbol{\theta}^T\mathbf{x}, \sigma^2)$
- Our goal: Given observations $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, find (1) ML estimates for the parameters $\boldsymbol{\theta} = \{\mathbf{W}, \boldsymbol{\mu}, \Psi\}$ and (2) the posterior $p(\mathbf{z}|\mathbf{x})$.

Model Analysis: The Marginal Distribution $p(\mathbf{x})$

- Note: since the product of Gaussians is Gaussian, the joint distribution $p(\mathbf{x}, \mathbf{z})$, the marginal $p(\mathbf{x})$ and the conditional $p(\mathbf{z}|\mathbf{x})$ are also Gaussian.
- The marginal distribution for the observed data is

$$\boxed{p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\,\boldsymbol{\mu}, \mathbf{W}\mathbf{W}^T + \Psi)}$$

since

$$\begin{aligned}\mathbb{E}[\mathbf{x}] &= \mathbb{E}[\mathbf{W}\mathbf{z} + \boldsymbol{\mu} + \boldsymbol{\epsilon}] = \mathbf{W}\mathbb{E}[\mathbf{z}] + \boldsymbol{\mu} + \mathbb{E}[\boldsymbol{\epsilon}] = \boldsymbol{\mu} \\ \mathrm{var}[\mathbf{x}] &= \mathbb{E}[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T] = \mathbb{E}[(\mathbf{W}\mathbf{z} + \boldsymbol{\epsilon})(\mathbf{W}\mathbf{z} + \boldsymbol{\epsilon})^T] \\ &= \mathbf{W}\mathbb{E}[\mathbf{z}\mathbf{z}^T]\mathbf{W}^T + \mathbb{E}[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T] = \mathbf{W}\mathbf{W}^T + \Psi\end{aligned}$$

- Apparently, *this (FA) model is just a multivariate Gaussian* $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with the restriction that $\boldsymbol{\Sigma} = \mathbf{W}\mathbf{W}^T + \Psi$.

FA vs. Multivariate Gaussian modeling

- The effective covariance is the low-rank outer product of two long skinny matrices plus a diagonal matrix.
- Number of free parameters
  - $D(D+1)/2$ for full Gaussian covariance $\boldsymbol{\Sigma}$
  - $D(M+1)$ for FA model where $\boldsymbol{\Sigma} = \mathbf{W}\mathbf{W}^T + \Psi$.
  - D for diagonal Gaussian covariance $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_i^2)$
- $\Longrightarrow$ Factor Analysis model provides Gaussian model of *intermediate complexity*.

The Factor Loading Matrix W is Not Unique

- The factor loading matrix $\mathbf{W}$ can only be estimated up to a rotation matrix $\mathbf{R}$. Namely, if we rotate $\mathbf{W} \to \mathbf{WR}$, then the observed variance

$$\mathbf{WR}(\mathbf{WR})^T + \Psi = \mathbf{WRR}^T\mathbf{W}^T + \Psi = \mathbf{WW}^T + \Psi$$

  does not change. (N.B., a rotation (or orthogonal) matrix $\mathbf{R}$ is a matrix such that $\mathbf{R}^T\mathbf{R} = \mathbf{RR}^T = I$).

- Two persons that estimate ML parameters for FA (or pPCA) on the same data are *not guaranteed to find the same parameters* (since any rotation of $\mathbf{W}$ is equally likely).

- $\Rightarrow$ we learn latent *subspaces* rather than individual components. (One has to be careful when interpreting the numerical values of $\mathbf{W}$ and $\mathbf{z}$).

Constraints on the Noise Variance Ψ

- When doing ML estimation for the parameters, a trivial solution for the covariance matrix $\mathbf{WW}^T + \Psi$ is setting $\hat{\mathbf{W}} = 0$ and $\hat{\Psi}$ equal to the sample variance of the data.

- In this case, all data correlation is explained as noise. (We'd like to avoid this).

$\Rightarrow$ The FA model is uninteresting without some restriction on the sensor noise covariance matrix Ψ.

- In *Factor Analysis* (FA), Ψ is restricted to be *diagonal*:

$$\Psi = \mathrm{diag}(\psi_i)$$

- If Ψ is diagonal, all correlations between the (D) components of $\mathbf{x}$ *must be explained* by the rank-M matrix $\mathbf{WW}^T$.

Further Constraints on Ψ

- In *(probabilistic) Principal Component Analysis* (pPCA), the variances are further restricted to be the same,

$$\Psi = \sigma^2 I$$

- $\Rightarrow$ *The difference between FA and pPCA lies entirely in the choice of* Ψ.

- If we go even further and discard of the noise model altogether by

$$\Psi = \lim_{\sigma^2 \to 0} \sigma^2 I$$

  and in addition constrain the columns of $\mathbf{W}$ to be orthonormal, then we obtain 'regular' *PCA*, which is a well-known deterministic procedure for dimensionality reduction (that predates pPCA).

⇒ FA, pPCA and PCA differ only by the model for the noise variance (namely, diagonal, isotropic and 'not').

Typical Applications

- In PCA (or pPCA), the noise variance is assumed to be the same for all components. This is appropriate if all components of the observed data are 'shifted' versions of each other.
- ⇒ *PCA is very widely applied to image and signal processing tasks!*
- Google: [PCA "face recognition"] > 59K hits; [PCA "noise reduction"] > 48K hits
- FA is insensitive to scaling of individual components in the observed data (see appendix).
- Use FA if the data are not shifted versions of the same kind.
- ⇒ *FA has strong history in 'social sciences'*

Factor Analysis vs. PPCA

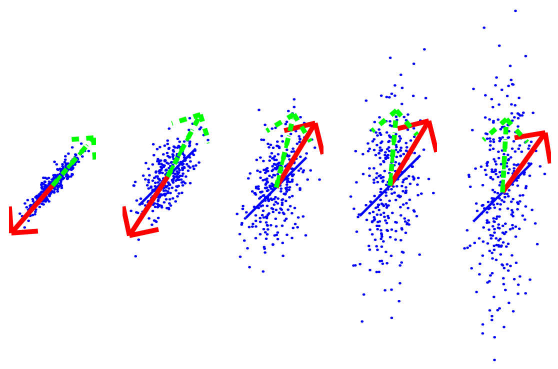


Figure 3: A comparison of factor analysis and PCA. The underlying data generating process is $y = x + \epsilon$, where ϵ is Gaussian noise of standard deviation σ. In the plots from left to right, σ takes the values 0.5, 1.2, 2, 3, 4. The FA solution is given by the solid arrow, and the PCA solution by the dashed arrow. The correct direction is given by the solid line. Note how the PCA solution "rotates" upwards as the noise level increases, whereas the FA solution remains a better estimate of the underlying correct direction.

ML estimation for FA Model

- The parameters to be optimized are $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \mathbf{W}, \Psi\}$

- (Inference for $\boldsymbol{\mu}$). $\hat{\boldsymbol{\mu}}$ is easy: $\mathbf{x}$ is a multivariate Gaussian with mean $\boldsymbol{\mu}$, so its ML estimate is
$$\hat{\boldsymbol{\mu}} = \frac{1}{N}\sum_n \mathbf{x}_n$$
Now subtract $\hat{\boldsymbol{\mu}}$ from all data points ($\mathbf{x}_n := \mathbf{x}_n - \hat{\boldsymbol{\mu}}$) and assume that we have zero-mean data.

- (solution 1). Work out the gradients for the log-likelihood
$$\log p(\mathcal{D}|\boldsymbol{\theta}) = -\frac{N}{2}\log|2\pi(\mathbf{W}\mathbf{W}^T + \Psi)| - \frac{1}{2}\sum_n \mathbf{x}_n^T(\mathbf{W}\mathbf{W}^T + \Psi)^{-1}\mathbf{x}_n$$
and optimize w.r.t. $\mathbf{W}$ and Ψ, subject to constraints. (Not possible to decouple $\mathbf{W}$ and Ψ).

- (Solution 2). Use EM

Inference for the Hidden Variables, $p(\mathbf{z}|\mathbf{x})$

- Both for dimensionality reduction and the E-step in EM-based parameter estimation, we need $p(\mathbf{z}|\mathbf{x})$.

- (Use Bayes rule). We first derive the joint density $p(\mathbf{x},\mathbf{z})$ and then condition to $p(\mathbf{z}|\mathbf{x})$

- The covariance between $\mathbf{x}$ and $\mathbf{z}$
$$\begin{aligned}\operatorname{cov}[\mathbf{z},\mathbf{x}] &= \mathbb{E}[(\mathbf{z}-0)(\mathbf{x}-\boldsymbol{\mu})^T] = \mathbb{E}[\mathbf{z}(\mathbf{W}\mathbf{z}+\boldsymbol{\epsilon})^T] \\ &= \mathbb{E}[\mathbf{z}\mathbf{z}^T]\mathbf{W}^T + \mathbb{E}[\mathbf{z}\boldsymbol{\epsilon}^T] = \mathbf{W}^T\end{aligned}$$

- Now we can write out the joint distribution $p(\mathbf{x},\mathbf{z})$,
$$p(\mathbf{x},\mathbf{z}|\boldsymbol{\theta}) = \mathcal{N}\left(\begin{bmatrix}\mathbf{z}\\ \mathbf{x}\end{bmatrix} \middle| \begin{bmatrix}0\\ \boldsymbol{\mu}\end{bmatrix}, \begin{bmatrix} I & \mathbf{W}^T \\ \mathbf{W} & \mathbf{W}\mathbf{W}^T + \Psi\end{bmatrix}\right)$$

- Direct application of the formula for gaussian-conditioning (SRG-5d) leads to the posterior $p(\mathbf{z}|\mathbf{x})$

$$p(\mathbf{z}|\mathbf{x}) = \mathcal{N}\left(\mathbf{z}|\,\mathbf{W}^T(\mathbf{W}\mathbf{W}^T + \Psi)^{-1}(\mathbf{x}-\boldsymbol{\mu}),\, I - \mathbf{W}^T(\mathbf{W}\mathbf{W}^T + \Psi)^{-1}\Psi\right)$$

A Computational Trick for $p(\mathbf{z}|\mathbf{x})$

- Note that $(\mathbf{W}\mathbf{W}^T + \Psi)^{-1}$ requires inversion of a $D \times D$ matrix. The computational load can be substantially reduced to inversion of a $M \times M$ matrix (size of $\mathbf{z}$), after application of the *matrix inversion identity*, (Bishop eqn. C.7),

$$(A + XBX^T)^{-1} = A^{-1} - A^{-1}X(B^{-1} + X^T A^{-1} X)^{-1} X^T A^{-1}$$

- Substitution of this identity into the previous formula for $p(\mathbf{z}|\mathbf{x})$ gives

$$\boxed{p(\mathbf{z}|\mathbf{x}) = \mathcal{N}\left(\mathbf{z}|\, \mathbf{V}\mathbf{W}^T\Psi^{-1}(\mathbf{x}-\boldsymbol{\mu}), \mathbf{V}\right)}$$

  where $\boxed{\mathbf{V} = (I + \mathbf{W}^T\Psi^{-1}\mathbf{W})^{-1}}$

Uncertainty about the Input z

- Given an observation $\mathbf{x}$, the mean value for the unobserved input $\mathbf{z}$ is a linear function of $\mathbf{x}$,

$$\mathbb{E}[\mathbf{z}|\mathbf{x}] = \mathbf{V}\mathbf{W}^T\Psi^{-1}(\mathbf{x}-\boldsymbol{\mu})$$

- Note however that when $\mathbf{x}$ is given, the unobserved input $\mathbf{z}$ is not known exactly; the uncertainty about input $\mathbf{z}$, as expressed by the variance $\mathbf{V} = (I + \mathbf{W}^T\Psi^{-1}\mathbf{W})^{-1}$, can be computed before the data point $\mathbf{x}$ has been seen.

- Compare this to linear regression, where we have full knowledge about an input-output pair $(\mathbf{x}, t)$.

EM for Factor Analysis

- Subtract sample mean $\hat{\boldsymbol{\mu}} = \frac{1}{N}\sum_n \mathbf{x}_n$ from all data points.

- (E-step). For each data point $\mathbf{x}_n$, compute the posterior distribution of hidden factors, given the observed data,

$$\begin{aligned} q_n^{\text{new}}(\mathbf{z}) &= p(\mathbf{z}|\mathbf{x}_n, \boldsymbol{\theta}^{\text{old}}) = \mathcal{N}(\mathbf{z}|\, \mathbf{m}_n, \mathbf{V}) \\ \mathbf{V} &= (I + \mathbf{W}^T\Psi^{-1}\mathbf{W})^{-1} \\ \mathbf{m}_n &= \mathbf{V}\mathbf{W}^T\Psi^{-1}\mathbf{x}_n \end{aligned}$$

- (M-step). Maximize $\mathcal{L}(\boldsymbol{\theta}, q^{\text{new}})$ w.r.t. $\boldsymbol{\theta}$, (see OPTIONAL slides at end of section for details)

$$\begin{aligned} \hat{\mathbf{W}}^{\text{new}} &= \left(\sum_n \mathbf{x}_n \mathbf{m}_n{}^T\right)\left(N\mathbf{V} + \sum_n \mathbf{m}_n\mathbf{m}_n{}^T\right)^{-1} \\ \hat{\Psi}^{\text{new}} &= \mathbf{W}^{\text{new}}\mathbf{V}(\mathbf{W}^{\text{new}})^T + \frac{1}{N}\sum_n(\mathbf{x}_n - \mathbf{W}\mathbf{m}_n)(\mathbf{x}_n - \mathbf{W}\mathbf{m}_n)^T \end{aligned}$$

Factor Analysis vs. Linear Regression

- Note again the close relationship between the M-step iteration for FA, $\mathbf{x}_n = \mathbf{W}\mathbf{z}_n + \mathcal{N}(0, \Psi)$,

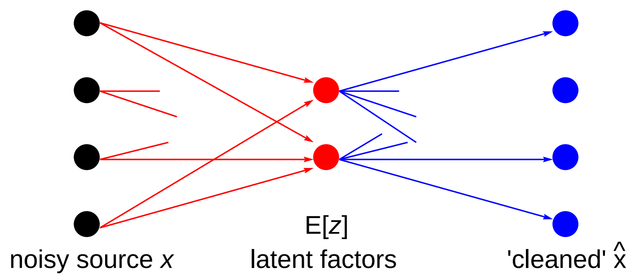
$$\hat{\mathbf{W}} = \left(\sum_n \mathbf{x}_n \mathbf{m}_n{}^T\right)\left(N\mathbf{V} + \sum_n \mathbf{m}_n \mathbf{m}_n{}^T\right)^{-1}$$

  and the ML solution for linear regression (in altered notation), $\mathbf{x}_n = \boldsymbol{\theta}^T \mathbf{u}_n + \mathcal{N}(0, \sigma^2 I)$,

$$\hat{\boldsymbol{\theta}} = (\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T\mathbf{x} = \left(\sum_n \mathbf{u}_n \mathbf{u}_n^T\right)^{-1}\left(\sum_n \mathbf{u}_n \mathbf{x}_n^T\right)$$

- The uncertainty about the inputs $\mathbf{z}_n$, expressed by variance $\mathbf{V}$ is the essential difference between FA and regression.
- In general, as $\mathbf{V} = (I + \mathbf{W}^T\Psi^{-1}\mathbf{W})^{-1} \to 0$, the equations for linear regression are recovered.

Example: Noise Reduction by Auto-Encoder



- [Q.] Suppress noise in observation $\mathbf{x}$ on the basis of observed data set $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$.
- [A.] Use $\mathcal{D}$ to train FA (or pPCA) network and recover cleaned version $\hat{\mathbf{x}}$ from $\mathbf{x}$ as

$$\hat{\mathbf{x}} = \underbrace{\mathbf{W}}_{(D\times M)} \cdot \underbrace{\mathbf{V}\mathbf{W}^T\Psi^{-1}}_{(M\times D)} \cdot \mathbf{x}$$

Mixtures of Factor Analyzers

- Do simultaneous clustering and dimensionality reduction.

- MoFA has *two sets of hidden variables*

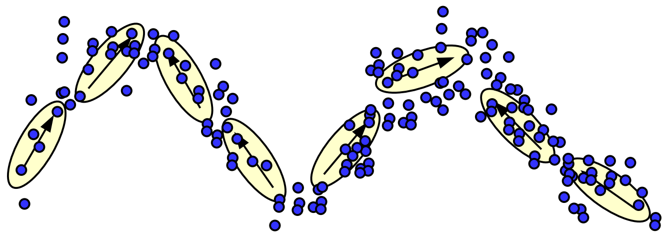
$$\begin{aligned} p(\mathbf{z}) &= \mathcal{N}(\mathbf{z}|0, I) && \text{(continuous hidden factors)} \\ p(k) &= \pi_k && \text{(discrete hidden components)} \\ p(\mathbf{x}|\mathbf{z}, k, \boldsymbol{\theta}) &= \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k + \mathbf{W}_k\mathbf{z}, \Psi) && \text{(conditional data model)} \end{aligned}$$

- This leads to a *constrained Gaussian mixture model*

$$\begin{aligned} p(\mathbf{x}|\boldsymbol{\theta}) &= \sum_k \int_{\mathbf{z}} p(\mathbf{x}|\mathbf{z}, k, \boldsymbol{\theta}) p(\mathbf{z}) p(k)\,\mathrm{d}\mathbf{z} \\ &= \boxed{\sum_k \pi_k \cdot \mathcal{N}(\mathbf{x}|\,\boldsymbol{\mu}_k,\, \mathbf{W}_k\mathbf{W}_k^T + \Psi)} \end{aligned}$$

- As before, train by EM
  - E-step: infer posterior for hidden variables, $p(\mathbf{z}_n, k_n|\mathbf{x}_n, \boldsymbol{\theta})$
  - M-step: maximize $\mathcal{L}$ w.r.t. $\boldsymbol{\theta}$

Mixtures of Factor Analyzers



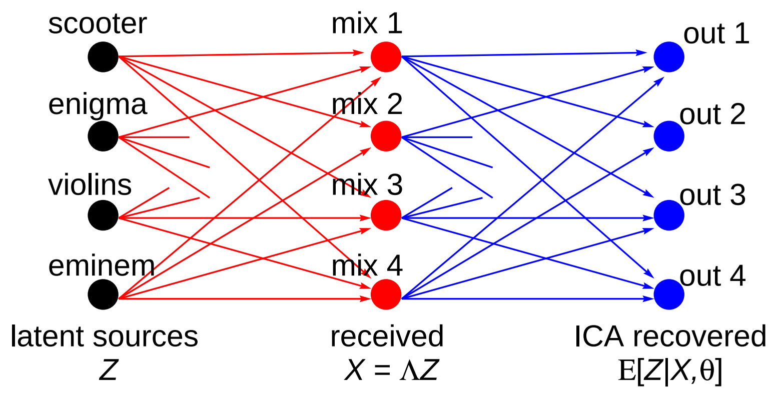
Independent Component Analysis (ICA)

- Often, in nature the observed data are mixtures of *independent non-Gaussian* sources, e.g., cocktail party
- The ICA (a.k.a. blind source separation) model is aimed at these kind of problems,

$$\begin{aligned} \mathbf{x} &= \mathbf{W}\mathbf{z} + \text{noise} \quad (\mathbf{x}\text{ observed}) \\ p(\mathbf{z}) &= \prod_k p(z_k) \quad \text{(unknown non-Gaussian factorized prior)} \end{aligned}$$

- Many sensor noise models and many algorithms (often EM-based) for estimation of the sources $\mathbf{z}$ and the mixing matrix $\mathbf{W}$ have been developed.

Demo ICA; Blind Signal Separation



(OPTIONAL) Derivation of EM for FA

- E-step has been done before,

$$q_n(\mathbf{z}) = p(\mathbf{z}|\mathbf{x}_n, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{z}|\,\mathbf{m}_n, \mathbf{V})$$

- Before executing the M-step, it will be useful at this point to establish that

$$\langle \mathbf{z} \rangle_{q_n^{\text{new}}} = \mathbf{m}_n$$
$$\left\langle \mathbf{z}\mathbf{z}^T \right\rangle_{q_n^{\text{new}}} = \mathbf{m}_n \mathbf{m}_n{}^T + \mathbf{V}$$

(OPTIONAL) Derivation of EM for FA (2)

- M-step; let's first compute the complete log-likelihood

$$L_c(\boldsymbol{\theta}) = \sum_n [\log p(\mathbf{z}) + \log p(\mathbf{x}_n|\,\mathbf{z}, \mathbf{W}, \Psi)] \propto \sum_n \log p(\mathbf{x}_n|\,\mathbf{z}, \mathbf{W}, \Psi)$$
$$= \sum_n \left[-\frac{1}{2}\log|\Psi| - \frac{1}{2}(\mathbf{x}_n - \mathbf{W}\mathbf{z})^T \Psi^{-1} (\mathbf{x}_n - \mathbf{W}\mathbf{z}) \right]$$
$$= -\frac{N}{2}\log|\Psi| - \frac{1}{2}\sum_n \left[\mathbf{x}_n^T \Psi^{-1}\mathbf{x}_n - 2\mathbf{x}_n^T \Psi^{-1}\mathbf{W}\mathbf{z} + \text{Tr}[\mathbf{W}^T\Psi^{-1}\mathbf{W}\mathbf{z}\mathbf{z}^T] \right]$$

- And take the expectation wrt 'responsibilities' q_n^{new}

$$\langle L_c(\mathbf{W},\Psi)\rangle_q =$$
$$= -\frac{N}{2}\log|\Psi| - \frac{1}{2}\sum_n \left[\mathbf{x}_n^T\Psi^{-1}\mathbf{x}_n - 2\mathbf{x}_n^T\Psi^{-1}\mathbf{W}\langle\mathbf{z}\rangle_q + \mathrm{Tr}[\mathbf{W}^T\Psi^{-1}\mathbf{W}\langle\mathbf{z}\mathbf{z}^T\rangle_q]\right]$$
$$= -\frac{N}{2}\log|\Psi| - \frac{1}{2}\sum_n \left[\mathbf{x}_n^T\Psi^{-1}\mathbf{x}_n - 2\mathbf{x}_n^T\Psi^{-1}\mathbf{W}\mathbf{m}_n + \mathrm{Tr}[\mathbf{W}^T\Psi^{-1}\mathbf{W}(\mathbf{m}_n\mathbf{m}_n{}^T + \mathbf{V})]\right]$$

(OPTIONAL) Derivation of EM for FA (3)

- Now take the derivatives, making good use of $\frac{\partial \mathrm{Tr}[AB]}{\partial B} = A^T$ [B-C.25] and $\frac{\partial \mathrm{Tr}[BA^TCA]}{\partial A} = 2CAB$ [SRM-5f],

$$\frac{\partial\langle L_c\rangle}{\partial\mathbf{W}} = \Psi^{-1}\sum_n \mathbf{x}_n\mathbf{m}_n{}^T - \Psi^{-1}(N\mathbf{V} + \sum_n \mathbf{m}_n\mathbf{m}_n{}^T)$$

- Set to zero to get

$$\boxed{\hat{\mathbf{W}} = \left(\sum_n \mathbf{x}_n\mathbf{m}_n{}^T\right)\left(N\mathbf{V} + \sum_n \mathbf{m}_n\mathbf{m}_n{}^T\right)^{-1}}$$

- As before, we differentiate to Ψ^{-1} rather than to Ψ

$$\frac{\partial\langle L_c\rangle}{\partial\Psi^{-1}} = \frac{N}{2}\Psi - \frac{1}{2}\sum_n \left[\mathbf{x}_n\mathbf{x}_n^T - 2\mathbf{W}\mathbf{m}_n\mathbf{x}_n^T + \mathbf{W}(\mathbf{m}_n\mathbf{m}_n{}^T + \mathbf{V})\mathbf{W}^T\right]$$

- and zeroing the derivative yields

$$\boxed{\hat{\Psi} = \mathbf{W}\mathbf{V}\mathbf{W}^T + \frac{1}{N}\sum_n (\mathbf{x}_n - \mathbf{W}\mathbf{m}_n)(\mathbf{x}_n - \mathbf{W}\mathbf{m}_n)^T}$$

(OPTIONAL) Rotational Invariance of pPCA

- Let $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{W}\mathbf{W}^T + \sigma^2 I)$, (pPCA model), and suppose that we observe a rotated data set $\mathcal{D}'$ with elements $\mathbf{x}' = Q\mathbf{x}$, where Q is a rotation matrix.
- Thus, the observed distribution is

$$\begin{aligned} p(\mathbf{x}') &= \mathcal{N}\left(\mathbf{x}'|\,Q\boldsymbol{\mu}, Q(\mathbf{W}\mathbf{W}^T + \sigma^2 I)Q^T\right) \\ &= \mathcal{N}(\mathbf{x}'|\,Q\boldsymbol{\mu}, (Q\mathbf{W})(Q\mathbf{W})^T + \sigma^2 QQ^T) \\ &= \mathcal{N}(\mathbf{x}'|\,\boldsymbol{\mu}', \mathbf{W}'\mathbf{W}'^T + \sigma^2 I) \end{aligned}$$

where we defined $\boldsymbol{\mu}' = Q\boldsymbol{\mu}, \mathbf{W}' = Q\mathbf{W}$ (and $\Psi' = \Psi = \sigma^2 I$).

- Then the likelihoods $L(\boldsymbol{\mu}, \mathbf{W}, \Psi; \mathcal{D})$ and $L(\boldsymbol{\mu}', \mathbf{W}', \Psi'; \mathcal{D}')$ are the same. I.o.w., *PCA is invariant to rotations in the observed data* (but not to scaling).
- Homework: Verify that this is not true for FA.

(OPTIONAL) FA is invariant to scaling of observed data components

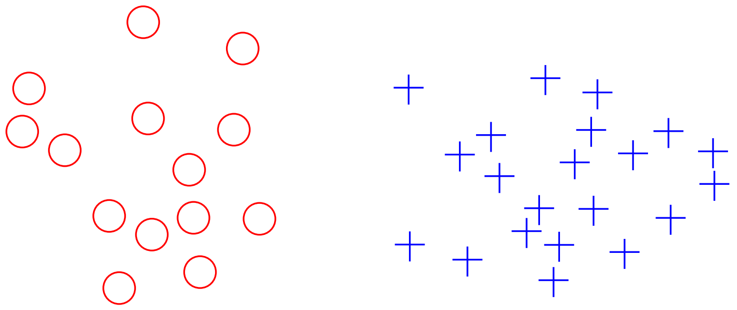
- Now let $\mathbf{x} \sim \mathcal{N}\left(\boldsymbol{\mu}, \mathbf{W}\mathbf{W}^T + \text{diag}(\psi_i)\right)$, (FA model), and suppose that we observe a scaled component in the observed data $x_i{}' = \alpha x_i$, (α scalar).
- If we set

$$\begin{aligned} \boldsymbol{\mu}_i' &= \alpha \boldsymbol{\mu}_i \\ \mathbf{W}_{ij}' &= \alpha \mathbf{W}_{ij} \quad \forall j \\ \psi_i' &= \alpha^2 \psi_i \end{aligned}$$

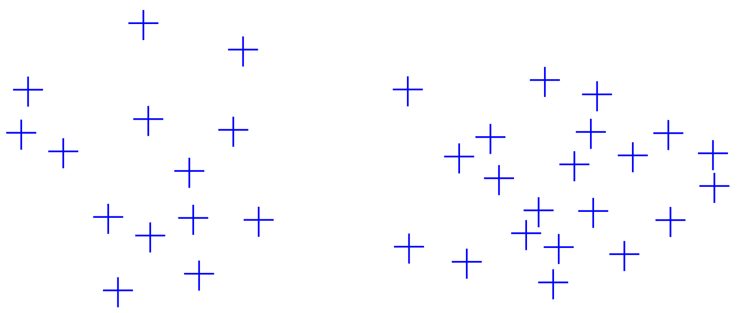
  then $L(\boldsymbol{\mu}, \mathbf{W}, \Psi; \mathcal{D}) = L(\boldsymbol{\mu}', \mathbf{W}', \Psi'; \mathcal{D}')$ and hence *FA is invariant to scaling of individual components in the observed data* (but not to rotations).
- Verify that pPCA is not invariant to scaling.

11 Temporal Latent Variable Models–Kalman Filtering and HMM

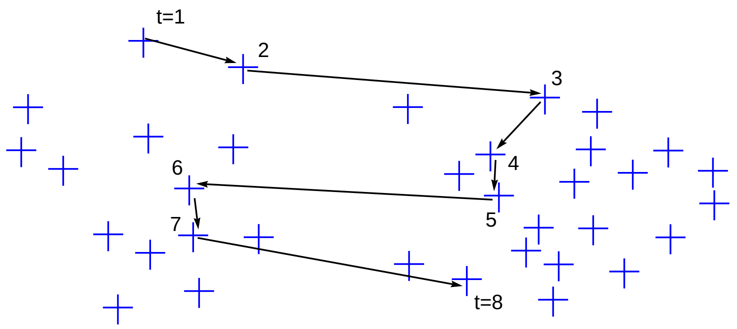
We've done this before (classification)



And we've covered this case (clustering)



Now the data points are temporally ordered



Temporal Models

- Consider the *ordered* observation sequence $\mathbf{x}_1, \dots, \mathbf{x}_N$.
- (Goal). We wish to develop a generative model

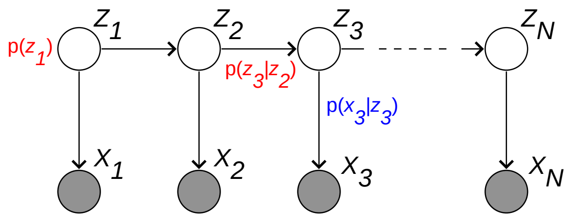
$$p(\mathbf{x}_1, \dots, \mathbf{x}_N | \theta)$$

  that 'explains' the time series.
- No more IID assumption; in general

$$\begin{aligned} p(\mathbf{x}_1, \dots, \mathbf{x}_n) &= p(\mathbf{x}_n | \mathbf{x}_1, \dots, \mathbf{x}_{n-1}) p(\mathbf{x}_1, \dots, \mathbf{x}_{n-1}) \\ &= p(\mathbf{x}_n | \mathbf{x}_1, \dots, \mathbf{x}_{n-1}) p(\mathbf{x}_{n-1} | \mathbf{x}_1, \dots, \mathbf{x}_{n-2}) p(\mathbf{x}_1, \dots, \mathbf{x}_{n-2}) \\ &= p(\mathbf{x}_n | \mathbf{x}_1, \dots, \mathbf{x}_{n-1}) p(\mathbf{x}_{n-1} | \mathbf{x}_1, \dots, \mathbf{x}_{n-2}) \cdots p(\mathbf{x}_2 | \mathbf{x}_1) p(\mathbf{x}_1) \end{aligned}$$

- Example: K-th order Auto-Regressive (AR) model

$$p(\mathbf{x}_n | \mathbf{x}_1, \dots, \mathbf{x}_{n-1}) = F(\mathbf{x}_{n-K}, \dots, \mathbf{x}_{n-1}, \theta) + \mathcal{N}(0, \sigma^2)$$



State Space Models

- In an AR model, we would need *large number of parameters* for a deep temporal memory
- Can we decouple memory depth from model size?
- Yes, store the past in a set of *latent* (unobserved) variables $\mathbf{z}_1, \dots, \mathbf{z}_n$. (called: state space)
- A general state space model is defined by

$$\begin{array}{ll} p(\mathbf{z}_n|\mathbf{z}_1, \dots, \mathbf{z}_{n-1}) & \text{(state transition model)} \\ p(\mathbf{x}_n|\mathbf{z}_n) & \text{(observation model)} \\ p(\mathbf{z}_1) & \text{(initial state)} \end{array}$$

- A very common computational limitation is to let state transitions be ruled by a *first-order Markov* chain

$$\boxed{p(\mathbf{z}_n|\mathbf{z}_1, \dots, \mathbf{z}_{n-1}) = p(\mathbf{z}_n|\mathbf{z}_{n-1})}$$

The Joint Probability Factorization

- Let $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ and $\mathbf{Z} = \{\mathbf{z}_1, \dots, \mathbf{z}_N\}$. The Markov assumption leads to the following joint probability distribution

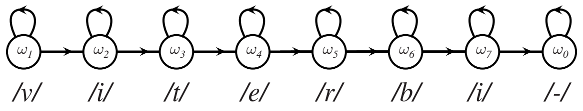
$$\boxed{p(\mathbf{X}, \mathbf{Z}) = p(\mathbf{z}_1) \prod_{n=2}^{N} p(\mathbf{z}_n|\mathbf{z}_{n-1}) \prod_{n=1}^{N} p(\mathbf{x}_n|\mathbf{z}_n)}$$

Hidden Markov Model

- In a *hidden Markov Model* (HMM), the *state space is discrete*

$$\begin{aligned} p(\mathbf{z}_{nk} = 1|\mathbf{z}_{n-1,j} = 1) &= A_{jk} && \text{(state transitions)} \\ p(\mathbf{x}_n|\mathbf{z}_{nk} = 1) &= p(\mathbf{x}_n|\psi_k) && \text{(observations)} \\ p(\mathbf{z}_{1k} = 1) &= \pi_k && \text{(initial state)} \end{aligned}$$

- Flexible state transition dynamics, but limited memory capacity (needs 2^M binary states to store M bits about past).
- Famous for applications to speech recognition, language modeling, protein and genetic sequence analysis, financial time series prediction, etc.



HMM Properties

- In a HMM, each state variable $\mathbf{z}_n$ holds K binary components. Only one state component is active at all times. (1-of-K coding scheme).
- The state sequence is similar to a 'multinomial-in-time',

$$p(\mathbf{z}_n|\mathbf{z}_{n-1},\mathbf{A}) = \prod_{k,j}(A_{jk})^{z_{n-1,j}\cdot z_{nk}}$$

$$p(\mathbf{z}_1) = \prod_{k}(\pi_k)^{z_{1k}}$$

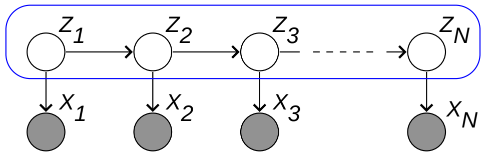
- The observations $\mathbf{x}_n$ may be either continuous or discrete, e.g.,

$$p(\mathbf{x}_n = j|\mathbf{z}_n) = \prod_k \psi_{jk}^{z_{nk}} \quad \text{or} \qquad \text{(discrete)}$$

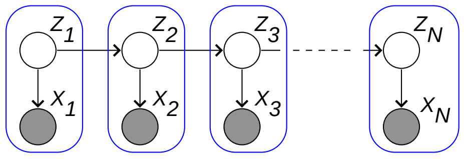
$$p(\mathbf{x}_n|\mathbf{z}_n) = \prod_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)^{z_{nk}} \qquad \text{(Gaussian)}$$

Model Interpretation

- HMM is *Markov-chain with stochastic observations*



- HMM is *mixture model over time*



Linear Dynamical Systems

- (Model Specification). In contrast to HMM, the state variable vector $\mathbf{z}$ is real-valued; furthermore, the 'standard' LDS model is linear with Gaussian noise,

$$\begin{aligned} p(\mathbf{z}_n|\mathbf{z}_{n-1}) &= \mathcal{N}(\mathbf{A}\mathbf{z}_{n-1}, \boldsymbol{\Gamma}) && \text{(state dynamics)} \\ p(\mathbf{x}_n|\mathbf{z}_n) &= \mathcal{N}(\mathbf{C}\mathbf{z}_n, \boldsymbol{\Sigma}) && \text{(observations)} \\ p(\mathbf{z}_1) &= \mathcal{N}(\boldsymbol{\mu}_0, \mathbf{V}_0) && \text{(initial state)} \end{aligned}$$

  or equivalently,

$$\begin{aligned} \mathbf{z}_n &= \mathbf{A}\mathbf{z}_{n-1} + \mathbf{w}_n && \mathbf{w}_n \sim \mathcal{N}(0, \boldsymbol{\Gamma}) \\ \mathbf{x}_n &= \mathbf{C}\mathbf{z}_n + \boldsymbol{\epsilon}_n && \boldsymbol{\epsilon}_n \sim \mathcal{N}(0, \boldsymbol{\Sigma}) \\ \mathbf{z}_1 &= \boldsymbol{\mu}_0 + \mathbf{u} && \mathbf{u} \sim \mathcal{N}(0, \mathbf{V}_0) \end{aligned}$$

- Again, dual interpretation as
  1. Factor Analysis over time, or
  2. Gauss-Markov chain with linear Gaussian observations

Properties of Linear Dynamical Systems

- Compared to HMM, rather limited state transitions (continuity constraints), but real-valued state vector can in principle store arbitrary large number of bits.
- All ARMA models can be written as state space models.
- Non-stationary models (time-varying ARMA or adaptive algorithms) are also state space models.
- Famous for applications to control and tracking (e.g. landing on the moon), adaptive filtering (RLS is a deterministic Kalman Filter).

Remaining Issues

- Once we specified the model, three inference problems remain:

1 State estimation. Compute the probability $p(\mathbf{z}_n|\mathbf{x}_a, \ldots, \mathbf{x}_b)$ of state $\mathbf{z}_n$, given the observed sequence $\mathbf{x}_a, \ldots, \mathbf{x}_b$.

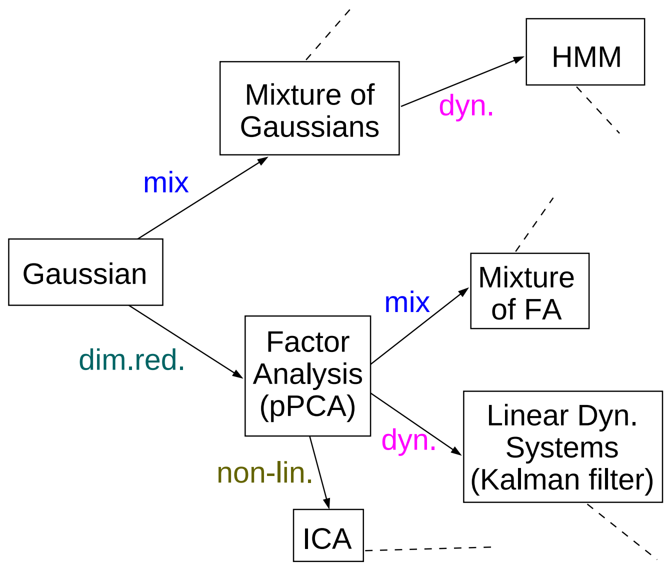
2 Learning. Determine the parameters from the data, $p(\boldsymbol{\theta}|\mathcal{D})$, with $\boldsymbol{\theta} = \{\mathbf{A}, \boldsymbol{\Gamma}, \mathbf{C}, \boldsymbol{\Sigma}, \boldsymbol{\mu}_0, \mathbf{V}_0\}$

3 Applications, e.g classification or prediction. Determine the probability $p(\mathbf{x}|\boldsymbol{\theta})$ that an observation $\mathbf{x}$ came from a given model $\boldsymbol{\theta}$.

- Note (again) that *all interesting problems can be specified as conditional probability statements* and can in principle be solved by straight probabiulity theory.
- Lots of efficient inference algorithms for temporal models have been developed (e.g. Viterbi decoder, Kalman filter, Baum-Welch recursion etc.)

12 Review and Perspective

Generative Gaussian Models



Probabilistic Modeling Review

- Given an observed data set $\mathcal{D}$; we want to predict (or classify etc.) a new data point $\mathbf{x}$
- [1. Model Specification]. Based on your knowledge, insight etc., propose a model,

$$p(\mathcal{D}|\boldsymbol{\theta}, h) \qquad \text{(likelihood)}$$
$$p(\boldsymbol{\theta}|h) \quad \text{and} \quad p(h) \qquad \text{(priors)}$$

- Thus far, we've assumed no uncertainties about the model structure h (this assumption will change in part-2), so we've worked with

$$p(\mathcal{D}|\boldsymbol{\theta}) \quad \text{and} \quad p(\boldsymbol{\theta})$$

- [2. Learning] Use Bayes rule to get posterior

$$p(\boldsymbol{\theta}|\mathcal{D}) \propto p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})$$

Probabilistic Modeling Review, cont'd

- Or alternatively, the maximum likelihood point estimate,

$$\hat{\boldsymbol{\theta}}_{ML} = \arg\max_{\boldsymbol{\theta}} p(\mathcal{D}|\boldsymbol{\theta})$$

  which is often a good approximation to the posterior if the data set $\mathcal{D}$ is large (and the posterior is unimodal)

- [3. Apply Model] E.g. data point prediction

$$\begin{aligned} p(\mathbf{x}|\mathcal{D}) &= \int p(\mathbf{x},\boldsymbol{\theta}|\mathcal{D})\,\mathrm{d}\boldsymbol{\theta} = \int p(\mathbf{x}|\boldsymbol{\theta},\mathcal{D})p(\boldsymbol{\theta}|\mathcal{D})\,\mathrm{d}\boldsymbol{\theta} \\ &= \int p(\mathbf{x}|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})\,\mathrm{d}\boldsymbol{\theta} \end{aligned}$$

- Or predict by ML: $p(\mathbf{x}|\mathcal{D}) = p(\mathbf{x}|\hat{\boldsymbol{\theta}}_{ML})$
- Part-2: how good were the model assumptions? Can we do better?

Final Slide

Lots of applications of machine learning in the SPS group, including

- Medical Signal Processing
  - fetal monitoring (together with MMC)
  - epilepsy
- Media Signal Processing
  - video processing
  - audio processing
- Communications
  - source coding
  - channel coding
- $\Longrightarrow$ Stop by if you want to do a traineeship or thesis work.