RUPRECHT-KARLS-
UNIVERSITAT HEIDELBERG
ZUKUNFT SEIT 1386

Machine Learning Models

Kiryl Batvinyeu

November 20, 2016

Contents

Chapter 1

Overview

This skript begins with an overview on graphical models as approximations to probability distributions. Each class of graphical models corresponds to a factorization property of the joint distribution. Most popular are the ***Bayesian belief networks*** and the ***Markov networks***.

A Bayesian network is represented by a directed graph. Since no cycles are modeled, the belief network corresponds to the Directed Acyclic Graph. Belief networks are only capable to encode conditional independencies $x \perp y|z$ whereas information on general dependencies between the variables cannot be read from them. For decision making processes one can extend the belief network to obtain ***influence diagrams***. Uncertain and unreliable evidence is modelled via ***Jeffrey's rule*** and ***virtual evidence***. The Markov network is a set of random variables having the Markov property. A mathematical unification of both is the ***chain graph***.

The graphical approximation is viable as long as marginal dependencies of variables of interests can be obtained. For optimized inference algorithms such as ***belief propagation (message passing)*** one transforms the graphical model to a factor graph. In case the graph is loopy one can use the ***junction tree*** representation instead and do again the belief propagation on the super nodes.

Usually in machine learning one chooses a known distribution to model the data and obtains the parameters of the model by some learning algorithm. Most popular is the ***maximum likelihood training***. Prior knowledge on the parameters can be accounted for in the ***ML-II method***.

For a belief network, maximum likelihood training under the i.i.d. assumption corresponds mathematically to setting the ***Conditional Probability Table*** entries by counting the relative number of occurrences. To reduce the number of parameters one can also use ***conditional probability functions***. For a convenient setting the learning can be done via Bayesian training on the families of nodes. The structure learning of a Belief network can be done with the ***PC algorithm***, starting with the fully connected skeleton of the DAG. For a Markov network maximum likelihood training can be equally applied having several constraints on the potentials.

The Bayesian method can additionally be used to estimate the ***model likelihood***, i.e. to choose a suitable model from a set of models. Common methods are the ***Laplace method*** and the ***BIC approximation***.

To account for imperfection of data, various learning methods with ***hidden variables*** have been developed. Te problem with hidden variables is that they prevent decoupling of marginals. Therefore variational approximation methods are needed, such as ***Variational Expectation Maximization***. The two-step EM method works both for belief and Markov networks. The generalization is the ***Variational Bayes*** method with numerous variations.

A class of models particularly suitable for modelling temporal and spatial correlation of multidimensional data is inspired by biology. Its advantage is that it utilizes distributed computation in an intrinsic way. ***Recurrent neural networks*** encode a temporal evolution by its directed connections that form a directed graph. In contrast to feed forward neural networks the graph is not acyclic. Recurrent neural network can use their internal state, their memory, to evaluate their input nodes. The model is Turing complete.

The ***Long Short Term Memory*** gates are designed to allow the information to be gated into the memory cell and then to be gated out when needed. In the intermediate period the gate is closed, so that the information that arrives does not interfere with the remembered state. This drastically reduces the number of parameters which are necessary to model long short-term correlations.

Stochastic neural networks introduce random variables into an artificial neural network to solve computationally intensive optimization problems with full utilization of fluctuations as a facilitizer. Stochastic can be the transitions or the weights. An example is the ***Restricted Boltzmann Machine***.

The ***Hopfield network*** is a recurrent neural model which has a large analogy to a magnetic field of individual spins in Fermi gas. It is also used to model human memory via its binary thresholds.

With ***Tractable Deterministic Continuous Latent Variable Models*** the framework is extended by a powerful setting of continuous hidden variables which follow non-linear dynamics with posterior inference being still tractable. An example is the ***Augmented Hopfield Network***.

The class of ***Multivariate Linear Models*** is capable of unsupervised data learning due to the fact that the models utilize only few parameters.

Popular model of ***Unsupervised Linear Dimension Reduction*** include ***PCA*** and ***SVD***, bot bilinear decomposition methods. Non-linearities can be accounted for via a suitable ***kernel***. Cross correlations can be accounted for in the ***Canonical Correlation Analysis***.

Supervised Linear Dimension Reduction methods utilize knowledge on class memberships for improved data description. SLDR can produce lower dimensional representations more suitable for subsequent classification than an unsupervised method. An example is ***Fishers Linear Discriminant Analysis*** and its generalization ***Canonical Variates***.

Latent Linear Models find a hidden structure in the data in form of internal dimen- sions along which the data is spread with the Gaussian variance. An example is ***Factor Analysis***. ***Probabilistic PCA*** and ***Canonical Correlation Analysis*** are variants of FA.

Dynamical Markov Models are used to model one dimensional time series data. Mixture models are used in this context. The ***Hidden Markov Model*** defines a Markov chain on hidden (or "latent") variables. The observed (or "visible") variables in HMM are dependent on the hidden variables through an emission. The applications are filtering, prediction, smoothing, likelihood inference and Viterbi alignment (decoding). Learning can be done via EM. ***HMM-GMM*** is a common model with the continuous observation Gaussian mixture emission.

An example of ***Continuous State Markov Models*** are the ***Observed Linear Dynamical Models***. Continuous State Markov Models though having discrete time evolution act upon a continuous state space. Therefore a discrete transition matrix has to be modified to a continuous (Gaussian) transition kernel. Another example are the ***Auto-regressive models*** and the ***Kalman fiter*** where the transitions are Gaussian.

Sometimes all that one needs is a fast and robust classification prescription. With the ***Statistical Decision Theory*** and the simple ***Bayes Classifier*** or ***Random Forest Classifier*** you will get the results.

Part I

Graphical Models

Chapter 2

Overview Graphical Models

In Essence: *Each class of graphical models corresponds to a factorization property of the joint distribution. Most popular are belief networks and Markov networks. A mathematical unification of both is the chain graph. For decision making processes one can extend the belief network to obtain influence diagrams.*

2.1 Belief Network

A BN is represented as a directed graph, with an arrow pointing from a parent variable to child variable. Since no cycles are modeled, the belief network corresponds to the Directed Acyclic Graph. Belief networks are only capable to encode conditional independencies $x \perp y|z$ whereas information on general dependencies between the variables cannot be read from them.

A Belief Network encodes the following factorization of the full distribution:

$$p(x_1, ..., x_D) = \prod_{i=1}^{D} p(x_i|\mathrm{pa}(x_i))$$

Given a DAG G and a set X that satisfies the local Markov property w.r.t. G, X is a Belief network w.r.t. G iff one of the two holds:

- The Markov blanket of a node x_i encodes all conditional independence information about the node:

  $$x_i \perp y|MB(x_i)$$

  where y is not in $MB(x_i)$.

- every two nodes A and B are conditionally independent of each other given a set Z that d-separates the nodes:

  $$A \perp B|Z$$

Every probability distribution can be encoded with a BN. A fully connected DAG, the 'cascade', is the most general case with the least expressibility:

$$p(x_1, x_2, x_3, x_4) = p(x_1|x_2, x_3, x_4)p(x_2|x_3, x_4)p(x_3|x_4)p(x_4)$$

Markov equivalence: Two DAGs are Markov equivalent if they both represent the same set of conditional independence statements. This definition holds for both directed and undirected graphs. Define an *immorality* in a DAG as a configuration of three nodes, A, B, C such that C is a child of both A and B, with A and B not directly connected. Define the *skeleton* of a graph by removing the directions on the arrows. Two DAGs represent the same set of independence assumptions (they are Markov equivalent) if and only if they have the same skeleton and the same set of immoralities.

Limited expressibility: Belief networks generally cannot graphically represent all the independence properties of a given distribution.

Causality: BNs only make independence statements, not the causal ones.

2.1.1 Soft Evidence

a) Uncertain Evidence

In Essence: *in soft or uncertain evidence, the evidence variable is in more than one state, with the strength of our belief about each state being given by probabilities. Performing inference with soft-evidence is straightforward and can be achieved using Bayes' rule.*

Soft evidence is a generalization, in contrast to the hard evidence it means that a variable y can be in a mixed state $\tilde{y} = 0.3, 0.15, 0.55$. Given evidence $\tilde{y}$ for the variable y we have $p(x|y) \to p(x|y, \tilde{y})$:

$$p(x|\tilde{y}) = \sum_y p(x|y, \tilde{y}) p(y|\tilde{y}) = \sum_y p(x|y) p(y|\tilde{y})$$

Basically we now sum over all different possible states y weighted with the evidence (the vector of soft probabilities for the states). So we first define the model conditioned on the evidence, and then average over the distribution of the evidence, which is known as **Jeffrey's rule**.

b) Unreliable Evidence (likelihood evidence)

An unreliable evidence can be replaced by a guess, *virtual evidence*:

$$p(G|A) \to p(H|A)$$

for an unreliable variable G. In a mixed case, first include the unreliable evidence into the model, then condition on the uncertain evidence.

Related topics and extensions: Bayes-Ball algorithm, additional variables, bothsided arrows, Simpson's Parodox - observational evidence vs. interventional evidence, Pearl's Do Operator, Influence Diagrams.

2.2 Markov Network

In Essence: *A Markov network is a set of random variables having a Markov property.*

Markov Networks are defined as products of potentials on maximal cliques of an *undirected* graph and form the probability distribution:

$$\boxed{p(x_1, ..., x_n) = \frac{1}{Z} \prod_{c=1}^{C} \phi_c(\mathcal{X}_c), \qquad Z = \sum_{\mathcal{X}} \prod_{c=1}^{C} \phi_c(\mathcal{X}_c)}$$

with C are subsets of variables and $\phi \geq 0$ are *potentials* between the sets of variables $\mathcal{X}_c \in \mathcal{X}$. Graphically this is represented by an undirected graph G with X_c; C = 1,...,c being the *maximal cliques* of G. Whilst a Markov network is formally defined on maximal cliques, in practice authors often use the term to refer to non-maximal cliques.

Markov Properties in descending strength:

- **Global Markov Property:** any two subsets of variables are conditionally independent given a separating subset:

  $$X_A \perp X_B | X_S$$

  where every path from a node in A to a node in B passes through S.

- **Local Markov Property:** a variable is conditionally independent of all other variables given its neighbours:

  $$X_v \perp X_{V\backslash\{X_v \cup N(v)\}} | X_{N(v)}$$

- **Pairwise Markov Property**: any two non-adjacent variables are conditionally independent given all other variables:

  $$X_v \perp X_u | X_{V\backslash\{v,u\}}$$

  Pairwise Markov Network: in the special case that the graph contains cliques of only size two, the distribution is called a pairwise Markov Network, with potentials defined on each link between two variables.

  Markov Random Fields: A MRF is Markov Network as a set of conditional distributions, one for each indexed 'location':

  $$\boxed{p(x_i|x_j) = p(x_i|\text{ne } (x_i))}$$

2.3 Chain Graph

In Essence: *CGs contain both directed and undirected links and are a generalization of BN and MN.*

Consider a model consisting of the nodes a, b, c, d where $pa(c) = a$, $pa(d) = b$ and c and d are connected via an undirected link. This link can be modelled via a potential ϕ. With this one gets the joint probability distribution:

$$p(a, b, c, d) = \frac{1}{Z} p(a)p(b)p(c|a)p(d|b)\phi(c, d)$$

the partition function is needed to normalize the potential and is equal to:

$$Z = \sum_{a,b,c,d} p(a)p(b)p(c|a)p(d|b)\phi(c, d) = \sum_{c,d} p(c|a)p(d|b)\phi(c, d) := \phi(a, b)^{-1}$$

The (connected) part

$$\phi(a, b)p(c|a)p(d|b)\phi(c, d) = \prod_{\mathcal{X}_c} p(x|pa(x)) \prod_{\mathcal{X}} \phi(y)$$

is a *Chain Component* of the chain graph.

The BN of this graph is a special case having singletons for each of the the CCs:

$$p(a, b, c, d) = p(a, b, f) = p(a)p(b)p(f|a)p(f|b)$$

A MN consists of the chain components alone:

$$p(a, b, c, d) = p(c, d) = \frac{1}{Z}\phi(c, d)$$

With this the CG provides a unified model.

Chapter 3

Inference in Graphical Models

In Essence: *Inference is computation of functions (e.g. marginals) on distributions. For optimized inference algorithms such as belief propagation one transforms the graphical model to a factor graph. In case the graph is loopy one can use the junction tree representation instead.*

3.1 Ascertaining independence in Markov and Belief Networks

Ancestral Graph: identify the ancestors $\mathcal{A}$ of the nodes $X \cup Y \cup Z$. Retain the nodes $X \cup Y \cup Z$ but remove all other nodes which are not in $\mathcal{A}$, together with any edges in or out of such nodes.

Moralisation: add a link between any two remaining nodes which have a common child, but are not already connected by an arrow. Then remove remaining arrowheads.

Separation: remove links neighbouring Z. In the undirected graph so constructed, look for a path which joins a node in X to one in Y. If there is no such path deduce that $X \perp Y | Z$.

3.2 Factor Graph

A model which preserves even more information about the distribution is a *Factor Graph.* Factor Graphs represent both BN and MN as well. Given a distribution

$$p(\mathcal{X}) \propto \prod_i \psi_i(\mathcal{X}_i)$$

one associates a square with each ψ_i and a circle with each variable from $\mathcal{X}$. An undirected link is drawn between each factor ψ_i and its variables $\mathcal{X}_i$. In case $\psi_i = p(x_i|pa(x_i))$ a directed link is drawn from parents and the factor and from factor to the children. Additional squares and directed links are associated with probability factors for the parent variables if they don't have predecessors. Factor graphs are used in many inference algorithms.

3.3 Message Passing

Consider the distribution:

$$p(a, b, c, d) = p(a|b)p(b|c)p(c|d)p(d)$$

Then:

$$p(a = a^*) = \sum_{b,c,d} p(a = a^*|b)p(b|c)p(c|d)p(d)$$

with 7 additions(for binary variables). More effectively one can rewrite:

$$p(a = a^*) = \sum_{b,c} p(a = a^*|b)p(b|c) \sum_{d} p(c|d)p(d)$$

which is:

$$p(a = a^*) = \sum_{b,c} p(a = a^*|b)p(b|c)\gamma_d(c) = \sum_{b} p(a = a^*|b) \sum_{c} p(b|c)\gamma_d(c) = \sum_{b} p(a = a^*|b)\gamma_c(b)$$

with only 5 additions. This procedure of passing *messages(unnormalized potentials)* $\gamma_i(j)$ from the leaves inwards is called *variable elimination.* The variable elimination operation scales linearly with the number of variables in the single connected graph (tree). It relies on the associativity of matrix multiplication which is *transfer matrix method*:

$$M_a = M_{ab}M_{bc}M_{cd}M_d = M_{ab}(M_{bc}(M_{cd}M_d)), \qquad [M_{ab}]_{i,j} := p(a = i|b = j)$$

3.4 Belief Propagation

The message passing works on FG as well and is called here *Belief Propagation* or Sum-Product algorithm.

3.4.1 Non-branching Tree (Chain)

Consider the factor graph:

$$p(a, b, c, d) = f_1(a, b)f_2(b, c)f_3(c, d)f_4(d)$$

The marginal $p(a, b, c)$ is:

$$p(a, b, c) = f_1(a, b)f_2(b, c) \sum_{d} f_3(c, d)f_4(d) = f_1(a, b)f_2(b, c)\mu_{d\to c}(c)$$

The marginal $p(a, b)$ is:

$$p(a, b) = \sum_{c} p(a, b, c) = f_1(a, b) \sum_{c} f_2(b, c)\mu_{d\to c}(c) = f_1(a, b)\mu_{c\to b}(b)$$

The relation:

$$\mu_{c\to b}(b) = \sum_{c} f_2(b, c)\mu_{d\to c}(c)$$

is a recursion.

3.4.2 Singly Connected Undirected Factor Graph

Given is the model:

$$p(\mathcal{X}) = \frac{1}{Z} \prod_{f} \phi_f(\mathcal{X}_f)$$

Variable to Factor message:

For a central variable node x and incoming messages from neighbouring factors g, the outgoing message to the factor f is:

$$\mu_{x\to f}(x) = \prod_g \mu_{g\to x}(x)$$

which is a sum of all incoming message paths.

Factor to Variable Message

For a central factor node f and incoming messages from variables y, the outgoing message to the variable x is:

$$\mu_{f\to x}(x) = \sum_{\mathcal{X}_f,\text{not x}} \phi_f(\mathcal{X}_f) \prod_y \mu_{y\to f}(y)$$

is the sum over all incoming variable potentials and the product over all incoming message paths.

Marginal

For a central variable node x and all incoming factors f the overall factor for the variable is:

$$p(x) \propto \prod_f \mu_{f\to x}(x)$$

is the product over all incoming message.

Normalization

The overall normalization of the above model is:

$$Z = \sum_x \prod_f \phi_f(\mathcal{X}_f) = \sum_x \prod_{f(x)} \mu_{f\to x}(x)$$

is the product of all incoming messages for an arbitrary chosen variable node x. Since the product over absolute values of messages is required, the propagation on the way to the node might encounter numerical issues for small numbers. To prohibit this one uses the *log messages*:

$$\lambda = \log \mu$$

Then the variable to factor messages become:

$$\lambda_{x\to f}(x) = \sum_g \lambda_{g\to x}(x)$$

The marginals are:

$$\log p(x) = \sum_{f\in ne(x)} \lambda_{f\to x}(x)$$

and factor to variable messages become:

$$\lambda_{f\to x}(x) = \lambda^*_{y\to f} + \log(\sum_{\mathcal{X}_f,notx} \phi_f(\mathcal{X}_f \exp(\sum_y \lambda_{y\to f}(y) - \lambda^*_{y\to f}))$$

with:

$$\lambda^*_{y\to f} = \max_y \lambda_{y\to f}(y)$$

which is the largest value of the incoming log messages.

Related algorithms: Max-Product, N-Max-Product Algorithms, Bucket elimination, Loop-count conditioning

3.5 Multiply Connected Graphs

Inference via Message Passing can be done only for *singly connected graphs(polytrees)* i.e. a graph with only one connection between any two nodes. For multiply connected graphs one absorbs certain nodes into one to obtain a singly connected graph and then perform belief propagation.

Junction Tree representation is based on the reparametrization of the distribution. The distribution can be written as a product of marginal distributions, divided by a product of the intersections of the marginal distributions:

$$p(a,b,c,d) = p(a|b)p(b|c)p(c|d)p(d)$$

$$p(a,b,c,d) = \frac{p(a,b)}{p(b)}\frac{p(b,c)}{p(c)}\frac{p(c,d)}{p(d)}p(d) = \frac{p(a,b)p(b,c)p(c,d)}{p(b)p(c)}$$

This representation gives an explicit prescription for computation of marginals that are part of the equation.

3.5.1 Clique Graph

A useful reparametrization, which allows for dealing with multiply connected graphs, is in terms of cliques. Consider the Markov network:

$$p(a,b,c,d) = \frac{\phi(a,b,c)\phi(b,c,d)}{Z}$$

One can rewrite this as:

$$p(a,b,c,d) = \frac{p(a,b,c)p(b,c,d)}{p(b,c)}$$

which has the same overall structure as the original distribution, but is now formulated with the use of marginals. This representation is called *Clique Graph* and is useful for inference of marginals. The definition is:

$$CG = \frac{\prod_c \phi_c(\mathcal{X}^c)}{\prod_s \phi_s(\mathcal{X}^s)}, \qquad \mathcal{X}^s = \mathcal{X}^{c'} \cap \mathcal{X}^{c''}$$

for some potentials ϕ, clique indices c and separator indices s. In the following are the transformation steps:

Absorption

Consider neighbouring cliques $\mathcal{V}$ and $\mathcal{W}$, sharing the variables $\mathcal{S}$. For $\mathcal{X} = \mathcal{V} \cup \mathcal{W}$, the distribution is:

$$p(\mathcal{X}) = \frac{\phi(\mathcal{V})\phi(\mathcal{W})}{\phi(\mathcal{S})}$$

Observe that:

$$p(\mathcal{W}) = \sum_{\mathcal{V}/\mathcal{S}} p(\mathcal{X}) = \phi(\mathcal{W})\frac{\sum_{\mathcal{V}/\mathcal{S}}\phi(\mathcal{V})}{\phi(\mathcal{S})}$$

$$p(\mathcal{V}) = \sum_{\mathcal{W}/\mathcal{S}} p(\mathcal{X}) = \phi(\mathcal{V})\frac{\sum_{\mathcal{W}/\mathcal{S}}\phi(\mathcal{W})}{\phi(\mathcal{S})}$$

Systematically one fist defines the separator/'message' in direction $\mathcal{V} \to \mathcal{W}$:

$$\phi^*(\mathcal{S}) := \sum_{\mathcal{V}/\mathcal{S}} \phi(\mathcal{V})$$

With this the updated $\mathcal{W}$-potential after 'absorbing' $\phi^*(\mathcal{S})$ is:

$$\phi^*(\mathcal{W}) = \phi(\mathcal{W})\frac{\phi^*(\mathcal{S})}{\phi(\mathcal{S})} = p(\mathcal{W})$$

For the opposite direction one defines the separator:

$$\phi^{**}(\mathcal{S}) = \sum_{\mathcal{W}/\mathcal{S}} \phi^*(\mathcal{W}) = p(S)$$

Then in the opposite direction update for $\mathcal{V}$ one gets:

$$\phi^*(\mathcal{V}) = \frac{\phi(\mathcal{V})\phi^{**}(\mathcal{S})}{\phi^*(\mathcal{S})} = p(\mathcal{V})$$

Now the local message passing is defined. The whole graph gets updated with the following rules. There is exactly one message between two nodes possible. The message can be sent only after the node gets all incoming messages from other neighbouring nodes. The algorithm is completed after the messages has been sent in both directions. Note that there are usually several message passing sequences possible.

Related topics and algorithms: Junction Tree Algorithm, Multiply connected clique graph, maximum weight spanning tree, Kruskal's algorithm.

Part II

Learning Methods

Chapter 4

Learning model parameters

In Essence: *Usually in machine learning one chooses a known distribution to model the data and obtains the parameters of the model $p(x|\theta)$ by some learning algorithm.*

i.i.d. data: Independent and identically distributed data (observations x^i of the variable x) has no conditional dependence conditioned on the parameters θ:

$$p(x^1, ..., x^N|\theta) = \prod_{n=1}^{N} p(x^n|\theta)$$

Bias and consistency: Bias measures if the estimate of θ is correct 'on average'. Consistency is the property of convergence to the true parameters as the sequence of data increases.

Unbiased estimator: Given data $X = x^1, ..., x^N$, formed from i.i.d. samples of a distribution $p(x|\theta)$ we can use the data X to estimate the parameter θ that was used to generate the data. *The estimator is a function of the data*, which we write as $\hat{\theta}(\mathcal{X})$. For an unbiased parameter estimator one has:

$$\left\langle \hat{\theta}(\mathcal{X}) \right\rangle_{p(\mathcal{X}|\theta)} = \theta$$

More generally, one can consider any function of the distribution p(x), with scalar value θ, for example the mean $\theta = \langle x \rangle_{p(x)}$. Then for a $\hat{\theta}(\mathcal{X})$ if

$$\left\langle \hat{\theta}(\mathcal{X}) \right\rangle_{\tilde{p}(\mathcal{X})} = \theta$$

$\hat{\theta}(\mathcal{X})$ is an unbiased estimator w.r.t. the *data distribution* $\tilde{p}(\mathcal{X})$. The *sample mean* is an unbiased estimator for i.i.d. data, not like that is the *sample variance.*

Bayes formula:

$$p(\theta|\mathcal{X}) = \frac{p(\mathcal{X}|\theta)p(\theta)}{p(\mathcal{X})}, \qquad \text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}$$

with $p(\mathcal{X})$ *evidence* or *marginal likelihood.* Conditioning on the model itself the *Bayes formula* becomes:

$$\boxed{p(\theta|\mathcal{X}, M) = \frac{p(\mathcal{X}|\theta, M)p(\theta|M)}{p(\mathcal{X}|M)}}$$

where $p(\mathcal{X}|\mathcal{M})$ is the *model likelihood.*

Hyperparameters α: are parameters of the prior distribution. The term is used to distinguish them from parameters of the model for the underlying probability distribution.

Conugate prior: the prior is the conjugate distribution for the likelihood distribution if the posterior is of the same parametric form as the prior. Then the posterior gives an update on the hyperparameters:

$$p(\theta|\mathcal{X}, \alpha) = p(\theta|\alpha')$$

Bayes vs empirical decision

In the Bayesian decision approach, an explicit model of the data is made, with the end decision/-classification being computed independently of the fitting of the model to the data.

In the empirical risk minimization, there is typically no explicit model of the data, and the focus is on the end use of the predictor.

4.1 Training Methods

4.1.1 Bayesian method

Examine the posterior:

$$p(\theta|\mathcal{X}) \propto p(\mathcal{X}|\theta)p(\theta)$$

There is no summarization procedure given.

4.1.2 Maximum A Posteriori (MAP)

This is a summarization of the posterior prescription:

$$\theta^{MAP} = \text{argmax}_\theta p(\theta|\mathcal{X})$$

MAP is the maximum expected utility:

$$U(\theta) = \langle U(\theta_{true};\theta)|\mathcal{X}\rangle = \sum_{\theta_{true}} \mathbb{I}[\theta_{true} = \theta]p(\theta_{true}|\mathcal{X}) = p(\theta|\mathcal{X})$$

with the utility function:

$$U(\theta_{true}, \theta) = \mathbf{I}[\theta_{true} = \theta]$$

which is zero for all but the correct θ.

4.1.3 Maximum Likelihood (ML)

This is a summarization of the posterior prescription:

$$\theta^{ML} = \text{argmax}_\theta p(\mathcal{X}|\theta)$$

For a flat prior Maximum Likelihood solution is equvalent to the MAP solution. The logarithm gives the same optimum (strictly increasing logarithm, positive p), which is the *log-likelihood* $L(\theta)$:

$$\theta^{ML} = \text{argmax}_\theta \log p(\mathcal{X}|\theta) = \text{argmax}_\theta L(\theta)$$

The log-likelihood is sometimes easier to handle w.r.t. derivatives, since for i.i.d. data one has:

$$L(\theta) = \sum_n \log p(x^n|\theta)$$

With that the posterior is easily computed:

$$\boxed{\log p(\theta|\mathcal{X}) = \sum_n \log p(x^n|\theta) + \log p(\theta) - \log p(\mathcal{X})}$$

4.1.4 Moment Matching

Based on empirical estimate of a moment, θ is set such that the moment of the distribution matches the empirical moment.

4.1.5 Pseudo Likelihood

$\theta = \text{argmax}_\theta \sum_{n=1}^{N} \sum_{i=1}^{D} \log p(x_i^n | x_{\text{not } i}^n, \theta)$ used for multivariate $\mathbf{x}$, when the true likelihood is difficult to compute.

4.1.6 Crude estimate of the posterior

A distribution with all its mass in a single most likely state $\delta(\theta, \theta^{MAP})$. However then the information on the reliability of the parameter estimate is lost.

4.2 Maximum Likelihood Training

Assuming the model is correct

Assuming that the data was generated by the underlying model $p(x|\theta^{tr})$, one fits a model $p(x|\theta)$ of the same form and hopes that for large data $\theta \to \theta^{tr}$, as θ is learned by ML. Assuming i.i.d. the **scaled log-likelihood** is:

$$sL(\theta) = \frac{1}{N} \log p(\mathcal{X}|\theta) = \frac{1}{N} \sum_{n=1}^{N} \log p(x^n|\theta)$$

We have:

$$\begin{aligned} sL(\theta) &=^{N\to\infty} \langle \log p(x|\theta) \rangle_{p(x|\theta^{tr})} = -KL\Big(p(x|\theta^{tr})|p(x|\theta)\Big) + \big\langle \log p(x|\theta^{tr}) \big\rangle_{p(x|\theta^{tr})} \\ &= -KL\Big(p(x|\theta^{tr})|p(x|\theta)\Big) + \text{const.} \end{aligned}$$

In the limit of large data one can therefore learn the correct parameters θ^{tr} with ML. Therefore ML is a *consistent estimator*.

Assuming the model is incorrect

Having the assumed model $q(x|\theta)$ and the correct generative model $p(x|\phi)$ of a different form, assuming our model is correct yields:

$$\begin{aligned} L(\theta)_{N\to\infty} &= \langle \log q(x|\theta) \rangle_{p(x|\phi)} = -KL\Big(p(x|\phi)|q(x|\theta)\Big) + \big\langle \log p(x|\theta^{tr}) \big\rangle_{p(x|\phi)} \\ &= -KL\Big(p(x|\phi)|q(x|\theta)\Big) + \text{const.} \end{aligned}$$

Since q and p are of different form, it is in this case not necessary that $\theta^{tr} = \phi$.

Empirical distribution

For a discrete empirical distribution $q(x)$ and a distribution $p(x)$ one has:

$$KL(q|p) = -\frac{1}{N} \sum_{n=1}^{N} \log p(x^n) + \text{const.} = -sL + \text{const.}$$

One recognizes the scaled log Likelihood of i.i.d. data under the model $p(x)$. This means that setting parameters by maximum Likelihood is equivalent to minimizing the KL divergence w.r.t. empirical distribution. In case $p(x)$ is unconstrained, the ML optimal choice is the empirical distribution.

4.3 Learning a Gaussian

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

with the *precision matrix* $\boldsymbol{\Sigma}^{-1}$.

Maximum Likelihood training (i.i.d.)

Optimal $\boldsymbol{\mu}$:

$$\nabla_{\boldsymbol{\mu}} L(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = 0$$

$\boldsymbol{\mu}^*$ is the sample mean.

Optimal $\boldsymbol{\Sigma}$: is the sample covariance.

Bayesian training (i.i.d., univariate)

For the Bayesian training we need the posterior, conjugate prior and Likelihood. The Likelihood $\mathcal{N}(\mathcal{X}|mu, \sigma^2)$ is a Gaussian. The posterior is:

$$p(\mu, \sigma^2|\mathcal{X}) \propto p(\mathcal{X}|\mu, \sigma^2)p(\mu, \sigma^2) = p(\mathcal{X}|\mu, \sigma^2)p(\mu|\sigma^2)p(\sigma^2)$$

The conjugate prior is:

$$p(\mu|\sigma^2)p(\sigma^2)$$

The convenient choice for the prior on the mean $p(\mu|\sigma^2)$ is that it is a Gaussian centered around a μ_0:

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

The posterior is correspondingly:

$$p(\mu, \sigma^2|\mathcal{X}) \propto \mathcal{N}(\mathcal{X}|\mu, \sigma^2)\mathcal{N}(\mu|\mu_0, \sigma_0^2)p(\sigma^2)$$

With the constraint $\sigma_0^2 = \text{const.} \cdot \sigma^2$ one finds for the $p(\sigma^2)$ part of the prior, that the Inverse-Gamma distribution is conjugate. Therefore the whole prior should take the **Gauss-Inverse-Gamma** form and has a Gauss-Inverse-Gamma conjugate posterior.

4.4 ML-II Method

Setting the parameters by maximum likelihood corresponds to finding the parameters θ that maximizes $p(x|\theta)$, where x are observations. In some cases however we may have an idea about which parameters θ are more appropriate and can express this prior preference using a distribution $p(\theta)$. This knowledge can be modeled with a parametrized prior $p(\theta|\alpha)$ where α are the hyperparameters. ML-II corresponds to finding optimal α that maximizes the likelihood $p(x|\alpha) = \int_\theta p(x|\theta)p(\theta|\alpha)$. By treating the parameters α as variables, one can view this then as learning under hidden variables.

Chapter 5

ML Belief Network Training

In Essence: *Maximum likelihood learning under the i.i.d. assumption corresponds mathematically to setting the CPT (Conditional Probability Table) entries by counting the relative number of occurrences.*

5.1 Counting

For a BN we have:

$$p(x) = \prod_{i=1}^{K} p(x_i|\mathrm{pa}(x_i))$$

To compute the maximum likelihood setting of each term $p(x_i|\mathrm{pa}(x_i))$, we can equivalently minimise the Kullback-Leibler divergence between the empirical distribution q(x) and the BN p(x). We have:

$$KL(q|p) = -\sum_{i=1}^{K} \langle \log p(x_i|\mathrm{pa}(x_i)) \rangle_{q(x_i,\mathrm{pa}(x_i))} + \text{const.}$$

using:

$$\langle f(\mathcal{X}_i) \rangle_{q(\mathcal{X})} = \langle f(\mathcal{X}_i) \rangle_{q(\mathcal{X}_i)} .$$

After adding an entropic term we have:

$$KL(q|p) = \sum_{i=1}^{K} \langle KL(q(x_i|\mathrm{pa}(x_i))|p(x_i|\mathrm{pa}(x_i))) \rangle_{q(\mathrm{pa}(x_i))}$$

This is a positive weighted sum of individual Kullback-Leibler divergences. The minimal Kullback-Leibler setting, and that which corresponds to maximum likelihood, is therefore:

$$p(x_i|\mathrm{pa}(x_i)) = q(x_i|\mathrm{pa}(x_i))$$

and:

$$p(x_i = s|\mathrm{pa}(x_i) = t) \propto \sum_{n=1}^{N} \mathbf{I}[x_i^n, \mathrm{pa}(x_i^n) = t]$$

This expression corresponds to the intuition that the table entry $p(x_i|pa(x_i))$ can be set by counting the number of times the state $\{x_i = s, \mathrm{pa}(x_i) = t\}$ occurs in the dataset (where t is a vector of parental states). The table is then given by the relative number of counts of being in state s compared to the other states s', for fixed joint parental state t.

5.2 Conditional probability functions

Consider a binary variable y with n binary parental variables x. This corresponds to 2^n entries in the CPT of $p(y|x)$. To reduce the complexity of the CPT we may constrain the form of the table. For example, one could use a function:

$$p(y=1|x,w) = \frac{1}{1+e^{-w^T x}}$$

where we only need to specify the n-dimensional parameter vector w. In this case, rather than using maximum likelihood to learn the entries of the CPTs directly, we instead learn the values of the parameters w.

Bayesian Belief Network Training: under several assumptions (i.i.d) and with a convenient prior (factorized globally and locally), the multidimensional posterior also factorizes into univariate distributions over parameters θ. The learning is then done by computing the posterior distributions $p(\theta_i|\mathcal{X}_i)$, with $\mathcal{X}_i$ being a set of data from the family of the variable (node) i.

Structure Learning: PC algorithm learns first the skeleton and orients the nodes to a DAG.

Chapter 6

ML Markov Network Training

Consider a Markov network $p(\mathcal{X})$ on cliques $\mathcal{X}_c$ with clique parameters θ_c:

$$p(\mathcal{X}|\theta) = \frac{1}{Z(\theta)} \prod_{c=1}^{C} \phi_c(\mathcal{X}_c|\theta_c), \qquad Z = \sum_{\mathcal{X}} \prod_{c=1}^{C} \phi_c(\mathcal{X}_c|\theta_c)$$

Assuming i.i.d data $\mathcal{X} = [x_1, ..., x_N]$, the log-likelihood is:

$$L(\theta) = \sum_n \log p(\mathcal{X}^n|\theta) = \sum_n \sum_c \log \phi_c(\mathcal{X}_c^n|\theta_c) - N \log Z(\theta)$$

In general case $Z(\theta)$ couples the parameters. For exact results one has to restrict to special cases. Gradient based techniques give overall informations on the distribution. The *likelihood gradient* can be used for numerical computation:

$$\frac{\partial}{\partial \theta_c} L(\theta) = \sum_n \frac{\partial}{\partial \theta_c} \log \phi_c(\mathcal{X}_c^n|\theta_c) - N < \frac{\partial}{\partial \theta_c} \log \phi_c(\mathcal{X}_c|\theta_c) >_{p(\mathcal{X}_c|\theta)}$$

6.0.1 Exponential Clique Potentials

A common model are exponential potentials:

$$\phi_c(\mathcal{X}_c) = \exp(\boldsymbol{\theta}_c^T \boldsymbol{\psi}_c(\mathcal{X}_c))$$

where $\boldsymbol{\theta}_c$ are the vector parameters and $\boldsymbol{\psi}_c(\mathcal{X}_c)$ is a fixed *feature function.* Using

$$\frac{\partial}{\partial \theta_c} \log \phi_c(\mathcal{X}_c|\theta_c) = \boldsymbol{\psi}_c(\mathcal{X}_c)$$

one finds that the likelihood derivative is zero when:

$$\frac{1}{N} \sum_n \boldsymbol{\psi}_c(\mathcal{X}_c^n) =< \boldsymbol{\psi_c(\mathcal{X}}_c) >_{p(\mathcal{X}_c)}$$

i.e. maximum likelihood solution is where the empirical average of a feature function matches the average of the feature function w.r.t. the model. With the definition of the empirical distribution on clique variables:

$$\epsilon(\mathcal{X}_c) := \frac{1}{N} \sum_{n=1}^{N} \mathbb{I}[\mathcal{X}_c = \mathcal{X}_c^n]$$

the condition can be written as:

$$< \boldsymbol{\psi}_c(\mathcal{X}_c) >_{\epsilon(\mathcal{X}_c)} =< \boldsymbol{\psi}_c(\mathcal{X}_c) >_{p(\mathcal{X}_c)}$$

6.0.2 Unconstrained Clique Potentials

One has to work with probability tables entries for each state of the potential. One can write the potentials conveniently:

$$\phi_c(\mathcal{X}_c^n) = \prod_{\mathcal{Z}_c} \phi_c(\mathcal{Z}_c)^{\mathbb{I}[\mathcal{Z}_c = \mathcal{X}_c^n]}$$

where $\mathcal{Z}_c$ are all states for potential ϕ_c and $\phi_c(\mathcal{Z}_c)$ is a probability table entry. With that one finds the log likelihood:

$$L(\theta) = \sum_c \sum_{\mathcal{Z}_c} \sum_n \mathbb{I}[\mathcal{Z}_c = \mathcal{X}_c^n] \log \phi_c(\mathcal{Z}_c) - N(\phi)$$

Its derivative w.r.t. a specific table entry is:

$$\frac{\partial}{\partial \phi_c(\mathcal{Z}_c)} L(\theta) = \sum_n \mathbb{I}[\mathcal{Z}_c = \mathcal{X}_c^n] \frac{1}{\phi_c(\mathcal{Z}_c)} - N \frac{p(\mathcal{Z}_c)}{\phi_c(\mathcal{Z}_c)}$$

The zero derivative solution are marginals of the form:

$$p(\mathcal{X}_c) = \epsilon(\mathcal{X}_c)$$

i.e. in general the ML optimal clique potentials are set such that the clique marginal distributions $p(\mathcal{X}_c)$ are equal the empirical distributions $\epsilon(\mathcal{X}_c)$ for each clique. To concretely get a closed analytical solution for the potentials once can use *iterative proportional fitting* as an iterative update:

$$\phi^{\text{new}}(\mathcal{X}_c) = \frac{\phi(\mathcal{X})_c \epsilon(\mathcal{X}_c)}{p(\mathcal{X}_c)}$$

This is a cordinate-wise optimization with the coordinate $\phi_c(\mathcal{X}_c)$ and other coordinates fixed. $p(\mathcal{X}_c)$ is recomputed each step, which is in general expensive for graphs with a wide junction tree.

Related topics: Decomposable Markov networks, CRFs

Chapter 7

Bayesian Model Selection

For a fixed set of models the model posterior probability of a model given some data is:

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

The normalization is:

$$p(\mathcal{D}) = \sum_{i=1}^{m} p(\mathcal{D}|M_i)p(M_i)$$

The relative probability of a model is:

$$\frac{p(M_i|\mathcal{D})}{p(M_j|\mathcal{D})} = \frac{p(\mathcal{D}|M_i)}{p(\mathcal{D}|M_j)} \frac{p(M_i)}{p(M_j)}$$

The model likelihood for continuous model parameters is given as:

$$p(\mathcal{D}|M_i) = \int p(\mathcal{D}|\theta_i, M_i)p(\theta_i|M_i)d\theta_i$$

For N i.i.d. data points the likelihood takes the form:

$$p(\mathcal{D}|M_i) = \int p(\theta_i|M_i) \prod_{n=1}^{N} p(x^n|\theta_i, M_i)d\theta_i$$

and the unnormalized log-posterior is:

$$\log p(\mathcal{D}|M_i)p(M_i) = \log p(\boldsymbol{\theta}|M) + \sum_{n=1}^{N} \log p(x^n|\boldsymbol{\theta}, M)$$

7.1 Laplace method

For computing the model likelihood the integral is usually intractable and has to be approximated. The Laplace method to approximate the model log-likelihood for a Gaussian-like posterior is to find $\boldsymbol{\theta}^*$ via MAP and to fit a Gaussian to this point in the parameter space based on the local curvature:

$$\boldsymbol{\theta}^* = \operatorname*{argmax}_{\boldsymbol{\theta}} \; p(\mathcal{D}|\boldsymbol{\theta}, M)p(\boldsymbol{\theta}|M)$$

$$\log p(\mathcal{D}|M) \approx \log p(\mathcal{D}|\boldsymbol{\theta}^*, M) + \log p(\boldsymbol{\theta}^*|M) + \frac{1}{2}\log\det(2\pi\mathbf{H}^{-1})$$

Hereby $\mathbf{H}$ is the Hessian of the negative log posterior evaluated at $\boldsymbol{\theta}^*$.

7.2 BIC Approximation

An even simpler method is to crudely approximate the Hessian $\mathbf{H}$ by taking $\mathbf{H} \approx N\boldsymbol{I}_k$ where $K = \dim(\boldsymbol{\theta})$ is the number of model parameters. With this Hessian the Laplace approximation is:

$$\log p(\mathcal{D}|M) \approx \log p(\mathcal{D}|\boldsymbol{D}|\boldsymbol{\theta}^*, M) + \log p(\boldsymbol{\theta}^*|M) + \frac{K}{2}(\log 2\pi - \log N)$$

One can further approximate the prior and take $p(\boldsymbol{\theta}|M) = \mathcal{N}(\boldsymbol{\theta}|\mathbf{0}, \mathbf{I})$, which penalizes the length of the parameter vector and favorizing a simple model. With that the log-likelihood is:

$$\log p(\boldsymbol{D}|M) \approx \log p(D|\boldsymbol{\theta}^*, M) - \frac{1}{2}(\boldsymbol{\theta}^*)^T\boldsymbol{\theta}^* - \frac{K}{2}\log N$$

By ignoring the penalty term for large N one arrives at the definition of the **Bayes Information Criterion (BIC)**:

$$BIC = K\log N - 2\log p(D|\boldsymbol{\theta}^*, M)$$

BIC is closely related to the **Akaike Information Criterion (AIC)** which is:

$$AIC = 2K - 2\log p(D|\boldsymbol{\theta}^*, M)$$

Chapter 8

Learning with Hidden Variables

In Essence: *New components of the model are hidden/latent variables x_h, i.e. parameters which are essential for the model but potentially cannot be observed due and missing variables which typically miss at random due to imperfection of data and are simply modelled by an index function $\boldsymbol{I}_m$.*

In general the probability of a special outcome of $x_v, \mathbf{I}_m$ will be:

$$p(x_v, \mathbf{I}_m|\theta) = \sum_{x_i} p(x_v, x_i, \mathbf{I}_m|\theta) = \sum_{x_i} p(\mathbf{I}_m|x_v, x_h, \theta)p(x_v, x_i|\theta)$$

One can further assume that the generation process for missing variables only depends on visible variables which is called **missing at random (MAR)**:

$$p(\mathbf{I}_m|x_v, x_h, \theta) = p(\mathbf{I}_m|x_v)$$

With that the probability of the observables of the model is:

$$p(x_v, \mathbf{I}_m|\theta) = p(\mathbf{I}_m|x_v)p(x_v|\theta)$$

and the parameters can be accessed from the marginal likelihood. MAR is an assumption that cannot be verified statistically.

The **missing completely at random assumption (MCAR)** assumes that the generation process of missing variables is completely independent of the model entities:

$$p(\mathbf{I}_m|x_v, x_h, \theta) = p(\mathbf{I}_m)$$

In MCAR case the model is completely unbiased. MCAR is almost never the case.

8.1 Variational Expectation Maximization

In Essence: *hidden variables prevent decoupling of marginals. The theoretical discussion in this chapter shows how EM is aimed on replacing the objective function by the lower bound where the coupling is removed.*

Consider a single visible variable v and a single hidden variable h. One wishes to set θ for the model $(v, h|\theta)$ by maximizing the marginal likelihood $p(v|\theta)$. The Kullback-Leibler divergence between a 'variational' distribution $q(h|v)$ and the model is:

$$KL(q(h|v)|p(h|v,\theta)) =< \log q(h|v) - \log p(h|v,\theta) >_{q(h|v)} \geq 0$$

With the use of:

$$p(h|v,\theta) = \frac{p(h, v|\theta)}{p(v|\theta)}$$

one can rewrite:

$$KL(q(h|v)|p(h|v,\theta)) =< \log q(h|v) >_{q(h|v)} - < \log p(h,v|\theta) >_{q(h|v)} + \log p(v|\theta) \geq 0$$

This gives a lower bound on the marginal likelihood for a single training example:

$$\log p(v|\theta) \geq - < \log q(h|v) >_{q(h|v)} + < \log p(h,v|\theta) >_{q(h|v)} = \text{Entropy} + \text{Energy}$$

The energy term is also called the 'expected complete data log likelihood'. Having more than one data point, i.i.d. data $\mathcal{V}$, the full log likelihood is:

$$\log p(\mathcal{V}|\theta) = \sum_{n=1}^{N} \log p(v^n|\theta)$$

And the lower bound is:

$$-\sum_{n=1}^{N} < \log q(h^n|v^n) >_{q(h^n|v^n)} + \sum_{n=!} < \log p(h^n, v^n|\theta) >_{q(h^n|v^n)} := B(\mathcal{Q}, \theta)$$

where $\mathcal{Q}$ is a set of variational distributions and the bound is again the sum of an energy term and an entropy term. The optimization of the lower bound is done via first optimizing the equation w.r.t. $\mathcal{Q}$ keeping θ constant **(E-step)** and then w.r.t. θ keeping $\mathcal{Q}$ constant **(M-step)**. The two steps are repeated until convergence. Since during the E-step q is fixed, it only maximizes the energy term (called classical EM). The steps are repeated until convergence. The fully optimal setting is:

$$q(h^n|v^n) = p(h^n|v^n, \theta) \implies \log p(\mathcal{V}|\theta) = B(\mathcal{Q}, \theta)$$

Due to the local optimization the global maximum can not always be reached. By design of EM the lower bound is never decreased. A positive side effect is that the log likelihood is optimized too, i.e. is higher with the updated θ.

8.2 EM for Belief networks

Given the general form of a BN:

$$p(x) = \prod_i p(x_i|pa(x_i))$$

and N variables $x = (v, h)$ with its visible and hidden part the **energy term** is:

$$\sum_n < \log p(x^n) >_{q_t(h^n|v^n)} = \sum_n \sum_i < \log p(x_i^n|pa(x_i^n)) >_{q_t(h^n|v^n)} = N < \log p(x) >_{q_t(x)}$$

This expression is obtained by using the notation:

$$q_t^n(x) := q_t(h^n|v^n)\delta(v, v^n)$$

and defining a **mixture distribution**:

$$q_t(x) := \frac{1}{N} \sum_{n=1}^{N} q_t^n(x)$$

where t is the iteration index. The E-step is (without derivation):

$$q_t(h^n|v^n) = p^{old}(h^n|v^n)$$

The M-step is:

$$p^{new}(x_i|pa(x_i)) = \frac{\sum_n q_t^n(x_i, pa(x_i))}{\sum_{n'} q_t^{n'}(pa(x_i))}$$

8.3 EM for Markov networks

Given the general form of a MN:

$$p(v,h|\theta) = \frac{1}{Z(\theta)} \prod_c \phi_c(h,v|\theta_c),\ Z(\theta) = \sum_{v,h} \prod_{c=1}^{C} \phi_c(h,v|\theta_c),\ \theta = (\theta_1,...,\theta_C)$$

with clique parameters θ_c, one finds the variational bound as being:

$$\log p(v|\theta) \geq -< \log p(x) >_{p(x)} + \sum_c < \log \phi_c(h,v|\theta_c) >_{q(h)} - \log Z(\theta)$$

where the first term is the entropy term. The bound only partially resolves the coupling, since the parameters are still coupled in the normalization term. One technique is to also put a bound on the normalization term.

8.4 Variational Bayes

The generalization of EM is Variational Bayes. Since it a a Bayesian method, the parameter posterior given a single observation v is:

$$p(\theta|v) \propto p(v|\theta)p(\theta) = \sum_h p(v,h|\theta)p(\theta)$$

In Variational Bayes one approximates by factorization:

$$p(h,\theta|v) \approx q(h)q(\theta)$$

The factors $q(h)$, $q(\theta)$ are found by minimizing the KL divergence between $p(h,\theta|v)$ and $q(h)q(\theta)$:

$$KL = -< \log q(h) >_{q(h)} + < \log q(\theta) >_{q(\theta)} - < \log p(h,\theta|v) >_{q(h)q(\theta)} \geq 0$$

this gives the bound:

$$\log p(v) \geq -< \log q(h) >_{q(h)} - < \log q(\theta) >_{q(\theta)} + < \log p(v,h,\theta) >_{q(h)q(\theta)}$$

with the E-step

$$q^{new}(h) = \underset{q(h)}{\text{argmin}} KL(q(h)q^{old}(\theta)|p(h,\theta|v))$$

and the M-step

$$q^{new}(\theta) = \underset{q(\theta)}{\text{argmin}} KL(q^{new}(h)q(\theta)|p(h,\theta|v))$$

One can find that

$$q^{new}(h) \propto \exp < \log p(v,h|\theta) >_{q(\theta)}$$

and

$$q^{new}(\theta) \propto p(\theta) \exp < \log p(v,h|\theta) >_{q(h)}$$

For i.i.d. data one obtains the bound for the whole data set as:

$$\log p(V) = \sum_n \log p(v^n)$$

EM is a special case of Variational Bayes with a flat prior $p(\theta) = const.$ and delta function approximation of the parameter posterior $q(\theta) = \delta(\theta, \theta^*)$.

More techniques exist which aim at resolving computational intractabilities:

Extensions: Partial E/M step, tractable factorizable or Gaussian classes of q, Viterbi training, stochastic EM

Part III

Neural Models

Chapter 9

Recurrent Neural Networks

In Essence: *Recurrent neural networks encode a temporal evolution by its directed connections that form a directed graph. In contrast to feed forward neural networks the graph is not acyclic. Recurrent neural network can use their internal state, their memory, to evaluate their **input** nodes. The model is Turing complete. An example is the Hopfield network.*

Think of handwriting recognition. The input x is temporal and highly correlated, as long as one word is written. Therefore it has to exhibit some kind of memory of continuous input $x_t - 1$.

The **recurrence formula** is:

$$h_t = f_W(h_{t-1}, x_t)$$

where h_t is the new state of the single (hidden) layer of the network, f_W is some function with parameters W, h_{t-1} is the old state and x_t is input vector at time stamp t. The easiest one layer recurrent neural network is (time is not depth!):

$$h_t = \tanh(U_{hh}h_{t-1} + W_{xh}x_t) = \tanh(W \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix})$$

$$y_t = V_{hy}h_t$$

where memory U and input weights W are combined into single weight matrix.

9.0.1 Training

For training we have the fixed vectors x_i, h_0 and the trained matrix W consists of three components: U_{hh}, W_{xh} and V_{hy}. U_{hh} is the only term that exhibits long-term memory and contains the information of all previous states. All weight matrices are **time independent**, i.e. at each time stemp the same coefficients w, u, v are used, which reduces the number of parameters.

Backpropagation one time stamp back, i.e. gradient of loss L w.r.t. h_t, multiplies by the same W_{hh}^T each time stamp. This is problematic, since by multiplication by the same number over and over again the gradient will eventually explode (largest singular value of W > 1) or vanish (largest singular value of W < 1).

9.1 Long Short Term Memory (LSTM)

In Essence: *The LSTM gates are designed to allow the information to be gated into the memory cell and then to be gated out when needed. In the intermediate period the gate is closed, so that the information that arrives does not interfere with the remembered state.*

To address the error propagation issue a new architecture of LSTM networks has been designed.

Memorizing long-term correlations is their default behaviour, they don't have to learn this. Instead they learn to forget, this is how they eventually resolve the long-term dependencies problem. The general recurrence structure for the hidden state is still the same:

$$h_t = f_W(h_{t-1}, x_t)$$

Additionally to h the network has a second hidden variable, which is not exposed to outside, the "cell state" c, with its own recurrence, exposed to the next cell:

$$c_t = f * c_{t-1} + i * \tilde{c}$$

$$h_t = o * \tanh c_t$$

where:

$$\begin{bmatrix} i \\ f \\ o \\ \tilde{c} \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix}$$

where:

- σ is sigmoid function [0,1]
- i is the Input gate (whether to write to cell)
- f is the Forget gate (whether to erase cell)
- o is the Output gate (how much to reveal cell)
- $\tilde{c}$ are the candidate values which should be added to the cell state from the memory and range from -1 to 1

The gates are technically the 0-1 gate vectors and the matrix parts of the weight matrix W_{hx}: W_f, W_i, W_o, $W_{\tilde{c}}$. All this components now have memory parts and depend on h_{t-1} as well as on x_t:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t])$$

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t])$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t])$$

$$\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}, x_t])$$

We see that the **input** of each cell is c_{t-1}, h_{t-1} and x_t. The **output** to the next cell is c_t, h_t.

The element-wise multiplication with the Forget matrix resolves the gradient problem, since the elements are not fixed, but vary with time, unlike the long-term memory case. Additionally the elements are normalized to be between 0 and 1, which regularizes the problem. But most importantly the + sign in the c_t equation replaces eventually the multiplication and is much more stable in terms of error propagation. This eventually solves the problem.

Chapter 10

Stochastic Neural Networks

In Essence: *stochastic neural networks introduce random variables into an artificial neural network to solve computationally intensive optimization problems with full utilization of fluctuations as a facilitator. Stochastic can be the transitions or the weights.*

10.1 Energy based models

For performance reasons one uses the stochastic gradient on the empirical log-likelihood of the training data:

$$\mathcal{L}(\theta, D) = \frac{1}{N} \sum_{x^{(i)} \in D} \log p(x^{(i)})$$

where θ are parameters of the model. The loss function is:

$$l(\theta, D) = -\mathcal{L}(\theta, D)$$

The stochastic gradient is:

$$-\frac{\partial \log p(x^{(i)})}{\partial \theta}$$

To increase the predictive power of the model and account for missing data, one introduces hidden variables:

$$p(x) = \sum_h p(x, h) = \sum_h \frac{e^{-E(x,h)}}{Z}$$

The free energy is:

$$\mathcal{F}(x) = -\log \sum_h e^{-E(x,h)}$$

With that the stochastic gradient separates into a positive phase and a negative phase:

$$-\frac{\partial \log p(x)}{\partial \theta} = \frac{\partial \mathcal{F}(x)}{\partial \theta} - \sum_{\tilde{x}} p(\tilde{x}) \frac{\partial \mathcal{F}(\tilde{x})}{\partial \theta}$$

The positive phase increases the probability of training data, the negative phase decreases the probability of samples generated by the model. An approximation is obtained by using *negative particles* $\mathcal{N}$:

$$-\frac{\partial \log p(x)}{\partial \theta} \approx \frac{\partial \mathcal{F}(x)}{\partial \theta} - \frac{1}{|\mathcal{N}|} \sum_{\tilde{x} \in \mathcal{N}} \frac{\partial \mathcal{F}(\tilde{x})}{\partial \theta}$$

The elements $\tilde{x} \in \mathcal{N}$ are sampled according to p as a MCMC.

10.2 The Ising Model

$$p(x_1, ..., x_n) = \frac{1}{Z}\prod_{i\sim j}\phi_{ij}(x_i, x_j),\ \ \phi_{ij}(x_i, x_j) = e^{-\frac{1}{2T}(x_i-x_j)^2},\ \ x_i \in \{-1, +1\}$$

with i, j are neighbours in the undirected graph, T is the temperature and the potential encourages aligned states. The Ising model is a special case of a MRF and a blueprint for some of the physical models in pattern recognition.

10.3 Gibbs Random Field

For the case in which clique potentials of the Markov Network are strictly positive, the probability distribution can be written as a **Gibbs distribution** with energy E:

$$p(x_1, ..., x_n) = \frac{1}{Z}\exp(-\beta E(\mathcal{X}_c)), \qquad E \propto \sum_{c=1}^{C} \ln \phi_c(\mathcal{X}_c)$$

For a Gibbs random field, the **local, pairwise and global Markov properties** are all equivalent and satisfied.

Hammersley-Clifford Theorem: a probability distribution that has a positive density satisfies one of the Markov properties with respect to an undirected graph G if and only if it is a Gibbs random field, that is, its density can be factorized over the cliques (or complete subgraphs) of the graph.

10.4 Restricted Boltzmann Machine

A Boltzmann machine is a MN, a particular log-linear MRF (energy function is linear in its free parameters), on binary variables with the probability distribution of the form $p(x) = \frac{1}{Z}e^{-E(v,h)}$:

$$p(x) = \frac{1}{Z}(\mathbf{w}, b)e^{\sum_{i<j} w_{ij}x_i x_j + \sum_i b_i x_i}$$

with the interactions w_{ij} (weights) and the biases b_i. The graphical model of the Boltzmann machine is an undirected graph with a link between nodes i and j for $w_{ij} \neq 0$. Consequently, for all but specially constrained W, the graph is multiply-connected and inference will be typically intractable. These are called *Restricted Boltzmann Machines.*

The simplest RBM is a bipartite graph with shared weights, i.e. two layers, a visible first layer i and a hidden second layer j. In that case the energy function is:

$$E(v, h) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_i \sum_j v_i w_{ij} h_j$$

which is in matrix notation:

$$E(v, h) = -a^T v - b^T h - v^T W h$$

with a being the biases for the hidden units. Training is done via *back-propagation*, i.e. the parameters (b, w) are trained in unsupervised manner by comparing the reconstruction, starting with the output neural activations and going backwards in the parametric model, with the initial signal and adjusting the parameters when the *Kullback-Leibler Divergence* between both is large.

The free energy is:

$$\mathcal{F}(v) = -b^T v - \sum_i \log \sum_{h_i} e^{h_i(c_i + W_i v)}$$

With conditional independence assumption one has:

$$p(h|v) = \prod_i p(h_i|v)$$

$$p(v|h) = \prod_j p(v_j|h)$$

Given binary units v_j and $h_i \in \{0, 1\}$ one has:

$$p(h_i = 1|v) = sign(c_i + W_i v)$$

$$p(v_j = 1|h) = sign(b_j + W_j^T h)$$

With this the free energy is:

$$\mathcal{F}(v) = -b^T v - \sum_i \log(1 + e^{(c_i + W_i v)})$$

The gradients are:

$$-\frac{\partial \log p(v)}{\partial W_{ij}} = \mathbb{E}_v[p(h_i|v)v_j] - v_j^{(i)} sign(W_i v^{(i)} + c_i)$$

$$-\frac{\partial \log p(v)}{\partial c_i} = \mathbb{E}_v[p(h_i|v)] - sign(W_i v^{(i)})$$

$$-\frac{\partial \log p(v)}{\partial b_j} = \mathbb{E}_v[p(v_j|h)] - v_j^{(i)}$$

10.4.1 Sampling

RBM uses block Gibbs sampling where h and v are sampled given each other fixed:

$$h^{(n+1)} \sim sign(W^T v^{(n)} + c)$$

$$v^{(n+1)} \sim sign(W h^{(n+1)} + b)$$

Contrastive Divergence is an approximation technique for speeding up known in the context of RBMs used for the likelihood gradient approximation of the backpropagation algorithm via a short MCMC. It is shown that even one cycle of MCMC is sufficient for convergence of the RBM.

Chapter 11

Neural Belief Networks with Markovian Dynamics

11.1 Hopfield Network

In Essence *In this section an example of a dynamic belief network model is discussed. The Hopfield network is a temporal recurrent neural model which has a large analogy to a magnetic field of individual spins in a fermi gas.*

Let's use a spin function $s[+1,-1]$, where -1 denotes a neuron that is quescient (not firing) and $+1$ denotes a neuron that is in a firing state. The overall potential at a neuron i at time t is then:

$$\phi_i(t) = b_i - \sum_{j=1}^{V} w_{ij} s_j(t)$$

where w is the coupling parameter and b is the activation bias. The energy of the system is then:

$$E = -\frac{1}{2}\sum_{ij} w_{ij} s_i s_j + \sum_i b_i s_i, \qquad E = -\frac{1}{2} s^T W s + b^T s$$

To introduce stochastic behaviour, one models the firing probability at i and $t+1$ as:

$$p(s_i(t+1) = +1|\mathbf{s}(t)) = f(\phi_i(t)), \qquad f(\phi) = \frac{1}{1+e^{-\beta\phi}}$$

which is the inverse of the step function of Fermi-Dirac distribution. The parameter $\beta = \frac{1}{k_B T}$ is controlling the stochastic behaviour of the neuron, where with higher temperature the step function is smoothed out. Additionally the probability of being in the quescient state is simply $1 - p(s_i(t+1) = +1|\mathbf{s}(t))$. Alltogether:

$$\boxed{p(s_i(t+1)|\mathbf{s}(t)) = f(s_i(t+1))\phi_i(t)}$$

In the limit of zero temperature $\beta \to \infty$ the neurons update deterministically (and similar to spins without latency):

$$s_i(t+1) = sgn(\phi_i(t))$$

In a synchronous Hopfield network all neurons are updated simultaneously and the time evolution can be represented as a dynamic Belief network:

$$p(\mathbf{s}(t+1)|\mathbf{s}(t)) = \prod_{i=1}^{S} p(s_i(t+1)|\mathbf{s}(t))$$

11.1.1 Standard Hebb's rule

Training is aimed on lowering the energy of the network. The weights are set in a simple rule, based only on individual interactions, hence most suitable for temporally uncorrelated systems:

$$w_{ij} = \frac{1}{S}\sum_{t=1}^{T-1} s_i(t+1)s_j(t)$$

The Hebb's rule is capable of storing an uncorrelated temporal sequence of length $0.269S$ time stamps.

11.1.2 Pseudo inverse rule

The pseudo inverse rule solves the equations:

$$\sum_j w_{ij}s_j(t) = s_i(t+1)$$

or in matrix notation:

$$\boldsymbol{W}\boldsymbol{S} = \boldsymbol{S}^{'}$$

This corresponds to the zero temperature deterministic limit, where the pattern is fully recalled. For $T < S$ the linear equation has multiple solutions, one of them is the pseudo inverse:

$$\boldsymbol{W} = \boldsymbol{S}^{'}(\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{S}^T$$

The PI can store any sequence of S linearly independent patterns. On the other hand it has very small basins of attraction for temporary correlated patterns, i.e. suffers from noise.

11.1.3 ML Hebb's rule

Given an initialization of the network in a state $\boldsymbol{s}(t=1)$ one looks for the maximum probability of the recall of all following states, i.e. one wishes to maximize the following quantity, the conditional likelihood:

$$p(\boldsymbol{s}(T), \boldsymbol{s}(T-1), ..., \boldsymbol{s}(2)|\boldsymbol{s}(1))$$

Since the time evolution is Markovian, one has:

$$p(\boldsymbol{s}(T), \boldsymbol{s}(T-1), ..., \boldsymbol{s}(2)|\boldsymbol{s}(1)) = \prod_{t=1}^{T-1} p(\boldsymbol{s}(t+1)|\boldsymbol{s}(t))$$

The conditional log-likelihood is :

$$L(\boldsymbol{w}, b) \equiv \log \prod_{t=1}^{T-1} p(\boldsymbol{s}(t+1)|\boldsymbol{s}(t)) = \sum_{t=1}^{T-1}\sum_{i=1}^{S} \log f(s_i(t+1)\phi_i(t))$$

The conditional log-likelihood is a convex function with a single global maximum, since the Hessian:

$$\frac{d^2L}{dw_{ij}dw_{kl}} = -\beta^2 \sum_{t=1}^{T-1} s_i(t+1)s_j(t)f(s_i(t+1)\phi_i(t)s_k(t+1)s_l(t)\delta_{ik}$$

is negative semidefinite. The simple gradient ascent can be used for learning and has to be solved numerically:

$$w'_{ij} = w_{ij} + \eta\frac{dL}{d_{ij}}, \qquad b'_i = b_i + \eta\frac{dL}{db_i}$$

$$\frac{dL}{dw_{ij}} = \beta \sum_{t=1}^{T-1} \gamma_i(t) s_i(t+1) s_j(t), \qquad \frac{dL}{db_i} = \beta \sum_{t=1}^{T-1} \gamma_i(t) s_i(t+1)$$

with

$$\gamma_i(t) \equiv 1 - f(s_i(t+1))\phi_i(t)$$

and η best chosen small for convergence. One can notice that with $\gamma_i(t) = 1$ this is again the pure Hebb's rule. For large potentials ϕ, i.e. in an advanced state of learning, the value of $\gamma_i(t)$ is typically close to 1 or 0, giving the largest update in case of disagreement:

$$sgn(\phi_i(t)) \neq sgn(s_i(t+1))$$

and no update in case of agreement:

$$sgn(\phi_i(t)) = sgn(s_i(t+1))$$

The storage capacity is S linearly independent patterns with best performance among mentioned methods. One can also use the Newton method or conjugate gradients. Given N independent sequences that has to be learned instead of one, the likelihood gradient is just the sum of the individual gradients for each sequence.

11.1.4 Stochastic ML Hebb's rule

The ML Hebb's learning can be also done as online training on the fly with incoming patterns. For this one writes the update w.r.t. w in the form:

$$\frac{dL}{dw_{ij}} = \sum_{t=1}^{T-1} \frac{1}{2}(s_i(t+1) - < s_i(t+1) >_{p(s_i(t+1)|\phi_i(t))}) s_j(t)$$

and samples a $\hat{s}_i(t+1)$ in state 1 with probability $f(\phi_i(t))$ and -1 otherwise:

$$\Delta w_{ij}(t) = \eta(s_i(t+1) - \hat{s}_i(t+1)) s_j(t)$$

This sampling approximates the ML Hebb's rule learning.

Chapter 12

Tractable Deterministic Continuous Latent Variable Models

In Essence: *The framework is extended by a powerful setting of continuous hidden variables which follow non-linear dynamics with posterior inference being still tractable.*

A dynamic belief network with hidden variables takes the form:

$$p(v(1:T), h(1:T)) = p(v(1))p(h(1)|v(1)) \prod_{t=1}^{T-1} p(v(t{+}1)|v(t), h(t))p(h(t{+}1)|v(t), v(t), v(t{+}1)h(t))$$

One assumes continuous $h(t)$. To have a tractable posterior model, one assumes that $h(T)$ is *deterministic*. The hidden variables follow a non-linear Markovian transition described by:

$$p(h(t+1)|v(t+1), v(t), h(t)) = \delta(h(t+1)) - g(v(t+1), v(t), h(t), \theta_h))$$

where θ_h is a parameter. The log likelihood of a single training sequence $v(1:T)$ is:

$$L = \log p(v(1)|\theta_v) + \sum_{t=1}^{T-1} \log p(v(t+1)|v(t), h(t), \theta_v)$$

where:

$$h(t+1) = g(v(t+1), v(t), h(t), \theta_h)$$

is recursive. The gradients are:

$$\frac{dL}{d\theta_v} = \frac{\partial}{\partial\theta_v} \log p(v(1)|\theta_v) + \sum_{t=1}^{T-1} \frac{\partial}{\partial\theta_v} \log p(v(t+1)|v(t), h(t), \theta_v)$$

$$\frac{dL}{d\theta_h} = \sum_{t=1}^{T-1} \frac{\partial}{\partial h(t)} \log p(v(t+1)|v(t), h(t), \theta_v) \frac{dh(t)}{d\theta_h}$$

$$\frac{dh(t)}{d\theta_h} = \frac{\partial g(t)}{\partial\theta_h} + \frac{\partial g(t)}{\partial h(t-1)} \frac{dh(t-1)}{d\theta_h}, \qquad g(t) \equiv g(v(t), v(t-1), h(t-1), \theta_h)$$

12.1 Augmented Hopfield Network

Consider as an example continuous vector hidden variables $\mathbf{h}(t)$ and discrete binary visible variables $\mathbf{s}(t) = \mathbf{v}(t),\ v_i(t)[-1, 1]$ and a simple linear deterministic dynamics:

$$\mathbf{h}(t+1) = 2\sigma(\mathbf{A}\mathbf{h}(t) + \mathbf{B}\mathbf{v}(t)) - \mathbf{1}$$

$$p(\mathbf{v}(t+1)|\mathbf{v}(t),\mathbf{h}(t)) = \prod_{i=1}^{V} f(v_i(t+1)\alpha_i(t)), \qquad \alpha(t) \equiv \mathbf{Ch}(t) + \mathbf{Dv}(t)$$

The parameters of the model are $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$. The general delivative of the log likelihood in this setting is:

$$\frac{d}{dx}L = \sum_i \beta_i(t)\frac{d}{dx}\alpha_i(t), \qquad \beta_i(t) \equiv (1-(v_i(t+1)\alpha_i(t)))v_i(t+1)$$

with this the deivatives w.r.t. $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ can be calculated and formulated recursively, given a start value $\mathbf{h}(1)$. This extension by hidden units is capable of learning attractor networks with ambiguous transitions such as a, b, a, c.

Part IV

Linear Models

Chapter 13

Regression

In Essence:
For linear functions $f(x)$ with additive Gaussian noise one has the following regression options:
1. for both x and y noisy take the **Total Least Squares**
2. in case only y is noisy take:
2.i. for an under specified system N « d take **Constrained Least Squares**
2.ii. for an over specified system take:
2.ii - in case the noise is the same for all y take the **Ordinary Least Squares**
2.ii - in case the noise is not the same take the **Weighted Least Squares**

For non-Gaussian linear noise take:
1.Poisson (counting) -> transform into Gauss -> use special regression
2. in case there are outliers (large measurement errors):
2.i. use robust loss functions (suppress)
2.ii. remove outliers

For non-linear noise take:
1.in case the noise is additive-Gaussian take **Non-Linear Least Squares**
2. otherwise take **Regression Forest**
3. **Particle Filtering** is another option

13.1 Traditional approach - OLS

The OLS regression model is:

$$y = h_\theta(x) = \theta^T x + \sigma$$

Errors are penalized with the Least Squares (minimize historical sum) cost function:

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^i) - y^i)^2$$

With the representation:

$$X = [(x^1)^T ...(x^m)^T]^T, \ y = [y^1...y^m]^T$$

the matrix differentiation of the cost function yields the normal equations:

$$X^T X\theta = X^T y$$

with the solution for the latent parameters θ:

$$\theta = (X^T X)^{-1} X^T y$$

The centroid of the data (μ_x, μ_y) lies always on the regression line. To get a numerically more stable result (smaller numbers) with also less parameters, centralize the data first $x' = x - \mu_x$, $y' = y - \mu_y$. One can also standardize the data to have scale invariance:

$$x''_{ij} = \frac{x'_{ij}}{std_j(x_{ij})}$$

with i-th element of the j-th feature and the standard deviation. With this one achieves $std(x'') = 1$, whereas $std(x) = x$.

The same results can be achieved with:

13.2 Numerical approach

Minimize $J(\theta)$ using the gradient descent:

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) = \theta_j + \alpha(y^i - h_\theta(x^i))x_j^i$$

where α is the learning rate, which is the Widrow-Hoff or Least Mean Squares (LMS). With the Batch Gradient Descent (all training examples in one shot) one has:

$$\theta_j \leftarrow \theta_j + \alpha \sum_{i=1}^{m} (y^i - h_\theta(x^i))x_j^i$$

With the Stochastic Gradient Descent one has:

$$\theta_j \leftarrow \theta_j + \alpha(y^i - h_\theta(x^i))x_j^i \ \forall \ i \in \{1, .., m\}$$

13.3 Probabilistic approach

For σ Normally and i.i.d. one has for the likelihood function:

$$L(\theta) = p(y|X;\theta) = \prod_{i=1}^{m} p(y^i|x^i;\theta) = \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\theta}} \exp(-\frac{(y^i - \theta^T x^i)^2}{2\theta^2})$$

The principle of Maximum Likelihood is to maximize the likelihood or the log-likelihood. With this the optimal parameters are:

$$\hat{\theta}_{ML} = arg \max_\theta l(\theta) = arg \max_\theta \log L(\theta) = arg \min_\theta \frac{1}{2} ||X\theta - y||^2$$

13.4 Regularization

13.4.1 Lasso

The absolute values of the coefficients θ are penalized via the L_1 norm:

$$J_L(\theta) = \sum_{i=1}^{n} ((h_\theta(x)^i - y^i)^2 + \alpha \sum_{i=1}^{n} |\theta^i|$$

thus taking only most important predictors into account. By increasing α one selects increasingly the most important features. This result can also be obtained through probabilistic reasoning, MAP (maximum a posteriori), with a Laplacian prior $f(\theta) \propto e^{-\alpha|\theta|}$. The posterior is then:

$$f(\theta|y) \propto f(\theta) \cdot f(y|\theta) = e^{-\alpha|\theta|}e^{-\frac{1}{2}||y-X\theta||^2}$$

then the MAP estimate for θ is:

$$\hat{\theta}_{MAP} = arg\max f(\theta|y) = arg\min ||y - X\theta||^2 + 2\alpha|\theta|$$

13.4.2 Ridge

The absolute values of the coefficients θ are penalized via the L_2 norm:

$$J_L(\theta) = \sum_{i=1}^{n}((h_\theta(x)^i - y^i)^2 + \alpha\sum_{i=1}^{n}(\theta^i)^2$$

This result can also be obtained through probabilistic reasoning, MAP (maximum a posteriori), with a Gaussian prior $f(\theta) \propto e^{-\frac{1}{2}||\theta||^2}$.

13.4.3 Elastic Net

The absolute values of the coefficients θ are penalized via the L_1 and the L_2 norm:

$$J_L(\theta) = \sum_{i=1}^{n}((h_\theta(x)^i - y^i)^2 + \alpha_1\sum_{i=1}^{n}|\theta^i| + \alpha_2\sum_{i=1}^{n}(\theta^i)^2$$

Chapter 14

Unsupervised Linear Dimension Reduction

In Essence: *A high dimensional data point x is projected down to a lower dimensional vector y by:*

$$y = \mathbf{F}x + \text{const.}$$

with a non-square dim(y*)* $\times$ *dim(*x*) matrix* $\boldsymbol{F}$*.*

14.1 PCA

Also Karhunen-Loeve decomposition/proper orthogonal decomposition/latent semantic analysis

If data lies close to a linear subspace, one can describe it as:

$$\boxed{x^n \approx \mathbf{c} + \sum_{j=1}^{M} y_j^n \mathbf{b}^j = c + \mathbf{BY}}$$

PCA corresponds to orthogonal Least Squares. For centered data one has c=0. What is the basis **B**? By minimizing the sum of squared differences between each vector x and its reconstruction w.r.t. **B**, one gets the optimality condition:

$$\mathbf{SB} = \mathbf{BL}$$

with sample covariance matrix of the data S. Requiring, that the matrix of Lagrange multipliers **L** (constraint $\mathbf{B}^T\mathbf{B} = \mathbf{I}$) is diagonal, one gets that the columns of **B** are the corresponding eigenvectors of **S**. Since least squares depend on the product **BY**, to break the scale/rotation invariance, one intuitively defines the vectors B in the directions, where the data has maximum spread (variance). To minimize the error, one defines the basis using the eigenvectors of **S** with largest corresponding eigenvalues. So for PCA one needs eigenvectors and eigenvalues of **S**.

14.1.1 PCA with high dimensional data, N<D

Eigen-decomposition

For high dimensional data, to determine eigenvectors and eigenvalues of **S** effectively, one can use:

$$\mathbf{XX}^T\mathbf{E} = \mathbf{E\Lambda} \Rightarrow \mathbf{X}^T\mathbf{X}\tilde{\mathbf{E}} = \tilde{\mathbf{E}}\mathbf{\Lambda}$$

with $\tilde{\mathbf{E}} = \mathbf{X}^T\mathbf{E}$ and the $N \times N$ matrix $\mathbf{X}^T\mathbf{X}$ (Kernel). One gets that the eigenvalues $\mathbf{\Lambda}$ are the same for $\mathbf{X}\mathbf{X}^T$ and $\mathbf{X}^T\mathbf{X}$ and the original eigenvectors can be obtained with:

$$\mathbf{E} = \mathbf{X}\tilde{\mathbf{E}}\mathbf{\Lambda}^{-1}$$

SVD

For SVD (singular value decomposition) one has:

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^T$$

for ordered singular values, the largest one is the upper left diagonal element of $\mathbf{D}$. Then we have:

$$\mathbf{X}\mathbf{X}^T = \mathbf{U}\tilde{\mathbf{D}}\mathbf{U}^T$$

with the $D \times D$ matrix $\tilde{\mathbf{D}} = \mathbf{D}\mathbf{D}^T$ which are the squared singular values. Then the PCA eigenvectors are given by $\mathbf{U}$ and the corresponding eigenvalues by $\tilde{\mathbf{D}}$.

14.1.2 Missing data

One constrains the squared distance minimization to only present data. One then preforms an iterative procedure, with first choosing a random $\mathbf{B}_0$.

1. Optimize Y for fixed B:

For a fixed guess B of dimensions $D \times M$, the optimization is a simple linear equation.

$$E(B, Y) = \sum^{N}\sum^{D}$$

2. Optimize B for fixed Y:

For a fixed obtained Y, the optimization is a simple linear equation. Repeat the steps iteratively until convergence. Online updating possible. With this algorithm, the basis $\mathbf{B}$ has to be additionally rotated along the direction of larest variance. Forming the SVD of $\mathbf{B}$, $\mathbf{B} = \mathbf{U}\mathbf{D}\mathbf{V}^T$, the principal directions $\tilde{\mathbf{B}}$ are given with:

$$\tilde{\mathbf{B}} = \mathbf{B}\mathbf{V}\mathbf{D}^{-1}$$

To make the above well defined, one may append $\mathbf{D}$ with the columns of the identity. Given B and Y one can form a reconstruction on all the entries of X, by using:

$$\tilde{\mathbf{X}} = \mathbf{B}\mathbf{Y}$$

Which is giving therefore a prediction for missing values!

14.2 Dual Representation and Kernels

For a linear parametric model

$$y = \mathbf{w}^T\mathbf{x}$$

only the dimensions spanned by the vectors $\mathbf{x}$ contribute to the model. Without loss of generality one can write:

$$\mathbf{w}^* = \sum_{p=1}^{N} a_p \mathbf{x}^p$$

If one now tries to model data points $\mathbf{x}'$ that are orthogonal to the previous data, i.e. $(\mathbf{x}')^T\mathbf{x}^n = 0$, $\forall n = 1, .., N$ this will have no effect on the optimal $\mathbf{w}^*$. The parameters a_p are called **dual parameters**. Therefore one has:

$$y^n = \sum_{p=1}^{N} a_p (\mathbf{x}^p)^T \mathbf{x}^n$$

Since the SKP involved is a scalar quantity, one can insert a transformation of the linear space via a arbitrary dimensional vector function $\phi(\mathbf{x})$. The result will be in the space spanned by the transformed vectors:

$$y = \mathbf{w}^T \phi(\mathbf{x}^n) = \sum_{p=1}^{N} a_p \phi(\mathbf{x}^p)^T \phi(\mathbf{x}^n) \equiv K(\mathbf{x}^p, \mathbf{x}^n) \equiv [\mathbf{K}]_{pn}.$$

with the **kernel** $K(\mathbf{x}^p, \mathbf{x}^n)$ which is a covariance function and a positive semidefinite $N \times N$ **Gram matrix K**. The prediction can be therefore be represented via the kernel which dimensions are number of data points, without explicitely computing the vector function of arbitrary dimension itself. This technique can be used to avoid the curse of dimensionality and for introducing non-linearity for many of the linear models.

14.3 Matrix decomposition methods: bilinear decomposition

SVD and PCA are matrix decomposition methods. For $M > D$ the basis is over-complete, there being more basis vectors than dimensions. In such cases additional constraints are placed on either the basis or components. For example, one might require that only a small number of the large number of available basis vectors is used to form the representation for any given x. Other popular constraints are that the basis vectors themselves should be sparse or positivity, energy, etc. constraints are used.

14.4 Canonical correlation analysis

Canonical correlation analysis attempts to find a low dimensional representation that jointly models the two related data spaces. CCA is a special case of the probabilistic factor analysis model.

Chapter 15

Supervised Linear Dimension Reduction

In Essence: *SLDR methods utilize knowledge on class memberships for improved data description. SLDR can produce lower dimensional representations more suitable for subsequent classification than an unsupervised method such as PCA.*

15.1 Fisher's Linear Discriminant Analysis

In Essence: *Model the data with Gaussians and find a projection where the means are maximally separate from each other.*

For simplicity consider two classes and a projection down to 1 dimension. One can model the data from 2 classes with two Gaussians:

$$p(\mathbf{x}_1) = \mathcal{N}(\mathbf{x}_1|\mathbf{m}_1, \mathbf{S}_1),\ p(\mathbf{x}_2) = \mathcal{N}(\mathbf{x}_2|\mathbf{m}_2, \mathbf{S}_2)$$

with $\mathbf{m}_{1,2}$ are the sample means and $\mathbf{S}_{1,2}$ the sample covariances. The projections $\mathbf{y}_{1,2}$ are given by the vector $\mathbf{w}$:

$$y_1^n = \mathbf{w}^T x_1^n,\ y_2^n = \mathbf{w}^T x_2^n$$

One looks for a projection that maximizes the following quantity:

$$\frac{(\mu_1 - \mu_2)^2}{\pi_1\sigma_1^2 + \pi_2\sigma_2^2}$$

where π_i is the fraction in class i. In terms of $\mathbf{w}$ the objective is:

$$F(\mathbf{w}) = \frac{\mathbf{w}^T(\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T\mathbf{w}}{\mathbf{w}^T(\pi_1\mathbf{S}_1 + \pi_2\mathbf{S}_2)\mathbf{w}} = \frac{\mathbf{w}^T\mathbf{A}\mathbf{w}}{\mathbf{w}^T\mathbf{B}\mathbf{w}}$$

The optimal $\mathbf{w}$ is found by the differentiating w.r.t $\mathbf{w}$ as:

$$\boxed{\mathbf{w} = k\mathbf{B}^{-1}(\mathbf{m}_1 - \mathbf{m}_2)}$$

The common scaling k is:

$$k = \frac{1}{\sqrt{(\mathbf{m}_1 - \mathbf{m}_2)^T\mathbf{B}^{-2}(\mathbf{m}_1 - \mathbf{m}_2)}}$$

In practice, however, $\mathbf{B}$ may not be invertible, and the above procedure requires modification (few or static data points).

15.2 Canonical Variates

Canonical Variates is a generalization of Fishers LDA to C classes and D dimensions. Then **W** is a $D \times L$ matrix and one has:

$$\boxed{\mathbf{y} = \mathbf{W}^T\mathbf{x}}$$

Assuming again that the classes and the projections are Gaussians, one has:

$$p(x) = \mathcal{N}(x|\mathbf{m}_c, \mathbf{S}_c)$$

and

$$p(y) = \mathcal{N}(y|\mathbf{W}^T\mathbf{m}_c, \mathbf{W}^T\mathbf{S}_c\mathbf{W})$$

With **m** is the mean of the whole dataset and N_c the number of datapoints in class c, one defines:

Between Class Scatter:

$$\mathbf{A} = \sum_{c=1}^{C} N_c(\mathbf{m}_c - \mathbf{m})(\mathbf{m}_c - \mathbf{m})^T$$

Within Class Scatter:

$$\mathbf{B} = \sum_{c=1}^{C} N_c\mathbf{S}_c$$

Assuming **B** is invertible, one defines the **Cholesky factor $\tilde{\mathbf{B}}$**, with:

$$\boxed{\tilde{\mathbf{B}}^T\tilde{\mathbf{B}} = \mathbf{B}}$$

Compute the L principal eigenvectors $[e_1, ..., e_L]$ of $\tilde{\mathbf{B}}^{-T}\mathbf{A}\tilde{\mathbf{B}}^{-1}$. Return $\mathbf{W} = [e_1, ..., e_L]$ as the projection matrix. In case B is not invertible, one expresses **all** training data **X** in a thin-SVD basis ($\Rightarrow$ orthonormal, non-square) **Q**. One then requires that **W** is also expressed in this basis, for some matrix **W**':

$$\mathbf{W} = \mathbf{Q}\mathbf{W}'$$

This leads to the modification of the procedure:

$$\mathbf{A}' = \mathbf{Q}^T\mathbf{A}\mathbf{Q}$$

$$\mathbf{B}' = \mathbf{Q}^T\mathbf{B}\mathbf{Q}$$

Now **B**' is guaranteed invertible.

Chapter 16

Latent Linear Models

In Essence: *Latent Linear Models find a hidden structure in the data in form of internal dimensions along which the data is spread with the Gaussian variance.*

16.1 Factor Analysis

Consider a set $\mathcal{V}$ of visible variables $\mathbf{v}_1, \mathbf{v}_N$ with $dim(\mathbf{v}) = D$. The factor analysis models data via a lower dimensional coordinate system with an additive Gaussian noise:

$$\boxed{\mathbf{v} = \mathbf{F}\mathbf{h} + \mathbf{c} + \boldsymbol{\epsilon},\ \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0}, \boldsymbol{\Psi})}$$

with the $D \times H$ *factor loading* matrix $\mathbf{F}$ with components λ_{ij} and hidden coordinates, or factors, $\mathbf{h}$. Whereas PCA has the covariance $\boldsymbol{\Psi} = \sigma^2\mathbf{I}$, the FA defines $\boldsymbol{\Psi} = diag(\psi_1, ..., \psi_D)$. This is a richer description for the off-subspace noise $\boldsymbol{\Psi}$ and allows the covariance to differ for each basis component. The marginal probability distribution is a Gaussian:

$$p(\mathbf{v}|\mathbf{h}) = \mathcal{N}(\mathbf{v}|\boldsymbol{\mu} = \mathbf{F}\mathbf{h} + \mathbf{c}, \boldsymbol{\Sigma} = \boldsymbol{\Psi})$$

One models the hidden coordinates also with a Gaussian:

$$p(\mathbf{h}) = \mathcal{N}(\mathbf{h}|\mathbf{0}, \mathbf{I})$$

with $\mathbf{0}$ being the most probable state. Then the observations are also normally distributed:

$$p(\mathbf{v}) = \int p(\mathbf{v}|\mathbf{h})p(\mathbf{h})d\mathbf{h} = \mathcal{N}(\mathbf{v}|\mathbf{c}, \mathbf{F}\mathbf{F}^T + \boldsymbol{\Psi})$$

FA has $D(H+1)$ free parameters, compared to $D(D+1)/2$ parameters of unconstrained covariance.

Since:

$$\mathbf{FR}(\mathbf{FR})^T + \boldsymbol{\Psi} = \mathbf{F}\mathbf{F}^T\boldsymbol{\Psi}$$

the solution for the loadings $\mathbf{F}$ is invariant under the rotation $\mathbf{FR}$, $\mathbf{RR}^T = \boldsymbol{I}$. One can look at the loadings as some kind regression coefficients w.r.t. the factors. Typically one looks for $\mathbf{F}$ with the minimal number of large entries (*Varimax algorithm*).

For i.i.d. data the optimal bias $\mathbf{c}$ is found by differentiating the log-likelihood

$$\log p(\mathcal{V}|\mathbf{F}, \boldsymbol{\Psi}, \boldsymbol{c}) = \sum_{n=1}^{N} \log p(\mathbf{v}^n|\mathbf{F}, \boldsymbol{\Psi}, \boldsymbol{c})$$

w.r.t. **c** and setting to zero. The optimal bias is:

$$\mathbf{c}^* = \frac{1}{N}\sum_{n=1}^{N}\mathbf{v}^n = \bar{\mathbf{v}}$$

which is the sample mean. With the sample mean bias the log likelihood is:

$$\boxed{\log p(\mathcal{V}|\mathbf{F}, \mathbf{\Psi}) = -\frac{N}{2}(tr(\mathbf{\Sigma}_D^{-1}\mathbf{S}) + \log\det 2\pi\mathbf{\Sigma}_D))}$$

with

$$\mathbf{S} \equiv \frac{1}{N}\sum_{n=1}^{N}(\mathbf{v}-\bar{\mathbf{v}})(\mathbf{v}-\bar{\mathbf{v}})^T$$

which is the sample covariance matrix and

$$\mathbf{\Sigma}_D := \mathbf{F}\mathbf{F}^T + \mathbf{\Psi}$$

The Factor loadings can then be learned via Maximum Likelihood with an eigenvalue-approach, SVD or an EM algorithm, depending on whether $\mathbf{\Psi} = diag(\psi_1, ..., \psi_D)$ is known or has to be learned too.

Probabilistic PCA is a variant of Factor Analysis with $\mathbf{\Psi} = \sigma^2\mathbf{I}_D$. **Canonical Correlation Analysis** is also a variant of FA.

Part V

Dynamical Models

Chapter 17

Discrete State Markov Models

17.1 First Order Markov Chain

In Essence: *In a first order markov chain the future state of the system depends only on the present state and is idependent of the past.*

Datapoints: T realizations $\{s_1, .., s_T\}$ of random variables $\{v_1, .., v_T\}$. Each random variable v is characterized by its probability distribution p(v), which can be discrete or continuous.

Timeseries: naturally ordered data points of T time steps. A time series joint can be decomposed with the natural *cascade decomposition* variant of the chain rule:

$$p(v_{1:N}) = \prod_{t=1}^{T} p(v_t|v_{1:t-1}) = p(v_1)p(v_2|v_1)...p(v_{T-1}|v_{1:T-2})$$

Markov property: the future state of the stochastic process depends only on the present state and is idependent of the past.

Markov process: a stochastic process with Markov property.

Markov chain: a Markov process with discrete realizations/states.

Markov chain order: a conditional independence assumption of *L-th order* on the joint:

$$\boxed{p(v_t|v_{1:t-1}) = p(v_t|v_{t-L:t-1})}$$

A **first order Markov chain** corresponds to singly connected nodes in the DGM. We have:

$$p(v_t|v_{1:t-1}) = p(v_t|v_{t-1})$$

and

$$p(v_t = i) = \sum_j p(x_t = i|x_{t-1} = j)p(x_{t-1} = j)$$

Stationary Markov chain: transition probabilities $p(v_t, v_{t-1}) \neq f(t)$ are time-independent

Transition matrix: Generally the probability table for $p(v_1, .., v_T) \equiv p(v_{1:T})$ has exponentially many entries. The probability table for the discrete stationary first order Markov chain can be encoded with the transition matrix $\mathbf{M}$ with $R \times R$ entries $M_{ij} = p(v_t = i|v_{t-1} = j)$. The probability table is then build up with:

$$p_t(i) = p(v_t = i) = \sum_j p(v_t = i|v_{t-1} = j)p(v_{t-1} = j) = \sum_j \mathbf{M}_{ij}p(v_{t-1} = j)$$

$$\mathbf{p}_t = \mathbf{M}\mathbf{p}_{t-1},\ dim(\mathbf{p}_t) = R$$
$$\mathbf{p}_t = \mathbf{M}^{t-1}\mathbf{p}_1$$

The non-zero entries of $\mathbf{M}$ can be visualized with a **state-transition diagram**.

Equilibrium and stationary distribution:

If $\mathbf{p}_\infty \perp \mathbf{p}_1$ in the sence of conditional independence then $\mathbf{p}_\infty$ is the **equilibrium distribution** of the chain.

The **stationary distribution** of the chain is defined as: $\mathbf{p}_\infty = \mathbf{M}\mathbf{p}_\infty$.

$\rightarrow$ *Page Rank algorithm*

Fitting a Markov chain: Given a sequence $v_{1:T}$, fitting a stationary first order Markov chain by maximum likelihood corresponds to setting the transitions by counting the number of observed (first-order) transitions in the sequence:

$$\boxed{p(v_t = i|v_{t-1} = j) \propto \sum_{t=2}^{T} \mathbb{I}[v_t = i, v_{t-1} = j]}$$

Bayesian Fitting: a simple assumption is a factorized prior on the transition:

$$p(\theta) = \prod_j p(\theta.|j)$$

Conveniently one chooses a Dirichlet distribution (conjugate to the categorical transition) with hyperparameters $\boldsymbol{u}_j$ for each conditional transition:

$$p(\theta.|j) = D(\theta_{.|j}|\boldsymbol{u}_j)$$

With that the posterior is:

$$\boxed{p(\theta|v_{1:T}) \propto \prod_j D(\theta_{.|j}|\hat{\boldsymbol{u}}_j),\ \hat{\boldsymbol{u}}_j = \sum_{t=2}^{T} \mathbb{I}[v_{t-1} = i, v_t = j]}$$

with $\hat{\boldsymbol{u}}_j$ being the number of $j \rightarrow i$ transitions in the dataset.

17.2 Mixture of Markov Models

The general idea is the clustering of data to a mixture of Markov chains with the cluster index being the hidden variable h. Assuming for simplicity that the chains are of the same length T and given a set of N sequences $\nu = \{v^n_{1:T}, k = 1, ..., K\}$, assuming i.i.d. data, so that:

$$p(\nu) = \prod_k p(v^n_{1:T}),$$

we have for each component chain:

$$p(v_{1:T}) = \sum_{h=1}^{H} p(h)p(v_{1:T}|h)$$

For the first order chains we have:

$$p(v_{1:T}) = \sum_{h=1}^{H} p(h) \prod_{t=1}^{T} p(v_t|v_{t-1}, h)$$

The optimal parameters p(h), $p(v_t|v_{t-1}, h)$ are found with the **EM algorithm**, the cluster assignment is done subsequently via: $p(h|v^n_{1:T})$.

17.3 Hidden Markov Models

The Hidden Markov Model (HMM) defines a Markov chain on hidden (or 'latent') variables $h_{1:T}$. The observed (or 'visible') variables are dependent on the hidden variables through an emission $p(v_t|h_t)$. The joint is:

$$\boxed{p(v_{1:T}, h_{1:T}) = p(v_1|h_1)p(h_1)\prod_{t=1}^{T} p(v_t|h_t)p(h_t|h_{t-1})}$$

Stationary HMM: The emissions and transitions are stationary in time. Then the transition distribution $p(h_{t+1}|h_t)$ is defined via the transition matrix:

$$A_{i'i} = p(h_{t+1} = i'|h_t = i)$$

and the initial distribution:

$$a_i = p(h_1 = i)$$

For discrete observed states v_i the $V \times H$ emission matrix is defined as:

$$B_{ij} = p(v_t = i|h_t = j)$$

For continuous outputs, h_t selects one of H possible output distributions $p(v_t|h_t), h_t \in \{1, ..., H\}$.

Classic inference problems are:

- **Filtering:** infer the present $p(h_t|v_{1:t})$
- **Prediction:** infer the future $p(h_t|v_{1:s})$, $s < t$
- **Smoothing:** infer the past $p(h_t|v_{1:T})$, $t < T$
- **Likelihood:** infer the Likelihood $p(v_{1:t})$
- **Viterbi alignment:** most likely path $\underset{h_{1:T}}{\text{argmax}} p(h_{1:T}|v_{1:T})$

17.3.1 Filtering:

With the conditional independence assumption we have:

$$p(h_t|v_{1:t}) \equiv \alpha(h_t) = p(v_t|h_t) \sum_{h_{t-1}} p(h_t|h_{t-1})p(h_{t-1}, v_{1:t-1}),$$

which defines the α-**recursion**:

$$\boxed{\alpha(h_t) = p(v_t|h_t) \cdot \sum_{h_{t-1}} p(h_t|h_{t-1})\alpha(h_{t-1}) = \textbf{corrector} \cdot \textbf{predictor}, \ \ t > 1}$$

$$\alpha(h_1) \equiv p(v_1, h_1) = p(v_1|h_1)p(h_1)$$

To avoid numerical problems it is therefore advisable to work with $\log \alpha(h_t)$. The desired ***filtered posterior*** $p(h_t|v_{1:t})$ is then computed from the normalization.

17.3.2 Smoothing:

a) Parallel Smoothing:

This most common method is equivalent to message passing on factor graphs.

$$\boxed{p(h_t|v_{1:T}) = p(h_t|v_{1:t}) \cdot p(v_{t+1:T}|h_t) = \textbf{past} \cdot \textbf{future} = \alpha(h_t)\beta(h_t)}$$

The $\alpha(h_t)$ is obtained via the 'forward' α-recursion, the 'backward' β-recursion for the second term can be run in parallel and is the following:

$$p(v_{t:T}|h_{t_1}) = \sum_{h_t} p(v_t|h_t)p(v_{t+1:T}|h_t)p(h_t|h_{t-1}),$$

which corresponds to:

$$\beta(h_{t-1}) = \sum_{h_t} p(v_t|h_t)p(h_t|h_{t-1})\beta(h_t),\ 2 \leq t \leq T$$

$$\beta(h_T) = 1$$

The smoothed posterior is:

$$p(h_t|v_{1:T}) \equiv \gamma(h_t) = \frac{\alpha(h_t)\beta(h_t)}{\sum_{h_t} \alpha(h_t)\beta(h_t)}$$

Together the α- and β-recursions are called the ***Forward-Backward algorithm***.

b) Correction Smoothing:

$$p(h_t|v_{1:T}) = \sum_{h_{t+1}} p(h_t, h_{t+1}|v_{1:T}) = \sum_{h_{t+1}} p(h_t|h_{t+1}, v_{1:t})p(h_{t+1}|v_{1:T})$$

This corresponds to:

$$\boxed{\gamma(h_t) = \sum_{h_{t+1}} p(h_t|h_{t+1}, v_{1:t})\gamma(h_{t+1}),}$$

with:

$$\gamma(h_T) \propto \alpha(h_T)$$

One finds additionally:

$$p(h_t|h_{t+1}, v_{1:t}) = \frac{p(h_{t+1}|h_t)p(h_t|v_{1:t})}{\sum_{h_t} p(h_{t+1}|h_t)p(h_t|v_{1:t})}$$

This is a form of dynamics reversal, as if we are reversing the direction of the hidden to hidden arrow in the HMM. This sequential procedure is also termed the ***Rauch-Tung-Striebel*** or $\alpha-\gamma$-**smoother**. In general we have: $\gamma(h_t) \propto \alpha(h_t)\beta(h_t)$.

17.3.3 Computing the pairwise marginal $p(h_t, h_{t+1}|v_{1:T})$

For the pairwise marginal (for later learning) we have:

$$p(h_t, h_{t+1}|v_{1:T}) \propto \alpha(h_t)p(v_{t+1}|h_{t+1})p(h_{t+1}|h_t)\beta(h_{t+1})$$

17.3.4 Computing the Likelihood $p(v_{1:T})$

$$p(v_{1:T}) = \sum_{h_T} \alpha(h_T)$$

Alternatively:

$$p(v_{1:T}) = \prod_{t=1}^{T} p(v_t|v_{1:t-1}),$$

with:

$$p(v_t|v_{1:t-1}) = \sum_{h_t} p(v_t|h_t) \sum_{h_{t-1}} p(h_t|h_{t-1})p(h_{t-1}|v_{1:t-1})$$

Alternatively for any t:

$$p(v_{1:T}) = \sum_{h_t} \alpha(h_t)\beta(h_t)$$

17.3.5 Sampling from $p(h_{1:T}|v_{1:T})$

$p(h_{1:T}|v_{1:T})$ can be itself expressed as a MC:

$$p(h_{1:T}|v_{1:T}) = p(h_1|h_2, v_{1:T})...p(h_{T-1}|h_T, v_{1:T})p(h_T|v_{1:T}),$$

with:

$$p(h_{t-1}|h_t, v_{1:T}) \propto \alpha(h_{t-1})p(h_t|h_{t-1})$$

One starts sampling from the end of the chain, this operation is called ***forward-filtering-backward-sampling***.

17.3.6 Viterbi most likely path $h_{1:T}$ of $p(h_{1:T}|v_{1:T})$:

The most likely path is the same as the most likely state of $p(h_{1:T}, v_{1:T})$ for fixed $v_{1:T}$. One has:

$$p(h_{1:T}, v_{1:T}) = \prod_t p(v_t|h_t)p(h_t|h_{t-1})$$

The most likely path can be found using the max-product version of the factor graph or max-absorption on the junction tree. Alternatively one uses a special case of the ***max-product algorithm***:

$$\max_{h_T} \prod_t p(v_t|h_t)p(h_t|h_{t-1}) = \prod_{t=1}^{T-1} p(v_t|h_t)p(h_t|h_{t-1}) \max_{h_T} p(v_t|h_t)p(h_t|h_{t-1}) = \prod_{t=1}^{T-1} p(v_t|h_t)p(h_t|h_{t-1})\mu(h_{T-1})$$

Which defines the recursion:

$$\boxed{\mu(h_{t-1}) = \max_{h_T} p(v_t|h_t)p(h_t|h_{t-1})\mu(h_t), \ 2 \leq t \leq T}$$

with:

$$\mu(h_T) = 1$$

With this one has:

$$h_1^* = \operatorname*{argmax}_{h_1} p(v_1|h_1)p(h_1)\mu(h_1)$$

$$h_t^* = \operatorname*{argmax}_{h_1} p(v_t|h_t)p(h_t|h_{t-1}^*)\mu(h_t)$$

This algorithm is called the ***Viterbi algorithm***.

17.3.7 Prediction $p(v_{t+1}|v_{1:t})$

The one-step ahead predictive distribution is given by:

$$p(v_{t+1}|v_{1:t}) = \sum_{h_t,h_{t+1}} p(v_{t+1}|h_{t+1})p(h_{t+1}|h_t)p(h_t|v_{1:t})$$

17.4 Learning HMMs

17.4.1 EM algorithm

Given a set of N sequences of data $\nu = \{\mathbf{v}^n\}_{n\in N}$, $\mathbf{v} = v_{1:T_n}$, we want to infer the transition matrix $\mathbf{A}$, the emission matrix $\mathbf{B}$ and initial vector $\mathbf{a}$, $a_i = p(h_1 = i)$, dim(a) = H, most likely to have have generated the data. H is the known number of hidden states, V is the number of observed states. Assuming i.i.d. data, learning can be done with the EM algorithm (*Baum-Welch algorithm*).

M-step

$$a_i^{new} = \frac{1}{N}\sum_{n=1}^{N} p^{old}(h_1 = i|\mathbf{v}^n)$$

$$A_{i',i}^{new} = \frac{\sum_{n=1}^{N}\sum_{t=1}^{T_n-1} p^{old}(h_t = i, h_{t+1} = i'|\mathbf{v}^n)}{\sum_{i'}\sum_{n=1}^{N}\sum_{t=1}^{T_n-1} p^{old}(h_t = i, h_{t+1} = i'|\mathbf{v}^n)}$$

$$B_{ji}^{new} \propto \sum_{n=1}^{N}\sum_{t=1}^{T_n} I[v_t^n = j]p^{old}(h_t = i|\mathbf{v}^n)$$

E-step

The quantities $p^{old}(h_1 = i|\mathbf{v}^n)$, $p^{old}(h_t = i, h_{t+1} = i'|\mathbf{v}^n)$ and $p^{old}(h_t = i|\mathbf{v}^n)$ are obtained by smoothed inference methods as discussed above.

Parameter initialization

The EM algorithm converges to a local maximum of the likelihood and, in general, there is no guarantee that the algorithm will find the global maximum. One of the techniques is A practical strategy is to initialise the emission $p(v|h)$ based on first fitting a simpler non-temporal mixture model $\sum_h p(v|h)p(h)$ to the data.

17.4.2 Mixture emission

It is straightforward to generalize to a continuous space of observations. The continuous vector observation $\mathbf{v}_t$ is a D-dimensional sample from a continuous distribution and requires a model $p(\mathbf{v}_t|h_t)$, which maps the discrete state h_t to the distribution over outputs. Conveniently, for filtering, smoothing and Viterbi, the normalization of the emission distribution is not required, not like that for learning. For a richer emission model in this context, one can use a ***mixture emission***:

$$\boxed{p(v_t|h_t) = \sum_{k_t}^{K} p(v_t|k_t, h_t)p(k_t|h_t)}$$

k_t are additional latent variables and the EM algorithm is straightforward (M-step):

$$p(k = \mathbf{k}|h = \mathbf{h}) \propto \sum_n \sum_{t=1}^{T} p^{old}(k_t = \mathbf{k}|h_t^n = \mathbf{h})p^{old}(h_t^n = \mathbf{h}|v_{1:T}^n)$$

17.4.3 HMM-GMM

A common continuous observation mixture emission model *component* is a Gaussian:

$$p(\mathbf{v}_t|k_t, h_t) = \mathcal{N}(\mathbf{v}_t|\boldsymbol{\mu}_{k_t,h_t}, \boldsymbol{\Sigma}_{k_t,h_t})$$

with $K \times H$ mean vectors and covariane matrices at each time step t. The EM update for the means and covariances is again straightforward.

Related topics: Discriminative training, Explicit Duration Model, Input-Output HMM Linear Chain Conditional Random Fields (CRFs), Dynamic Bayesian Network

Chapter 18

Continuous State Markov Models

In Essence: *Continuous State Markov Models though having discrete time evoution act upon a continuous state space. Therefore a discrete transition matrix P has to be modified to a continuous (Gaussian) transition kernel.*

18.1 Observed Linear Dynamical Systems

In Essence: *Continuous State Markov Models though having discrete time evoution act upon a continuous state space. Therefore a discrete transition matrix P has to be modified to a continuous (Gaussian) transition kernel.* The physical model of a **deterministic OLDS** describes the deterministic temporal evolution of a vector $\mathbf{v}_t$:

$$\boxed{\mathbf{v}_t = \mathbf{A}^{t-1}\mathbf{v}_1 = \mathbf{P}\mathbf{\Lambda}^{t-1}\mathbf{P}^{-1}\mathbf{v}_1}$$

with $dim(\mathbf{v}_t) = V$, $\Lambda = diag(\lambda_1, ..., \lambda_V)$ and $\mathbf{P}$ is the corresponding eigenvector matrix of $\mathbf{A}$. If any $\lambda_i > 1$, then v_∞ explodes. If $\lambda_i < 1$, then λ_i^{t-1} tends to zero. So for stable systems we need for all $\lambda_i \leq 1$ and for long term solutions only $|\lambda_i = 1|$ will contribute. Complex eigenvalues correspond to rotational behavior. A **stochastic OLDS** considers additive noise on $\mathbf{v}$:

$$\mathbf{v}_t = \mathbf{A}_t\mathbf{v}_{t-1} + \boldsymbol{\eta}_t$$

where $\boldsymbol{\eta}_t$ is sampled from $\mathcal{N}(\boldsymbol{\eta}_t|\boldsymbol{\mu}_t, \mathbf{\Sigma}_t)$. This is equivalent to a first order Markov model with transition:

$$p(\mathbf{v}_t|\mathbf{v}_{t-1}) = \mathcal{N}(\mathbf{v}_t|\mathbf{A}_t\mathbf{v}_{t-1} + \boldsymbol{\mu}_t, \mathbf{\Sigma}_t)$$

18.1.1 e.g. 1-dim stationary distribution with noise

Consider the one-dimensional linear system:

$$v_t = av_{t-1} + \eta_t,\ \eta_t \sim \mathcal{N}(\eta_t|0, \sigma_v^2)$$

What is the distribution for v_t? The answer is:

$$v_t \sim \mathcal{N}(v_t|a\eta_{t-1}, a^2\sigma_{t-1}^2 + \sigma_v^2)$$

If $a \geq 1$, then the variance an the mean increase indefinitely with time. If $a < 1$, assiming a finite variance σ_∞^2 for the infinite time, the stationary distribution satisfies:

$$\sigma_\infty^2 = \frac{\sigma_v^2}{1 - a^2}$$

$$\mu_\infty = a^\infty \mu_1 \longrightarrow 0$$

The additive noise counteracts the decrease of the magnitude of v_t. More generally for convergence one requires $\lambda < 1$ for all eigenvalues of $\mathbf{A}$.

18.2 Auto-Regressive Models

A scalar time-invariant auto-regressive model is defined by:

$$\boxed{v_t = \sum_{l=1}^{L} a_l v_{t-1} + \eta_t, \ \eta_t \sim \mathcal{N}(\eta_t|\mu, \sigma^2)}$$

$\mathbf{a} = (a_1, ..., a_L)^T$ are the AR coefficients and σ^2 is the *innovation noise*. The future is predicted based on the linear combination of the previous observations. This is equivalent to the L^{th} order Markov model:

$$p(v_{1:T}) = \prod_{t=1}^{T} p(v_t|v_{t-1}, .., v_{t-L}), \ v_{i\leq 0} = \emptyset$$

$$p(v_t|v_{t-1}, ..., v_{t-L}) = \mathcal{N}\Big(v_t|\sum_{l=1}^{L} a_l v_{t-l}, \sigma^2\Big) = \mathcal{N}\Big(v_t|\mathbf{a}^T\hat{\mathbf{v}}_{t-1}, \sigma^2\Big)$$

with the definition:

$$\hat{\mathbf{v}}_{t-1} \equiv [v_{t-1}, v_{t-2}, ..., v_{t-L}]^T$$

18.2.1 Training an AR model

Maximum likelihood training of the AR coefficients is:

$$\mathbf{a} = \Big(\sum_t \hat{\mathbf{v}}_{t-1}\hat{\mathbf{v}}_{t-1}^T\Big)^{-1} \sum_t v_t \hat{\mathbf{v}}_{t-1}$$

which can be solved by Gaussian elimination. The linear system has a **Toeplitz form** that can be more efficiently solved using the **Levinson-Durbin method**. Furthermore, optimally:

$$\sigma^2 = \frac{1}{T}\sum_{t=1}^{T}\Big(v_t - \hat{\mathbf{v}}_{t-1}^T\mathbf{a}\Big)^2$$

With a trained $\boldsymbol{a}$, future predictions can be made with the use of $v_{t+1} = \hat{\boldsymbol{v}}_t^T\boldsymbol{a}$. This model is capable of capturing the trend in the data.

18.2.2 Time-varying AR model

A scalar time-invariant AR model can be written as a stochastic OLDS:

$$\boxed{\hat{\mathbf{v}}_t = \mathbf{A}_t\hat{\mathbf{v}}_{t-1} + \boldsymbol{\eta}_t, \ \boldsymbol{\eta}_t \sim \mathcal{N}(\boldsymbol{\eta}_t|\mathbf{0}, \boldsymbol{\Sigma})}$$

with:

$$\mathbf{A} = \begin{pmatrix} a_{1:L-1} & a_L \\ \mathbf{I} & \mathbf{0} \end{pmatrix}, \ \boldsymbol{\Sigma} = \begin{pmatrix} \sigma^2 & 0_{1,1:L-1} \\ 0_{1:L-1,1} & o_{1:L-1,1:L-1} \end{pmatrix}$$

The LDS form can be used for learning the latent AR coefficients, viewing the term:

$$v_t = \hat{\mathbf{v}}_{t-1}^T\boldsymbol{a}_t + \eta_t, \ \eta_t \sim \mathcal{N}(\eta_t|0, \sigma^2)$$

as the emission distribution of a latent LDS in which the hidden variable is $\boldsymbol{a}_t$ and the time-dependent emission matrix is given by $\hat{\mathbf{v}}_{t-1}^T$. One encourages the AR coefficients to change slowly with time by placing a simple latent transition:

$$\boldsymbol{a}_t = \boldsymbol{a}_{t-1} + \boldsymbol{\eta}_t^a, \ \boldsymbol{\eta}_t^a \sim \mathcal{N}(\eta_t^a|0, \sigma_a^2\boldsymbol{I})$$

With this we have the model:

$$p(v_{1:T}, \boldsymbol{a}_{1:T}) = \prod_t p(v_t|\boldsymbol{a}_t, \hat{\mathbf{v}}_{t-1})p(\boldsymbol{a}_t|\boldsymbol{a}_{t-1})$$

From the conditional $p(\boldsymbol{a}_{1:T}, v_{1:T})$ one can compute the a-posteriori most likely sequence of AR coefficients. Standard smoothing algorithms is the choice to obtain the time-varying AR coefficients.

18.3 Switching AR models

The model is allowed to switch between different AR models at each time stamp. Each AR model is associated with a discrete switch coefficient $\boldsymbol{a}(s)$, $s = 1, ..., S$:

$$v_t = \hat{\boldsymbol{v}}_{t-1}^T \boldsymbol{a}(s_t) + \eta_t, \eta_t \sim \mathcal{N}(\eta_t | 0, \sigma^2(s_t))$$

The switch variables have themselves a Markov transition:

$$p(s_{1:T}) = \prod_t p(s_t | s_{t-1})$$

With this the full model is :

$$p(v_{1:T}, s_{1:T} | \theta) = \prod_t p(v_t | v_{t-1}, ..., v_{t-L}, s_t | \theta) p(s_t | s_t - 1)$$

18.4 Latent Linear Dynamical Systems (Kalman Filter)

In Essence: *A LLDS defines a stochastic linear dynamical system in a latent space on a sequence of vectors $\boldsymbol{h}_{1:T}$ with Gaussian transitions and emissions.*

Each observation $\mathbf{v}_t$ is a linear function of the latent vector $\boldsymbol{h}_t$. This setting also goes under the name *linear Gaussian state space model.* The model can be seen as an ordinary LDS on the joint variables $x_t = (v_t, h_t)$ with componets of the vector missing.

One has the transition model $\mathbf{h}_t$ and the emission model $\mathbf{v}_t$:

$$\mathbf{h}_t = \mathbf{A}\mathbf{h}_{t-1} + \boldsymbol{\eta}_t^h, \ \boldsymbol{\eta}_t^h \sim \mathcal{N}(\boldsymbol{\eta}_t^h | \bar{\mathbf{h}}_t, \boldsymbol{\Sigma}_T^H)$$

$$\mathbf{v}_t = \mathbf{B}\mathbf{h}_{t-1} + \boldsymbol{\eta}_t^h, \ \boldsymbol{\eta}_t^h \sim \mathcal{N}(\boldsymbol{\eta}_t^h | \bar{\mathbf{v}}_t, \boldsymbol{\Sigma}_T^V),$$

where $\mathbf{A}$ is the transition matrix and $\mathbf{B}$ the emission matrix. The terms $\bar{\mathbf{h}}_t$ and $\bar{\mathbf{v}}_t$ are the hidden and output biases. With this emission and transition models one obtains the following first order Markov model:

$$\boxed{p(\boldsymbol{h}_{1:T}, \mathbf{v}_{1:T}) = p(\boldsymbol{h}_1) p(\mathbf{v}_1 | \boldsymbol{h}_1) \prod_{t=2}^{T} p(\boldsymbol{h}_t | \boldsymbol{h}_{t-1}) p(\mathbf{v}_t | \boldsymbol{h} t)}$$

with:

$$p(\boldsymbol{h}_t | \boldsymbol{h}_{t-1}) = \mathcal{N}(\boldsymbol{h}_t | \boldsymbol{A}_t \boldsymbol{h}_{t-1} + \bar{\boldsymbol{h}}_t, \boldsymbol{\Sigma}_t^H), \ p(\boldsymbol{h}_1) \mathcal{N}(\boldsymbol{h}_1 | \boldsymbol{\mu}_\pi, \boldsymbol{\Sigma}_\pi)$$

$$p(\mathbf{v}_t | \boldsymbol{h}_t) = \mathcal{N}(\mathbf{v}_t | \boldsymbol{B}_t \boldsymbol{h}_t + \bar{\mathbf{v}}_t, \boldsymbol{\Sigma}_t^V)$$

This model can be represented as a belief network. One may also include an external driving input at each time which will add a $\boldsymbol{\delta}$ to the mean of the hidden variable and the mean of the observation at each time.

18.5 Inference

Due to the underlying structure of a belief network, similar to HMM, the independence assumptions can be re-derived, but now with continuous variables. Meanwhile Gaussian distributions allow exact treatment.

18.5.1 Filtering

The filtered distribution is represented in the *moment representation* as a Gaussian with mean $\boldsymbol{f}_t$ and covariance $\boldsymbol{F}_t$:

$$\boxed{p(\boldsymbol{h}_t|\mathbf{v}_{1:t}) \sim \mathcal{N}(\boldsymbol{h}_t|\boldsymbol{f}_t, \boldsymbol{F}_t)}$$

Assuming time-invariance and zero biases one finds the mean and covariance updates:

$$\boldsymbol{f}_t = \boldsymbol{A}\boldsymbol{f}_{t-1} + \boldsymbol{P}\boldsymbol{B}^T(\boldsymbol{B}\boldsymbol{P}\boldsymbol{B}^T + \boldsymbol{\Sigma}_V)^{-1}(\mathbf{v}_t - \boldsymbol{B}\boldsymbol{A}\boldsymbol{f}_{t-1}),$$

$$\boldsymbol{F}_t = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{B})\boldsymbol{P}$$

with:

$$\boldsymbol{P} = \boldsymbol{A}\boldsymbol{F}_{t-1}\boldsymbol{A}^T + \boldsymbol{\Sigma}_H$$

and the *Kalman gain* matix:

$$\boldsymbol{K} = \boldsymbol{P}\boldsymbol{B}^T(\boldsymbol{\Sigma}_V + \boldsymbol{B}\boldsymbol{P}\boldsymbol{B}^T)^{-1}$$

The iteration is expected to be numerically stable when the noise covariances are small. To avoid numerically instable difference of two positive definite matrices one can write using the *Woodbury identity*:

$$\boldsymbol{F}_t = (\boldsymbol{P}^{-1} + \boldsymbol{B}^T\boldsymbol{\Sigma}_V^{-1}\boldsymbol{B})^{-1}$$

Alternatively with the observation:

$$\boldsymbol{K}\boldsymbol{\Sigma}_V\boldsymbol{K}^T = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{B})\boldsymbol{P}\boldsymbol{B}^T\boldsymbol{K}^T$$

one obtains the numerically less expensive *Joseph's symmetrized update.*

18.5.2 Smoothing

The smoothed posterior $p(\boldsymbol{h}_t|\boldsymbol{v}_{1:T})$ is Gaussian with mean $\boldsymbol{g}_t$ and covariance $\boldsymbol{G}_t$ as conditional marginal of a Gaussian. The *Rauch-Tung-Striebel correction* method is commonly used.

18.5.3 The likelihood

The log-likelihood is given by:

$$\log p(\mathbf{v}_{1:T}) = -1/2\sum_{t=1}^{T}[(\mathbf{v}_t - \boldsymbol{\mu}_t)^T\boldsymbol{\Sigma}_t^{-1}(\mathbf{v}_t - \boldsymbol{\mu}_t) + \log\det(2\pi\boldsymbol{\Sigma}_t)]$$

18.5.4 The most likely hidden state

The most likely hidden state sequence is equivalent to the smoothed mean sequence. The filtered and smoothed covariances $\boldsymbol{F}_t$ and $\boldsymbol{G}_t$ are depending only on the parameters of the model. In practice one even approximates the time-dependent covariances with a single time-independent one using the *algebraic Ricatti equations.*

Related topics: Structured LDSs, Bayesian LDSs, Switching LDSs

Part VI

Hands On

Chapter 19

Statistical Decision Theory

Statistical Decision Theory is the "Schroedinger Equation" of classification answers questions.

What does the best conceivable classifier look like? *Bayes classifier.*
How good is this classifier? *Bayes risk.*

Define the *Loss function* $L(y, \hat{y})$ where y is the true and $\hat{y}$ predicted class, e.g. symmetric/asymmetric *0-1-loss.* Aim of classification is to minimize the expected loss="risk" R of classifier f:

$$R(f) := \mathbb{E}_x \mathbb{E}_y L(y, f(x)) = \int_x \mathbb{E}_y L(y, f(x)) p(x) dx = \sum_{y \in Y} \sum_{z \in Y} L(y, z) I[f(x) = z] p(y|x)$$

where $\mathbb{E}$ is taken w.r.t. true distributions. For symmetric *0-1-loss*, i.e. loss of 1 for wrong predictions, one has:

$$\mathbb{E}_y L(y, f(x)) = \sum_{y \in Y/f(x)} 1 \cdot p(y|x) = 1 - p(f(x)|x)$$

since

$$\sum_y p(y|x) = 1$$

with $y \in Y/f(x)$ is y except for the predicted class and $p(y|x)$ is the true posterior which one does not have. One can interpret the result as: the posterior probability of predicted class must be large to minimize the expected risk, i.e. MAP. For 0-1-loss function the ideal classifier is therefore:

$$f(x) = \underset{z}{\operatorname{argmax}} p(z|x)$$

i.e. Bayes classifier. The actual quality of the classification however largely depend on the data overlap in the feature space.

Chapter 20

Simple Classifiers

20.1 Naive Bayes

Idea: use the independence assumption to construct a simple Gaussian classifier.

A classification algorithm for K classes C_k. With the Bayes rule one has:

$$p(C_k|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|C_k)p(C_k)}{p(\boldsymbol{x})}$$

The *'naive'* assumption is that the variables are not correlated: $p(x_i|x_j, C_k) \sim p(x_i|C_k)$. Therefore:

$$p(\boldsymbol{x}|C_k) = p(x_1|C_k)p(x_2|x_1, Ck)...p(x_n|x_{n-1}, ..., x_1, C_k) \sim \prod_i p(x_i|C_k)$$

One assumes:

$$p(x_i|C_k) \sim \mathcal{N}$$

The label is given with Maximum a posteriori:

$$\hat{y} = \underset{k}{\text{argmax}} p(C_k|\boldsymbol{x}) = \underset{k}{\text{argmax}} p(C_k) \prod_{i=1}^{n} p(x_i|C_k)$$

Extensions: Gaussian Naive Bayes, Multinomial Naive Bayes, Bernoulli Naive Bayes.

20.2 Random Forest

Idea: use the randomized sum of binary trees to classify in a robust and localized way.

20.2.1 CART

Classification and regresssion trees (CART) use series of simple yes/not questions to divide a sample into sectors that represent classes as their boundaries and can be straightforwardly represented as a binary tree. The goal is to construct a dividing effectively via a *greedy strategy* by iteratively asking a question that minimizes the impurity in each sector the most.

Impurity criteria

1) Entropy:

$$i(N) = -\sum_j P(x_j) \log P(w_j)$$

where i is the impurity in a node N and p is the fraction of patterns is class j.

2) Misclassification impurity:

$$i(N) = 1 - \max_j P(w_j)$$

3) Gini impurity:

$$i(N) = \sum_i \sum_{j \neq i} P(w_i)P(w_j) = \sum_i P(w_i) \sum_{j \neq i} P(w_j) = 1 - \sum_i P^2(w_i)$$

which is an expected error if you randomly pick a sample and predict the class of entire node based on itwhere $P(w_i)$ is the probability of selecting a sample of class i and $P(w_j)$ is the probability of making an error. The last equation is get using:

$$\sum_i P(w_i) \sum_j P(w_j) = \sum_i P(w_i) \sum_{j \neq i} P(w_j) + \sum_i P(w_i)^2 = 1$$

Attractiveness of splitting

Attractiveness of splitting a node into left-right child is:

$$i(N) - P_L \cdot i(N_L) - (1 - P_L) \cdot i(N_R)$$

where P_L/P_R are data fractions that go left and right and N_L/N_R are the child nodes. This encourages simple features.

Properties

CART is scale invariant, has a low bias and VERY high variance

Regularization techniques

Regularization techniques to prevent overfitting:

- early stopping if impurity reduction too small
- early stopping based on CV
- early stopping if too few samples left in node
- early stopping if $\sum_N i(N) + c \cdot$(tree size) no longer decreases

Problems

- no parameters
- horizon effect (XOR): a node is only informative after split$\rightarrow$prunning required
- obliqueness$\rightarrow$good data investigation required
- monothetic trees have difficulty with correlated data

Techniques

Prunning - first grow complete tree, then start cutting back according to some criterion

20.2.2 Random Forest

Random Forest merges two techniques for smoother and more stable decision boundary:

1) Bagging - build multiple training sets by sampling with replacement (put back in place: replace), which reduces the variance of ensemble

2) Random Split Selection - at each node, consider only random subset of all features, which makes tree construction less greedy, i.e. also weak features have a chance to be decisive, then average over tress

Feature importance

Feature importance is a multivariate measure:

1) Gini importance - sum of the impurity decreases for each variable over all nodes and trees. Gini prefers categorical variables with many levels.

2) Permutation importance - difference in oob-error is the only available test of classifier:

in-bag samples: used to build a tree and all trees from the training set by random sampling with replacement

out-of-bag samples: have been never sampled, can be used for evaluating prediction accuracy via oob-error

Permutation importance has problems with correlated data.

More simple classifiers: LDA/QDA, SVM

Chapter 21

Sampling Methods

21.1 Metropolis-Hastings Update

Design the transition probability $p(x \to x')$.

Detailed balance condition

$$p(x)p(x') = p(x')p(x' \to x)$$

demands:

$$\frac{p(x \to x')}{p(x' \to x)} = \frac{p(x')}{p(x)}$$

The probability of an update is the *proposal probability* g and *acceptance probability* A:

$$p(x \to x')g(x \to x')A(x \to x')$$

With that one has has the ratio R:

$$R = \frac{A(x \to x')}{A(x' \to x)} = \frac{p(x')g(x' \to x)}{p(x)g(x \to x')}$$

The *Metropolis choice* is then the following: accept $A(x \to x')$ if $R \geq 1$ otherwise use lower acceptance:

$$A(x \to x') = \min(1, \frac{p(x')g(x' \to x)}{p(x)g(x \to x')})$$

21.2 Gibbs sampling

Gibbs sampling is a special case of Metropolis-Hastings update. Instead of sampling a vector $\mathbf{x}^{(i+1)}$ from the joint distribution, better take the conditional on the most recent values:

$$p(x_j^{(i+1)}|x_1^{(i+1)}, ..., x_{j-1}^{(i+1)}, x_{j+1}^{(i)}, ..., x_n^{(i)})$$

with j being the component index of the vector. With that one get a first order Markov chain.

21.3 Hybrid Monte Carlo

Sample distribution $p(x)$. Define the 'physical' Hamiltonian with a kinetic term K:

$$H(s, \phi) = E(s) + K(\phi) = E(s) + \frac{1}{2}\sum_i \phi^2$$

Then the distribution is sampled from:

$$p(s, \phi) = \frac{1}{Z} \exp(-H(s, \phi)) = p(s)p(\phi), \qquad p(\phi) \sim \mathcal{N}$$

Since a product is involved, one gets samples from $p(x)$ by simple marginalization. The Hamiltonian dynamics allows the use of *leapfrog update*. The model dynamics is therefore a mix of Hamiltonian dynamics via leapfrog and a Gibbs sampling for velocity ϕ from $\mathcal{N}$. One uses Metropolis/Hastings accept/reject after n leapfrog states:

$$p_{acc}(x, x') = \min(1, \frac{p(s', \phi')}{p(s, \phi)})$$

Chapter 22

Kernels

RBF - radial basis function

$$k(x_i, x_j) = \exp(-\|x_i - x_j\|^2/2\sigma^2)$$

Polynomial

$$k(x_i, x_j) = (x_i^T x_j + 1)^T$$

Sigmoid with steepness k and offset Θ, not p.s.d. in general

$$k(x_i, x_j) = \tanh(k x_i^T x_j + \Theta)$$

Fischer kernel

$$k(x_i, x_j) = u_{x_i}^T u_{x_j}$$

where $u_x = \nabla_\theta \ln p(x|\theta)$ and θ are the model parameters.

More kernels: Matern kernel, Intersection kernel, kernel Alignment, γ-exponential kernel, Ornstein-Uhlbeck kernel, Inverse Quadrics, Thin Plate Spline, Wendland Splines, B-splines, Cardinal function, sinc, Catmull-Rom spline..

Kernel trick applications: Laplacian Eigenmaps, Multidimensional Scaling, kernel density estimation, kernel PCA (Spectral Clustering), kernel k-Means, kernel SVM, Multiple Kernel Learning.

Chapter 23

Covariance measures

23.1 Univariate variables

For variance and covariance of univariate RV's x and y there are such *linear* assosciations as variance and covariance:

$$V(x) := \mathbb{E}_x((x - \mathbb{E}(x))^2)$$

$$Cov(x, y) := \mathbb{E}_{x,y}((x - \mathbb{E}(x))(y - \mathbb{E}(y)))$$

The more computationally efficient expression of covariance is:

$$\begin{aligned}
Cov(x, y) &= \mathbb{E}(x, y) - \mathbb{E}(x)\mathbb{E}(y) && (23.1)\\
&= \mathbb{E}_{x,y}(xy - x\mathbb{E}_y(y) - \mathbb{E}_x(x)y + \mathbb{E}_x(x)\mathbb{E}_y(y)) && (23.2)\\
&= \mathbb{E}_{x,y}(xy) - \mathbb{E}_y(y)\mathbb{E}_{x,y}(x) - \mathbb{E}_x(x)\mathbb{E}_y(y) + \mathbb{E}_x(x)\mathbb{E}_y(y) && (23.3)\\
&= \mathbb{E}(xy) - \mathbb{E}(x)\mathbb{E}(y) && (23.4)
\end{aligned}$$

Properties are:

$$Cov(\sum_i a_i x_i, \sum_j b_j y_j) = \sum_i \sum_j a_i b_j Cov(x_i, y_j)$$

Notice that $V(\sum_i (X_i)) \neq \sum_i V(X_i)$:

$$\begin{aligned}
V(\sum_i X_i) &= Cov(\sum_i X_i, \sum_j X_j) && (23.5)\\
&= \sum_i \sum_j Cov(X_i, X_j) && (23.6)\\
&= \sum_i V(X_i) + 2\sum_{i>j} Cov(X_i, X_j) && (23.7)\\
&= \text{ diag} + \text{ off-diag} && (23.8)
\end{aligned}$$

23.2 Multivariate variables

For multivariate $X \in \mathbb{R}^p$:

$$Cov(X)^{p\times p} = \mathbb{E}((X - \mathbb{E}(X))(X^T - \mathbb{E}(X^T)))$$

$$\begin{aligned}
Cov(A^{q\times p}X^{p\times 1}) &= \mathbb{E}(AX - \mathbb{E}(AX))(X^T A^T - \mathbb{E}(X^T A^T))) && (23.9)\\
&= A\mathbb{E}((X - \mathbb{E}(X))(X^T - \mathbb{E}(X^T)))A^T && (23.10)\\
&= ACov(X)A^T && (23.11)
\end{aligned}$$

23.3 Independence

Independence implies $Cov(X,Y) = 0$:

$$X \perp Y <=> P(X,Y) = P(X)P(Y) => Cov(X,Y) = 0$$

23.4 Sample covariance matrix

An empirical estimate for covariance is:

$$Q = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})^T$$

where i is column, based on a sample from tzhe true distribution.

Remark: estimated mean deviates from the true mean in the sense that it is "optimized" w.r.t. the sample. Therefore the seen spread is lower than the true one, hence the normalization $1/(n-1)$. There are two exceptions, where $1/n$ should be used instead:
1) the mean is known
2) Gaussian distribution, then it gives a ML estimator of Q
3) for $n >> 1$ doesn't make a difference

23.5 Correlation

Since covariance is scale dependent:

$$Cov(aX,Y) = aCov(X,Y)$$

one defines the scale-independent (Pearson-) correlation:

$$Corr(X,Y) := \rho(X,Y) = \frac{Cov(X,Y)}{\sqrt{Var(X)}\sqrt{Var(Y)}} = \frac{Cov(X,Y)}{\sigma_x \sigma_y}$$

scaled to $-1 \leq \rho \leq 1$. Covariance is a measure if a linear model is good, however gets irritated by outliers.

More measures: Rank correlation coefficient, Spearman-correlation, Kendall-τ rank correlation coefficient

23.6 Capturing non-linear dependence

23.6.1 Distance correlation

"Distance correlation" $R(X,Y)$ is also defined for X,Y of different dimensions:

$R(X,Y) = 0$ iff X,Y independent
$R(X,Y) = 1$ at least if $Y = a + bX\mathcal{O}$, where $\mathcal{O}^T\mathcal{O} = I$ is an orthogonal matrix
$0 \leq R(X,Y) \leq 1$

23.6.2 Fourier transform

"Characterisitic function" $\mathcal{F}(p.d.f.)$ continuous transform:

$$X \perp Y <=> f_{X,Y} = f_X \cdot f_Y$$

Measure of dependence:

$$m = ||f_{X,Y} - f_X \cdot f_y||_w$$

with some weight function w.

23.6.3 Empirical distance covariance

Take empirical distance covariance as measure of dependence for n data points:

$$m_n^2(X,Y) = ||f_{X,Y}^n(t,s) - f_X^n(t) f_Y^n(s)||_w^2$$

where n are data points, (t,s) are dimensions of $\mathcal{F}$. The weighting function is:

$$w(t,s) = \frac{1}{c_p \cdot c_q \cdot |t|^{1+p} \cdot |s|^{1+q}}$$

where c_p, c_q are half the surface of unit sphere in p/q dimensions (dimensions of random variables). Empirical distance covariance is simply calculated from the centered distance matrix:

$$m_n(X,Y)^2 = \frac{1}{n^2} \sum_{k=1}^{n} \sum_{l=1}^{n} [D_X]_{kl} [D_Y]_{kl}$$

$$D_X = C A_X C$$

$$[A_X]_{kl} = ||x_k - x_l|| = \frac{1}{n^2} \mathbb{E}^T S_X S_Y \mathbb{E}$$

By using $w(t,s) = 1$ one gets the Pearson-covariance.

Chapter 24

Convex Optimization

In Essence: *To handle generally more complicated optiization problems, one approximates the problem with the convex problem in the dual domain. The duality gap is the resulting slack. Convex optimization is minimizing convex functions over convex sets. Hence a local minimum is the global minimum.*

24.1 Lagrange dual

Consider the foolowing optimization problem:

Objective: find $\min f_0(x)$ under the constraints $f_i(x) \leq 0$, $i = 1, ..., m$ and $h_j(x) = 0$, $j = 1, ..., p$.

Feasible set: values of x, where the constraints are fulfilled.

The Lagrangian is:

$$\mathcal{L}(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{j=1}^{p} \nu_j h_j(x)$$

with Lagrange multipliers λ, ν.

The Lagrange dual approximates the problem:

$$g(\lambda, \nu) = \inf_{x \in D} \mathcal{L}(x, \lambda, \nu)$$

where one considers the infimum (largest lower bound) and closes the gap. This corresponds to minima of $\mathcal{L}$ as function of the Lagrange multipliers in domain D where f_0, f_i, h_j are defined.

Since $f(x) \leq 0$, the minima of $g(\lambda, \nu)$ are infima of the true minimum $f_0^*(x^*)$ which is called the **Duality Gap**.

The dual problem is simple and always a *convex optimization problem* independent of the original problem. The dual function is always concave. With the dual problem one finds the 'tightest lower bound' d^*. The 'price of the duality' is measured by the duality gap $x^* - d^*$.

Weak Duality: $d^* \leq p^*$.

Strong Duality: $d^* = p^*$, lower bound is tight.

24.2 Constraint qualifications

24.2.1 Slater's theorem

Is a *sufficient* condition for strong duality to hold for a convex optimization problem. It is a specific example of constraint qualification. In particular, if Slater's condition holds for the primal problem, then the duality gap is 0, and if the dual value is finite then it is attained.

Slater's theorem: for convex $f_0(x)$, $f_i(x)$ strong duality holds in case the feasible region has an interiour point (and all the non-linear constraints are slack).

24.3 Optimality conditions

Karush-Kuhn-Tucker Conditions are *necessary* conditions for a solution in nonlinear programming to be optimal, provided that some constraint qualifications are satisfied. This can be a Slater condition or the requirement of differentiability of all constraints and zero duality gap. For convex optimization problems the KKT-Conditions are also sufficient for global minimum.

Karush-Kuhn-Tucker Conditions: For a local mimimum x^* which satisfies regularity conditions there exist $(\lambda^*,\ \mu^*)$ so that (x^*, λ^*, ν^*) is a **KKT-Point**. The KKT-Conditions can be formulated as following.

A *point* (x^*, λ^*, ν^*) is a **KKT-Point** if it satisfies:

1. **Stationarity**:

   $\nabla_x \mathcal{L}(x^*, \lambda^*, \nu^*) = 0$
2. **Primal feasibility**:

   x^* is is the feasible set.
3. **Dual feasibility**:

   $\lambda_i^* \geq 0$
4. **Complementary slackness**:

   $\lambda_i^* f_i^*(x^*) = 0$

Part VII

Math Basics

Chapter 25

Statistics Basics

25.1 Variables and Probability

Variable: can take states from its domain.

Domain: $dom(x) = \{...\}$ are all the states, the variable x can take.

Set of variables: $\mathcal{V} = \{a, B, c\}$.

Probability: $p(state)$ is shorthand for $p(x = state)$.

Summation over variable: $\sum_x f(x) \equiv \sum_{s \in dom(x)} f(x = s)$.

Distribution for x: a variable x, its $dom(x)$ and the probability values p(x) for each of the states.

Normalization condition: $\sum_x p(x) \equiv \sum_{\times \in dom(x)} p(x = \times) = 1$.

Variable interaction: $p(x\, or\, y) = p(x) + p(y) - p(x\, and\, y)$.

Marginal: $\sum_x p(x, y) = p(y)$.

Bayes rule: posterior $= p(\theta|\mathcal{D}) = \frac{p(\theta,\mathcal{D})}{p(\mathcal{D})} = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} = \frac{p(\mathcal{D}|\theta)p(\theta)}{\int_\theta p(\mathcal{D}\theta)p(\theta)} = \frac{\text{generative model (likelihood)}\times\text{prior belief}}{\text{evidence}}$.

Independence $x \perp y$:

a) $p(x, y) = p(x)p(y)$

b) for $p(x) \neq 0$ and $p(y) \neq 0$ $p(x|y) = p(x) \Leftrightarrow p(y|x) = p(y)$

c) p(x,y)=kf(x)g(y) for constant k and positive f, g.

Conditional independence $\mathcal{X} \perp \mathcal{Y}|\mathcal{Z}$: $p(\mathcal{X}, \mathcal{Y}|\mathcal{Z}) = p(\mathcal{X}|\mathcal{Z})p(\mathcal{Y}|\mathcal{Z})$, in case $\mathcal{Z} = \emptyset$: $\mathcal{X} \perp \mathcal{Y}$.

Independent and identically distributed: For a variable x and a set of i.i.d. observations $x^1, ..., x^N$ conditioned on θ we assume no dependence between the observations: $p(x^1, ..., x^N|\theta) = \prod_{n=1}^N p(x^n|\theta)$.

Probability table: encodes the joint over the discrete variables in a table, e.g. $p(x, y) = p(x|y)p(y)$,
$P_{ij} = p(x = i|y = j)p(y = j)$. Sum over columns gives $p(y)$, sum over rows gives $p(x)$. From this one can easily get $p(y|x) = p(x, y)/p(x)$.

25.2 Functions and Distributions

Indicator function: $\mathbf{I}[A] = 1$ if A=true, 0 otherwise.

Dirac delta: $\delta(x - x_0) = \lim_{\sigma \to 0} \mathcal{N}(x|x_0, \sigma^2)$, $\int \delta(x - x_0)dx = 1$, $\int \delta(x - x_0)f(x)dx = f(x_0)$.

Kronecker delta: $\delta_{x,x_0} = \mathbf{I}[x = x_0]$.

25.3 Statistics

Mean value (first moment):

$\langle f(x) \rangle_{p(x)} \equiv \int f(x; ..)p(x; ..)dx$

$\langle f(x)|y \rangle \equiv \int f(x; ...)p(x; ...|y; ...)dx$

$\langle f(\mathcal{X}_i) \rangle_{q(\mathcal{X})} = \langle f(\mathcal{X}_i) \rangle_{q(\mathcal{X}_i)}$ *which says that if the function f only depends on a subset of the variables, we only need to know the marginal distribution of this subset of variables in order to carry out the average.*

Change of variables:

$p(y) = p(x)\left(\frac{df}{dx}\right)^{-1}$, $y = f(x)$ for a univariate continuous random variable x with distribution p(x) and monotonic $f(x)$.

$p(\mathbf{y}) = p(\mathbf{x} = f^{-1}(\mathbf{y}))\left|\det\left(\frac{\partial f}{\partial x}\right)\right|^{-1}$ for multivariate $\mathbf{x}$ and bijection $\mathbf{y} = f(\mathbf{x})$. The **Jacobian** has the elements $\left[\frac{\partial f}{\partial x}\right]_{ij} = \frac{\partial f_i(x)}{\partial x_j}$.

Moments:

k^{th} moment $= \langle x^k \rangle_{p(x)}$.

Moment generating function:

$g(t) = \langle e^{tx} \rangle_{p(x)}$ with $\lim\limits_{t \to 0} \frac{d^k}{dt^k} g(t) = \langle x^k \rangle_{p(x)}$.

Cumulative distribution function:

$cdf(y) \equiv \int p(x \leq y) = \langle \mathbf{I}[x \leq y] \rangle_{p(x)}$.

Mode x_*:

$x_* \equiv \text{argmax}_x p(x)$, term: 'multi-modal'.

Variance:

$\sigma^2 \equiv \langle (x - \langle x \rangle)^2 \rangle_{p(x)} = \langle x^2 \rangle - \langle x \rangle^2$ with *standard deviation* σ

Covariance matrix:

$\Sigma_{ij} = \langle (x_i - \mu_i)(x_j - \mu_j) \rangle = \langle x_i x_j \rangle - \langle x_i \rangle \langle x_j \rangle$ with $\mu_i = \langle x_i \rangle$ for a multivariate distribution. $x_i \perp x_j \Rightarrow \Sigma_{ij}, \rho_{ij} = 0$ (only in this direction). If x_i and x_j are Gaussian distributed, then the arrow is bothsided.

Skewness:

$\gamma_1 \equiv \frac{\left\langle (x-\langle x\rangle)^3 \right\rangle_{p(x)}}{\sigma^3}$ with σ^2 is variance of $p(x)$. $\gamma_1 > 0 \Rightarrow$ heavy tail to the right and vice-versa.

Kurtosis:

$\gamma_2 \equiv \frac{\left\langle (x-\langle x\rangle)^4 \right\rangle_{p(x)}}{\sigma^4} - 3$ is a measure of how peaked around the mean a distribution is. A distribution with positive kurtosis has more mass around its mean than would a Gaussian with the same mean and variance. These are also called *super Gaussian*. Similarly a negative kurtosis (*sub Gaussian*) distribution has less mass around its mean than the corresponding Gaussian. The kurtosis is defined such that a Gaussian has zero kurtosis (which accounts for the -3 term in the definition).

Correlation matrix:

$\rho_{ij} = \left\langle \frac{(x_i-\mu_i)}{\sigma_i} \frac{(x_j-\mu_j)}{\sigma_j} \right\rangle$ which is the normalized form of covariance with $-1 \leq \rho_{ij} \leq 1$.

Kullback-Leibler Divergence KL(q|p):

$< \log q(x) - \log p(x) >_{q(x)} \leq 0$ measures the non-overlapping regions of two distributions.

Entropy H(p):

$- < \log p(x) >_{p(x)}$, $H(p) = -KL(p|u) + \text{const.}$, with $u =$ uniform distribution. The more similar the distribution is to the uniform distribution, the greater is the entropy. Therefore the entropy is the measure of the a-priori uncertainty in the state occupancy (uniform = no information, every state has the same probability).

Mutual information:

$MI(\mathcal{X};\mathcal{Y}|\mathcal{Z}) =< KL\big(p(\mathcal{X},\mathcal{Y}|\mathcal{Z})|p(\mathcal{X}|\mathcal{Z})p(\mathcal{Y}|\mathcal{Z})\big) >_{p(\mathcal{Z})} \leq 0$ is the measure of dependence between sets of variables $\mathcal{X}$, $\mathcal{Y}$ conditioned on $\mathcal{Z}$. If $\mathcal{X}$ is ind. of $\mathcal{Y}$ cond. on $\mathcal{Z}$, then $MI(\mathcal{X};\mathcal{Y}|\mathcal{Z}) = 0$ and vice versa.

25.4 Information Criteria

Bayes Information Criterion (BIC):

$$BIC = K \log N - 2 \log p(D|\boldsymbol{\theta}^*, M)$$

Akaike Information Criterion (AIC):

$$AIC = 2K - 2 \log p(D|\boldsymbol{\theta}^*, M)$$

25.5 Empirical Distribution

Given n samples one describes the data with a simple mean.

Empirical distribution:

$p(x) = \frac{1}{N} \sum_{n=1}^{N} \mathbf{I}[x = x^n]$ which is $p(x) = \frac{1}{N} \sum_{n=1}^{N} \delta(x - x^n)$ for continous distributions and $\mathbf{I}[x = x^n] = \prod_i \mathbf{I}[x_i = x_i^n]$ in the case x is a vector of discrete variables.

Sample mean:

The mean of the empirical distribution is $\hat{\mu} = \frac{1}{N}\sum_{n=1}^{N} x^n$. For vectors one has $\hat{\mu} = \frac{1}{N}\sum_{n=1}^{N} x_i^n$.

Sample variance:

The variance of the empirical distribution is $\hat{\sigma}^2 = \frac{1}{N}\sum_{n=1}^{N}(x^n - \hat{\mu})^2$. Consider the unbiased normalization $\frac{1}{N-1}$ for small samples.

Sample covariance matrix:

For vectors one has $\hat{\Sigma}_{ij} = \frac{1}{N}\sum_{n=1}^{N}(x_i^n - \hat{\mu}_i)(x_j^n - \hat{\mu}_j)$, $\hat{\mathbf{\Sigma}} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}^n - \boldsymbol{\mu})(\mathbf{x}^n - \boldsymbol{\mu})^T$. For centered data one has: $\hat{\mathbf{\Sigma}} \equiv \mathbf{S} = \frac{1}{N}\mathbf{X}\mathbf{X}^T$. Consider the normalization $\frac{1}{N-1}$ of *unbiased covariance* for small samples. Sample covariance matrix is the maximum likelihood estimation for the covariance of a multinomial normal distribution given n samples.

25.6 Basic Distributions

25.6.1 Bernoulli

The Bernoulli distribution models the outcome of a single Bernoulli trial (yes/no question). In other words, it models whether flipping a (possibly biased) coin one time will result in either a success (obtaining a head) or failure (obtaining a tail). We have the probability mass function $f(k, p)$ equal p if $k = 1$ or equal $1 - p := q$ if $k = 0$. This can be written as:

$$f(k, p) = p^k(1 - p)^{1-k}$$

for binary k which is the number of successes.

25.6.2 Binomial

The binomial distribution generalizes this to the number of heads from performing n independent flips (Bernoulli trials) of the same coin, which is equivalent to binary sampling *with replacement.*

$$f(k, n, p) = \binom{n}{k} p^k(1 - p)^{n-k}$$

with $k < n$ being the number of successes in n trials.

25.6.3 Categorical

The categorical distribution is the generalization of the Bernoulli distribution, or a special case of the Multinomial distribution with $n = 1$, for a categorical random variable, i.e. for a discrete variable with more than two possible outcomes, such as the roll of a dice.

$$f(k, \boldsymbol{p}) = \prod_{i=1}^{k} p_i^{x_i}$$

with $\sum_{i=1}^{k} p_i = 1$ and the vector $\boldsymbol{x}$ being restricted to only have one element equal 1 and others equal 0.

25.6.4 Multinomial

The multinomial distribution models the outcome of n experiments, where the outcome of each trial has a categorical distribution, such as rolling a k-sided die n times.

$$f(k, \boldsymbol{n}, \boldsymbol{p}) = \binom{n}{x_1, ..., x_k} \prod_{i=1}^{k} p_i^{x_i}$$

with $\sum_{i=1}^{k} p_i = 1$, $\sum_{i=1}^{k} x_i = n$, p_i being the probability for each of the mutually exclusive events k (dice side) for each of n trials and x_i being the number of events k for all n trials.

25.7 Mahalanobis Distance

Intuitively Mahalanobis distance is the probability that a point belongs to a distibution, taking correlations into account. Mahalanobis distance is a distance between a point P and a distribution D, or how many standard deviations away is P from the mean of D along each of the principal component axes. Mahalanobis distance is unitless, scale invariant and takes into account correlation $\mathbf{S}$ of the data:

$$D_M(\mathbf{x}) = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^T \mathbf{S}^{-1} (\mathbf{x} - \boldsymbol{\mu})}$$

For $\mathbf{S} = \mathbf{E}$ the Mahalanobis distance corresponds to the Euclidean distance. Manalanobis distance can also be defined between any two points of the distribution, i.e. $\boldsymbol{\mu} \rightarrow \mathbf{y}$. Expressed with the Mahalanobis distance, the multivariate normal distribution is:

$$\mathcal{N}(D_M(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{S}), \boldsymbol{S}) = \frac{1}{\sqrt{(2\pi)^p \det \boldsymbol{S}}} e^{-\frac{1}{2} D_M(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{S})^2}$$

Chapter 26

LA Basics

26.1 Vector

Vector x, $n \times 1$ matrix, a column

$$E^T_{(1\times n)}E_{(n\times 1)} = n$$

Scalar product

$$x^Ty = y^Tx \equiv < x, y > \ (1 \times n \cdot n \times 1 = 1 \times 1)$$

Dyadic product

$$v \cdot w^T \equiv v \otimes \ (n \times 1 \cdot 1 \times n = n \times n)\ , c_{ik} = \sum_j a_{ij}b_{jk}$$

26.2 Matrix Definitions

Regularity

$$\det(A^{n\times n}) \neq 0 \Rightarrow \exists A^{-1} \vee A \text{ regular/invertible}$$

Singularity

$$\det(A^{n\times n}) = 0 \Rightarrow A \text{ singular/not invertible}$$

Orthogonality

$$< Av, Aw >=< v, w > \text{ or } A^{-1} = A^T \text{or } AA^T = E \textit{ reflections, rotations, reflection-rotations}$$

Unitarity

$$A \in \mathbb{C} \vee < Av, Aw >=< v, w > \ \text{ or } A^{-1} = A^T \text{ or } A^{-1} = A^* \text{ or } AA^* = E$$

Normality

$$A \in \mathbb{C} \vee A^*A = AA^* \Rightarrow \text{A is unitarily diagonilizable: } A = UDU^*$$

Every unitary or hermitic matrix is normal

Symmetricity

$$A^T = A$$

Hermiticity

$$A = A^*, A \in \mathbb{C}$$

Skew symmetricity/Antisymmetricity

$$-A^T = A$$

Positive/negative (semi) definite

$$v^T Av < / > (\leq / \ \geq)\ 0$$

$$v^* Av < / > (\leq / \ \geq)\ 0 \vee A \in \mathbb{C}$$

Corresponding are for $A^T = A$ ($A^* = A \vee A \in \mathbb{C}$) the eigenvalues.
A real matrix $A^{n\times n}$ is -//- if its symmetric part $A_s = 1/2(A + A^T)$ is -//-.

For orthogonal (unitary) A:

- A is regular
- $\|Ax\|_2 = \|x\|_2$
- $\|AB\|_2 = \|B\|_2$
- $|\lambda| = 1$ for all (complex) eigenvalues
- $|\det(A)| = 1$

26.3 Matrix Norms

Frobenius norm

$$\|A\|_F = \sqrt{\sum_i \sum_i |a_{ij}|^2} = (A^T A)$$

Spectral norm

$$\|A\|_2 = \max_{x\neq 0} \frac{\|Ax\|_2}{\|x\|_2} = \max_{\|x\|_2=1} \|Ax\|_2$$

Largest stretch factor of a vector with $\|X\|_2 = 1$

- $\|A\|_2 = \sqrt{\lambda_{max,A^*A}}$
- $\|A\|_2 \leq \sqrt{\|A\|_1 \cdot \|A\|_\infty}$
- $\|A\|_2 \leq \|A\|_F$

Spectral radius is the largest eigenvalue of a matrix

26.4 Matrix Properties

$AB \neq BA$ in general
$(AB)^T = B^T A^T$
$(A^{-1})^T = (A^T)^{-1}$

26.5 Trace

$$tr(A^{n\times n}) = \sum_{j=1}^{n} a_{jj}$$

$$tr(AB) = tr(BA)$$

$$tr(A^T) = tr(A)$$

$$tr(ABC) = tr(CAB) = tr(BCA)$$

$$\frac{\partial trABC}{\partial B} = trA^T C^T$$

$$\frac{\partial trAB^T BA}{\partial B} = 2BAA^T$$

26.6 Rank

A matrix A has full rank, if $\det(A) \neq 0$ or $\lambda_i(A) \neq 0$ or there are only linearly independent rows/columns (column rank).

26.7 Matrix Differentiation

Assuming $A \neq f(x)$, $A \neq f(X)$:

$$\frac{\partial AB}{\partial B} = A^T$$

$$\frac{\partial x^T Ax}{\partial x} = 2x^T A$$

$$\frac{\partial x^T a}{\partial x} = \frac{\partial a^T x}{\partial x} = a^T$$

$$\frac{\partial b^T Ax}{\partial x} = b^T A$$

$$\frac{\partial tr(XAX^T)}{\partial X} = (A^T + A)X$$

$$\frac{\partial tr(X^T AX)}{\partial X} = X^T(A^T + A)$$

$$\frac{\partial tr(AX)}{\partial X} = A$$

$$\frac{\partial tr(AX^T)}{\partial X} = A^T$$

$$\frac{\partial x^T x}{\partial x} = 2x^T$$

26.8 Special Matrices

26.8.1 Data matrix

$X^{p\times n}$ is matrix of observations, centered

Scatter matrix:

$$\tilde{\mathbf{S}}^{p\times p} = XX^T$$

Scalar product matrix:

$M_{SKP}^{n\times n} := X^T X$ is the matrix of scalar products of all observations.

26.8.2 SVD - Singular Value Decomposition

$$A^{m\times n} = U \cdot S \cdot V^T$$

with matrix of singular values $S = diag(s_1, ..., s_n), s_1 \geq s_2 \geq ... \geq s_n$ and $V^T V = U^T U = \mathbb{I}$.
The properties are:

$$|\det(A)| = \prod_i s_i$$

$\det(A^{n\times n}) \neq 0$ if A is square

$$||A||_F = \sum_i s_i^2$$

$$A^{-1} = VS^{-1}U \text{ pseudo inverse}$$

$$Av_i = s_i u_i$$

$$A^T u_i = s_i v_i$$

where s_i are length changes and u_i are directions of the linear transformation A in the image space of the orthonormal vectors in the original space.

A helpful picture is the following. If U, V are unitary then they represent a rotation and $|\text{eigenvalue}| = 1$, $\det(U) = \det(V) = 1$. S is then a stretching/compression without rotation. U,S,V act on an orthonormal system.

26.8.3 Centering Matrix

$$C_n = (\mathbb{I} - \frac{1}{n} 1_n 1_n^T)$$

with centered dataset:

$$XC_n = X - \bar{X} 1_n^T$$

where $\bar{X}$ is the mean of each row. A centered data set is:

$$X \cdot \mathbb{I} = 0$$

Chapter 27

Graph Basics

Loop: path containing more than two nodes, that irrespective of edge direction starts and returns to the same node.

Cycle: *directed* path that starts and returns to the same node.

Chord: edge that connects two non-adjacent nodes in a loop.

Markov Blanket: a Markov blanket of a node x_i are its parents, children and the parents of its children (excluding itself). In a belief network for any other variable y that is not in the Markov blanket of x_i one has $x_i \perp y | MB(x_i)$. That is, the Markov blanket of x_i carries all information about x_i.

Family of a node: the node itself and its parents.

Clique: given an undirected graph, a clique is a fully connected subset of nodes. All the members of the clique are neighbours.

Maximal clique: for a clique to be maximal, there is no larger other clique that contains this clique. Cliques play a central role in both modelling and inference. In modelling they will describe variables that are all dependent on each other. In inference they describe sets of variables with no simpler structure describing the relationship between them and hence for which no simpler efficient inference procedure is likely to exist.

Cliquo: non-maximal clique.

Connected graph: undirected graph is connected if there is a *path* between every pair of nodes (i.e. there are no isolated islands). For a graph which is not connected, the connected components are those subgraphs which are connected.

Singly connected graph/a tree: graph is singly connected if there is only one *path* from any node A to any other node B. This definition applies regardless of whether or not the edges in the graph are directed.

Multiply connected/loopy graph: otherwise the graph is multiply connected.

Spanning tree: spanning tree of an undirected graph G is a singly-connected subset of the existing edges such that the resulting singly connected graph covers all nodes of G.

Maximum weight spanning tree: a maximum weight spanning tree is a spanning tree such that the sum of all weights on the edges of the tree is at least as large as any other spanning tree of G.

Directed Acyclic Graph (DAG): a DAG is a graph G with directed edges between the nodes such that by following a path of nodes from one node to another along the direction of each edge no path will revisit a node. In a DAG the ancestors of B are those nodes who have a directed path ending at B. Conversely, the descendants of A are those nodes who have a directed path starting at A.

Separation: a subset S separates a subset A from a subset B (for disjoint A and B) if every path from any member of A to any member of B passes through S. If there is no path from a member of A to a member of B then A is separated from B. If $S = \emptyset$, then provided no path exists from A to B, A and B are separated.

d-separation: A aund B are d-separated if for all paths between A and B there is an intermediary variable Z s.t.:

1. $A \to Z \to B$ (serial) or $A \leftarrow Z \to B$ (diverging) and Z is observed
2. $A \to Z \leftarrow B$ (converging) and neither Z nor its children are observed

Collider: given a path P, a collider is a node c on P with neighbours a and b on P such that $a \to c \leftarrow b$. A collider is path specific.

More definitions: edge list, adjacency matrix powers, ancestral order with triangular adjacency matrix, clique matrix, incidence matrix

Bibliography

[1] Lectures "Pattern Recognition" and "Machine Learning" at HCI, by Prof. Hamprecht, Heidelberg

[2] Lecture "Fundamentals of Simulation Methods" at HITS, by Prof. Springel, Heidelberg