# Machine Learning Summer School 2018

Miquel Perello-Nieto

August 27–September 7, 2018

# Contents

# Bibliography


[1] M. Artetxe, G. Labaka, E. Agirre, and K. Cho. Unsupervised neural machine translation. *arXiv preprint arXiv:1710.11041*, 2017.

[2] F. Bach, R. Jenatton, J. Mairal, G. Obozinski, et al. Optimization with sparsity-inducing penalties. *Foundations and Trends® in Machine Learning*, 4(1):1–106, 2012.

[3] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[4] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[5] T. Blom and J. M. Mooij. Generalized strucutral causal models. *arXiv preprint arXiv:1805.06539*, 2018.

[6] P. Bojanowski and A. Joulin. Unsupervised learning by predicting noise. *arXiv preprint arXiv:1704.05310*, 2017.

[7] S. Bongers and J. M. Mooij. From random differential equations to structural causal models: the stochastic case. *arXiv preprint arXiv:1803.08784*, 2018.

[8] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[9] M. A. Castano, F. Casacuberta, and E. Vidal. Machine translation using neural networks and finite-state models. *Theoretical and Methodological Issues in Machine Translation (TMI)*, pages 160–167, 1997.

[10] C. Dann, G. Neumann, and J. Peters. Policy evaluation with temporal differences: A survey and comparison. *The Journal of Machine Learning Research*, 15(1):809–883, 2014.

[11] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier. Language modeling with gated convolutional networks. *arXiv preprint arXiv:1612.08083*, 2016.

[12] M. De Mooij. *Global marketing and advertising: Understanding cultural paradoxes*. Sage Publications, 2013.

[13] P. Dyer and S. McReynolds. Optimization of control systems with discontinuities and terminal constraints. *IEEE Transactions on automatic Control*, 14(3):223–229, 1969.

[14] D. Ernst, P. Geurts, and L. Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(Apr):503–556, 2005.

[15] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

[16] O. Firat, K. Cho, and Y. Bengio. Multi-way, multilingual neural machine translation with a shared attention mechanism. *arXiv preprint arXiv:1601.01073*, 2016.

[17] O. Firat, B. Sankaran, Y. Al-Onaizan, F. T. Y. Vural, and K. Cho. Zero-resource translation with multi-lingual neural machine translation. *arXiv preprint arXiv:1606.04164*, 2016.

[18] R. A. Fisher. The design of experiments. 1935.

[19] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional Sequence to Sequence Learning. *ArXiv e-prints*, May 2017.

[20] A. Graves, G. Wayne, and I. Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

[21] J. Gu, G. Neubig, K. Cho, and V. O. Li. Learning to translate in real-time with neural machine translation. *arXiv preprint arXiv:1610.00388*, 2016.

[22] F. Hieber, T. Domhan, M. Denkowski, D. Vilar, A. Sokolov, A. Clifton, and M. Post. Sockeye: A Toolkit for Neural Machine Translation. *arXiv preprint arXiv:1712.05690*, Dec. 2017.

[23] S. Jean, O. Firat, K. Cho, R. Memisevic, and Y. Bengio. Montreal neural machine translation systems for wmt'15. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 134–140, 2015.

[24] A. Joulin, L. van der Maaten, A. Jabri, and N. Vasilache. Learning visual features from large weakly supervised data. *CoRR*, abs/1511.02251, 2015.

[25] N. Kalchbrenner, L. Espeholt, K. Simonyan, A. v. d. Oord, A. Graves, and K. Kavukcuoglu. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*, 2016.

[26] Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[27] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. OpenNMT: Open-source toolkit for neural machine translation. In *Proc. ACL*, 2017.

[28] S. Lange, T. Gabel, and M. Riedmiller. Batch reinforcement learning. In *Reinforcement learning*, pages 45–73. Springer, 2012.

[29] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196, 2014.

[30] J. Lee, K. Cho, and T. Hofmann. Fully character-level neural machine translation without explicit segmentation. *arXiv preprint arXiv:1610.03017*, 2016.

[31] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[32] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[33] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. *CoRR*, abs/1603.09246, 2016.

[34] A. Owens, J. Wu, J. H. McDermott, W. T. Freeman, and A. Torralba. Ambient sound provides supervision for visual learning. *CoRR*, abs/1608.07017, 2016.

[35] J. Pearl. Causal inference without counterfactuals: Comment. *Journal of the American Statistical Association*, 95(450):428–431, 2000.

[36] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[37] J. Peters and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697, 2008.

[38] M. Riedmiller, T. Gabel, R. Hafner, and S. Lange. Reinforcement learning for robot soccer. *Autonomous Robots*, 27(1):55–73, 2009.

[39] S. Schaal and C. G. Atkeson. Constructive incremental learning from only local information. *Neural computation*, 10(8):2047–2084, 1998.

[40] H. Schwenk, D. Dchelotte, and J.-L. Gauvain. Continuous space language models for statistical machine translation. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 723–730. Association for Computational Linguistics, 2006.

[41] R. Sennrich, O. Firat, K. Cho, A. Birch, B. Haddow, J. Hitschler, M. Junczys-Dowmunt, S. Läubli, A. V. M. Barone, J. Mokry, et al. Nematus: a toolkit for neural machine translation. *arXiv preprint arXiv:1703.04357*, 2017.

[42] S. Sukhbaatar, R. Fergus, et al. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, pages 2244–2252, 2016.

[43] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus. Weakly supervised memory networks. *CoRR*, abs/1503.08895, 2015.

[44] R. S. Sutton, A. G. Barto, et al. *Reinforcement learning: An introduction*. MIT press, 1998.

[45] K. Tran, A. Bisazza, and C. Monz. Recurrent memory networks for language modeling. *arXiv preprint arXiv:1601.01272*, 2016.

[46] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802, 2015.

[47] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015.

[48] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola. Deep sets. In *Advances in Neural Information Processing Systems*, pages 3391–3401, 2017.

# Chapter 1

# Introduction

- 150 out of 500
- Aug 29 19:30 guided tour through downtown Madrid
- Sep 1 9:00am Visti to segovia
- Sep 5 20:00

# Chapter 2

# Bayesian Deep Learning: Planting the seeds of probabilistic thinking by Shakir Mohamed

- Shakir Mohamed - Cambridge, CIFAR, DeepMind

## 2.1 Introduction 27 Aug. Mon. 9:30–11:00

- Different views of probability
  - Statistical probability: frequency ratio of items
  - Logical Probability: Degree of confirmation of an hypothesis based on logical analysis
  - Probability as Propensity: probability used for predictions
  - Subjective probability: probability as a degree of belief
    - Probability is a measure of a belief in a proposition given evidence. A description of a state of knowledge.
    - Different observers with different information will have different beliefs
- $p(x)$ probability of $x$, $p^*(x)$ true probability of $x$
- Conditions: $p(x) \geq 0$, $\int p(x)dx = 1$
- Bayes rule: $p(z|x) = \frac{p(x|z)p(z>)}{p(x)}$
- Parametrisation: $p_\theta(x|z) = p(x|z;\theta)$
- Gradient: missed
- Generalised Linear Regression

$$\mu = w^T x + b \tag{2.1}$$

- Recursive Generalised Linear Regression
- Recursively copose the basic linear functions
- Gives a deep neural network
- Probabilistic model
- $p(y|x) = p(y|h(x); \theta)$
- likelihood function (log of the previous one)
- $\mathcal{L}(\theta) = \sum_n \log p(y_n|x_n; \theta)$
- Sometimes the likelihood is not computationally tractable because of an integral
- Likelihoods can give you estimates of parameters
- They are efficient estimators, are asymptotically unbiased
- It is possible to make statistical tests on the likelihood, this can give you confidence intervals
- Pool information: combine the outcomes of different data sources
- Biggest problem: misspecification, inefficient estiamtes or confidence intervals/test can fail completly (assuming Gaussian while data is not)

A likelihood function with regularization term

$$\mathcal{L}(\theta) = \sum_n \log p(y_n|x_n; \theta) + \frac{1}{\lambda}\mathcal{R}(\theta) \tag{2.2}$$

Other names for the regularization are ...

The Maximum a Posteriory estimation (MAP)

Shows an example of two instantiations for a MAP estimate where the answer changes from 0 to 1. This problem arises because of the variable scale. In order to solve that, we need to learn more than just the mean.

In Bayesian analysis there are two core components: the evidence

$$p(y|x) = \int p(y|h(x); \theta)p(\theta)d\theta \tag{2.3}$$

and the posterior

$$p(\theta|y, x) \propto p(y|h(x); \theta)p(\theta) \tag{2.4}$$

In Bayesian analysis, all the things that are not obseved need to be integrated over (averaged out)

**Intractable Integrals** do not have a closed form, or very high-dimensional quantities that can't be computed (e.g. using quadrature)

### 2.1.1 Learning and inference

In statistics there is no distinction between learning and inference, only inference (or estimation). While in Bayesian statistics, all quantities are probability distributions, so there is only the problem of inference.

**TODO Look at this with more detail**

- Deep Learning
  - Rich non-linear
  - Scalable learning
  - Easily composable
  - negative point
  - negative point
- Bayesian Reasoning
  - negative point: Rich non-linear
  - negative point: Scalable learning
  - negative point: Easily composable
  - positive point
  - positive point

Some additional examples of Bayesian reasoning combined with Deep learning are Density Estimation, Decision Making and Reinforcement Learning.

Deep and Hierarchical models can be decomposed into a sequence of conditional distributions in the form

$$p(z) = p(z_1|z_2)p(z_2|z_3)\dots p(Z_{L-1}|z_L)p(z_L) \tag{2.5}$$

A list of different models

- Directed vs undirected
- observed vs unobserverd variables
- parametreic vs non-parametric

A list of learning principles (Statistical inference)

- Direct
  - Laplace approximation
  - Maximum likelihood
  - maximum a posteriori
  - variational inference

  - cavity methods
  - integr. nested laplace approximation
  - expectation maximisation
  - markov chain monte carlo
  - noise contrastive
  - sequential monte carlo
- Indirect (missing these ones)

Question about calibrated probabilities. In general in classification it is not but in healthcare is an example of aplication where this is important.

## 2.2 Inferential questions: Mon. 11:30–13:00

Evidence estimation

$$p(x) = \int p(x, z)dz \tag{2.6}$$

Moment computation

$$\mathbb{E}[f(x)|x] = \int f(z)p(z|x)dz \tag{2.7}$$

Identity trick to compute unobserved variables?
Integral problem

$$p(x) = \int p(x|z)p(z)dz \tag{2.8}$$

Probabilistic one

$$p(x) = \int p(x|z)p(z)\frac{q(z)}{q(z)}dz \tag{2.9}$$

Importance sampling: Monte carlo estimator

$$p(x) = \frac{1}{S}\sum_s w^{(s)}p(x|z^{(s)}) \tag{2.10}$$

$$w^{(s)} = \frac{p(z)}{q(z)}\dots \tag{2.11}$$

### 2.2.1 Hutchinson's Trick

Example with computing the trace of a matrix, KL between two gaussians, gradient of a log-determinant

The trace problem $Tr(A)$ with zero mean unit variance $\mathbb{E}[zz^T] = I$. Applying the identity trick we obtain

$$Tr(AI) = Tr(A\,\mathbb{E}[zz^T]) \tag{2.12}$$

With linear operations allow us to obtain

$$\mathbb{E}[Tr(Azz^T)] \tag{2.13}$$

And because of the trace property

$$\mathbb{E}[z^T Az]. \tag{2.14}$$

Then, with montecarlo methods the expected value is converted into a sumation

### 2.2.2 Probability Flow Tricks

Given a distribution and sample

$$\hat{x} \sim p(x) \tag{2.15}$$

With a transformation

$$\hat{y} = g(\hat{x}; \theta) \tag{2.16}$$

Change of variables

$$p(y) = p(x) \left| \frac{dg}{dx} \right|^{-1} \tag{2.17}$$

Example: compute the

$$\log\det\left(\frac{\partial f(z)}{\partial z}\right) \tag{2.18}$$

This method is seen in the literature as *Normalising Flows* (see more in the blogpost)

### 2.2.3 Stochastic Optimization

Compute the gradient of the common problem

$$\nabla_\phi \mathop{\mathbb{E}}_{q_\phi(z)}[f_\phi(z)] = \nabla_\phi \int q_\phi(z) f_\phi(z) dz \tag{2.19}$$

With some reparametrisation tricks a distribution can be expressed as a transofmration of other distributions

$$z \sim q_\phi(z) \tag{2.20}$$

Other names for these methods are samples, one-liners and change-of-variables

### 2.2.4 Pathwise Estimator

Also known as Unconscious statistician, stochastic backpropagation, perturbation analysis, reparametrisation trick, affine-independent inference.

When to use this trick

- Function $f$ is differen
- Density $q$ is known with a suitable transorm of a simpler base distribution: inverse CDF, location-scale transform, or other co-ordinate transform
- Easy to sample from base distribution

### 2.2.5 Log-derivative trick

Score function is the derivative of a log-likelihood function

$$\nabla_{\phi} \log q_{\phi}(z) = \tag{2.21}$$

### 2.2.6 Score-function estimator

Also known as Likelihood ratio method, reinforce and policy gradients, automated and black-box inference

We should use this method when

- Function is not differentiable, not analytical
- Distribution $q$ is easy to sample from
- Density $q$ is known and differentiable

### 2.2.7 Evidence Bounds

Integral problem

$$p(x) = \int p(x|z)p(z)dz \tag{2.22}$$

Proposal

$$p(x) = \int p(x|z)p(z)\frac{q(z)}{q(z)}dz \tag{2.23}$$

Importance weight
Jensen's inequality
Lower bound: evidence ower bound

$$\mathop{\mathbb{E}}_{q(z)}[\log p(x|z)] - KL[q(z)||p(z)] \tag{2.24}$$

### 2.2.8 Density Ratio Trick

The ratio of two densities can be computed using a classifier of using samples drawn from the two distributions. (TODO there is a type here, find out why)

$$\frac{p^*(x)}{q(x)} = \frac{p(y=1|x)}{p(y=-1|x)} \tag{2.25}$$

## 2.3 Unsupervised Learning Thu. 9:30–11:00

- Move beyond of associating inputs to outputs
- Generative models
- It allows to perform density estimation
  - Probabilistic models
  - High-dimensional data
  - Data distribution is targeted

Some examples of aplications for generative models is the compression of images with high resolution GANs. Some artists used GANs to create pieces of video art. In an example the artist makes a video of a piece of cloth and gives the video as an input to a GAN that is trained with a particular set of images (eg. waves on the sea, or fire), then the GAN needs to generate new images that resemble the training data.

Types of generative models are:

- Fully-observed models
- Latent variable models (where there is a direction (from z to x?)
- Undirected models: where there is no known direction from the observed variables X to the hidden variables Z
- https://arxiv.org/abs/1202.3732

Some points to consider when designing a generative model

- Data: binary, real-valued, nominal, strings, images
- Dependency: independent, sequential, temporal, spatial
- Representation: continuous or discrete
- Dimension: parametric or non-parametric
- Computational complexity
- Modelling capacity
- Bias, uncertainty, calibration
- Interpretability

### 2.3.1 Fully-observed models

This are models that operate on the observed data directly. It does not assume any hidden variables that may interact with our observed variables. As an example, Markov models assume a dependency with past events, depending on the number of dependencies with past events it has different orders of dependency.

$$x_1 \sim \text{Cat}(x_1|\pi) \tag{2.26}$$
$$x_2 \sim \text{Cat}(x_2|\pi(x_1)) \tag{2.27}$$
$$\dots \tag{2.28}$$
$$x_i \sim \text{Cat}(x_i|\pi(x_{<n})) \tag{2.29}$$
$$p(x) = \prod_i p(x_i|f(x_{<i};\theta)) \tag{2.30}$$

Some of the properties of these models are

- Can directly encode how observed points are related
- Any type of data can be used
- ...

Directed and discrete: NADE, EoNADE... Directed and continuous: Normal means... Undirected and discrete:... Undirected and continuous:...

### 2.3.2 Latent variable models

These models introduce unobserved local random variables that represent a hidden cause. One of the most common assumptions is to assume some random hidden noise in the called Prescribed models. On the other hand, ....

An example of a prescribed model is a Deep Latent Gaussian Models in wich all the hidden variables follow a Gaussian distribution that are connected one to each other and with dependencies to previous variables.

$$z_3 \sim \mathcal{N}(0, \mathbf{I}) \tag{2.31}$$
$$z_2 \sim \mathcal{N}(\text{some dependency on} z_3) \tag{2.32}$$
$$z_1 \sim \mathcal{N}(\text{some dependency on} z_3 \text{and} z_2) \tag{2.33}$$
$$\dots \tag{2.34}$$
$$x_1 \sim \mathcal{N}(\text{some dependency on} z_i) \tag{2.35}$$

Some of the properties of Latent variable models are

- ...

Some dimensions in order to separate different latent models are, discrete vs continuous, deep vs direct/linear, and parametric vs non-parametric. Some examples are Buffet process, Sigmoid Belief Nets, Deep Gaussian processes, Hidden Markov Model, Sparse LVMs, Nonlinear factor Analysis.

### 2.3.3 Implicit Models

These are models that asume a random hidden variable that adds noise to the observed variables. One of the most common examples is to assume random Gaussian noise in a signal.

TODO the following equations need to be checked:

$$z \sim \mathcal{N}(\mu, \sigma^2) \tag{2.36}$$
$$x \sim f(z) \tag{2.37}$$

Some of the important properties are

- Easy to sample
- easy to compute expectations
- Can exploit on large-scale classifiers and Convolutional networks (I think this is the regularisation part?)

We can separate this models mostly on functions discrete time and difffusions in continous time.

## 2.4 Model-Inference-Algorithm

We will understand VAriational Autoencoders and Generative ...

## 2.5 Variational Inference

The variational principle is a general family of methods for the approximation of a complicated density by a simpler class of densities. In most of the cases we can asses the similarity between our approximated distribution and the original one by using the Kullback–Leibler divergence.

IN variational inference there is always a variational bound that (Evindence Lower Bound, a.k.a. ELBO)

$$F(x, q) = \mathop{\mathbb{E}}_{q(z)} [\log p(x|z)] - KL[q(z)||p(z)] \tag{2.38}$$

## 2.6 Mean-Fields

These methods assume that the distribution is factorised (the hidden variables are independent). This means that we can compute their probability by multiplying every independent hidden variable probability.

The most expresive model with the true posterior would be $q^*(z|x) \propto p(x|z)p(z)$ (true posterior), in the least expressive side we have $q_{MF}(z|x) = \prod_z q(z_k)$ (fully-factorised model). There is a huge vairety of models in between like Hidden Markov models, Autoregresive models, Gaussian processes?

## 2.7 Variational Optimisation

In the variational Expectation Maximisation consists on an alternating optimization for the variational parameters and the model parameters (VEM).

$$\phi \propto \underset{\phi}{\nabla} \text{missing equation} \tag{2.39}$$

In the E-step instead of computing q with every sample of our dataset, we will simulate the answer with an Inference network $q(z/x)$ that will give us a sample $z \sim q(z/x)$. This may be an encoder or inverser. TODO need to see this previous paragraph with more detail.

## 2.8 Reinforcement learning as generative models

- An unknown likelihood
- not known analytical
- something more

Applying all the techniques seen in the three lectures it is possible to learn a policy learning that can be used in reinforcement learning.

# Chapter 3

# An Introduction to Gaussian Processes by Richard Turner

Example with a non-linear regression

Start with an example of a multivariate Gaussian

$$p(y|\Sigma) \propto \exp(\frac{-1}{2} y^T \Sigma^{-1} y) \tag{3.1}$$

If we condition the probabilities of one variable we obtain Gaussian shaped distributions as well

$$p(y_2|y_1, \Sigma) \propto (\exp(\frac{-1}{2} y^T (y_2 - \mu_0) \Sigma^{-1} y (y_2 - \mu_0)) \tag{3.2}$$

TODO: revise previous equation

If we map any sample from the original distribution, it is possible to draw a line with the x-axis the different variables (variable index), and the y-axis the coordinates. If the variables are correlated, the lines should be quite horizontal.

In the examples it goes from 2 variables to 20, and show how the farther is the variable less correlated is to the first variable.

In the example, there is a coviariance matrix with a Gaussian with fixed variance in the diagonal with the mean going from the first variable to the last one.

Then, if some of this variables are actual samples (fix values), then the sampling process should force the other variables to converge to these samples. (TODO: check if the following equation is an "l" or an "I")

$$\Sigma(x_1, x_2) = \mathbf{K}(x_1, x_2) + l\sigma_y^2 \tag{3.3}$$

$$\mathbf{K}(x_1, x_2) = \sigma^2 e^{-\frac{1}{2l^2}(x_1 - x_2)^2} \tag{3.4}$$

I a non-parametric method (infinit parameters)

$$p(y|\theta) = \mathcal{N}(y; 0, \Sigma) \tag{3.5}$$
$$\tag{3.6}$$

where $\sigma^2$ is the scale (vertical scale), and $l$ is the horizontal-scale (this scales can be seen in the previous lineplot).

In a parametric model

$$y(x) = f(x; \theta) + \sigma_y \epsilon \tag{3.7}$$
$$\epsilon \sim \mathcal{N}(0, 1) \tag{3.8}$$

Definition of a Gaussian process: generalisation of a multivariate Gaussian distribution to infinitely many variables

A Gaussian distribution is fully specified by a mean vector, $\mu$ and a covariance matrix $\Sigma$.

$$f = (f_1, \ldots, f_n) \sim \mathcal{N}(\mu, \Sigma), \text{ indices } i = 1, \ldots, n \tag{3.9}$$

$$y(x) = f(x) + \epsilon\sigma_y \tag{3.10}$$
$$p(\epsilon) = \mathcal{N}(\epsilon; 0, 1) \tag{3.11}$$

place a GP prior over the non-linear function

$$p(f(x)|\theta) = GP(f(x); 0, K_\theta(x, x')) \tag{3.12}$$
$$\mathbf{K}(x, x') = \sigma^2 \exp\left(-\frac{1}{2l^2}(x - x')^2\right) \tag{3.13}$$

sum of a Gaussian variable into a GP is still a multivariate gaussian.

$$\text{missing equation} \tag{3.14}$$

The marginalisation property of Gaussian distributions is

$$p(y_1) = \int p(y_1, y_2) dy_2 \tag{3.15}$$
$$p(y_1, y_2) = \mathcal{N}\left([] \text{ missing}\right) \rightarrow p(y_1) = \mathcal{N}(y_1 : a, A) \tag{3.16}$$

How do twe make a prediction

$$p(y_1|y_2) = \frac{p(y_1, y_2)}{p(y_2)} \tag{3.17}$$
$$\rightarrow p(y_1|y_2) = \mathcal{N}(y_1 : a + BC^{-1}(y_2 - b), A - BC^{-1}B^T) \tag{3.18}$$

Where $y_1$ are the predicted positions and $y_1$ are the samples

The predictive mean (first part of $\mathcal{N}$ is linear in the data $= Wy_2$

The predictive covariance (second part of the $\mathcal{N}$) ca be interpreted as the predictive uncertanity = prior uncertainty $A$ - the reduction in uncertainty $BC^{-1}B^T$.

What are the implications of the hyper-parameters?

- $\sigma$ missing implications
- $l$ missing implications

We can use the probability distributions to represent the plausibility of the hyper-parameters (uncertainty) given the data

Bayes theorem with posterior probability of data given the parameters (3.19)

Shows an example of modifying the lengh-scale variable $l$ and showing how the likelihood of the parameter value starts increasing and decreasing, showing a peak at lenghtscale 2.

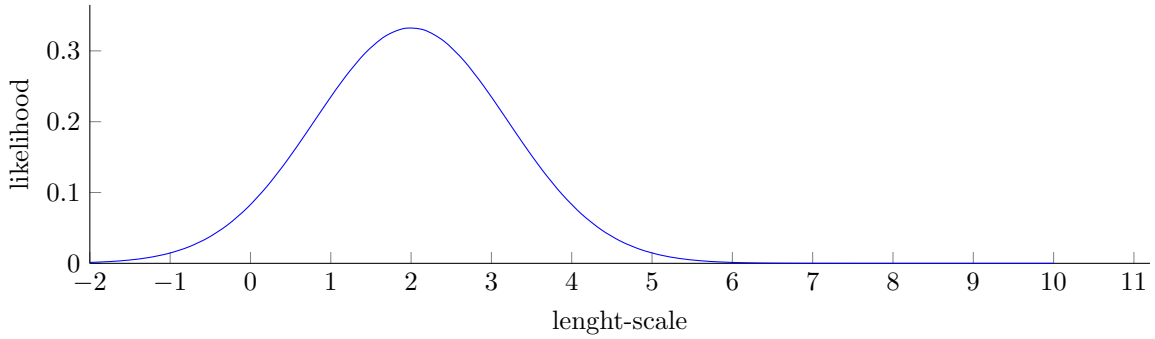


Figure 3.1: Likelihood of the data guiven the value of the lenght-scale $l$

## 3.1 Covariance functions

Warning. Difficult to compare different models. If you try to compute the posterior probability of the different models guiven the data, it is hard to compute, it needs approximations and the results are really sensitive to the priors.

### 3.1.1 References

- Gaussian Processes for Machine Learning, Rasmussen and Williams, 2006
- Gaussian Process Summer School, Neil Lawrence and collegues
- Software

– GPy: Gaussian processes in Python
– GPflow: Gaussian Processes and tensorflow
– GPML: Gaussian Processes in Matlab
– GP Stan: Gaussian Processes in probabilistic programming

## 3.2 Using Gaussian Processes: Models, applications and connections 16:30–18:00

Question 1: Addition of two GPs

$$f(x) = f_1(x) + f_2(x) \tag{3.20}$$
$$f_1(x) \sim GP(0, \Sigma_1(x, x')) \tag{3.21}$$
$$f_2(x) \sim GP(0, \Sigma_2(x, x')) \tag{3.22}$$

The expected value is

$$m(x) = \mathbb{E}(f(x)) = \mathbb{E}(f_1(x) + f_2(x)) \tag{3.23}$$
$$= \mathbb{E}(f_1(x)) + \mathbb{E}(f_2(x)) \tag{3.24}$$

And the covariance is

$$\Sigma(x) = \mathbb{E}(f(x)f(x')) - \mathbb{E}(f(x))\,\mathbb{E}(f(x)) \tag{3.25}$$
$$= \mathbb{E}\left[(f_1(x) + f_2(x'))(f_1(x) + f_2(x'))\right] \tag{3.26}$$
$$= \mathbb{E}[f_1(x) + f_1(x')] + \mathbb{E}[(f_2(x) + f_2(x')] + \mathbb{E}[f_1(x) + f_2(x')] + \mathbb{E}[(f_2(x) + f_1(x')] \tag{3.27}$$
$$= \mathbb{E}[f_1(x) + f_1(x')] + \mathbb{E}[(f_2(x) + f_2(x')] \tag{3.28}$$

More generally: GPs closed under linear transofmration / combination:

- GP multiplied by a deterministic function = GP
- derivatives of GP = GP
- integral of a GP = GP
- convolution of a GP by a deterministic funtion = GP

Question 2: Random linear model

$$g(x) = mx + c \tag{3.29}$$
$$m \sim \mathcal{N}(0, \sigma_m^2) \tag{3.30}$$
$$c \sim \mathcal{N}(0, \sigma_c^2) \tag{3.31}$$

The expected value

$$m(x) = \mathbb{E}[g(x)] = \mathbb{E}(m)x + \mathbb{E}(c) = 0 + 0 \tag{3.32}$$

The covariance

$$\begin{aligned}
\Sigma(x, x') =& \sigma_m^2 x x' + \sigma_c^2 && (3.33)\\
=& \mathbb{E}(g(x)g(x')) && (3.34)\\
=& \mathbb{E}((mx + c)(mx' + c)) && (3.35)\\
=& \mathbb{E}(m^2)xx' + \mathbb{E}(c^2) + \mathbb{E}(cm)x' + \mathbb{E}(mc)x && (3.36)\\
=& \mathbb{E}(m^2)xx' + \mathbb{E}(c^2) + 0 + 0 && (3.37)
\end{aligned}$$

Question 3: random sinusoid model

$$\begin{aligned}
h(x) &= a\cos(wt) + b\sin(wt) && (3.38)\\
a &\sim Normal(0, \sigma^2) && (3.39)\\
b &\sim Normal(0, \sigma^2) && (3.40)
\end{aligned}$$

$$m(x) = 0 \tag{3.41}$$

$$\Sigma(x, x') = \sigma^2 \cos(w(x - x')) \tag{3.42}$$

Bochner's theorem: Any stationary covariance function can be written as

$$\Sigma(x - x') = \int \sigma^2(x) \cos(w(x - x'))dw \tag{3.43}$$

roughly, the function comprises and uncontably infinite sum of random sins and cosines

| linear mappings $f(x) = Wx$ | neural network | Gaussian Process mappings |
|---|---|---|
| linear regression | neural network regression | Gaussian Process regression |
| PCA or Factor analysis | variational auto-encoder (VAE) | Gaussian Process latent variable model |
| Gaussian auto-regressive (or Markov) model | neural auto-regressive density estimation (NADE) | Gaussian process auto-regressive model (GPAR) |
| linear Gaussian state space model (LGSSM) | recurrent neural latent variable model | Gaussian process state-space model (GP-SSM) |

Strenghts of Gaussian Processes

- Interpretable machine learning
- data-efficient machine learning
- decision making
- automated machine learning including probabilistic numerics

Weaknesses

- Large numbers of datapoints ($N \leq 10^5$)
- High-dimensional inputs spaces ($D \leq 10^2$)

In the speakers opinion, GPs are not good for large image recognition tasks, but for small tasks where it is cruciall to get uncertainty estimates.

Example of an Interpretable auto-ML: the automatic statistician

Given some airline data it detects four underlying patterns, linearly increasing factor, a periodicity at every year, some incresing noise.

Shows an example with a video of an inverted pendulum and how in a small number of iterations (around 7?) a GP is learned.

### 3.2.1 Deep Gaussian Processes

Allowing non-parametric kernel spaces

$$y(x) = f(x) + \sigma_y \epsilon \tag{3.44}$$
$$\text{missing this part} \tag{3.45}$$

with a composition of GPs

$$y(x) = f(g(x)) + \sigma_y \epsilon \tag{3.46}$$
$$f(x) = GP(0, K_f(x, x')) \tag{3.47}$$
$$g(x) = GP(0, K_g(x, x')) \tag{3.48}$$

Deep GP may perform automatic kernel design

A Neural Network with one hidden layer and infinite number of units in the hidden layer is a Gaussian Process Nial 1996

A Neural Network with multiple hidden layers with infinite number of hidden units on each layer is also a GP (the variance on the weights need a specific variance $\sigma^2/D$ Matthews et al. 2018. The specific variance is the reason why with finite number of units the regularisation needs to be readjusted and follows the values found by Matheews et al.

### 3.2.2 References

- Gaussian process latent variable models for visualisation of high dimensional data by Lawrence
- Local distance preservation in the GP-LVM throught Back constraints
- the automatic statistician
- PILCO: a model-based and data-efficient approach to policy serch

## 3.3 Large data and non-linear models Tue. 11:30–13:00

Shows a few examples of sounds generation using Gaussian Processes.

## 3.4 A brief history of gaussian process approximation

One of the first approaches was to modify the original samples in a way that the computational complexity of making exact inference was reduced from $O(n^3$ to $O(n^2)$.

See following publications:

- Sparse Gaussian Processes using Pseudo-inputs
- Local and global sparse gaussian process approximations
- Sparse-posterior Gaussian Processes for general likelihoods
- Variational Learning of Inducing variables in sparrse Gaussian Processes
- Fast Forward selection to speed up sparse Gaussian Process Regression

Some examples of Factor graphs

$$p(x_1, x_2, x_3) = g(x_1, x_2, x_3) \tag{3.49}$$
$$p(x_1, x_2, x_3) = g_1(x_1, x_2) g_2(x_2, x_3) \tag{3.50}$$
$$\tag{3.51}$$

Where in the first one all the nodes are connected with one factor (square), while the second one $x_1$ is only connected to $x_2$ and this one to $x_3$.

Shows an example of a multivariate gaussian where the covariance matrix has some numbers, and the inverse of the covariance matrix presents some zeros. By looking at the inverse of the covariance matrix we can see what nodes are independent guiven the rest (positions whith the value zero).

$$\mathcal{N}(x,\mu,\Sigma) \propto \exp[-1/2(x-\mu)^T\Sigma^{-1}(x-\mu)] \tag{3.52}$$
$$= \exp[-1/2\sum_{i,j}(x_i-\mu_i)\Sigma^{-1}_{i,j}(x_j-\mu_j)] \tag{3.53}$$
$$= \prod_{i,j}\exp[-1/2(x_i-\mu_i)\Sigma^{-1}_{i,j}(x_j-\mu_j)] \tag{3.54}$$
$$= \prod_{i,j} g_{i,j}(x_i,x_j) \tag{3.55}$$

The previous equations show that when the inverse of the covariance is 0, the exponent value goes to one and the multiplicative factors are not affected.

The Kullback-Leibler divergence has the Gibb's inequality property, that means that $KL(p_1(z)||p_2(z)) \geq 0$ and has the equality at $p_1(z) = p_2(z)$. In order to prove the previous property apply the Jensen's inequality or differentiation. It is also Non-symmetric $KL(p_1(z)||p_2(z)) \neq KL(p_2(z)||p_1(z))$.

### 3.4.1 Fully independent training conditional (FITC) approximation

1. Augment model with $M < T$ pseudo data

$$p(f,u) = \mathcal{N}(\text{f u, with mean 0 and covariance K}) \tag{3.56}$$

2. Remove some of the dependencies (results in a simpler model)
3. Calibrate model (e.g. using KL divergence, many choices)

- This method introduces a pareametric bottleneck into a non-parametric model
- Should we add extra pseudo-data when more data is avilable?

### 3.4.2 Variational free-energy method (VFE)

In this case we want to lower bound the likelihood

$$\mathcal{L}(\theta) = \log p(y|\theta) = \log \int df p(y,f|\theta) \tag{3.57}$$
$$= \log \int df p(y,f|\theta)\frac{q(f)}{q(f)} \geq \int df q(f)\log\frac{p(y,f|\theta)}{q(f)} = F(\theta) \tag{3.58}$$
$$= \int df q(f)\log\frac{p(f|y,\theta)p(y|\theta)}{q(f)} = \log p(y|\theta) - \mathbf{KL}(q(f)||p(f|y)) \tag{3.59}$$

$$F(\theta) = \int df q(f)\log\frac{p(y,f|\theta)}{\text{missing}} \tag{3.60}$$

At the end we get a lower bound of the likelihood.

$$F(\theta) = \int df q(f) \log \frac{p(y, f|\theta)}{p(f_{\neq u}|u)q(u)} \tag{3.61}$$

$$= \int df q(f) \log \frac{p(y|f, \theta)p(f_{\neq u}|u)p(u)}{p(f_{\neq u}|u)q(u)} \tag{3.62}$$

$$= \int df q(f) \log \frac{p(y|f, \theta)p(u)}{q(u)} \tag{3.63}$$

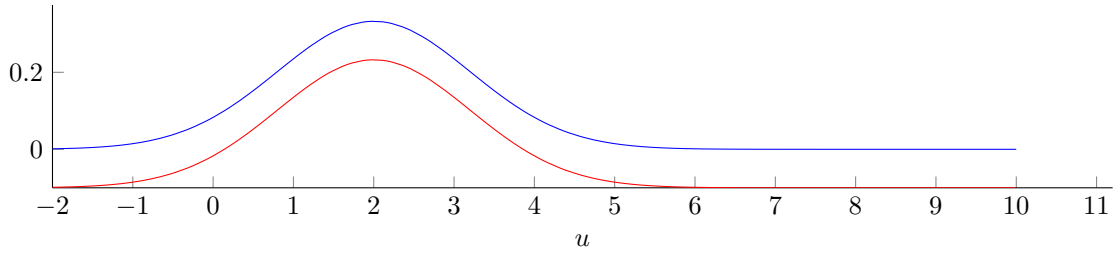


Figure 3.2: Where the blue line is the true likelihood and the red line is the approximated $q(u)$

They may reduce the time complexity from $O(M^3)$ to $O(NM^2)$, however $N$ is porportional to $M$ and in certain way is just a small improvement.

Some extra references about approximate inference in GPs:

- Sparse online gaussian processes
- A unifying view of sparse approximate gaussian process regression
- variational learning of inducing variables in sparse gaussian processes
- on sparse variational methods and the kullback-leibler divergence between stochastic processes
- A unifying framework for Gaussian Process Pseudo-Point approximation using Power Expectation propagation
- Streaming sparse gaussian process aproximations
- Efficient deterministic approximate Bayesian Inference for Gaussian Process Models

and about Deep Gaussian processes:

- Deep gaussian processes for regression using approximate expectation propagation
- Doubly stochastic variational inference for deep gaussian processes

# Chapter 4

# Deep Learning by Rob Fergus 14:30–16:00

"Rob Fergus is an Associate Professor of Computer Science at the Courant Institute of Mathematical Sciences, New York University. He is also a Research Scientist at Facebook, working in their AI Research Group. He received a Masters in Electrical Engineering with Prof. Pietro Perona at Caltech, before completing a PhD with Prof. Andrew Zisserman at the University of Oxford in 2005. Before coming to NYU, he spent two years as a post-doc in the Computer Science and Artificial Intelligence Lab (CSAIL) at MIT, working with Prof. William Freeman. He has received several awards including a CVPR best paper prize, a Sloan Fellowship & NSF Career award and the IEEE Longuet-Higgins prize." – Personal Page from New York University

## 4.1 Deep Supervised Learning

### 4.1.1 History of Neural Nets

The connectionsims started around 40's with Hebb's work, McCulloch and Pitts and Rosenblatt 50's.

Second era started around 80's with Backpropagation to train multi-layered networks with the work of Rumelhart, Hinton and Williams. And the architecture of the Convolutional neural networks by Yann LeCun.

The era of Deep learning starts around 2011 with tasks focused on vision, speech understanding and natrual language processing. The main ingredients were (1) supervised training of deep networks with (2) faster GPUs and (3) big labeled datasets.

### 4.1.2 Deep learning vs traditional approaches

The traditional approach consists on the hand-designed feature extraction that may be used by a simple classifier in order to predict the labels of a set of data. On the oposite hand, deep neural networks are focused on learning useful feature representations that improve the performance of the model on the dataset.

From 2010 to 2015 there is a clear decrease on the error rate in the ImageNet tasks by ussing more complex neural networks. The number of layers goes from 8 layers with the AlexNet in 2012 to 152 layers with the ResNet in 2016.

There are similar jumps in the performance of speech recognition systems. As an example, in 2015 Baidu0's Deep Speach 2 system used 100 million parameteres in 11 layers of a Recurrent Neural Network. It was trained in around 11.940 hours of English.

In the task of natural language processing and Machine Translation the work by Sutskever et al. and Cho et.al 2014. Tests of perplexity show a linear decrease from 2013?

One of the benefits of Deep learning is that altought there is a huge computational cost during training, it is really lightweight during deployment. This means that small devices like smartphones are able to perform real-time predictions. One example is the detection of cars, road and pedestrians on self-driving cars.

Other interesting areas where deep learning is being applied are: astronomy, healthcare (e.g. skin cancer classification, or diabetic retinopathy).

### 4.1.3 Some issues with Deep Learning

(1) There is a missing theoretical understanding and performance guarantees. (2) Difficult to inspect the models. (3) Need lots of labeled data (4) We are still hand-designing the network architecture (instead of the features). There are some attempts to learn automatically the architecture (meta-learning) like Neural architecture search with reinforcement learning. (5) The most generic architectures (i.e. fully connected) does not seem to work, and it seems that their architecture is really dependent on the domain (e.g. for images exploit 2D)

### 4.1.4 Convolutional Neural Networks

First designed in the work of Yann LeCun (and previously Fukushima) by trying to mimic the visual system (of cats in Fukushima). It tries to exploit the spatial relation of pixels. In the work of Fukushima there was no back-propagation and the architecture could not be learned automatically. It was after back-propagation was applied into neural networks in 1986 that Yann LeCunn could show a practical application by classifiying handwritten digits from a dataset generated by some american post offices with Yann LeCunn as an investigator (TODO rephrase previous sentence).

If the traditional activation functions were sigmoid shaped like the hyperbolic tangent or the logistic function, in more recent years activations with linear regions started to become more popular (e.g. ReLu).

Anothre useful trick is the Batch Normalisation, that has been empirically proved to improve the performance of CNNs. This method consists on normalizing the input features with 0 mean and 1 standard deviation on every mini-batch. There are two extra parameters (TODO missed theses parameters).

### 4.1.5 Stochastic Gradient Descent

This is an iterative learning method that usually trains with mini-batches.

$$\text{missing} \tag{4.1}$$

Another training method is the AdaGrad

$$\theta_{t+1} = \theta_t - \text{missing} \tag{4.2}$$

RMSProp

$$\text{missing} \tag{4.3}$$

and ADAM

$$\text{missing} \tag{4.4}$$

### 4.1.6 Some practical debugging tips from M. Ranzato

- Train on small subset of data: the training erro should go to zero
- Training diverges: learnoing rate may be too large (decrease learning rate), or the back-propagation is buggy because of numerical gradient issues.
- The parameters collapse, the loss is minimized but the training accurayc does not increase: check the loss function
- TODO some other tricks missing

### 4.1.7 Weights' initialization

With a simple example with a linear activation with 1 layer neural network we have

$$Var[y] = (n^{in} Var[w]) Var[x] \tag{4.5}$$

While in a multilayer network

$$Var[y] = (\prod_d n_d^{in} Var[w_d]) Var[x] \tag{4.6}$$

This value during the forward propagation will tend to explode with the depth $d$. And the backward propagation will make the gradients vanish.

$$\text{missing derivative of previous equation} \tag{4.7}$$

This makes the initialization really important for a good convergence of the weights into a good region within reasonable number of epochs (shows empirical comparison with good and bad initializations).

### 4.1.8 Deep Residual Learning

It allows some layers to bypass the next layer. This allows the network to skip layers if these are not necessary for the precition. Empirical experiments show a decrease from 7.4 error rate (with 34 layers) to 5.7 (with 152 layers).

Some additional information is described in "Tradeoffs of Large Sclae Learning, Bottou & Bousquet, 2011".

### 4.1.9 Weakly supervised pretraining

Some recent work on using weakly labeled images from the internet in order to pretrain deep neural networks.

Learning Visual Features from Large Weakly Supervised Data [24]

## 4.2 Unsupervised Learning

Learning without labels the intrinsic structure of the data. This is practically important as the real-world categories follow a Zipf's law, meaning that lots of categories appear very few times.

Some of the benefitrs of unsupervised learning are that there is a vast amount of free data available, and it is potentially useful as a regularisation method.

The basic idea is to be able to build a model of $p(X)$ (density modeling) given just data $\{X\}$.

### 4.2.1 Self supervised learning

This method consists on using the input data as some part of the target data

$$y : X \to Y \tag{4.8}$$

$$x \to y(x) \tag{4.9}$$

and use the same techniques used in supervised learning to train a model

$$\arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} l(f_\theta(x_i), y(x_i)) \tag{4.10}$$

This approach usually requires some knowledge of the domain. Some examples are word2vec in wich the word in the central position of a sentence is considered the target prediction.

### 4.2.2 Auto-Encoder

This method tries to find a hidden representation that keeps as much information of the input data as possible by reducing the reconstruction error. The network is divided between a decoder and a encoder part.

### 4.2.3 Variational Auto-Encoder

Similar to the simple autoencoder but makes

### 4.2.4 Generative Adversarial Networks

This consists in a decoder-only model but that uses an advesarial loss term using a discriminator $D$. Mini-max game between G and D.

### 4.2.5 Stacked Auto-Encoders

The Ladder Networks (Rasmus et al. 2015) adds a reconstruction constrain at every layer. In this way, there is a loss between every layer that can be minimized. This is like being able to generate hidden variable states apart from the input data.

### 4.2.6 Other approaches

Autoencoders, restricted / Deep Boltzmann Machines, ...

### 4.2.7 Application examples

Stochastic video generation (Denton and Fergus 2018) tries to predict the following frames from a video using encoders, decoders and Long Short Term Memories. The underlaying method is a Variational Auto-Encoder.

An application example is shown with synthetically generated videos using MNIST dataset and two bouncing numbers in a closed square. The task of the model is to predict the position of the numbers in the future (pixel by pixel?). It is possible to sample from the learned distribution, possibly generating an un/certainty heat-map.

Training a generative model to add color to images from just their luminance levels.

Training a generative model to predict the word in the middle of a sentence.

Cut an image into different sections (3x3 squares?) and train a model to predict to which part of the image every pice belongs to. Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles [33].

Automatic learning of images and audio from videos [34]. This method shows how it is possible to know the source of a sound from an image, some examples show people talking and rainfalls.

Map every image into a hypersphere [6], hopefully similar images will fall into nearby regions?

Unsupervised learning of visual representations using videos [46]

## 4.3 Deep Learning Models Rob Fergus 9:30–11:00

Most of the deep learning models are exploiting some structural bias of the data. For example, images have a 2D correlation pixelwise, while speech recognition and natural language processing exploits a 1D time correlation.

### 4.3.1 Memory in Deep networks

Deep neural networks are not able to store memory to solve sequential problems. Some networks are able to store implicit internal memory on their connections like Recurrent Neural Networks and Long Short Term Memory networks.

#### Implicit internal memory

Recurrent neural networks have the computational and the memory integrated in their weights. However, one of the main problems of RNNs is how to prevent the network from forgetting during the

training. In order to solve that problem Mikolov et al. 2014 separated the state into a fast and slow changing part. Other approaches are by using gated units for the internal state like Long-short term memory (LSTM) (see Hochreiter & Schmidhuber 1997, Graves, 2013), or Simplified minimal gated unit variations for RNNs. Also GRU light-weight version from Cho et al 2014.

RNN search: attention in machine translation [3] investigates an encoder and decoder model in a RNN, similar method was used later for image caption generation with attention [47], this last method allows the network to be queried and create heatmaps on the original images to correlate the generated words with their corresponding patch.

**External Global Memory**

Some work tries to split the computational memory from the problem solving memory aspect. These two parts are called the *memory module* and the *controller module*. The controller is able to write and read to the memory module. The memory needs some addressing mechanism, and can be soft or hard. The benefit of soft addressing is that it can be trained by backpropagation, while the hard addressing is not differentiable.

The end-to-end memory network (MemN2N) [43] is an architecture. The memory module is able to store some input vectors and hidden states, the controller module is able to search in the memory for similar patterns and obtain the corresponding associated output. This can be a new hidden state that will be combined with the current hidden state of the controler.

1. Embed each word (word to vector?)
2. Sum embedding vectors ($v_{Sam} + v_{drops} + v_{apple} = m_i$)
3. It generates one memory per sentence
4. The controller takes the vector encoding of the question and makes a dots product with the memory sentences and applies a softmax.
5. This guives weights to each sentence it computes a weighted sum of the sentences' representations.
6. The representation is guiven to the controller and generate an answer from it.

Another example is the Neural Turing Machine [20] in which the authors design an end-to-end diferentiable architecture that may be trained by backpropagation to solve tasks involving memory. Some example applications are coping, sorting and associative recall of inputs with outputs.

## 4.4 Deep Nets for sets

There ar problems where there is permuation invariance, dynamic sizong, single output, or output for each element.

In the Communication Neural Network (CommNet) [42] the inputs and outputs are sets, another example is the DeepSets [48]. The CommNet is a special case of a Graph NN in which a set can be represented as a complete graph.

# Chapter 5

# Statistical machine learning and convex optimization by Francis Bach

## 5.1 Introduction 14:30–18:00

We will use $n$ for the number of observations and $d$ for de dimension size. When working with big data we will seek ideally for a running-time complexity of $O(dn)$.

## 5.2 Classical methods for convex optimization

In supervised learning we have a set of observations $(x_i, y_i) \in X, Y, i = 1, \ldots, n$, i.i.d. (this assumption is almost never true).

For regression $y \in \mathbb{R}$ and prediction $\hat{y} = \theta^T \Phi(x)$, for classification $y \in -1, 1$ and the prediction is $\hat{y} = \text{sign}(\theta^T \Phi(x))$.

Empirical risk minimization to find a $\hat{\theta}$ solution to

$$\arg\min_{\theta \in \mathbb{R}^d} 1/n \sum_{i=1}^{n} l(y_i, \theta^T \Phi(x_i)) + \mu \Omega(\theta) \tag{5.1}$$

the training cost (or empirical risk) is

$$\hat{f}(\theta) = 1/n \sum_{i=1}^{n} l(y_i, \theta^T \Phi(x_i)) \tag{5.2}$$

and the testing cost or expected risk is

$$\hat{f}(\theta) = \mathop{\mathbb{E}}_{(x,y)} l(y_i, \theta^T \Phi(x_i)) \tag{5.3}$$

This imposes two fundamental questions: (1) how are we finding the optimal set of parameters $\theta$ and (2) how to analyse $\hat{\theta}$.

The loss for a single observation is $f_i(\theta) = l(y_i, \dots$ TODO missing

Jensen inequality says that $g(\mathbb{E}(\theta) \leq \mathbb{E}\, g(\theta)$

The global definition of convexity if we assume differrntiability is

$$\forall \theta_1, \theta_2, g(\theta_1) \leq g(\theta_2) + g'(\theta_2)^T(\theta_1 - \theta_2) \tag{5.4}$$

With twice differentiable functions $\forall \theta, g''(\theta) \succeq 0$ (positive semi-definite Hessians).

We ideally want a convex problem because the local minimum is the same as the global minimum, with an optimal condition in $g'(\theta) = 0$. See also the convex duality and recognizing convex problems in [8].

### 5.2.1 Lipschitz continuity

Bounded gradients of g (lipschitz-continuity) if the function g is convex, differentiable and has a

### 5.2.2 Smoothness and strong convexity

**Theorem 5.1.** *A function $g : \mathbb{R}^d \to \mathbb{R}$ is* L-smooth *if and only if it is differentiable and its gradient is L-Lipschitz-continuous*

$$\forall \theta_1, \theta_2 \in \overset{d}{\mathbb{R}}, ||g'(\theta_1) - g'(\theta_2)||_2 \leq L||\theta_1 - \theta_2||_2 \tag{5.5}$$

Adding a regularization $\mu/2||\theta||^2$ with a value of $\mu$ on the order of $1/n$.

### 5.2.3 Analysis of empirical risk minimization

most important slide from the first part is summarized in slide 47

### 5.2.4 Accelerated gradient methods (Nesterov, 1983)

Algorithm

$$\theta_t = \eta_{t-1} - \frac{1}{L} g'(\eta_{t-1}) \tag{5.6}$$

$$\eta_t = \theta_t + \frac{t-1}{t+2}(\theta_t - \theta_{t-1}) \tag{5.7}$$

with bound

$$g(\theta_t) - g(\theta_*) \leq \frac{2L||\theta_0 - \theta_*||^2}{(t+1)^2} \tag{5.8}$$

### 5.2.5 Optimization fro sparsity-inducing norms

See Bach, Jenatton, Mairal, and Obozinski, 2012b [2].

**Newton method**

Minimize the second-order Taylor expansion

### 5.2.6 Summary about minimization of convex functions

- Gradient descent: $\theta_t = \theta_{t-1} - \gamma_t g'(\theta_{t-1})$
  - $O(1/\sqrt{t})$ convergence rate for non-smooth convex functions
  - $O(1/t)$ convergence rate for smooth convex functions
  - $O(e^{-pt})$ convergence rate for strongly smooth convex functions
- Newton method: $\theta_t = \theta_{t-1} - g''(\theta_{t-1})^{-1} g'(\theta_{t-1})$
  - $O(e^{-p2^t})$ convergence rate
- Key insights from Bottou and Bousquet (2008)
  - In machine learning, it is not necessary to optimize below statistical error
  - In machine learning the cost functions are averages
  - Testing errors are more important than training errors

## 5.3 Convex stochastic approximation

There are known global minimax? rates of convergence for non-smooth problems (Nemirovsky and Yudin, 1983; Agarwal et al., 2012).

- Least-squares regression is easy to analyze, and has an explicit relationship to bias/variance trade-offs (see Défossez and Bach (2015); Dieuleveut et al. (2016).
- Many important loss functions are not quadratic (see Bach and Moulines (2013)).

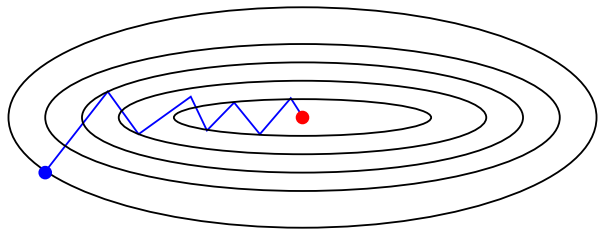

Figure 5.1: Batch Gradient Descent: it needs to decrease the error on every step, if not, then the parameters are not right

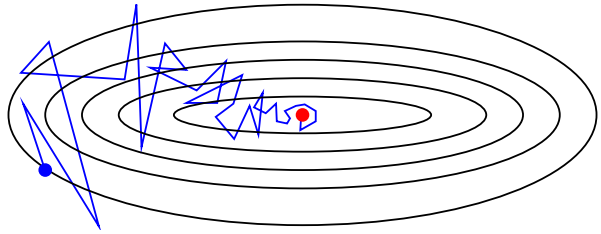

Figure 5.2: Stochastic Gradient Descent: the error may increase ocasionally

## 5.4 Summary of rates of convergence

Given the problem parameters

- $D$: diameter of the domain
- $B$ Lipschitz-constant
- $L$ smoothnesss constant
- $\mu$ strong convexity constant

| | **convex** | **strongly convex** |
|---|---|---|
| nonsmooth | deterministic: $BD/\sqrt{t}$<br>stochastic: $BD/\sqrt{n}$ | deterministic: $B^2/(t\mu)$<br>stochastic: $B^2/(n\mu)$ |
| smooth | deterministic: $LD^2/t^2$<br>stochastic: $LD^2/\sqrt{n}$<br>finite sum: $n/t$ | deterministic: $\exp(-t\sqrt{\mu/L})$<br>stochastic: $L/(n\mu)$<br>finite sum: $\exp(-t/(n+L/\mu))$ |
| quadratic | deterministic: $LD^2/t^2$<br>stochastic: $d/n + LD^2/n$ | deterministic: $\exp(-t\sqrt{\mu/L})$<br>stochastic: $d/n + LD^2/n$ |

Table 5.1: Summary of rates of convergence

## 5.5 Conclusions

- Statistics with our without optimization
  - Significance of mixing algorithms with analysis
  - Benefits of mixing algorithms with analysis
- Open problems

– Non-parametric stochastic approximation (Dieuleveut and Bach, 2014)
– Characterization of implicit regularization of online methods item Structured prediction
– Going beyond a single pass over the data (/testint performance)
– Parallel and distributed optimization
– Non-convex optimization (Reddi et al., 2016)

# Chapter 6

# Supervised Learning and Text Classification by Kyunghyun Cho

## 6.1 Introduction to supervised learning with ANNs Wed. 11:30–13:00

An overview on supervised learning, in which the input is described as a validation set $D_{val}$ and a test set $D_{test}$

At the end we need to choose the hypothesis that best adjusts to our problem between the availables. Examples of hyperparameters sets are in SVMs the regularization parameters and the kernel, simiarly with Gaussian Processes with the kernel and the parameters $\sigma^2$ and *lenght-scale*.

In Artificial Neural Networks the hypothesis set is the set of architectures, and hyperparameters. The architecture is commonly an directed acyclic graph (DAG) with parameters, inputs, outputs and compute nodes (functions that are often differentiable).

An example of architecture is a logistic regression where

$$p_\theta(y = 1|x) = \sigma(w^T x + b = \frac{1}{1 + \exp(-w^T x - b)} \tag{6.1}$$

or a 3rd-order polynomial function.

The inference is done by forward propagation of the input to the output trhough all the hidden layers.

Supervised learning tries to find a function $f_\theta(x)$, while in the case of Neural Networks we can usually interpret it as computeing the conditional probabilities guiven the input $p(y =' |x)$. This is achieved by computing any arbitrary output values from a network, then exponentiating every individual prefdiction and dividing them by their sum (soft-max function).

The objective is that the training data is maximally maximized, by ensuring that the maximum individual probabilities is maximized at the same time.

$$\arg\max_\theta \log p_\theta(D) = \arg\max_\theta \sum_{n=1}^{N} \log p_\theta(y_n|x_n) \tag{6.2}$$

We can also use the log-likelihood

$$\text{Missing equation} \tag{6.3}$$

### 6.1.1 Loss minimization

In order to minimize the loss it is necessary to use optimization techniques, in the case of Artificial Neural Networks (ANNs) is by using backpropagation to compute the partial derivatives of the parameters with respect the input using the chain rule of derivatives

$$\frac{\partial(fog)}{\partial x} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x} \tag{6.4}$$

This differentiation is done automatically by autograd which will implement the Jacobian-vector product of each P node.

By doing backpropagation we obtain the gradient of the loss with respect to every parameter $\theta$. Instead of computing the gradient for the full dataset, it is common to use mini-batches with a method called Stochastic Gradient Descent.

1. Random subset of M training examples
2. Compute the minibatch gradient

$$\nabla L \sim 1/N' \sum_{n=1}^{N'} \text{missing part} \tag{6.5}$$

3. Update the paraemters missing equation
4. repeat until the validation loss stops improving

However, this method will find a local minima in the training set, in order not to overfit to the training data, a validation set is used to do an early stop. This is one of the most important parts of the training as we want the minima to be as near as possible to the dataset near.

## 6.2 Text classification

In text classificatio the input are sentences or paragraphs and the output is a category to which the input belongs to (commonly a fixed number of $C$ categories).

Some of the particularities of using sentences as imputs is that they are of variable size $X = (x_1, x_2, \ldots, x_T)$ where every $x_t$ is a token from a vocabulary $V$. This is done by automatically (or manually) splitting all the individual words and creating an index of words that will form our vocabulary $V$. Then every sentence is encoded by a sequence of integers. In this case we are not tokenizing the words first by extracting any meaning of the word, in this sense the words "cat" and "cats" have completly independent indices.

At the end the words are encoded as one-hot-encoding with a bit to 1 for the corresponding index. This will be given to the ANN and will be fordard propagated to obtain a hidden representation $e_i$.

This can be interpreted as a table lookup from a word to a hidden embedding (eg. a fixed matrix $W$ of dimmension #tokens $\times$ #arbitrary dimesion that is multiplied by the binary token).

The representation of a sentence is a continuos bag-of-words, this means that we ignore the order of the words in the sentence, and average the corresponding token vectors. The method has been proven to be very useful in the works Iyyer2016, cho2017, and in FastText by Bojanowski2017.

### 6.2.1 How to represent a sentence

When the order of the words is important, it is possible to encode every set of words. For example, Relation Network: skip bigrams consider all the possible pairs of tokens and averages all the relationship vectors. The relations between words can be encoded depending on the problem, we could consider that the order of the bigrams is not important, only interested on bigrams of contiguous words, all the possible pairs, or skip some words.

With the previous idea is possible to use Convolutional Neural Networks in order to convolve the the "look up table" trough the input [26].

The CNN can also represent the bigram representations, and it is possible to learn a weight vector that will choose how many words are important. Look at the relation between Relational Networks and Convolutional Neural Networks.

#### Self-attention 11:30–13:00

Self-attention is a generalization of a CNN and RN that is able to capture long-range dependecies with a single layer. It can also ignore irrelevant long-term dependencies. Also mention generalization with multi-head and multi-hop attention.

Using a Recurrent Neural Network we can create an online representation of a sentence by reading every word and storing their representation in the recursive hidden representation. This allows a cost of $O(T)$ instead of the $O(T^2)$ necessary to do all the word pairs.

The representation that is generated by the RNN can encode the representation of a region of the text given it's context. This can then be feeded to the previously seen atention model that can learn the weighted sum of the context.

In "Fully character-level neural machine translation without explicit segmentation" [30] the authors stack a RNN on top of CNNs.

## 6.3 Natural Language Models

### 6.3.1 Autoregressive language modelling

The autoregressive sequence modelling assumes that the past tokens influence the current token as

$$p(X) = p(x_1)p(x_2|x_1)\cdots p(x_T|x_1,\ldots,x_{T-1}) \tag{6.6}$$

this holds true given the conditional distribution assumption.

With this method, an unsupervised method is transormed in $T$ smaller supervised problems.

One think to keep in mind from a question that was asked is that altough the marginalisation of $p(X)$ should sum to one, in real cases with RNNs this is not always true. Possibly because of the parametrization.

The autoregressive sequence modelling can be represented as

$$p(X) = \prod_{t=1}^{T} p(x_t | x_{<t}) \tag{6.7}$$

In order to score a sentence we can compare the output of the Autoregressive model for several sentences and use softmax to obtain "probabilities" (values between 0 and 1 that sum to one) for each sentence.

### 6.3.2 N-Gram language models

Before ANNs were used, the idea about ussing the conditional probabilities was already used on a smaller scale with N-gram models. In this case the $N$ needs to be decided in advance.

$$p(x|x_{-N}, x_{-N+1}, \ldots, x_{-1}) = \frac{p(x_{-N}, x_{-N+1}, \ldots, x_{-1}, x)}{\sum_{x \in V} p(x_{-N}, x_{-N+1}, \ldots, x_{-1}, x)} \tag{6.8}$$

The process then is to get the dataset and count the frequencies of every occurrence of the tokens $\ldots, x_{N-1}$ followed by all the possible tokens $x$.

There are two main issues that araise by ussing frequentist N-Grams

1. Data sparsity: lack of generalization
    - If using a 3-gram model you find 3 words that never happened again, the product of probabilities will be zero independently on the rest.
    - One possible solution is to use smoothing by adding a small constant
    - Another solution is to try with all $n \in \{N, \ldots, 1\}$
2. Inability to capture long-term depenfdencies
    - By choosing a fixed $N$ we may lose long term dependencies.

### 6.3.3 Neural N-Gram Language Model

[4] solved some of the previous issues by using the hidden representation of a Neural Network instead of the tokens. In the continuous vector space the similar tokens or phrases are nearby.

Some other work on the same direction is [32], , [36], [29].

An example of this application is the generalization of sentences that never happened by realizing that some words share some similarities (eg. numbers). An example was shown in which sentences with the 2-grams "three teams", "four teams" and "four groups" are able to generalise to the bigram "three groups" by realizing that three and four share a similar continuous space, and that before groups there could be a number.

In practice

1. Collect TODO missing steps

### 6.3.4 Convolutional Language Models

Convolutional Neural Networks allow to extend the context size by applying the convolution through larger parts of the text (see kalchbrenner et al, 2015 and Dauphin et al 2016, ByteNet by [25], PixelCNN, WaveNet, ...)

Gated convolutional language model by Dauphin 2016 [11]

### 6.3.5 CBoW Language Models (infinite context)

The idea is similar to the LM of using averages instead of concatenation.

### 6.3.6 Recurrent Language Models

An RNN can summarize all the tokens seen until $x$ into a continuous vector representation Mikolov et al, 2010 [31].

### 6.3.7 Recurrent Memory Networks

The work of Tran et al., 2016 [45] combines RNNs to compress the context into a continuous vector representation with the attention model that learns the weighting of the context.

## 6.4 Recurrent Networks and Backpropagation

Consider the full path from a parameter $\theta$ to the loss $l$. The backpropagation consists on computing the gradient of the $l$ with respect to the previous node and multiply by the Jacobian matrix of every step back to the parameter $\theta$.

$$\underset{h^{t+1}}{\overset{h^t}{\mathbf{Jac}}} = W^T \text{diag}(\tanh'(a^t)) \tag{6.9}$$

Because the Jaciobians are multiplied by every backpropagation step, this means that

- If $W > 1$ (the upperbound of ) the norm blows up, exploding gradient
- If $W < 1$ (the upperbound of) the norm shrinks to zero, vanishing gradient

$$|| \prod_{t'=t} || \leq || \mathbf{Jac} || \text{TODO missing part} \tag{6.10}$$

### 6.4.1 Gated recurrent units

These type of networks allow to skip some of the paths in order to avoid the exploding or vanishing gradient.

- Adaptive shortcut:
- Candidate update + pruning

- Update gate: $u_t = \sigma(W_u h_{t-1} + U \cdots$
- reset gate $r_t = \sigma(W_r h_{t-1} + U_r x + b_r)$

### 6.4.2 Lessons from GRU/LSTM

- Credit assignment over a long path of computation is difficult
- Adaptive shorcut or skip-connection helps avoid credit dilution
- Gates are an effective way to focus credit assignment

## 6.5 Neural Machine Translation 14:30–16:00

### 6.5.1 History of machine translation

The original idea was to get a text from one language, (1) perform a morphological analysis, (2) a syntactic analysis, (3) semantic analsysis, (4) a semantic composition and obtein an interlingua text that can be transformed back to any other language Borr, Hovy and Levin 2006 **TODO missing citation**.

Allen 1987 ieee icnn, a brief resurrection of Neural Networks in 1997 by Castano and Casacuberta 1997 [9], then in 2006 Schwenk 2006 [40] as a filter source to SMT to ANN to target sentence, then Devlin 2014 from source to SMT + ANN to target sentence, then source to ANN to target sentence.

### 6.5.2 Encoding: Token representation

First it is necessary to build a source and target vocabulary of unique tokens (for each language). Then transform the text into the set of tokens. Then encode the token sentences into sentence representation vectors, being careful not to compress the sentences into small vectors that may lose useful information.

### 6.5.3 Decoding: conditional language modeling

Using autoregressive networks we are interested in predicting the posterior probability

$$p(Y|X) = \prod_{t=1}^{T} p(y_t|y_{t-1}, X) \tag{6.11}$$

Look at the RNN Neural Machine Translation by Bahdanau et al., 2015 [3]. The model uses the target sentence (and what has been translated until the current moment) in order to generate the following prediction.

1. Encode: read the entire source sentence to know what to translate
2. Attention:
3. Decode:
4. Repeat 2 to 3 until the end-of-sentence (token) is achieved

This method achieved performance as good as the current state-of-the-art alternative at the moment phrase-based machine translation (PBMT).

At translation time every predicted word of the sentence had an associated vector of weights that indicted the source words involved, altough the model was trained only with pairs of text without any extra supervision. We should consider that every token of the source sentence is at the same time associated with a context in its language (see [23]).

### 6.5.4 In practice

Available frameworks

- Nematus [41]
- OpenNMT-py [27]
- FairSeq [19]
- Sockeye [22]

## 6.6 Current and ongoing projects

Multilingual translation, real-time translation, and character level translation.

### 6.6.1 Multilingual translation

A common approach has been to use a pivot language to translate languages without lots of examples. For example, translate korean to Japanese and then to English as the corpus between them is larger than the Korean to English.

With this idea, there has been some work to generate a pivot common language that is automatically learned in an ANN. For example, an encoder/decoder aproach in [16, 17] creates one encoder per source and decoder per target, and a model is learned for every pair. Later Johnson et al, 2016, Ha et al, 2016, Lee et al 2017, and Gu et al, 2018 work on an Universal ...

A current limitation is these methods drastically depend on the different amount of available data in each language. The models start ignoring the less frequent language. However, if given the same proportions of samples for every language they may overfit to the less frequent one because of the multiple epochs run on one language compared to the other.

Some work to solve this problem is being done with Meta-Learning MAML in [15]. The difference with multitask learning is that ...

Another idea is to create the unsupervised lookup tables to convert word tokens into continuous vectors ussing big corpus of different languages. In this manner, words with simmilar meanings will fall into similar regions (see [1]). In this case, the overfiting is less probable to happen as the lookup table is only learning a mapping of tokens to a continuoius space, but not the translation. Then, the translation model is trained on top of it.

### 6.6.2 Real-Time Translation (learning to decode)

learning to translate in real-time with neural machine translation [21]

In order to generate the best translation it is possible to generate several and then select the most probable one. However, ocasionally it may happen that after several words are translated the topic changes and the best translation may be a different one. In order to solve this the Beam Search keeps track of parallel best translations and is able to switch to another one at any point if necessary.

**Exploiting the hidden activation**
In a Deep Neural Network huge information in the hidden layers is usually discarded in order to predict a class; obtaining at the end a binary prediction. However, the hidden information has rich information about the original input that can be exploited.

By using the hidden information the performance of several methods was increased [21].

# Chapter 7

# Causality by Joris Mooij

There are many questions in science that are casual (eg. climatology, healthcare, ...).

## 7.1 Introduction

Causation is not correlation (gives example with chocolate consumption being correlated with number of nobel prizes in different countries, while there is no actual correlation between both).

In order to represent causal relations we can use *causal graphs* (directed graphs) in wich nodes are variables $X_n$ from (a vocabulary?) $V$. While directed edges indicate that the first variable causes direclty another variable respect to $V$. An example of a cyclic graph where every adjacent variable is directly connected to its neighbours is the encoding of standing domino pieces.

It is possible to modify the *causal graph* with an *intervention*, in the example the domino piece $X_2$ is glued to the floor, removing the direct connections from other pieces to $X_2$, but possibly keeping a connection from $X_2$ to the adjacent (in case an external user can still force $X_2$ to fall).

A *perfect ("surgical", "atomic") intervention* on a set of variables $X \subseteq V$, is an external enforced change of the system (eg. the previous example).

A *confounder* is a latent common cause:

> $H$ is a confounder of $X$ and $Y$ if:
>
> 1. $H$ causes $X$ directly w.r.t. $\{X, Y, H\}$
> 2. $H$ causes $Y$ directly w.r.t. $\{X, Y, H\}$

We will denote latent confounders by *bidirected edges* in a causal graph.

## 7.2 Defining causality in terms of probabilities

*Simpson's paradox* shows that if we interpret the probabilities as causes we may make wrong decisions. As an example, shows the recovery rate of a drug test in which depending on the groups separation

the most probable outcome changes.

1. The probability of recovery is higher for patients that took the drug

$$p(\text{recovery}|\text{drug}) > p(\text{recovery}|\text{no drug}) \tag{7.1}$$

2. For both male and female patients the relation was oposite

$$p(\text{recovery}|\text{drug, male}) < p(\text{recovery}|\text{no drug, male}) \tag{7.2}$$
$$p(\text{recovery}|\text{drug, female}) < p(\text{recovery}|\text{no drug, female}) \tag{7.3}$$

*endogenous variables* are binary variables that we are interested in.

*Exogenous variables* are latent, independent binary variables that affect externally the state of our endogeneous variables.

A *Structureal Causal Model (SCM)*, also knwon as *Structural Equation Model (SEM)*, is a tuple ...

There is one Structural equations per endogenous variable.

1. a product of standard measures ...
2. a product of standard measures ...
3. Measurable mapping ...
4. A product probbility measure...

An augmented functional graph $G^a(M)$ depicts the exogenous variables while the functional graph $G(M)$ doesn't.

If $M$ has no *self-loops*, the causal graph of $M$ is a subgraph of the functional graph $G(M)$.

**Definition 7.1.** We call the family of sets of probability distributions of the solutions of $M_{do(l,E_l)}$ ...

some of the previous points are teh basic difference between causal models and probabilistic models.

We donote a marginalization of the model $M$ with respect two variables $X_2$ and $X_4$ as $M^{\setminus\{2,4\}}$.

See the following extra references [12], and [7], [5], Bongers et al., 2018.

**Definition 7.2.** Definitions of: Independence and conditional independence

**Definition 7.3.** Definition of: nodes blocking a path

**Theorem 7.4.** *For an* acyclic *SCM, ...*

*Reichenbach's* principle of common cause, the dependence $X|Y$

The Reichenbach's Principe may fail in case of *selection bias* (related with the explaining away problem)

## 7.3 Causal Inference: Predicting Causal Effects

**Theorem 7.5.** *Back-Door Criterion [35]*

## 7.4 Resolving Simpson's paradox

It is important to realize that "seing" is not the same as "doing".

- $p(R = 1|D = 1)$: the probability that somebody recovers, given the observation that the person took the drug.
- $p(R = 1|\text{do}(D = 1))$: the probability that somebody recovers, if we force the person to take the drug.

In practice randomized control trial

> A **randomized controlled trial** (or randomized control trial;[2] RCT) is a type of scientific (often medical) experiment which aims to reduce bias when testing a new treatment. The people participating in the trial are randomly allocated to either the group receiving the treatment under investigation or to a group receiving standard treatment (or placebo treatment) as the control. Randomization minimises selection bias and the different comparison groups allow the researchers to determine any effects of the treatment when compared with the no treatment (control) group, while other variables are kept constant. The RCT is often considered the gold standard for a clinical trial. RCTs are often used to test the efficacy or effectiveness of various types of medical intervention and may provide information about adverse effects, such as drug reactions. Random assignment of intervention is done after subjects have been assessed for eligibility and recruited, but before the intervention to be studied begins. – **Wikipedia**

## 7.5 Causal Discovery: from data to causal graph

Randomized controlled trials [18] are one solution to avoid previously seen problems.

1. Divide patients into two groups: treatment and control randomly
2. Patients with the treatment group are forced to take a drug, and patients in the group are forced to not take the drug (but to take a placebo instead): $D = C$
3. Estimating the causal effect of the drug now becomes a standard ...
4. ...

### 7.5.1 Local Causal Discovery (LCD)

Simple method that Joris Mooij likes

## 7.6 Practical aplication

It is possible to apply $k$-fold Cross-validation to the observational data and interventional data in order to estimate the test performance.

## 7.7 Conclusions

Additional readings: Causality: Models reasoning and inference, pearl 2000. Constraint-based causal discovery for non-linear

- Elements of causal inference, foundations and learning algorithms by Peters, Janzing, Scholkopf 2017
- Causation, prediction and search by spirtes, glymour, scheines 2000
- correlation and causation by Wrights 1921
- Causal inference in statistics: an overview by Pearl 2009
- Simpson's paradox: an anatomy by Pearl 1999
- Causality: models, reasoning and inference by Pearl 2000
- Theoretical aspects of cyclic structural causal models by Bongers, Peters, Scholkopf, Mooij 2018
- Markov properties for graphical models with cycles and latent variables by Forré, and Mooij 2017

Some possible aplications of Causal inference could be:

- Transfer learning
- Domain adaptation
- Reinforcement learning

What tools or framewors: **there are no tools yet**, R package for individual methods. Literature and the implementations are scattered, **necessary to unify!**.

# Chapter 8

# Reinforcement Learning by Jan Peters, Fri. 14:30–16:00

## 8.1 Optimal Control Systems

Data to model to value function to policy back then to data.

### 8.1.1 Markov Decision Problems

A stationary MDP is defined as

- a state space $s \in S$
- action space $a \in A$
- transition dynamics $P(s_{t+1}|s_t, a_t)$
- reward function $r(s, a)$
- initial state probabilities $\mu_0(s)$

The Markov property says that the transition dynamics depends only on the current time step.

$$P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \dots) = P(s_{t+1}|s_t, a_t) \tag{8.1}$$

### 8.1.2 Basic reinforcement learning loop

The objective is to maximize the expected long-term reward

$$J_\theta = \mathop{\mathbb{E}}_{\mu_0, P, \pi}[\sum_{t=1}^{T-1} \gamma^t r(s_t, a_t)] \tag{8.2}$$

The algorithmic description of the value iteration

- Init: $V_T^*(s) \leftarrow r_T(s), t = T$
  - Compute Q-Function for time step t (for each state action pair)

$$Q_t^*(s,a) = r_t(s,a) + \gamma \sum_{s'} P_t(s'|s,a) V_{t+1}^*(s') \tag{8.3}$$

  - Compute V-Function for time step t(for each state) (TODO, check if next equation is max or argmax)

$$V_t^*(s) = \max_a Q_t^*(s,a) \tag{8.4}$$

- Repeat: $t = t - 1$
- Until $t = 1$
- REturn: Optimal policy for each time step

$$\pi_t^*(s) = \arg\max_a Q_t^*(s,a) \tag{8.5}$$

The Bellman Equation (Bellman Principle of Optimality)

$$V^*(s) = \max_a (r(s,a) + \gamma \mathop{\mathbb{E}}_p [V^*(s')|s,a) \tag{8.6}$$

See Policy evaluation with temporal differences: a survey and comparison [10]
When the max is expensive is possible to use the *Policy Iteration* method:

1. Policy evaluation: Estimate quality of states (and actions) with current policy
2. Policy improvement: Improve policy by taking actions with the highest quality

### 8.1.3 Linear Quadratic Gaussian Systems

A Linear Quadratic Regulator (LQR) system is defined as

- state space $xin\,\mathbb{R}^n$
- action space $uin\,\mathbb{R}^m$
- linear transition dynamics with Gaussian noise

$$p_t(x_{t+1}|x_t, u_t) = \mathcal{N}(x_{t+1}|A_t x_t + B_t u_t + b_t, \Sigma_t) \tag{8.7}$$

- quadratic reward function

$$r_t(x,u) = (x - r_t)^T R_t (x - r_t) + u_t^T H_t u_t \tag{8.8}$$

$$r_t(x) = (x - r_T)^T R_T (x - r_T) \tag{8.9}$$

- initial state density

$$\mu_0(x)\mathcal{N}(x|\mu_0, \Sigma_0) \tag{8.10}$$

See Stefan Schaal, Christopher G. Atkeston 1998 [39]

The LQR systems need the initial point to be linearly related to the optimal point (eg. keeping a stick balanced upwards). However, they can not solve situations in which it is necessary a non-linear component (eg. if the stick starts from a hanging position, and it needs some sinuidal movement before it can reach the upward position).

See work by Emo Todorov and Yuval Tassa on Incremental LQG (a simplification of Differential Dynamic Programming by Dyer and McReynolds 1969 [13])

With Optimal control we can compute optimal policies but only on

1. Discrete Systems: (but world is not discrete)
2. Linear Systems, Quadratic Reward, Gaussian Noise (LQR): (but the world is not linear)
3. Along an optimal trajectory: (it is really hard to find)

For these reasons we need to approximate.

## 8.2 Value Function Methods

Data to value function to policy to data.

One of the principles is that "all models are wrong, but some are useful". In the previous section we created perfect models, but they may not be match the real world. In that case, there can be drastical problems.

The *Classical Reinforcement Learning* postulate is to solve the optimal control problem by learning the value function and not the model.

### 8.2.1 Markov Decision Processes (MDP)

Infinite Horizon with a discounted reward parameter $\gamma$

Updates the value function based on samples $D = \{s_i, a_i, r_i, s_i'\}$

### 8.2.2 Temporal difference learning

We are incorporating an error value into the prediction of the states (see Reinforcement learning, rich sutton and andy barto, 1998 [44].

Withe our estimate we can compute the TD error and make a decision

- if the estimation was higher, we decrease the prediction
- if the estimation was lower, we increase the prediction

1. Init $V_0^*(s) \leftarrow 0$
2. Repeat $t = t + 1$

- Observe transition $(s_t, a_t, r_t, s_{t+1})$
- Compute TD error $\delta_t = r_t + \gamma V_t(s_{t+1}) - V_t(s_t)$
- Update V-Function $V_{t+1}(s_t) = V_t(s_t) + \alpha\delta_t$

3. until convergence of $V$

But we do not want deterministic policies as these will not explore the space, for that reason there are at least two policy for exploration

1. Epsilon-Greedy Policy
2. Soft-Max Policy (it has an important temperature parameter $\beta$)

Update equations for learning the Q-function $Q(s, a)$

$$Q_{t+1}(s_t, a_t) = \dots \tag{8.11}$$

In wich it is necessary to estimate the future action $a_?$. There are two methods

1. Q-learning: $a_? = \arg\max_a Q_t(s_{t+1}, a)$
2. SARSA: $a_? = a_{t+1}$, where $a_{t+1} \sim \pi(a|s_{t+1})$

### 8.2.3 Approximating the value function

Instead of creating the matrix $V$ we can approximate with any function approximation method (see Dann et al: Policy evaluation with temporal differences: a survey and comparison, JMLR, 2014 [10]).

Some remarks on temporal difference learning

- It si not a proper stochastic gradient descent
- As the target values $V^\pi(s)$ change after each parameter update
- This "ignoreance" introduces a bias in our optimization
- ...

### 8.2.4 Batch-Mode Reinforcement Learning

Online methods are typically data-inefficient as they use only once every sample. The rehuse of the samples has been done in Least-squares temporal difference learning and fitted q-iteration (Tree-Based batch mode reinforcement learning [14], Batch reinforcement learning [28]).

In Q-iteration we do as in Value-iteration, but we use Supervised Learning methods to approximate the ? function.

See Reinforcement learning in robot soccer, 2009 [38].

## 8.3 Policy Search

From data to policy and back to data.

see Reinforcement learning of motor skills with policy gradients, 2008 [37].

### 8.3.1 Black-box approaches

- Perturb the parameters of your policy: $\delta J = J(\theta + \delta\theta) - J(\theta)$
- Approximate J by the first order Taylor approximation $J(\theta + \delta\theta) = J(\theta) + \frac{\partial J(\theta)}{\partial \theta}\delta\theta$
- Solve for $\frac{\partial J(\theta)}{\partial \theta}$ in a least squares sense (linear regresssion).

See a large classs of algorithms includes: Kiefer-Wolfowitch procedure, Robbins-Monroe, Simultaneous Perturbation Stochastic Approximation SPSA,...

### 8.3.2 Likelihood-Ratio Policy Gradient methods

The expected long term reward $J(\theta)$ can be weritten as expectation over the trajectory distribution.

## 8.4 Key problems

1. no notion of data in the generic problem formulation
2. optimization bias problematic with data
3. role of tfeatures is unclear in most methods