COMS30007 - Machine Learning

Lecture Summary

Carl Henrik Ek

September 29, 2018

Abstract

This document will try to summarise the key concepts, equations, variables, etc of each lecture during the unit. This will be a living document so make sure that you keep an up-to-date version as I'll be adding things

1 Machine Learning

The key material from this lecture is all philosophical, where we tried to convey "what is it that allows us to learn?" and "what has made the rapid progress in machine learning possible?". The key thing that we tried to convey was that in order to learn something you have to make assumptions. Laplace demon convey's the message that it is infeasible to think about truths for every system therefore we have to reason about the world, we have to make decisions based on beliefs. Machine learning is the science of doing just that and even though it was founded a long time ago it is just very resently that the field has become a main stream computer science topic. The reason behind this is that now, for the first time we are actually able to solve important problems. The motivation behind this is not what the romantic mind would assume, that we have made enormous strides in theoretical machine learning that has unlocked these possibilities, nope thats not it. What has allowed us to prosper is the amazing work done by people designing hardware that allows us to not have to be so clever, we can just compute a lot, and the people who made it feasible to store enourmous amounts of data. This development have allows us to apply our rather naive learning systems at scale and solve some quite impressive tasks. As a final note I mentioned the narrative of artificial intelligence, this field is contains a lot more than machine learning but it is machine learning that is responsible for its resurgence. AI has promised a lot and always failed, during the 80:ies there was a phase referred to as the `AI-winter` where people stoped believing as the promises never materialised. Personally I think it is important to not fall into the trap and listen to the public narrative about AI, this is anthropomorphism at its worst. If there is one thing that we need to fear it is the sentient human that controlls the stupid AI that will do whatever its master commands.

1.1 Equations

None really but Baye's rule was mentioned

2 Probabilities

If we agree with the idea that we do not deal with truths we have to accept the concept of learning in an uncertaint world. Traditional mathematics deals with truths and will therefore not be applicable as a language, the strand of mathematics that encapsulate uncertain statements is probability theory and this is what we talked about in lecture 2.

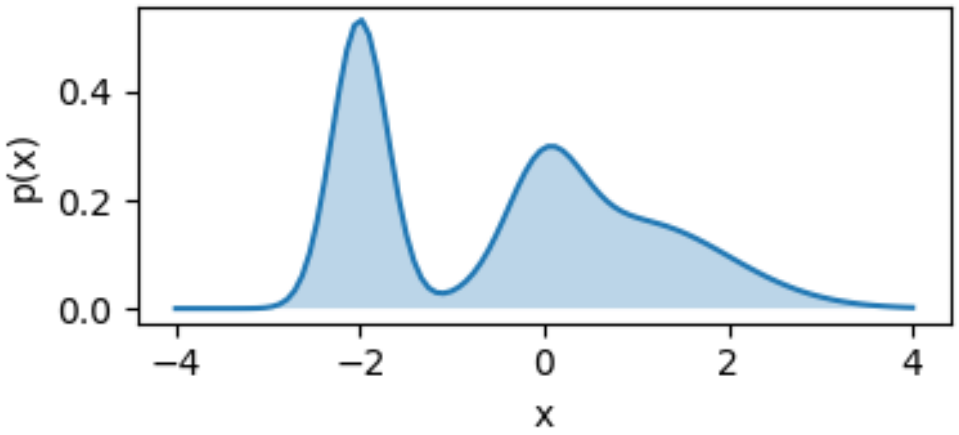

Figure 1: The above figure shows the distribution of the random variable x. The higher the probability is of a specific location the more likely the variable is to take that value.

2.1 Concepts

Random Variable A random variable is the outcome of some random phenomena, it is a variable that cannot be described by a value but rather a distribution that encodes "how likely" each value of the variable is. When we don't know something for certain we model them as random variables and therefore the system becomes stochastic rather than deterministic.

Bayesian Probabilities The rules of probabilities is just a language, a set of rules, but they do not mean anything they are like words without a semantic. The traditional way to see probabilities are as frequencies of events, this is what is called a frequentist approach. This is too limited when it comes to machine learning as if we want to say

something about something that we haven't observed, or about something that can only happen once ever (sun exploding). Therefore in a Bayesian setting we interpret a probability as a **belief** in a variable, and it is very plausible to have beliefs about something that hasn't happened. It is this interpretation as beliefs that allows us to place semantics onto the terms in Baye's rule as *posterior*, *likelihood*, *prior*, *marginal likelihood* or *evidence*.

Models A model is something that relates something to something else. As we are in the stochastic uncertain world this means a distribution over a variable (something) that is parametrised by another variable (something else). This means that it is a *conditional distribution*

Probability Mass Function when we are talking about systems that have a discrete outcome the function that is defined over the outcome space is called a *probability mass function*.

Probability Density Function when we have a continous random variable the function that specifies the probability over this variable is called a *probability density function*.

Probability Mass/Density Functions it is important to note, these are just like any function, and you can deal with them in the same way, the difference is just that they have the additional constraints

$$p(x) \geq 0, \quad \int_{-\infty}^{\infty} p(x)\mathrm{d}x = 1$$

2.2 Equations

- Distributions

  Joint $p(x, y)$

  Marginal $p(x)$, $p(y)$

  Conditional $p(y|x)$, $p(x|y)$

- Rules of Probability

  Sum $p(x) = \sum_y p(y, x)$, $p(x) = \int p(y, x)\mathrm{d}y$

  Product $p(x, y) = p(y|x)p(x)$

Often you will see Baye's rule mentioned as a "rule" of probability. I don't think it should be as it is just a consequence of the product rule. The interesting and important thing with Baye's is not the rule but the interpretation of what a probability is.

The rules above and the names of the distributions is something that you should know by heart, it is something that we need in order to keep the conversation on a good level.

Expectation The expectation of a variable is taking a an average of all possible values of the variable weighted according to their probability, i.e. in a Bayesian setting how likely I believe the variable is to take this value.

$$\mathbb{E}[x] = \sum_x p(x)x$$

Expectations We can also take expectations over probability distributions,

$$p(y) = \int p(y, x)\mathrm{d}x = \int p(y|x)p(x)\mathrm{d}x.$$

In the case above this means, what do I believe the function is over only y if I create a new function $p(y)$ which is a weighted average of all possible settings of the variable x.

Variance The expectations of how a variable varies around its expectations,

$$\mathrm{var}[x] = \mathbb{E}\left[(x - \mathbb{E}[x])^2\right]$$

Covariance The expectation on how two variables varies *together* in relation to their means,

$$\mathrm{cov}[x, y] = \mathbb{E}\left[(x - \mathbb{E}[x])(y - \mathbb{E}[y])\right]$$

3 Distributions

In the first lecture we tried to justify that truth is not particularly intereting and that we therefore need to deal with uncertainty and make assumptions. In the second lecture we therefore went ahead and looked at what the language/tools is for general probability distributions. In this lecture we will take a look at what form these probabilities take. If we are supposed to encode our assumptions into probabilities they have to somehow match my assumptions. We looked at several distributions, Bernoulli, Multinomial, Beta, Dirichlet, Gaussian, etc. The only one of them you should know by heart is the Gaussian, but you need to know that the others exists. The key concept in this lecture is the concept of conjugacy.

3.1 Concepts

Conjugate prior most commonly we will specify models in terms of a likelihood first. We have defined a generative process that can create our data now by defining the "error/noise" we believe we have in the observations we get a likelihood. So in the likelihood we will now have parameters controlling the generative process and these

are the ones that we want to reason about. So for a line, $y_i = \mathbf{w}^{\mathrm{T}}\mathbf{x}_i$ our likelihood is a distribution over y and the likelihood is parameterised by $\mathbf{w}$ and $\mathbf{x}$ as these are the things that is needed to generate the data. Now we want to make an assumption about $\mathbf{w}$ so that we have a prior and are able to "do" Baye's rule to get our updated assumption, i.e. learn from data. Now conjugacy becomes important, if we pick a prior such that the posterior becomes a function in the *same family* as the prior we call the prior *conjugate*. For an example of this look at page 13-14 in the lecture and also in the Bernoulli trial example write-up that is in the repo. If we have conjugacy we do not have to compute the denominator in Baye's Rule we can simply multiply likelihood and prior and then *identify* the parameters of the posterior distriubiton as we know its form. This is **really** useful.

Central limit theorem the average/sum of i.i.d. samples from **any** distribution will follow a Gaussian distribution. This is one of these magical things when you see if first and I do recommend testing it with some code. When you think a bit further it makes perfect sense. It can be very useful to motivate the use of Gaussians as *if I don't know which distribution something comes from, but I assume I get samples i.i.d. then I'll take an average* and this will be Gaussian.

Gaussian identities these are just the form of "all" the interesting Gaussian distributions. If I know the likelihood and the prior, what is the posterior, if I know the joint what is the marginal and the conditional. They are tedious to derive but I find seeing them very good practice as if you will develop a new model you are likely to have to do very similar computations. Importantly though, you do not have to know them, you only have to be able to use their result.

3.2 Equations

- The Gaussian distribution,

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

- The Gaussian distribution

$$p(x_1, x_2) = \mathcal{N}\left(\begin{array}{c} \mu_1 \\ \mu_2 \end{array}, \left[\begin{array}{cc} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{array} \right] \right)$$

- Gaussian Marginal

$$p(x_2) = \int p(x_1, x_2) \mathrm{d}x_1 = \mathcal{N}(\mu_2, \sigma_{22})$$

- Gaussian Conditional

$$p(x_1|x_2) = \frac{p(x_1, x_2)}{p(x_2)} = \mathcal{N}\left(\mu_1 + \sigma_{21}\sigma_{22}^{-1}(x_2 - \mu_2), \sigma_{11} - \sigma_{12}\sigma_{22}^{-1}\sigma_{21}\right)$$

4 Linear Regression

Now we know how to formulate our assumptions in terms of specific distributions so we have all the tools to build our first simple model and learn from data. The key ideas in this lecture is to take the simples possible model, a line in two-dimensions to test all our knowledge and really do proper machine learning on this task. The motivation for this is that machine learning often is a quite a hurdle to get through in the beginning if taught to depth and then at some point it says "click" and we get everything. Therefore really try to get the key concepts of this and you will find the rest of the unit easier. The key thing to get from this lecture is the modelling proceedure.

4.1 Concepts

Modelling when we have been given data, how do we got about learning, this proceedure is general and will work everywhere.

1. we define a model $t = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) + \epsilon$
2. our assumption of the type of noise, that it is addative and that it has a specific distribution leads to the likelihood
3. Now we need to specify our prior to reach the posterior
4. If the likelihood is Gaussian, and the parameter that we want the posterior over is in the mean, the conjugate prior is another Gaussian
5. Specify prior
6. Now pick the posterior distribution from the Gaussian identities
7. Job done!

Prediction when we have an updated posterior we want to see what this says about new data. What we want to do is see what is the *expected value* of the new data point under the posterior over the weights.

$$p(t_*|\mathbf{t}, \mathbf{x}_*, \mathbf{X}, \alpha, \beta) = \int p(t_*|\mathbf{x}_*, \mathbf{w}, \beta)p(\mathbf{w}|\mathbf{t}, \mathbf{X}, \alpha, \beta)\mathrm{d}\mathbf{w}$$

- t_* is the output value at location $\mathbf{x}_*$
- $\mathbf{x}_*$ is the location we are interested in evaluating the function
- $\mathbf{w}$ is the weights that multiplied with $\mathbf{x}$ gives the output
- α, β are parameters of the prior and the likelihood
- $\mathbf{X}$, $\mathbf{t}$ are the pairs of input-output training data pairs

So this formula looks scary but lets remove the variables that are not involved really,

$$p(t_*|\mathbf{x}_*) = \int p(t_*|\mathbf{x}_*, \mathbf{w})p(\mathbf{w})\mathrm{d}\mathbf{w} = \mathbb{E}_{p(\mathbf{w})}\left[p(t_*|\mathbf{x}_*, \mathbf{w})\right]$$

this is just a simple expectation, as we do not know $\mathbf{w}$ but we have a **belief** in its value we take the weighted average of *all* possible weights $\mathbf{w}$ and see what the value is at $\mathbf{x}_*$ and weigh this average by hom *likely* we think the specif $\mathbf{w}$ is. Makes sense?

5 Dual Linear Regression

In the previous lecture we did our first model, it was a rather simple one as it could only represent lines. We also did a showed that you can first mapped the data to another space and then do the regression there, if we then plotted the line in that space but along the original axises then it would for a non-linear mapping be non-linear. An example of this is shown in Figure 2, where we have data which is on a circle, now we want to represent this and clearly a line is not sufficient, but if we map the data to polar-coordinates now we can fit a simple line and when mapped back it looks like a circle. So clearly there is a good reason to pre-process the data and map it somewhere where we can solve it easier.

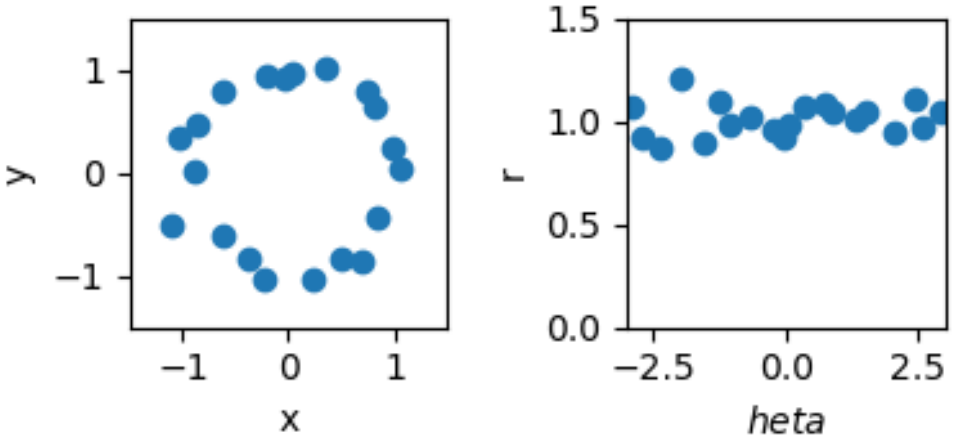

Figure 2: *On the left we have the original data which we cannot map a line to, on the right we have the same data but descibed in polar coordinates instead, where it is possible to model the data as a line.*

Lets make this more specific,

$$y = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{z},$$

where $\phi(\cdot)$ is a mapping from,

$$\phi : \mathcal{X} \to \mathcal{Z} \tag{1}$$
$$\mathbf{x} \in \mathcal{X} \tag{2}$$
$$\mathbf{z} \in \mathcal{Z}. \tag{3}$$

The question now is how should the mapping $\phi(\cdot)$ be choosen, what should the dimensionality be of the space that we map to? Ideally we would like to have a mode that adapts its complexity to the data, so if we just see one data-point we have a simpler model compared to the potential complexity of seeing hundreds of data-points. We can reach just such a model by performing the following steps,

Dual linear regression steps this derivation looks challenging and is quite long **but** you only need to know the results and have a vauge idea of the way we got there, such as this,

1. formulate posterior
2. find stationary point of posterior
3. re-write the variable $\mathbf{w}$ in terms of the data
4. $\rightarrow$ kernel regression

The important thing to understand about kernel regression is that rather than having a fixed set of parameters, we now do not have parameters in the traditional sense, instead when we predict we relate the new data to the data that we already have, based on their similarity we predict the point.n

5.1 Concepts

Kernels kernels are functions that define inner-products (dot-products) in "some" space. The key use of these is that often it is much easier to define the inner-product compared to what it is to define spaces. Further, they allow us to never have to realise the space, therefore we can work with things such as infinite dimensional spaces and put things such as non-vectorial data in a vector space. The spaces that the kernel defines are referred to as *induced* spaces.

Kernel Regression performin linear regression in an induced space, the problem is linear in the induced space but non-linear in the original space. This therefore allows us to learn non-linear functions using lines.

Duals very general idea, sometimes we can reformulate a problem in a different variable and work with this instead. Sometimes, as in the kernel regression case it is very beneficial.

5.2 Equations

Kernel a function that computes the inner-product in some induced space

$$k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^{\mathrm{T}} \phi(\mathbf{x}_j)$$

- $\phi(\cdot)$ is the feature mapping
- $\mathbf{x}$ is a vector in the original space

Kernel regression below is the formula that we derived to perform kernel regression for predicting a new data points $\mathbf{x}_*$

$$\mathbf{y}(\mathbf{x}_*) = k(\mathbf{x}_*, \mathbf{x})(\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{t}$$

- $\mathbf{y}_*(\mathbf{x}_*)$ is the output location of the previously unseen data-point $\mathbf{x}_*$
- $k(\mathbf{x}_*, \mathbf{x})$ is the kernel function evaluated between the test data input location $\mathbf{x}_*$ and the training data $\mathbf{x}$
- $\mathbf{K} = k(\mathbf{X}, \mathbf{X})$ the kernel function evaluate between all the training data points
- λ is a parameter, that is related to the noise we assume that we have in the data. It is unclear on purpose what it is, but code this up and test with different values. We will see in the next lecture how we can provide it with semantics.

6 Gaussian Processes

Now we have made a couple of steps towards more complicated models, our linear regression model is now capable of having adaptive complexity capable of generating non-linear functions. However, there are two things which makes the kernel regression model hard to use, first it is not really a model, it provides us with a function but no *uncertainty* measure around that function, this is bad. Secondly, it is not clear how to choose the type of kernel and how we should set the kernel parameters that controls the behaviour of the function. In this lecture we will look at the first concept and then in the second we will move on and sort out the second.

We know from our previous model that in order to get a measurment of the uncertainty in our prediction we need to make a prior assumption. In this case this means that we need to make a prior assumption over the space of functions. What is even the space of functions, what would the basis be? This is really strange to think of so instead we make an assumption of the *instantiation* of a function at a specific input location. In specific,

$$y_i = f(\mathbf{x}_i) + \epsilon = f_i + \epsilon \tag{4}$$

$$\epsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{I}), \tag{5}$$

where we have introduced a new random variable f_i which is the output of the function at point $\mathbf{x}_i$. This new random variable now provides us with a "handle" to specify our assumptions over, i.e. we want to specify, $p(f_i)$ or rather $p(f|\mathbf{x})$ where the input location just indexes the distribution. The assumption that is made by a Gaussian process is that every instantiation follows a Gaussian distribution, importantly, what this means is that the joint distribution of ever possible instantiation (which is infinite) is also a Gaussian distribution, this is what we call a *Gaussian process*. Now importantly, if each instantiation is a Gaussian, i.e. each slice along the x-axis and as a Gaussian never goes down to 0 we have a probability mass over every possible instatiation of the function. Further, as the input space itself, for a one dimensional case the real line, contains infinetly many values, and goes between $(-\infty, \infty)$ the Gaussian process actually specifies a probability distribution over the whole infinite xy-plane which is never zero. Think about what this means, are there functions with a zero probability?

If we agree with the above reasoning now the one step that remains is to actually specify the prior. First we need a mean, this we will in practice always assume to be zero, the interesting thing is what our covariance should be. As we need to be able to compute the covariance between all input locations this cannot be a value, rather it has to be a function that takes the input locations in and computes *how much each location along the input axis co-varies with each other* or *how much information about the function at one location can I infer from knowing the function value at another location.* We implement this using a covariance function $k(\mathbf{x}_i, \mathbf{x}_j)$. A simple choice which leads to smooth functions is to make the assumption that the closer things are in input space the more their output should covary, i.e. a smooth function. Below you can see the prior *evaluated* at two locations $\mathbf{x}_1$ and $\mathbf{x}_2$, but importantly it is over the whole function as we can take and input any location and can compute it for any number of outputs.

$$\begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu(\mathbf{x}_1) \\ \mu(\mathbf{x}_2) \end{bmatrix}, \begin{bmatrix} k(\mathbf{x}_1, \mathbf{1}_1) & k(\mathbf{x}_1, \mathbf{x}_2) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) \end{bmatrix}\right) \tag{6}$$

This is really it, GPs are very very useful as they allow for full Bayesian treatment of functions. Often they are a bit hard to get and I believe this is because they do something really exiting therefore we have a bias that they must somehow be advanced, they are not. Try to think through what we said above and then test to sample from a GP, it is three lines of code in python,

1. call the covariance function
2. draw samples from the resulting Gaussian
3. plot functions

6.1 Concept

Kernels the class of kernel functions are actually the same class of functions that generates covariances. Therefore kernel functions can also be thought of as covariance functions. One way of getting an intuitive idea about this is on Lecture slide 3, the idea that a kernel is actually the covariance between data-points not between data-dimensions. We can get to that explanation by making each of the data-points a dimension.

Process a process is the generalisation of a distribution such that one instantiation of the process is a distribution. Compare this with a distribution where the instantiation is a value.

6.2 Equations

GP Prior the Gaussian process prior

$$p(f|\mathbf{x}, \theta) = \mathcal{N}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$$

- f is the instantiation of the function at location $\mathbf{x}$
- θ is the parameters that controls the behaviour of the covariance function (more on that in lecture 7)

GP Predictive Posterior the Gaussian process posterior is trivial to derive, we know that all instantiations are jointly Gaussian so now we want to get a conditional distribution over a new data point given the training data in just the same way as we did with the normal Gaussian

$$\begin{aligned} p(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{f}, \boldsymbol{\theta}) = \mathcal{N}(&k(\mathbf{x}_*, \mathbf{X})^{\mathrm{T}} K(\mathbf{X}, \mathbf{X})^{-1}\mathbf{f}, \\ &k(\mathbf{x}_*, \mathbf{x}_*) - k(\mathbf{x}_*, \mathbf{X})^{\mathrm{T}} K(\mathbf{X}, \mathbf{X})^{-1} K(\mathbf{X}, \mathbf{x}_*)) \end{aligned} \tag{7}$$

7 Gaussian Processes and Unsupervised learning

In the previous lecture we looked at Gaussian processes as a means of specifying priors over the space of functions. With very simple means they allow us to introduce assumptions on something which is very hard to even think of "the space of functions". They are strange to think about and hard to get an understanding of, my advice is to actually implement them and try to sample from it. The code below does just for a simple squared exponential covariance. Implement this and test the parameters of the covariance, try to get what is really going on.

```
import numpy as np
import matplotlib.pyplot as plt
```

```
from scipy.spatial.distance import cdist

# create data
x = np.linspace(-5,5,200)
x = x.reshape(-1,1)
mu = np.zeros(x.shape)

# compute covariance matrix
K = np.exp(-cdist(x,x)**2/5)

# now we have the mean function and the covariance function
# the GP is fully described
f = np.random.multivariate_normal(mu.flatten(),K,10)

# plot the data
plt.plot(x,f.T)
plt.show()
```

Even though we now have the possibility of inserting our assumptions by setting the parameters of the GP, i.e. the parameters of the covariance and mean function, this can sometimes be really tricky. What we would ideally like to do is to infer them from data. From the unit so far we know how to do this, make an assumption, formulate the prior, get the posterior. However, for the covariance parameters this is sometimes rather challenging to do (you have to trust me that it is at this point) so instead we try and find a point estimate, i.e. a single value. We could do this by maximising the likelihood, however, we can do something better. It is possible to marginalise out our prior over f and then maximise the marginal likelihood instead. This way we have treated some of our assumptions with principle

$$p(\mathbf{Y}|\mathbf{X},\theta) = \int p(\mathbf{Y}|\mathbf{f})p(\mathbf{f}|\mathbf{X},\theta)\mathrm{d}f$$

We looked at the behaviour of the Type II maximum likelihood did, what you should notice here is that, the likelihood only cares about f being close to y it does not care what the function looks like at all. But as we have integrated out the function, our preference is there, and it chooses a function that is a balance between being close to the observed data and matching our assumption.

In the second part of the lecture we looked at unsupervised learning, what we want you to get from this is simply this,

1. unsupervised learning is a really ill-constrained problem
    - we have observed data and we want to find "another" representation of it

2. to proceed we have to make assumptions (same story again)

3. for the linear model we can specify a distribution over the representation that we want, i.e. the latent coordinates

4. we have two things to determine, both the weights of the mapping and the latent representation, treating them both in a Bayesian fashion and reaching the posterior is sadly intractable (you have to trust me on this here) therefore we do the next best thing, integrate out one of them and maximise the other. This is called a Type II maximum likelihood.

The key concept here is to see that this is just the same again, we specify assumptions and then we churn the handle. If you can relate the linear unsupervised model to the supervised model and see the similarities then you have gotten it and this is the message of this lecture.

7.1 Concepts

Type II Maximum Likelihood if we cannot marginalise out all variables, but we can some. Rather than performing maximum likelihood on all, we integrate out the ones we can and maximise for the remaining.

Latent variable a latent variable is a variable that is not observed it is latent in the process that generates what we can see.

General the key thing in this lecture is to see that we can make assumptions about anything and as long as we make them then we can do machine learning on them, however ill-conditioned the problem is. Our assumptions places structure on the possible solutions so it allows us to choose a sensible one.

7.2 Equations

PCA the formula below is the marginal likelihood of linear regression when we have integrated out the latent locations. We will work with this formula in the assignment to solve Question 19

$$\log p(\mathbf{X}|\boldsymbol{\mu}, \mathbf{W}, \sigma^2) = \sum_{n=1}^{N} \log p(\mathbf{x}_n|\mathbf{W}, \boldsymbol{\mu}, \sigma^2) \tag{8}$$

$$= -\frac{ND}{2}\log(2\pi) - \frac{N}{2}\log|\mathbf{W}\mathbf{W}^{\mathrm{T}} + \sigma^2\mathbf{I}| \tag{9}$$

$$- \frac{1}{2}\sum_{n=1}^{N}(\mathbf{x}_n - \boldsymbol{\mu})^{\mathrm{T}}(\mathbf{W}\mathbf{W}^{\mathrm{T}} + \sigma^2\mathbf{I})^{-1}(\mathbf{x}_n - \boldsymbol{\mu}) \tag{10}$$

7.3 Code

Here is some help code to plot a 2D gaussian distribution as a contour plot that might come in handy for the assignment.

```
# create multivariate distribution
pdf = multivariate_normal(mu,K)
# generate points along axis
x = np.linspace(-4,4,N)
y = np.linspace(-4,4,N)
# get all combinations of the points
x1p,x2p = np.meshgrid(x,y)
pos = np.vstack((x1p.flatten(),x2p.flatten()))
pos = pos.T
# evaluate pdf at points
Z = pdf.pdf(pos)
Z = Z.reshape(N,N)
# plot contours
pdf_c = ax.contour(x1p,x2p,Z,3,colors='k')
```

8 Bayesian Optimisation

The material in the lecture is intended to be understood on an abstract level, not at all in the same detail as the previous work. The intention is for you to see the benefits that we can reap by spending all this effort thinking of uncertainty and what we can do with "proper" models of the world. First lets think of decision that we have to take in different scenarios, you are trying to figure out what to study for an exam, you are designing a product to sell, etc. in all the scenarios you have no idea what the "right" thing is to do. However, you can try something and see what happens but each try is quite costly. Studying the wrong material for an exam, or building something that no one wants can in some ways be seen as failures. However, they are only failures if we do not learn from them and act differently in the future. This is how humans act and now we want to think about how we can make these things explicit and formulate mathematics that allows us to automate this process. This is as you can see something incredibly general and very useful if we could do. This is the focus of an emerging field of machine learning called Bayesian optimisation. It was first suggested 30-40 years ago but it is really taking off right now, if you ask me it is the most exciting thing that is happening in machine learning at the moment.

The setting for Bayesian optimisation is the following, we want to find the minima of a function $f(x)$,

$$x_M = \text{argmin}_{x \in \mathcal{X}} f(x)$$

Compared to standard optimisation, where by standard we mean the optimisations we are

taught in school, we do not have $f(x)$ written on explicit form, it is a black-box, we can evaluate it at a certain point but thats it. We cannot assume that the answers we get from the black-box is correct so we also have to take this into account. In order to proceed we need to make some assumptions, if we do not we are stuck and cannot learn anything. We have learnt how we can make assumptions about the space of functions using Gaussian processes which means we have exactly the right tools to proceed. By specifying a prior we can now ask the black-box and query the function at a set of locations, from the data we observe we can now get a posterior distribution over what we think the function is based on our new observations. Now the question is, where should we query the function next in order to learn as much as possible about where the minima of the function is? This is where uncertainty comes into play, we can now use the uncertainty about the function at different locations to guide our search. In order to decide where to search we construct what is known as an aquisition function, a function that tells us what we believe the benefit would be to query the function at this specific point. We now find the maxima of the aquisition function, query the objective function and then update the posterior distribution. Now we are back

1. formulate prior of objective function $f(x)$
2. query objective function
3. derive posterior distribution using the known values of the objective function
4. evaluate aquisition function and find best place (according to your strategy)
5. Go to (2) and repeate until you are happy with result or you have to produce a result

And thats about it. BO is a really exciting and emerging field that I think will have a huge impact on how machine learning is applied. It is very much at its infacy though so there is lots of work to be done and if you are keen to pursue machine learning I would suggest its a great thing to get involved in more. It also relates to something called Probabilistic nummerics which goes even one step further and treats the process of computation as a stochastic process. Again, remember the intention of this lecture, get the idea why uncertainty is important, and see what very useful things you can do with it.

8.1 Concepts

Aquisition function In Bayesian optimisation we work in two stages we model what our current belief is of the function using a Gaussian process, then we have another function which we use to try and figure out where to sample the function from. This function uses the information that we get from the GP. The function is referred to as an aquisition function.

Explotation vs Exploration a good aquisition function is one that balances the exploration of the space x to gain new knowledge with the exploitation of what we already know. Designing such functions is one of the main challenges with Bayesian optimisation.

8.2 Equations

Nothing new, we use the Gaussian process prior and posterior

9 Dirichlet Processes

In this lecture we looked at our second non-parametric prior the Dirichlet process. The DP is a prior over countable infinite sets compared to a GP which is over uncountable infinite sets. This means we can use DPs as priors over partitionings, the traditional example is to think of a mixture model, how can we specify how many clusters/components that the mixture is built up of? We ideally want to have a non-parametric such that the number of components can grow when we see more data. DPs are not at all as simply to write down as a GP, instead we write them down in a constructive manner. To see why this makes sense we first have to think slightly differently about models. When we write down a model that describes the probability distribution over a set of data $\mathbf{Y}$ as a set of conditional distributions we are describing the generative process of the data. As an example lets write down the joint distribution defined by the graphical model defined below,

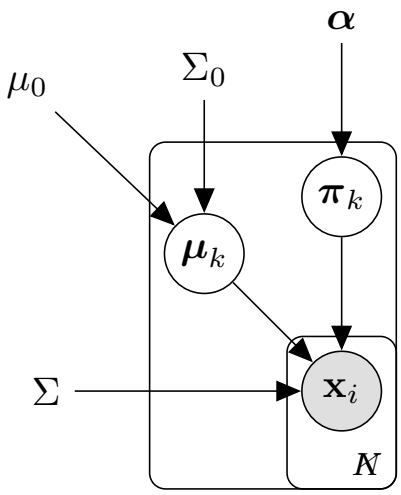

Figure 3: *Above is a graphical model describing the joint distribution for a Mixture model*

The observed data is shown as a shaded node while the stochastic variables are shown as white nodes and the variables we set do not have a circle. The model in Figure 3 describes the following graphical joint distribution,

$$p(\mathbf{X}, \mu_k|\mu_0, \Sigma, \Sigma_0, \alpha) = \prod_{i=1}^{N}\sum_{k=1}^{K} p(\mathbf{x}_i|\Sigma, \mu_k, \pi_k)p(\mu_k|\mu_0, \Sigma_0)p(\pi_k|\alpha).$$

Do not worry too much about what the model actually describes but lets try to see what generative process it describes. The four variables $\boldsymbol{\alpha}$, Σ_0, μ_0 and Σ are not treated as stochastic variables so they take on a specific value, while μ_k and π_k are stochastic variables parametrised by the and $\mathbf{x}_i$ is the observed data. This means that if we know the input variables we can sample and generate μ_k and π_k once we have these variables we can generate the data $\mathbf{x}_i$. Due to the choices that has been made, the form of the distributions, the parameter values etc. this will be a restricted distribution. The task of learning is to make this distribution tighter so that it really fits the data. However, this is the next part of the unit, so far we have only done rather simple learning and the main thing is being able to write up distributions/models such as this once we have formulated the generative process it is time to think about how to learn.

The Dirichlet process is best formulated constructively by describing how we can sample from the distribution. We discussed two different formulations, the Chinese resturant process and the stick breaking construction. The important thing to get from this lecture is, to see that formulation of a joint distribution in terms of conditionals defines a generative model and that there exists a non-parametric model called a Dirichlet process that specifies partitionings into possibly an infinite number of components.

9.1 Concepts

Generative models a generative model describes how the data we see can be generated. This means that it describes the distribution of the data as a set of conditional distributions.

Dirichlet process a non-parametric distribution that describes a distribution of a partitioning into possibly infinite number of components. It can be used to construct discrete models where we do not have to set the number of components such as infinite mixture models. [1]

Stick-breaking construction a constructive formulation of the dirichlet distribution that rather intuitively sees samples from a DP as breaking a stick recursively with the stick length drawn from a Beta distribution.

Chinese Resturant Process another constructive way to generate samples from a Dirichlet process. There is also an extension of this to hierarchical DP and then it is referred to as an Indian Buffet Process.

9.2 Equations

None

9.3 Code

Here is some simple code to generate samples from a Chinese Resturant Process

```
def chrp(N, alpha):
    table_assignments = np.zeros(N)
    next_open_table = 0
    for i in range(0,N):
        r = np.random.random()
        if r < (alpha/(i+alpha)):
            table_assignments[i] = next_open_table
            next_open_table += 1
        else:
            index = int(np.round((i-1)*np.random.random()))
            table_assignments[i] = table_assignments[index]

    return table_assignments
```

10 Topic Models

In this lecture we take a look at the most celebrated model that makes use of the Dirichlet Prior that we discussed during the previous lecture namely, Latent Dirichlet Allocation, or LDA for short [2]. Importantly there is an algorithm called LDA as well but then it stands for Linear Discriminant Analysis, don't mix these up. One lecture is way too little to really get the full grasp of what LDA really is and how it can be used, the motivation to still have a lecture on this was the idea of how we build models by making hierarchical assumptions and specifying conditional distributions. Try to get this concept and you have done well.

The key concepts that underpins LDA is the idea that certain types of data can be well described as collection of atomic units, text is one of these, you can think of words such as the unit, sentences or letters, the same with DNA. Importantly we make the assumption that the order that these units appear does not make a difference, it is only there relative frequence that is important. Text is a good example of this, say for example that we have a text about football and one about cricket, the word "silly" is likely to appear in both text while the word "mid-on" is probably a lot more likely for the cricket article while "offside" is more likely to belong to a football text. Now lets generalise this and make it into a hierarchy, lets say that each document is made up of a distribution of topic and that each topic is made up of a distribution of words. This will be our generative model of a text, if we know the topic distribution, we can sample from it, this gives us the word distribution,

now we can sample the words and we have a text. So what is the benefit, well, we have a description of texts in a rather intuitive manner, if we find a topic distribution that is 100% nonsensical dravel we might infer that it is an article from the Daily Mail while if we find a more balanced distribution we might infer that it is probably from a serious newspaper that we might pick up and read. The key idea here is what we have been on about a lot, with a model that can generate data we can say that at least to some extent we *understand* the data, and if we do we can do a lot of tasks from it, if we just learn the task we can at best only do that specific task. Now lets get a bit more precise and define some notation,

$w_{d,n}$ n:th word in d:th document

β_k topic-word distribution for k:th topic

θ_d topic distribution for d:th document

$z_{d,n}$ topic assignment for n:th word in d:th document.

Now we can write down what the distributions that we need to define in order to build the model,

- To generate the n:th word in the d:th document we need to know the topic that has been assigned to this word and the topic-word distribution

$$p(w_{d,n}|\beta, z_{d,n})$$

- To generate a topic assignment $z_{d,n}$ we need to know the topic distribution for document d

$$p(z_{d,n}|\theta_d)$$

- If we know the topic assignment and the topic-word distribution we can assume the words to be independent

$$p(w_d|\beta, z_d) = \prod_{n=1}^{N} p(w_{d,n}|\beta, z_{d,n})$$

- We can assume that the topic-document distribution is not dependent between documents

$$p(\theta) = \prod_{d=1}^{D} p(\theta_d)$$

- We can assume that topic word distribution is independent

$$p(\beta) = \prod_{k=1}^{K} p(\beta_k)$$

Look at these one by one, they might look scary at first, but hopefully each assumption does make sense one by one. Now we can put this together and write down the join distribution of the model,

$$p(w, z, \theta, \beta) = \underbrace{\prod_{k=1}^{K} p(\beta_k) \underbrace{\prod_{d=1}^{D} p(\theta_d) \underbrace{\prod_{n=1}^{N} p(w_{d,n}|\beta, z_{d,n}) p(z_{d,n}|\theta_d)}_{\text{word}}}_{\text{document}}}_{\text{corpus}} .$$

As we can see it factorises exactly as the hierarchy we started of assuming, well that is hardly surprising as we defined the whole thing but it is nice to see that the words and the math come together and say the same thing. We can also draw this as a graphical model,

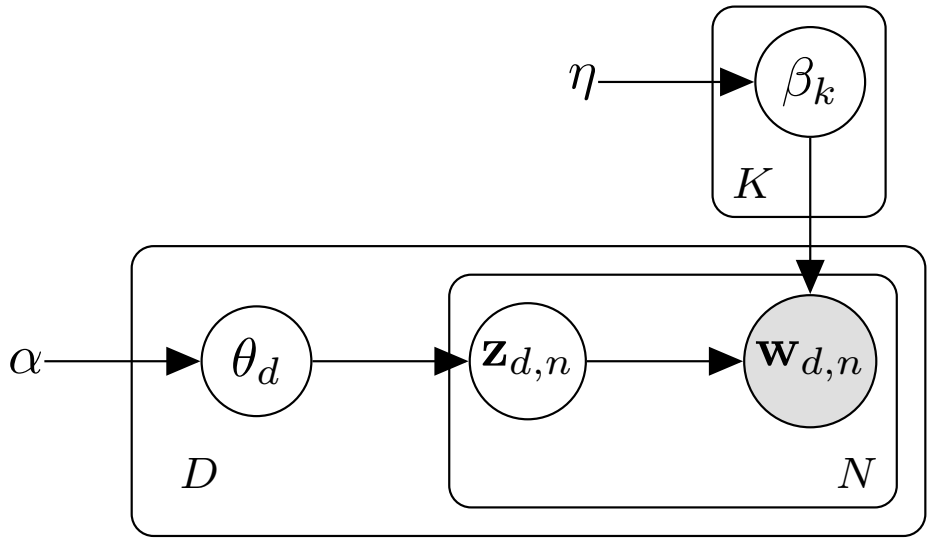

Figure 4: *Above is the graphical model for Latent Dirichlet Allocation in this figure the plates are rather important as there are many factorisations in the model, make sure you can go from the joint distribution to this model.*

Importantly we still haven't choosen the specific form of the distribution, this is the last thing we will do. We will assume that we can set α and η to sensible values that parametrises the distributions below.

- Topic-word: $\beta_k \sim \text{Dir}(\eta)$

- Topic-proportions: $\theta_d \sim \text{Dir}(\alpha)$
- Topic assignment: $z_{d,n} \sim \text{Mult}(\theta_d)$
- Word assginment: $w_{d,n} \sim \text{Mult}(\beta_{z_{d,n}})$

So in all the model is built up by two beta distributions that controls the proportions of the topic/word and the topic propotions and then multinomials for the actual assignments.

The last bit of the lecture highlighted the infeasibility of performing exact inference in this model and also gave some interesting examples of experiments.

10.1 Concepts

Generative models same thing as in the previous lecture, the idea that we build a model that can generate/syntesise data that looks like real data, if we can do that then we say that we believe that the data is actually created from our model and it can be used as a proxy for understanding the data.

Hierarchical Models the idea that we can build interesting models by stacking assumptions together. We have seen plenty of this but this is a very good example.

10.2 Equations

None

11 Graphical Models

This lecture was a bit of an extra depth into the language of graphical models. Graphical models is a visual language for writing down joint distributions, importantly it does not add anything but sometimes it is easier to see things visually rather than in equations especially when the number of variables grows. First and foremost we divide graphical models into two different classes, *directed* and *undirected*. What you have seen in the unit so far have been directed models which are built up of conditional distributions while here we also add undirected models which specifies joint distributions of variables.

Directed models are specified by conditional distributions which are described by arrows. If we have an arrow from node x_1 to x_2 this specifies the distributions $p(x_2|x_1)$. If we have a large network of many different nodes we can always factorise the joint distribution according to the following formula,

$$p(\mathbf{x}) = \prod_i p(x_i|\text{pa}_i),$$

where pa_i is the *parents* of the node x_i, i.e. the nodes from which arrows point *in* towards x_i.

This also leads us to a very important concept in machine learning namely the concept of *explaining away*. We have used this a lot but we haven't put a name to it till now. When we say that we have a linear regression task we might parametrise it as follows,

$$y_i = \mathbf{w}^{\mathrm{T}}\mathbf{x}_i + \epsilon.$$

In this case we can say that we have added the variable ϵ to explain away the noise so that the variable $\mathbf{w}$ can explain the relationship between y and $\mathbf{x}$. If we didn't $\mathbf{w}$ would be poluted by also having to explain the noise in the data. Think about this for a minute, relate it to a bigger model and hopefully you will see how important this is.

Next we looked at some specific graphical rules that one can use to see and figure out independence and dependency assumptions. Know that there are ways of doing these graphically but do not remember them.

tail-to-tail when not observed the nodes are **not** independent, when observed makes them conditionally independent

head-to-tail when not observed the nodes are **not** independent, when observed makes them conditionally independent

head-to-head when not observed the nodes are independent, when observed makes them dependent

Another concept that can be very important especially when implementing graphical models (to make them run in sensible time) is the concept of a *markov blanket* it is defined as follows,

- *Node x_i conditioned on all the remaining nodes in the graph can be written as conditioned on only its /parents* and *co-parents* these nodes make up the markov blanket/.

It is the only here which is important, if you want to update x_i you only have to consider its markov blanket not all the nodes in the graph. For a directed graph the markov blanket contains only its neighbours.

11.1 Concepts

The key ideas in this lecture is just to try and become a bit more abstract about building models, we don't just have to write equations, sometimes it is much easier to think of them in terms of graphical models instead. Importantly if you felt at ease with the LDA model, that you can see the connection between the joint distribution and the graphical model, you are more than fine.

11.2 Equations

None

12 Laplace Approximation

This is the first lecture of the third and final part of this unit, we now know about probabilities, the language of assumptions, and we know how to build models, the task of combining assumptions. Now comes the final bit, how can we combine assumptions with data when we have models which are intractable to perform inference in. Intractability comes from two different aspects, analytical intractability we often have when our models are not conjugate, i.e. we actually have to compute the evidence in Bayes Rule so that we can reach the posterior. Analytical intractability is often when the evidence cannot be written as elementary functions, or is a combination of infinite number of components. Computational intractability is when it is simply to expensive to compute, say that it is a sum of too many elements. In this lecture we will look at the first approximation, the Laplace approximation, it is simple and rather useful. We will explain it for a classification task.

The first part of the lecture was a derivation from a previous unit (SPS) that showed where logistic regression comes from, I hope that this part explains itself, therefore lets assume that we want to build a model of class from data, this is therefore not a generative model. We have some data x and we want to build a distribution over two classes $\mathcal{C}_1$ and $\mathcal{C}_2$ we say that the distribution over the former looks like this,

$$p(\mathcal{C}_1|x) = \frac{1}{1 + exp(-f(x))} = \sigma(x).$$

What we really have done is to specify the form of the posterior that we want. However, we can still treat the inference over that posterior in a Bayesian fashion. The first thing we will do is to assume that the function $f(x)$ is linear and then we will perform Bayesian inference over this variable. Now what we really have is linear regression which we know well, however, the target variable is no longer contious, it is discrete and therefore the likelihood changes which means that we have broke conjugacy if we use the same Gaussian prior over the weights. Before we write down the likelihood we need some notational clarity, for the target variable let us use (because that is Bishops notation) t where $t = 1 \rightarrow t \in \mathcal{C}_1$ and $t = 0 \rightarrow t \in \mathcal{C}_2$. This allows us to write the likelihood on the following form,

$$p(\mathbf{t}|\mathbf{w}, \mathbf{x}) = \prod_i^N \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i)^{t_i} \cdot (1 - \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i))^{1-t_i}, \tag{11}$$

where we have assumed each t_i to be independent. We want to use the same prior as we did for linear regression which was,

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{m}_0, \mathbf{S}_0).$$

This prior is not the conjugate prior to the likelihood which means we cannot reach the posterior using conjugacy. To still get somewhere we will seek an approximation to the posterior in the form of a Laplace approximation.

The idea behind the Laplace approximation is simple, we will approximate the posterior with a distribution that we know, this means that we can normalise it. Lets write this down in a general form for a general distribution,

$$p(z) = \frac{1}{Z} f(z) = \frac{f(z)}{\int f(z) dz}$$

where $p(z)$ is unknown as we cannot compute Z while $f(z)$ is possible to compute if we have likelihood and prior as,

$$f(z) = p(x|z)p(z).$$

The idea now is that we will find the mode of the posterior and then fit a Gaussian distribution to this mode Figure 5.

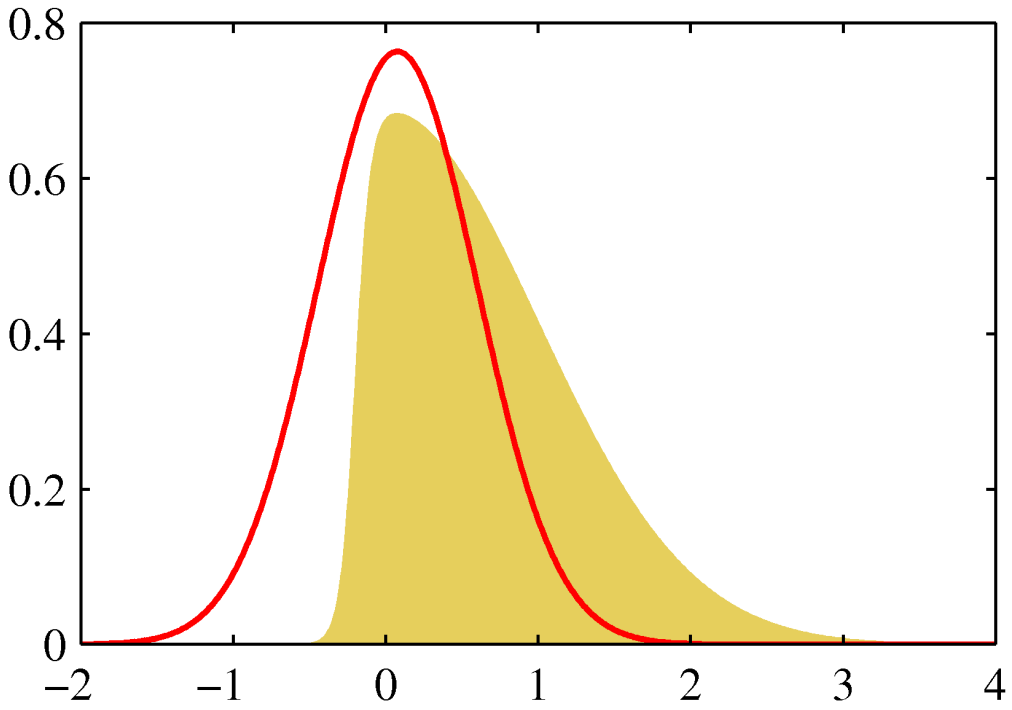

Figure 5: *Figure 4.14a from [3] showing the true posterior in yellow and the approximation in red.*

In our case we have both the likelihood and the prior so we can compute the unormalised posterior what remains is to find its mode and center a Gaussian around this. We can do this by using a Taylor expansion of the function. A Taylor expansion is a simple series

expansion of a function around a specific value (x_0 below)as follows,

$$f(x) = f(x_0) + \frac{\partial}{\partial x} f(x_0)(x - x_0) + \frac{1}{2}\frac{\partial^2}{\partial x^2} f(x_0)(x - x_0)^2 + \mathcal{O}((x - x0)^3).$$

As we know that we want to expand around the mode of the posterior we know that the first derivative is zero,

$$\frac{\partial}{\partial x} f(x_0) = 0.$$

We also know that the second derivative has to be negative as we are at a maxima therefore we have the following will be the expansion,

$$f(x) = f(x_0) - \frac{1}{2}\left|\frac{\partial^2}{\partial x^2} f(x_0)\right|(x - x_0)^2 + \mathcal{O}((x - x0)^3).$$

Now lets look at the above expression, we have a constant term and a quadratic term, just what a Gaussian have, so now we can identify the terms in the expression above and directly read out what our Gaussian should be. Now lets do this with our function that is the likelihood time prior,

$$\log f(z) \approx \log f(z_0) - \frac{1}{2}\frac{\partial^2}{\partial^2} \log(f(z_0))(z - z_0)^2.$$

Now to get the actual function we need to take the exponential,

$$f(z) \approx f(z_0) e^{\underbrace{-\frac{1}{2}\frac{\partial^2}{\partial^2}\log(f(z_0))}_{A}(z-z_0)^2} = f(z_0) e^{-\frac{1}{2}A(z-z_0)^2}.$$

This in itself looks an awful lot like a Gaussian if A was the precision. So we can now directly write down our approximation as,

$$p(z) \approx q(z) = \left(\frac{A}{2\pi}\right)^{\frac{1}{2}} e^{-\frac{A}{2}(z-z_0)^2}.$$

This is the steps that we walked through to get this approximation,

1. Find mode of $p(z)$
$$\frac{\partial}{\partial z} p(z_0) = \frac{\partial}{\partial z} f(z_0) = 0$$
2. Make Taylor Expansion around mode
$$\log f(z) \approx \log f(z_0) - \frac{1}{2}\frac{\partial^2}{\partial^2} \log(f(z_0))(z - z_0)^2$$

3. Take exponential to get function

$$f(z) \approx f(z_0)e^{-\frac{1}{2}\underbrace{\frac{\partial^2}{\partial^2}\log(f(z_0))}_{A}(z-z_0)^2} = f(z_0)e^{-\frac{1}{2}A(z-z_0)^2}$$

4. we want to find an approximation, to $p(z)$ so we need to normalise to a distribution

$$p(z) = \frac{1}{Z}f(z) \approx q(z)$$

5. assume that $q(z)$ is Gaussian, i.e. $f(z_0) = p(\text{mean})$

$$q(z) = \left(\frac{A}{2\pi}\right)^{\frac{1}{2}} e^{-\frac{A}{2}(z-z_0)^2}$$

In the above example everything was assumed to be one-dimensional but the extension to the multi-dimensional case is trivial. Now we can follow the same idea for our logistic regression example and derive the Laplace approximation as,

1. Apprixiate posterior

$$q(\mathbf{w}|\mathbf{t}) = \mathcal{N}(\mathbf{m}_N, \mathbf{S}_N) \approx p(\mathbf{w}|\mathbf{t}) \propto p(\mathbf{t}|\mathbf{w})p(\mathbf{w}) = f(\mathbf{w})$$

2. Compute $f(\mathbf{w})$

$$\log\, p(\mathbf{w}|\mathbf{t}) = \log\left(\prod_i^N \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i)^{t_i} \cdot (1-\sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i))^{1-t_i}\right) \tag{12}$$

$$- \frac{1}{2}(\mathbf{w}-\mathbf{m}_0)^{\mathrm{T}}\mathbf{S}_0^{-1}(\mathbf{w}-\mathbf{m}_0) - \log(Z) \tag{13}$$

3. The stationary point is the MAP estimate, we can compute the Hessian around $\mathbf{w}_{\text{MAP}}$

$$\mathbf{S}_N^{-1} = -\nabla\nabla\log\, p(\mathbf{w}|\mathbf{t})|_{\mathbf{w}=\mathbf{w}_{MAP}} = \mathbf{S}_0^{-1} + \sum_{n=1}^{N}\sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x})(1-\sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}))\mathbf{x}\mathbf{x}^{\mathrm{T}}$$

4. this leads to the final approximation

$$q(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{w}_{\text{MAP}}, \mathbf{S}_N)$$

The above might have seen a bit tedious and complicated but if you look through the details it isn't actually particularly much, look through the general steps and make sure that they make sense. The key thing is to see how well suited the Taylor expansion is to fit a Gaussian to.

12.1 Concepts

Intractability Intractability appears during inference as we cannot compute the evidence, i.e. we cannot compute sum out all possible parameter values and see how well they explain the data. This is either because of analytical or computational reasons.

Laplace approximation approximating an intractable posterior using a distribution that we know. In specific taking a Gaussian distribution and matching the modes of the posterior.

12.2 Equations

None

13 Sampling

In this lecture we will look at the second technique of performing approximative inference namely sampling. Sampling is a stochastic approach to computing the posterior. The general idea comes by the idea of a Riemann sum[1]. This is how at least I was taught integration as the limit of a Riemann sum, interestingly there are apparently people who where not taught this which lead to this rather amusing article[2] its a good read. So the idea is as follows, what we want to do is to marginalise out the unobserved variables to compute the evidence which is taking an expectation as,

$$\mathbb{E}_{p(z)}[f] = \int f(z)p(z)\mathrm{d}z \approx \frac{1}{L}\sum_{l=1}^{L} f(z^{(l)}) \tag{14}$$

$$\mathbf{z}^{(l)} \sim p(\mathbf{z}) \tag{15}$$

The idea of sampling is then to replace the intractable integration with a sum as,

$$\hat{f} = \frac{1}{L}\sum_{l=1}^{L} f(z^{(l)})$$

where we assume that the points z^l where we evaluate the function is drawn from the true distribution,

$$z^{(l)} \sim p(z).$$

However, before we move onto this stage we will begin by looking at how we can draw samples from a distribution that we can actually formulate. A Computer does not know

[1] https://en.wikipedia.org/wiki/Riemann_sum

[2] https://www.forbes.com/sites/alexknapp/2011/11/10/apparently-calculus-was-invented-in-1994/#5c068bd32792

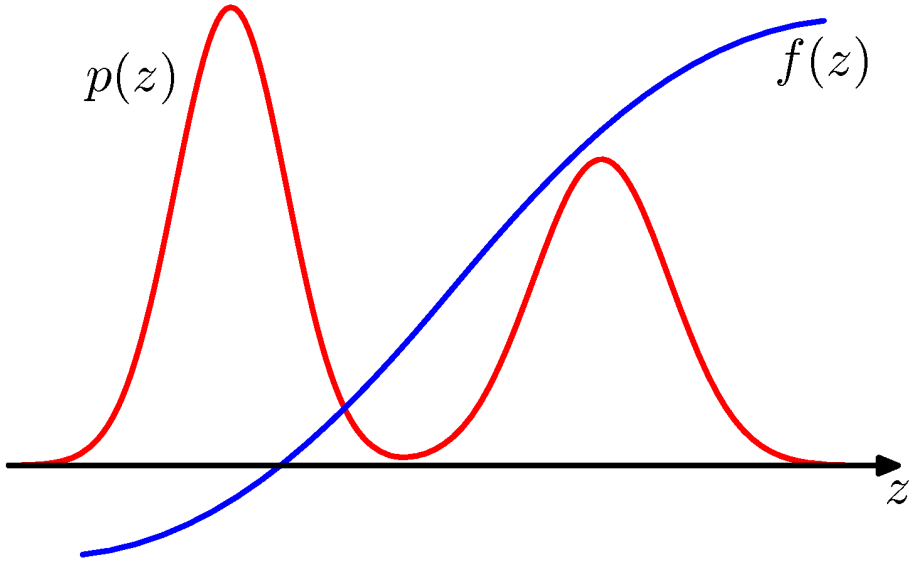

Figure 6: *Example of an expectation, where we want to take the expectation of the function f over the distribution p*

about Gaussian or Beta distributions at best it knows a little bit about uniform random values. The way we get random values from known distributions is through a process called *change-of-variables*. To me this is best described visually. Say that we want to draw samples from the blue distribution below. We can first formulate the comulative distribution function or *cdf* for short,

$$z = f^{-1}(y) = \int_{-\infty}^{y} p(y)\mathrm{d}y.$$

If we now think of this as a means of normalise the current distribution to a uniform distribution we can use this function in reverse. If we now flip the figure, (right pane) we can see that if we sample from the uniform distribution (green) and push these samples through the red function we will get samples from the blue distribution. This is the basics of sampling. So with this technique we are able to draw random samples from distributions that we can formulate the CDF for.

However, say that we cannot actually formulate the cumulative distribution then the above will not actually help us particularly much. This is when we need to approximate. There exists a myriad of different sampling techniques and they are mostly based around the same principle, if we can draw samples from a known distribution how can we convert these samples so that they come from the desired. So it is the same principle as above except that we do not know the translation function. There exists a lot of different techniques, their benefit is that most often with infinte amount of samples they will converge to the

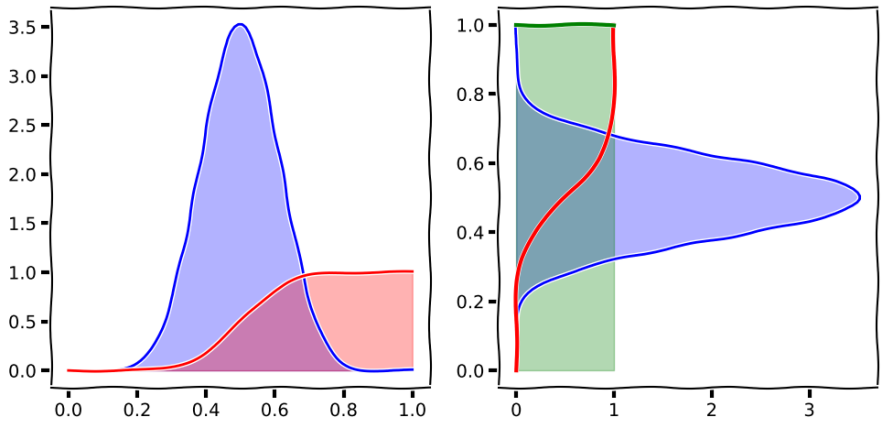

Figure 7: *Probability Density Function (blue) and Cumulative Density Function (red)*, uniform PDF (green).

correct answer but we have very little in terms of guarantees until we get there. This means that sampling is a bit of a black-art and can come out as rather hacky. There is no need to study the different techniques we looked at during the lecture, they where there as a means of exemplifying the field but rather understand the concept of change-of-variable and understand the idea of approximating an integral with a sum.

13.1 Concept

Sampling we want to compute an expectations over a function but we cannot do this analytically. The idea of sampling is to replace the integration with a sum.

Change-of-Variables we want to draw samples from a distribution by drawing samples from another distribution and transforming these to be samples from the desired distribution.

13.2 Equations

Change-of-Variables if f(y) is the cummulative distribution function of $p(y)$ then we can use this to transform samples from $p(z)$ to $p(y)$

$$z = f^{-1}(y) = \int_{-\infty}^{y} p(y)\mathrm{d}y.$$

14 Variational Inference

In this lecture we looked at approximative inference from a third perspective as an optimisation problem. Can we reformulate inference into a simple objective function that we can optimise? This is the idea that underlies variational inference. The idea is as follows, we have a posterior distribution $p(x|y)$ that is intractable, can we formulate a distribution $q(x)$ which has some parameters θ and then try and set these so that $q(x)$ becomes as close to $p(x|y)$ as possible? This doesn't seem sensible at all, we can't even write down $p(x|y)$ how are we supposed to fit something to it? The idea of variational inference is that rather than fitting this directly we will formulate the optimisation as a bound and this we can express, as long as we minimise this bound we will find an approximation. The idea is can be seen in the Figure below where $p(\mathbf{Y})$ is the distribution that we cannot formulate, the thing that is stopping us from reaching the posterior $p(X|Y)$. We will try and rewrite this as two terms, one which is a distance measure between our approximation $q(X)$ and the true posterior $p(X|Y)$ and a remaining term $\mathcal{L}(q)$. The first term is intractable as we do not have the posterior but if the second is and we maximise this we will automaticall minimise the former as it is bounded by $p(\mathbf{Y})$.

So how do we go about doing this? Let us first try and expand the evidence term,

$$\log\, p(\mathbf{Y}) = \log \int p(\mathbf{Y}, \mathbf{X}) \mathrm{d}\mathbf{X} = \log \int p(\mathbf{X}|\mathbf{Y}) p(\mathbf{Y}) \mathrm{d}\mathbf{X}.$$

Now we will add a distribution $q(\mathbf{X})$,

$$\log p(\mathbf{Y}) = \log \int \frac{q(\mathbf{X})}{q(\mathbf{X})} p(\mathbf{X}|\mathbf{Y}) p(\mathbf{Y}) \mathrm{d}\mathbf{X}.$$

The above might seem non-sensical but now we can employ something called the Jensen inequality which will lead us to our bound. This inequality in our setting says that if we move the logarithm inside the integral this will be a lower-bound on the integral itself. Therefore we will now proceed to move one of the $q(\mathbf{X})$ distributions inside the log,

$$\log p(\mathbf{Y}) = \log \int \frac{q(\mathbf{X})}{q(\mathbf{X})} p(\mathbf{X}|\mathbf{Y}) p(\mathbf{Y}) \mathrm{d}\mathbf{X} \geq \int q(\mathbf{X}) \log \frac{p(\mathbf{X}|\mathbf{Y}) p(\mathbf{Y})}{q(\mathbf{X})} \mathrm{d}\mathbf{X}.$$

We can now move the $p(\mathbf{Y})$ term out as we do not integrate over it which leads us to the following expression,

$$\log p(\mathbf{Y}) \geq \int q(\mathbf{X}) \log \frac{p(\mathbf{X}|\mathbf{Y})}{q(\mathbf{X})} \mathrm{d}\mathbf{X} + \int q(\mathbf{X}) \mathrm{d}\mathbf{X} \log\, p(\mathbf{Y}) \tag{16}$$

$$= -\mathrm{KL}\left(q(\mathbf{X}) || p(\mathbf{X}|\mathbf{Y})\right) + \log\, p(\mathbf{Y}). \tag{17}$$

The first term is known as the *Kullback-Leibler divergence* or KL-divergence for short, and is a measurement between distributions which is always positive except for in the situation

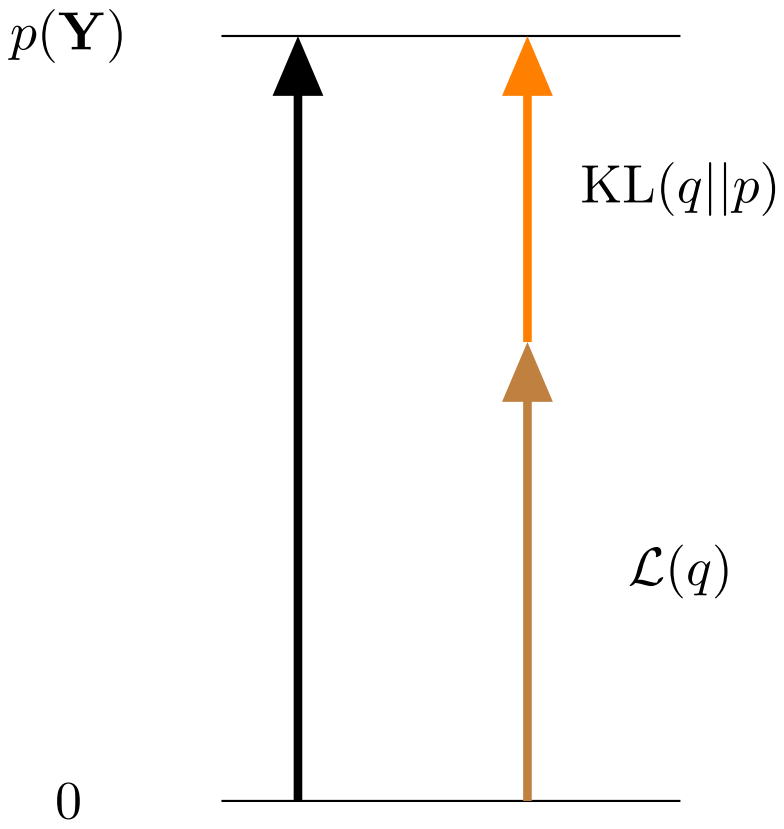

Figure 8: *Visualisation of the variational bound.*

where $p(\mathbf{X}|\mathbf{Y}) = q(\mathbf{X})$. This means that using the bound above if the divergence is zero $q(\mathbf{X})$ is the exact posterior. So therefore it makes sense to think of minimising this measure. We could have started with saying, "The KL-divergence is a measure of similarity between distributions, so lets minimise it" but to me that only tells half the story, the KL divergence appears as a consequence of applying Jensen's inequality and I think that is important to keep in mind.

So lets agree that we should be looking for an approximate distribution $q(\mathbf{X})$ that minimises the KL-divergence. But this has not done us any good at all as this is impossible as it requires us to know the true posterior which is intractable. Lets expand the KL

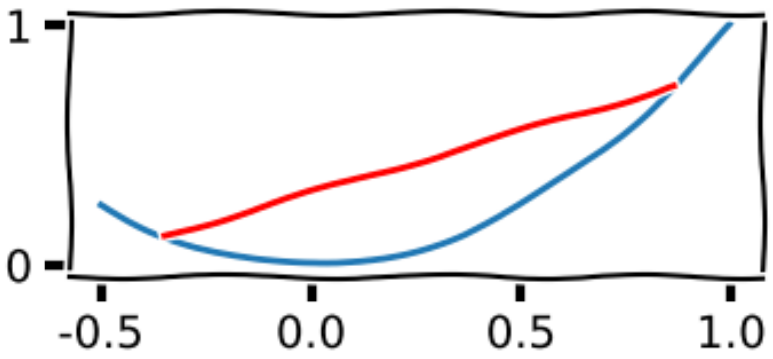

Figure 9: Jensen inequality.

divergence,

$$\mathrm{KL}(q(\mathbf{X})||p(\mathbf{X}|\mathbf{Y})) = \int q(\mathbf{X}) log \frac{q(\mathbf{X})}{p(\mathbf{X}|\mathbf{Y})} \mathrm{d}\mathbf{X}.$$

If we now use Bayes rule to rewrite the posterior as the joint and the evidence we can break out the latter just as we did before,

$$\mathrm{KL}(q(\mathbf{X})||p(\mathbf{X}|\mathbf{Y})) = \int q(\mathbf{X}) log \frac{q(\mathbf{X})}{p(\mathbf{X},\mathbf{Y})} \mathrm{d}\mathbf{X} + log\ p(\mathbf{Y}).$$

We can then break the logarithm inside the integral to reach,

$$\mathrm{KL}(q(\mathbf{X})||p(\mathbf{X}|\mathbf{Y})) = \int q(\mathbf{X}) log\ q(\mathbf{X}) \mathrm{d}\mathbf{X} - \int q(\mathbf{X}) \log\ p(\mathbf{X},\mathbf{Y}) \mathrm{d}\mathbf{X} + log\ p(\mathbf{Y}),$$

where for those of you who know their information theory can identify the first term as the Shannon entropy,

$$\mathrm{KL}(q(\mathbf{X})||p(\mathbf{X}|\mathbf{Y})) = H(q(\mathbf{X})) - \mathbb{E}_{q(\mathbf{X})}\left[log\ p(\mathbf{X},\mathbf{Y})\right] + log\ p(\mathbf{Y}).$$

Now we are starting to get somewhere, we have rewritten the KL-divergence as the entropy of a distribution that we can define and an expectation plus the evidence. Now let us go all the way back to our initial idea of finding a bound by re-arranging the terms,

$$log\ p(\mathbf{Y}) = \mathrm{KL}(q(\mathbf{X})||p(\mathbf{X}|\mathbf{Y})) + \mathbb{E}_{q(\mathbf{X})}\left[log\ p(\mathbf{X},\mathbf{Y})\right] - H(q(\mathbf{X}))$$

As we know that the KL-divergence is always positive we know that the last two terms will be a *lower-bound* on the evidence. Therefore if we maximise this we will in effect minimise the KL-divergence! This leads to the final expression which is what we want to optimise,

$$log\ p(\mathbf{Y}) \geq \mathbb{E}_{q(\mathbf{X})}\left[log\ p(\mathbf{X},\mathbf{Y})\right] - H(q(\mathbf{X})) = \mathcal{L}(q(\mathbf{X})).$$

This is often refered to as the *Evidence Lower BOund - ELBO*. Now we are left with trying to figure out a family of distributions $q(\mathbf{X})$ parametrised by some parameters θ and try to find the values that maximises the ELBO, and that is variational inference.

We then discussed one approach to this which is called Mean-Field Variational Inference which chooses a family of distributions which completely factorises over the unknown variables, this means in effect that we are matching the marginals of the posterior distribution. This as you can imagine is a very crude approximation and that is generally what variational inference has to be, it will never be exact but we will get an answer rather quickly as it is much simpler compared to sampling and we will also have an idea how well we are doing, if we take two different approximations and get two different bounds we can compare them. There was a lot written about variational inference in the second coursework so I do recommend having a look at this there as well.

14.1 Concepts

Variational Inference formulate inference as an optimisation problem where we try to find an approximation to the true posterior of analytical form.

Lower Bound the idea is that rather than minimising the KL divergence measure we find a formulation as a bound so that we indirectly minimise the divergence.

Approximation variational inference will never be exact and will also suffer from everything optimisation generally suffers from such as local minima. However, we know a lot about optimisation and there are very quick methods for it so computationally it is often very cheap.

14.2 Equations

None

15 Neural Networks

In this lecture we discussed from a very high level perspective what Neural Networks is. It is something that has gotten a lot of media attention recently and is associated with a significant amount of hype and buzzwords. The aim of this lecture was therefore to take this down a notch and explain what they are in actual machine learning terms. Neural networks where initially described in the 1940:s where researchers tried to describe mathematically how a neuron worked, I do not know enough neuroscience to say if the model is a good match but the general idea is that we use very simple computational units called neurons that are connected with each other in an intricate manner. The neurons are very simple functions, ideally a threshold function, such that if they get enough stimuli from its connected neurons they "fire". The parameters of the network is a weight for each connection that is multiplied

with the incomming signal. This is the idea, so lets be machine learners about this, what is this actually mathematically? It is a composite function, several functions connected to each other. Clearly these things can do amazing things, so this means that there has to be something special about such functions. A composite $g(\mathbf{x})$ of K functions is is simply this,

$$\mathbf{y} = g(\mathbf{x}) = f_K(f_{K-1}(f_{K-2}(\dots f_1(\mathbf{x})\dots))) = f_{k-1} \circ \dots \circ f_2 \circ f_1(\mathbf{x}).$$

Where in the case of a neural network each function is a linear function and a threshold (which is very nonlinear). Before we look at what they do lets revisit some old algebra, in specific we will need to definitions,

- Kernel of a function is all elements in the input that maps to the same point in the output
$$\text{Kern}(f_k) = \left\{(\mathbf{x}, \mathbf{x}')|f_k(\mathbf{x}) = f_k(\mathbf{x}')\right\}$$
- Image of a function is all the output values the input can take
$$\text{Im}(f_k(\mathbf{x})) = \{\mathbf{y} \in Y|\mathbf{y} = f_k(\mathbf{x}), \mathbf{x} \in X\}$$

The interesting thing is that if we look at composite functions we have this behaviour,

- Kernel of a composite function
$$\text{Kern}(f_1) \subseteq \text{Kern}(f_{k-1} \circ \dots \circ f_2 \circ f_1) \subseteq \text{Kern}(f_k \circ f_{k-1} \circ \dots \circ f_2 \circ f_1)$$
- Image of a function
$$\text{Im}(f_k \circ f_{k-1} \circ \dots \circ f_2 \circ f_1) \subseteq \text{Im}(f_k \circ f_{k-1} \circ \dots \circ f_2) \subseteq \dots \subseteq \text{Im}(f_k).$$

What this says is that the kernel will only increase or stay the same while the image will only decreas and stay the same. Intuitively I think about this as knots, say that $f_1(2) = f_1(3)$ this now ties a not of $x = 2$ and $x = 3$ this not can never be untied by f_2 as $f_2(f_1(2)) = f_2(f_1(3))$ this means that we have reduced the representative capacity of the composite function. Try to visualise this. So why does this turn out to be such a useful thing, well it is because these nots are very very useful because it allows us to learn very simple functions that when put together can collapse the input space in very intricate manners and say that if you are doing something like classification, this is exactly what you want to do. If you take all images of dogs and cats, you have a huge space now you want to collapse this to two points, cats and dogs and the more compositions you add the more knots you tie.

Another way to think about this is in terms of a change-of-variable that we used to do sampling. Let us assume that we have a distrubtion of images of cats and dogs $p(x)$ now we want to transform this to a distribution over classes. If we now push the distribution to

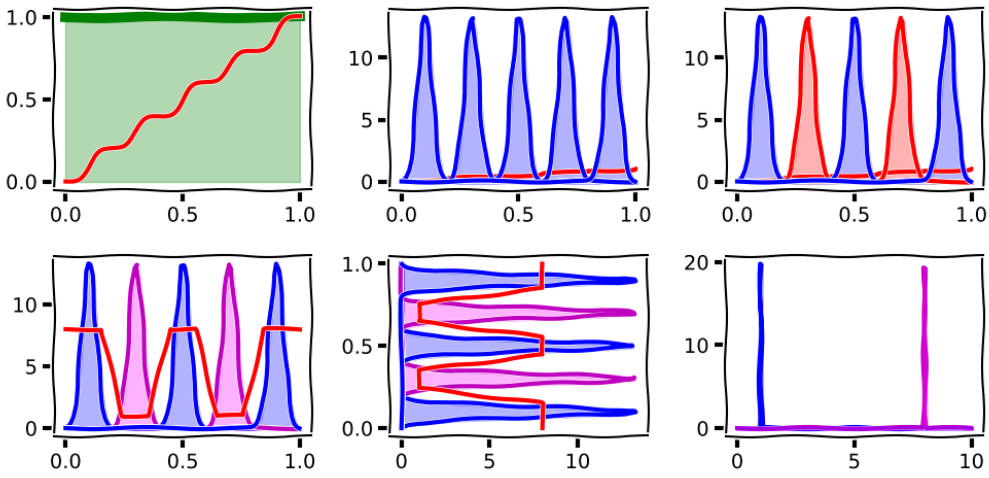

Figure 10: *Pushing a uniform distribution (green) through a set of non-linear and non-monotonic functions, from left to right and top to bottom*

a function which is not monotonic, which is what the cumulative distribution always will be we will collapse the distribution onto itself. Again its easier to see this visually.

Here we can see that we have taken a uniform distribution and made it pretty much into two delta functions by completely collapsing the distribution in many places, i.e. tying knots. This is the principle of composite functions, they can take an input space and collapse it very easily, this is big benefit as lets say in the image above the blue and magenta are different classes, they would be very hard to classify in the original dataspace, however, in the last plot it is very easy to classify them as they have been reduced to two points. In here though also lies the danger, these things are more prone to overfitting than any other method and this is why the require enormous amounts of data to not do so. The techniques where developed 30-40 years ago but it is first with the advent of big data we have been able to learn anything useful using composite functions. Further they are very hard to make principled assumptions about as it is very hard to define priors over composite functions in a sensible manner. Initial work on Deep Gaussian Processes [4] and [5] show that these methods interpretability do not extend well to compositions.

In the conclusion of the lecture we discussed how you set the parameters, i.e. the weights that describes the functions. This is done using a simple algorithm called 'back-propagation'. In general it consists of describing the function and then performing the chain rule of derivatives in succession through each layer and by that propagating the error on the output through the network.

15.1 Concepts

Composite functions Neural networks consists of composite functions. The key aspect that makes them useful is that the kernel of the function can only increase when layers are added while the image can only decrease. This means that the values that can be represented *decreases* when depth is added, i.e. that the output becomes more and more restricted.

15.2 Equations

- Kernel of a function is all elements in the input that maps to the same point in the output

$$\text{Kern}(f_k) = \left\{(\mathbf{x}, \mathbf{x}')|f_k(\mathbf{x}) = f_k(\mathbf{x}')\right\}$$

- Image of a function is all the output values the input can take

$$\text{Im}(f_k(\mathbf{x})) = \{\mathbf{y} \in Y|\mathbf{y} = f_k(\mathbf{x}), \mathbf{x} \in X\}$$

- Kernel of a composite function

$$\text{Kern}(f_1) \subseteq \text{Kern}(f_{k-1} \circ \ldots \circ f_2 \circ f_1) \subseteq \text{Kern}(f_k \circ f_{k-1} \circ \ldots \circ f_2 \circ f_1)$$

- Image of a function

$$\text{Im}(f_k \circ f_{k-1} \circ \ldots \circ f_2 \circ f_1) \subseteq \text{Im}(f_k \circ f_{k-1} \circ \ldots \circ f_2) \subseteq \ldots \subseteq \text{Im}(f_k).$$

16 Conclusion

I hope after reading this you feel like you can distil the unit down to its core concepts. Importantly it is left to you to fill in the details from the lectures and put them into this context. If you feel comfortable with this then you have a good knowledge of the topic. My aim with creating the material for this unit is that you should feel that there is an underlying philosophy that ties it all together, that really all these different methods and problems are just examples of the same principle.

17 References

References

[1] Carl Edward Rasmussen. The infinite gaussian mixture model. In *In Advances in Neural Information Processing Systems 12*, pages 554–560. MIT Press, 2000.

[2] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. 3:993–1022, March 2003.

[3] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[4] Andreas C Damianou and Neil D Lawrence. Deep Gaussian Processes. In *International Conference on Airtificial Inteligence and Statistical Learning*, pages 207–215, 2013.

[5] David Duvenaud, Oren Rippel, Ryan P Adams, and Zoubin Ghahramani. Avoiding pathologies in very deep networks. *arXiv.org*, February 2014.