COMS30007 - Machine Learning

Lecture Summary

Carl Henrik Ek

October 23, 2017

Abstract

This document will try to summarise the key concepts, equations, variables, etc of each lecture during the unit. It will be a living document as we progress through the lectures so do keep this file in sync with the repository.

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.

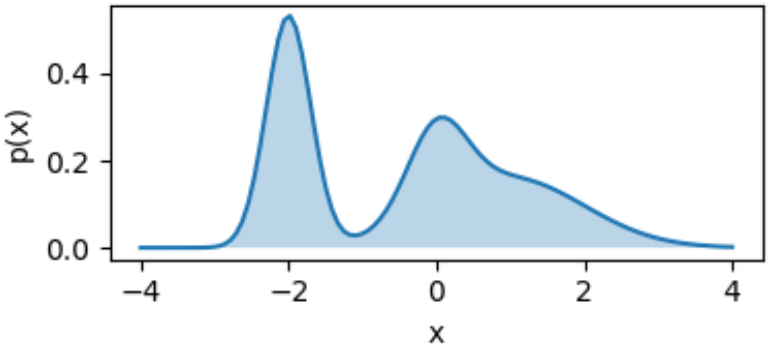

Figure 1: The above figure shows the distribution of the random variable x. The higher the probability is of a specific location the more likely the variable is to take that value.

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 **??**, 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.

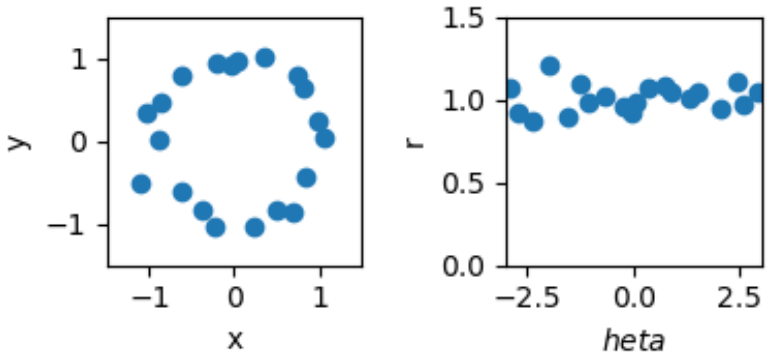

Figure 2: *On the left we have the original data which we cannot map a line to, on the right we have the same data but descibed in polar coordinates instead, where it is possible to model the data as a line.*

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

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

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,

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

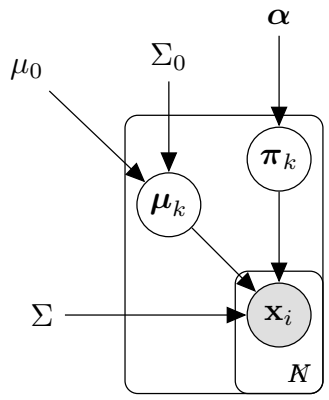

Figure 3: *Above is a graphical model describing the joint distribution for a Mixture model*

model in Figure **??** 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 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.