**Deep Learning**

**Week 1-2-3: Feed-forward neural networks**

* **0. Overview:**

Data science is a combination of computer science, maths, and domain expertise on the subject that you study. This course is focused on maths and computer science.

**Statistical artificial intelligence**: We fit statistical models and large quantities of data to learn from data to make systems that can generalize to datasets outside the training data.

**Reinforcement language is when** you take actions in an environment that will affect you in the future.

* **1. Deep Learning:**

DL is a rebranding of artificial neural networks name. Lacan has defined it as that it is deep learning if there is more than one step of a nonlinear feature transformation.

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

* **2.1 Feed-forward neural networks:**

Diagrama

Descripción generada automáticamenteIn **supervised learning**, we have an input and an output and what we do in the neural network is that we train a conditional model, so given an input, the output is predicted.

Each of the **units Zm** in the hidden layers has a linear model. It takes all the inputs, and it takes a weight. Then is applied a non-linearity, a so-called activation function and then you repeat the process in the second layer with new weights and then you compute the output.

In each layer, you apply a non-linear transformation, but you can also apply the identity.

X0 is the bias term. It is used to shift the activation function. For example, being sigmoid the activation function, the result without bias can be:

Gráfico, Histograma

Descripción generada automáticamente

Imagen que contiene reloj, dibujo

Descripción generada automáticamente

But when a bias term is added you can shift the activation function:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Gráfico, Diagrama

Descripción generada automáticamente

Texto, Carta

Descripción generada automáticamente

The first layer multiplies the weight by the input and then adds the bias term.

And the output of a 2 layers network

Non-linearity and training:

There are different options for these activations’ functions.

Esquemático

Descripción generada automáticamente con confianza media

**Logistic function** gets a real-valued number and then it transforms between 0 and 1 and interpret it as a probability number. It is computationally expensive, causes vanishing gradient problem and not zero-centred. This method is generally used for binary classification problems.

**Softmax** is a more generalised form of the sigmoid. It is used in **multi-class classification problems**. Like sigmoid, it produces values in the range of 0–1 therefore it is used as the final layer in classification models.

**Hyperbolic tangent** (blue) was very popular but when you do gradient descent-based learning you can get the learning process stuck even if your output is incorrect. Compared to sigmoid, it solves just one problem of being zero-centred.

**Rectified linear,** in the code, is fast and it has better convergence properties.It is easy to compute and does not saturate and does not cause the Vanishing Gradient Problem. It has just one issue of not being zero centred. Since the output is zero for all negative inputs. It causes some nodes to completely die and not learn anything.

**\*\*\* Research: Desirable features of an activation function\*\*\***

**\*Vanish gradient problem:** Certain activations functions, like sigmoid, squishes a large input space into a small input space between 0 and 1.

When the inputs of the sigmoid function become larger or smallerGráfico, Gráfico de líneas

Descripción generada automáticamente, the derivative becomes close to zero.

Working with only a few layers is not important, but when more layers are added it can cause the gradient to be too small for training to work effectively. A small gradient means that the weights and biases of the initial layers will not be updated effectively with each training session.

Solutions:

* **Other activation functions**, such as ReLU, which doesn’t cause a small derivative.
* **Residual networks** are another solution, as they provide residual connections straight to earlier layers.
* Finally**, batch normalization** layers can also resolve the issue. Reduces this problem by simply normalizing the input so |x| doesn’t reach the outer edges of the sigmoid function.

**\*Zero-Centered:** The output of the activation function should be symmetrical at zero so that the gradients do not shift in a particular direction.

* **2.2 Feed-forward neural networks:**

**Supervised learning**: We have a training set of n examples, so which is a pair of inputs and the associated target value, which is called labelled training. In **unsupervised learning,** we only would have X (input).

Gráfico

Descripción generada automáticamente

Conditionals models are trained to minimize the error. So, a **cost function** is defined to minimize that by **stochastic gradient** **descent**.

**\*\*\*** SGD randomly picks one data point from the whole data set at each iteration to reduce the computations enormously. It is also common to sample a small number of data points instead of just one point at each step and that is called “mini-batch” gradient descent. Mini batch tries to strike a balance between the goodness of gradient descent and the speed of SGD. **\*\*\***

**Overfitting:** when you fit well to the training situation, but you see new examples not making good predictions.

Example network:

Diagrama

Descripción generada automáticamenteX is a long vector unfolding the image, 28\*28.

We have 2 **hidden layers**, each one with its own **units**, and they are both **relu** activation functions.

Finally, **SoftMax** is used because we want a classification system and it has the property that can interpret the output as probabilities, due to the function definition.

Also, in each layer is defined a vector with intercept term (**bias**).

SoftMax:

Has 2 properties:

* + Because of the exponential function it means that **the output** for each dimension is **always non-negative**.
  + Because of the denominator the **output will always sum to one** (Probability).

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente Very nice to construct a network which is a **probabilistic classifier**.

On activations functions:

Gráfico, Gráfico de líneas

Descripción generada automáticamenteWe don’t use linear activation functions. Replacing 2 relus with linear models and calculating at the end it says that h(2) is just a linear function so that means that the feature transformation we make here can only be linear. That is why we should use non-linear activation functions.

Texto

Descripción generada automáticamente con confianza baja

**Weight matrix W(1):** the second dimension of the weight matrix of the first layer has the dimension of the input. And the first dimension are the weights for each specific unit.

After training it the network has pick kind of features that corresponds to generic tools. In that way can extract features of sub-parts.

Calendario

Descripción generada automáticamente

It can be seen for example that, the ones that appear in the input has similarities in the second hidden layer, because those has been passed through the same activation functions. That is how numbers can be predicted.

On sparsity:

Gráfico, Histograma

Descripción generada automáticamenteWe can see the output of the first and second hidden layers for a specific example. What is non showed are all the outputs that are zero.

* **3. Neural network training: Backpropagation**

Texto

Descripción generada automáticamente

**Why negative log-likelihood is used?** The derivative of a positive function is 0 exactly where the derivative of its logarithm is 0, and the log of that function has a much simpler derivative.

Texto

Descripción generada automáticamente

**One hot encoding** is the idea in which we take our class and make that into a sparse vector which has the same dimensionality as the number of classes.

The output is the softmax, which means that each term represents a probability.

In C we have - sign because we want to transform a maximization problem into a minimization problem. Being n the number of examples, and k the number of classes.

Texto, Carta

Descripción generada automáticamenteWe use **Gradient descent-based learning**, which means that we have some training criterion, for example log likelihood, and then we compute the gradient (taking the derivative of the cost function).

And the basic operation is to take a small step opposite the gradient, making down moves.

We have a learning rate which might depend on which step we have taken.

Diagrama

Descripción generada automáticamente con confianza mediaSimplest non-trivial network: linear outputs with 1 HL and 1 unit.

The cost function will depend on the weights of 1st and 2nd layers.

Taking derivatives, we can see that the cost function depends only implicitly on the weights, and explicitly on the activation of the second layer.

Texto

Descripción generada automáticamenteWhen having more layers there are reused computations through the layers, so if we store these computations, we will simplify it.

**Texto

Descripción generada automáticamenteIn the backpropagation algorithm,** we say that the derivatives of the cost function on one layer are defined in terms of the derivatives of the activations on the next layer.

Forward propagation is when we propagate the inputs to the outputs and calculate the cost.

When having done backpropagation in the middle equation we can compute the actual gradient simply by the last of the rule.

The important thing here is that we must store a lot of things in memory, and this is usually a problem if we have like recurrent NN which can actually be a little bit memory-consuming.

* **4. Neural network training: Optimisation**

Diagrama

Descripción generada automáticamenteWe have one training point and a network with a linear output unit. We set the input to 1 and the target to 1.5.

The output of the network is simply w1 times w2. Then we have the square error looking at the difference of this w1 w2 to 1.5, and a penalty is added to the square length of the weights. This is the plot.

Gráfico

Descripción generada automáticamente

We only have 2 params. The gradients are always pointing to the minimum

Diagrama

Descripción generada automáticamenteDiagrama

Descripción generada automáticamente

Selecting **n=0.25** we can see small steps that take some time to get the minimum wasting a lot of time, we used to avoid this kind of spectra.

Increasing step size, have a tendency that goes really far of the minimum overshooting our target.

Diagrama

Descripción generada automáticamenteUsing the curvature, by the second derivative (Hessian), this method takes this into account.

In Newton Rapson rule we can replace the learning rate by the inverse of this H matrix. This works very well when we are closer to a quadratic minimum. But, as can be seen in the corner, gradient points in the opposite direction.

This method is very computationally intensive because the inverse matrix must be calculated. But it can give you a good intuition about which step size to use.

Diagrama

Descripción generada automáticamenteOne thing to avoid the oscillatory behaviour is to remember where you went in the previous step (Momentum method).

This m parameter remembers where the gradients were in the previous step and then you only modify this by having this alpha in front which is below one so you can slowly forget your old gradients and then do new steps.

This has a smoother behaviour.

Diagrama

Descripción generada automáticamente

A noisy gradient is something that we get when we have mini-batch learning. It looks like is not a really good idea but with large systems is very nice.

Gráfico, Gráfico de líneas

Descripción generada automáticamenteWe calculate the gradient on not the whole data but on a subset of data.

This means that maybe in the beginning you can be so far (initialize with some random parameter setting), so you need a few data that give you a general sense of where you want to go.

Secondly, having a stochastic gradient introduces some noise that helps you to get out of local minima.

Third thing is that, if you have a large dataset and you use your GPU you will not be able to store all data in your memory and therefore it’s nice only to have to store some of the subsets of the data.

We need to change step size through learning.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamenteThe step that we take is never going to be larger than alpha. That is a way to kind of control that in all directions we have more or less the same step size and we know how big it is and it is easy to know more or less how big the steps need to be.

Fixed step size makes the training stop the convergence at some point. This types of algorithms try to avoid these problems.

**Week 1-2-3: Convolutional neural networks**

* **1. Introduction to CNNs:**

CNNs are used for image classification. The idea in CNNs is to do a lot of **weight sharing** in a way that we can extract features from an image in different locations in the image. So, the same kind of weights are applied in different parts of the input. We are going to apply the same “**filter**” for all areas of the image.

We apply a filter (3x3 weigths) to the image and produce an output image. This is done starting in one corner with the filter and then we do the **element-wise product between grey values in the image and then the values of the weights in the filter**. If we do that with this type of filter (weights example) it will pick up edges that are 45 degrees.Diagrama

Descripción generada automáticamente

These weights are learned from data. Another operation that we have in CNNs is called **pooling**. The general idea is that we have these filter layers, we learn many different filters usually all of them have the same size for a specific layer, and then we have pooling layers and then we repeat it again.

Diagrama, Dibujo de ingeniería

Descripción generada automáticamenteThe most famous deep learning paper. Numbers indicate dimensions of each particle filter that is being applied.

Una captura de pantalla de un celular con texto e imágenes

Descripción generada automáticamente con confianza mediaCarta

Descripción generada automáticamente con confianza media

* **CNNs the details:**

Texto

Descripción generada automáticamente con confianza mediaApplying 3x3 greyscale 2D convolution is possible to detect edges in different orientations.

Mathematically, we sum over all the elements in the matrix multiplied by weights.

Texto

Descripción generada automáticamente

MNIST dataset is nice because we have 28x28 images. Imagine we have 1000x1000 images then we really need a lot of weights.

The main things with CNNs are that we have local connectivity and parameter sharing.

Diagrama

Descripción generada automáticamente1D input with 5 pixels connected with 5 units in next layer. If they are connected all in all we have 25 parameters.

**First restriction** to have less parameters by **only connecting the next layer weights to the kind of the neighbouring weights** as it’s shown in the 2nd image reducing number of params.

Diagrama, Esquemático

Descripción generada automáticamenteWe can make a **parameter sharing** as shown by these colours, so we can say we will use **same weights in kind of the connections in the same direction** having only 3 parameters.

If you think in each hidden unit extracting some feature from the original image, then we can extract this feature in all places of the image with these 3 params.

So, if we are looking for edges in a specific direction, with these 3 weights can extract the kind of pattern in the input.

Imagen que contiene Diagrama

Descripción generada automáticamenteBoxes indicate dimensions of our inputs and our filter images. The convolutional filter chosen is 5x5x1 (if we had a RGB image, the filter would be 5x5x3).

We choose to have 9 of these filters. Applying **zero padding**, which means adding zeros in the edge of the image to preserve the size of the input image, the output of the filters would be 28x28x9 (we call 9 channels).

Texto, Carta

Descripción generada automáticamenteWe can see that each of the filter has extracted some features from the original 8.

We sum over the channels in the input layer

Diagrama

Descripción generada automáticamentePooling is an operation that makes the output smaller.

We can use different types of pooling. For example, average pooling means that the output of the pooling layer is the average of the elements. Also we can use a max pooling meaning we look at the maximum activation in this 2x2 area and then we simply pass only the max.

The reason we use pooling is: 1. We want to reduce the computation in the subsequent layers so we will implement a kind of approximate local translational invariance.

Diagrama

Descripción generada automáticamente

From the bottom to the top. A convolutional layer is applied with 9 filters. Then, we have a 2x2 pooling so we have only half the number of pixels with same channels. Then, a convolutional layer is applied. Another 2x2 pooling layer. And finally, we can take all these 7x7x16 values that we make into a long vector that we feed into a feed forward NN.

We train all this by back-propagation as before. In the output we can see a big probability of the class 8.

Gráfico

Descripción generada automáticamente

The closer we get to the output, the more we hope that the features we have will look alike if we push images from the same class.

If we look at pixel wise distance at the beginning distance between 9-4 is smaller than the one between 4-4. But when we get to the 2nd convolutional layer, we get more accurate.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

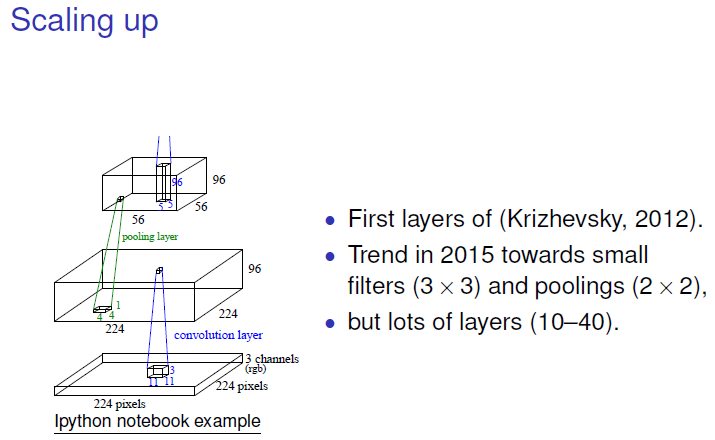
Another important point about weight sharing is that we can do some counting both on the number of weights in the network, but also the number of features that we extract in different layers.

So, in 1st layer we had 5x5x9 filters, so we had 5·5·9 parameters. And the same for next layers. For the 2nd layer

we still have 5x5x9 and we have 16 of those filters.

So, finally we have **11665 parameters which are quite low**.

Compared to FFNN weights are less now than before, so that means that with FFNN we could risk overfitting with this type of NN.

Trend is that a lot of people use smaller pools and lot of layers (150 Microsoft).

* **2. Res-, Dense- and Wave Nets:**

Gráfico, Histograma

Descripción generada automáticamente

When having more layers in a NN is more difficult to train it because when doing backpropagation, you can get stuck at some suboptimal solution.

What is proposed in resnet is to sum the output of the L-1 layer to the output of the L layer. If initially our F is close to 0 our NN is close to implementing a linear network, what means that we can start learning more linear features and then slowly we can make F different from 0 and then turn on the feature maps that we need.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente Left plot is a regular NN and the right one a resnet. We can see how resnet with more layers improve its accuracy.

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteDenseNets** uses the output of previous layers to compute the output for the actual layer.

Diagrama

Descripción generada automáticamente

* **3. New activation functions**

**Gráfico

Descripción generada automáticamente con confianza bajaGLU**: has 2 linear models. The 1st is applied only to the input, and the 2nd one is applied to the input and a logistic function is applied.

**MaxOut**: Each hidden unit is connected to K inputs.

**Leaky Relu:** you allow a slope in the negative side.

* **4. Image segmentation**

Diagrama

Descripción generada automáticamenteYou take an image, and you want to classify each of the pixels.

Diagrama, Dibujo de ingeniería

Descripción generada automáticamenteFirst you go from the large image to a representation that has fewer pixels but more channels, combining local and non-local information to perform the segmentation. Then you have to go from the low-resolution image with many channels as to the original resolution to classify objects in the image.

Gráfico, Diagrama

Descripción generada automáticamenteIn last layer, the one with the low-resolution image, we want to get the original size. So, one method is to multiply the size x2 and divide the channels by 4. Getting a larger image.

Gráfico, Gráfico de cajas y bigotes

Descripción generada automáticamente

Used in modern image segmentation. The idea is that you want to perform something that use non-local information, going down to low resolution images after conv and pooling layers, but for having a fine-tuned segmentation you also need information from the original high-resolution image.

High and low resolution information is combined.