**Deep Learning**

**Week 1 - Feed-forward neural networks - do it yourself pen and paper**

* **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

First layer multiplies the weight by the input and then add 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 mediaFor example, if we are working with binary classification (1 or 0) we can use the **logistic function** because it gets a real-valued number and then it transforms between 0 and 1 and interpret it as a probability number.

**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.

**Rectified linear,** in the code is fast and it has better convergence properties.

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

**Supervised learning**: We have a training set of n examples so that is a pair of inputs and the associated target value, that 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**.

**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 their owns **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

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 in 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 where 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 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.