BUILDING UNIVERSAL UNDERSTANDING

Deep Learning Meetup @Unbabel

Unbabel

Expectations

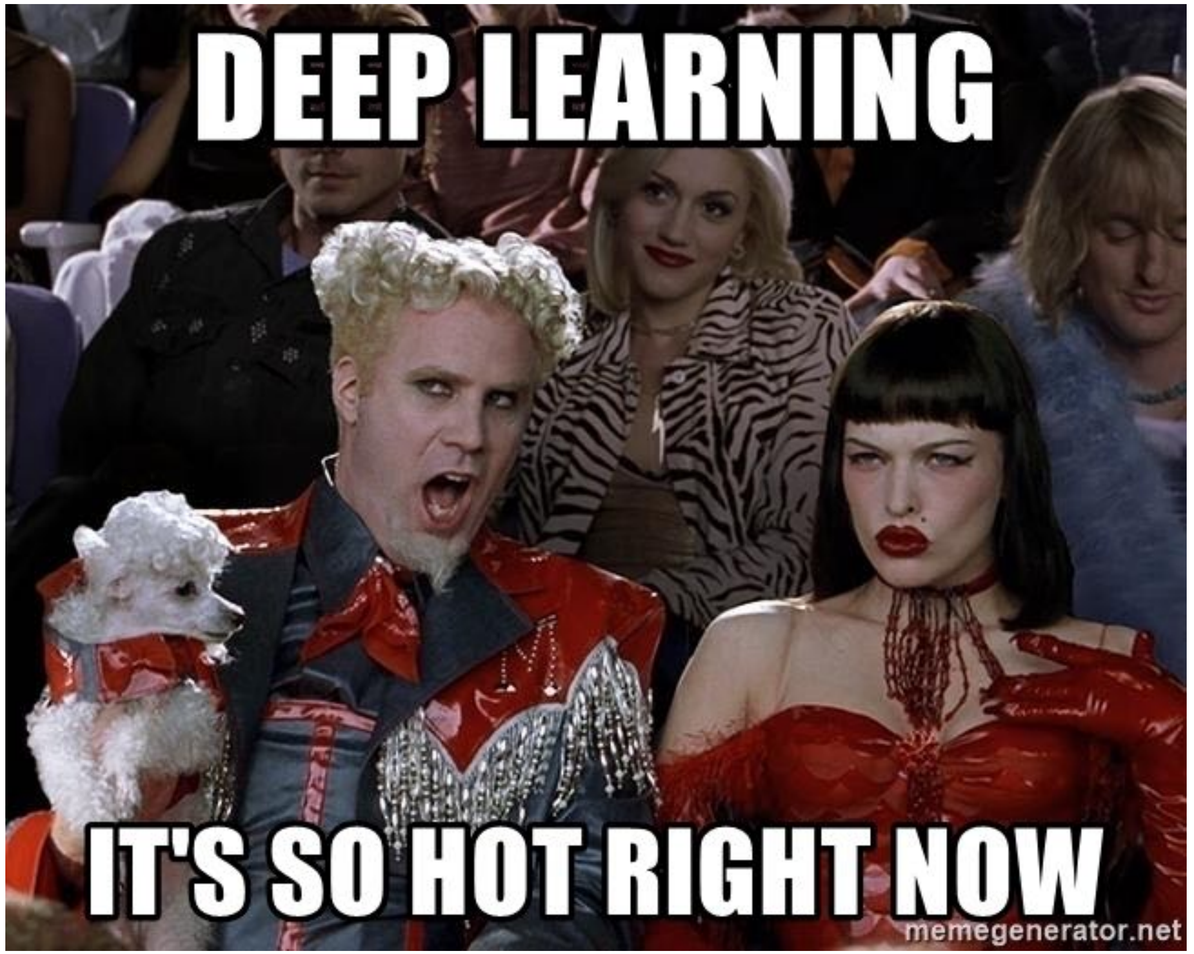
What this presentation **will** be

- A brief overview of the field
- A review of some machine learning concepts with a step up to deep learning
- A light presentation with some pointers
- A practical meetup where you can get started with simple code examples

What this presentation **won’t** be

- A mathematics class
- A comprehensive overview of machine learning
- An intensive course in deep learning
- A deep dive into the presented techniques

What is deep learning?


DEEP LEARNING
IT'S SO HOT RIGHT NOW
memegenerator.net

The AI/deep learning hype

Facebook kills AI that invented its own language because English was slow

Google AI platform Neural Machine Translation develops internal language

Artificial intelligence can now emulate human behaviors – soon it will be dangerously good

11.42am BST

OpenAI Writes So Well That Creators Won't Release It, Fearing Flood of Disinformation

What is deep learning

Artificial Intelligence

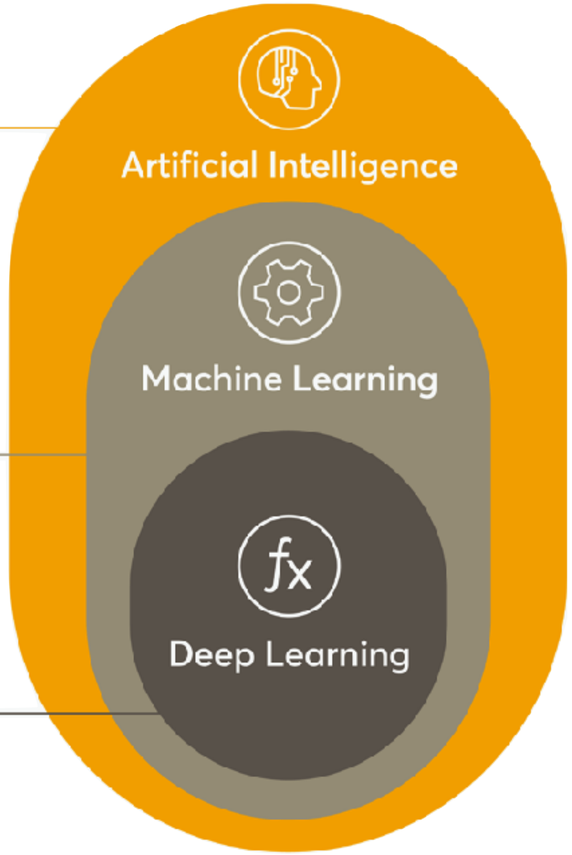
Any technique which enables computers to mimic human behaviour.

Machine Learning

Subset of AI techniques which use statistical methods to enable machines to improve with experiences.

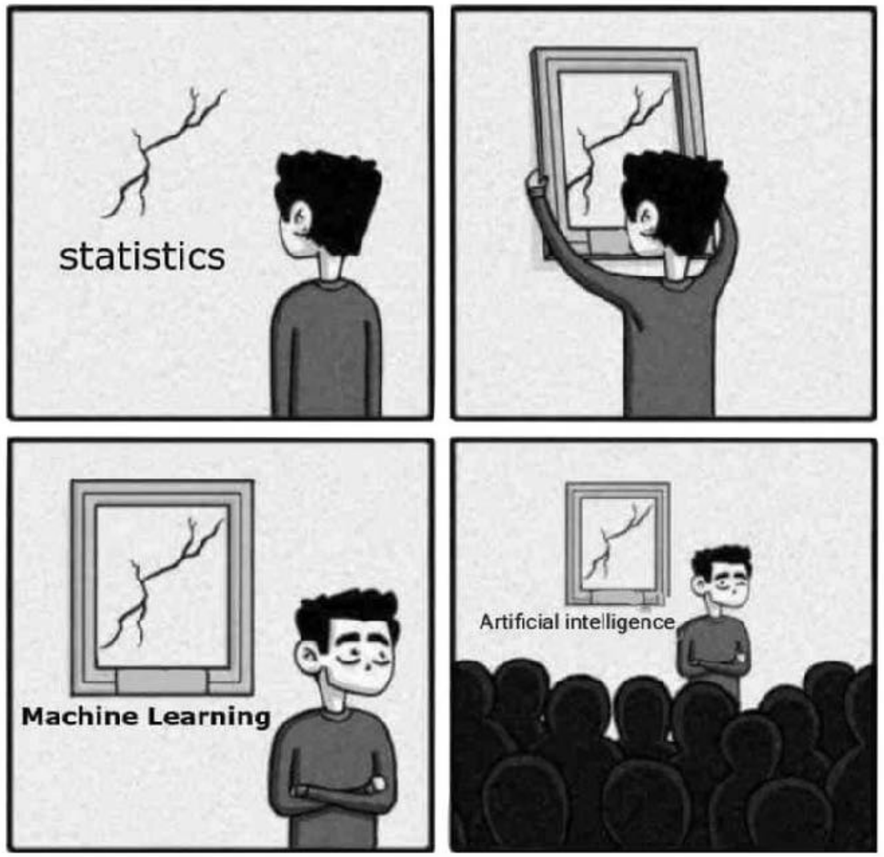
Deep Learning

Subset of ML which makes the computation of multi-layer neural networks feasible.



source: https://www.nastel.com/blog/simple-guide-ai-machine-learning-deep-learning/

Is machine learning just statistics?



source: https://towardsdatascience.com/no-machine-learning-is-not-just-glorified-statistics-26d3952234e3

- Most methods rooted in statistics, but it is a different set of techniques/algorithms
- Some methods more related than others
- At the end of the meetup, maybe you'll make up your own opinion about it

An introduction to machine learning

Supervised learning

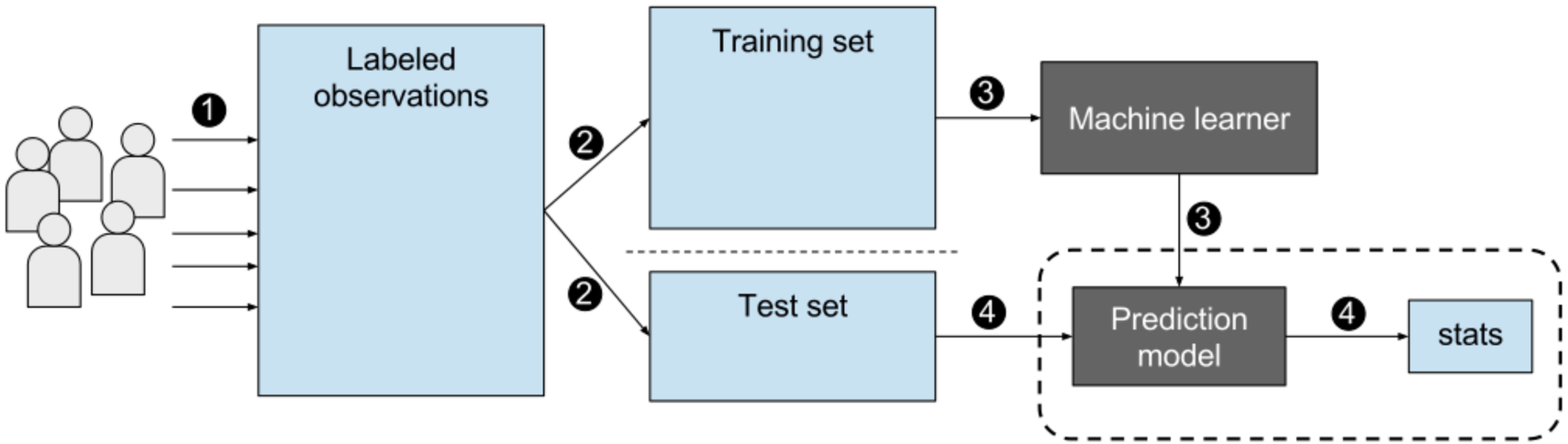
- **Supervised learning**: model learns from labeled data, this is, several examples where you have the desired output - the true answer

- **Unsupervised learning**: dataset does not have an explicit solution, and it is just a collection of example data with no label. The methods need to learn automatically some sort of structure in the data purely by analysing features

- **Reinforcement learning**: The model attempts some answer on a particular goal, and receives a reward signal depending how well it performs. Overall aim is to maximise this reward.

Supervised learning

- We will focus only on supervised learning: for our tasks we have labeled observations, with which we can train or model to learn to perform well in its predictions



source: https://blogs.nvidia.com/blog/2018/08/02/supervised-unsupervised-learning/

Simple tasks - regression

- Let’s say everytime you go out with your friends, you register the amount of money you spend, together with some information you consider useful, like the number of people that went out

- After a few times, you that maybe you can use this data to predict how much you’re going to spend on your next night out

Number of people	Money spent
4	20.4
6	13.5
3	24.13
6	14.12
…	…
8	13.12

Simple tasks - regression

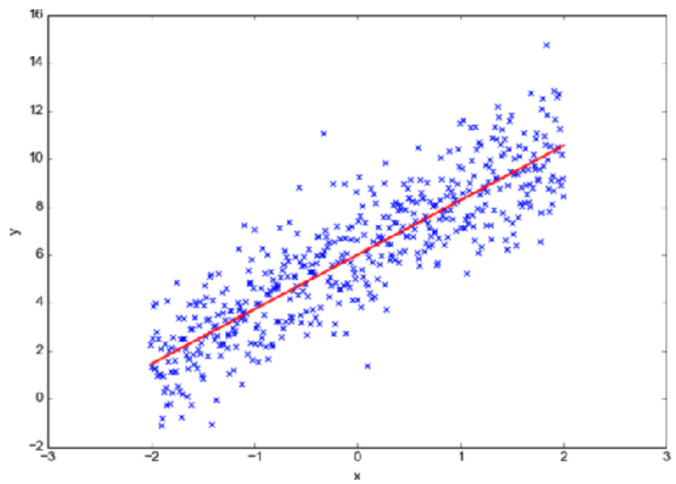
Or let’s say you did the same exercise for the money you spend on the supermarket and the number of days since you last went there

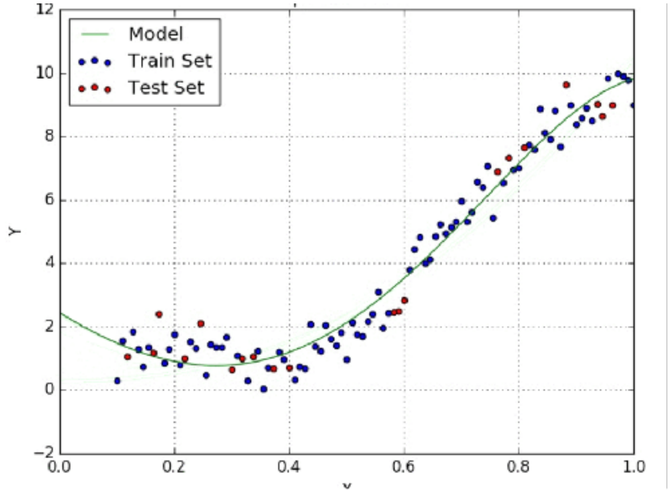
- You also want to predict how much you will spend on your next trip to the supermarket

Number of days	Money spent
2	10.1
1	6.50
6	24.13
6	14.12
…	…
5	45.02

Simple tasks - regression

- Both these tasks are simple **regression** tasks. **Regression predictive tasks** try to approximate a mapping from input variables **X** to a continue output variable **y**. You probably already heard at some point of this concept, for example in its simple form of linear regression or polynomial regression, the figures show below.





source: https://towardsdatascience.com/introduction-to-linear-regression-and-polynomial-regression-f8adc96f31cb

Start with linear models

We can define our regressions as a function a(x) that we want to model and that can be represented by a linear combination of our inputs **x**:(x_0, x_1, ... x_n), also called features, through weights w_0, w_1, ..., w_{n+1} :

$$a_w(x) = w_0 + w_1 x_1 + w_2 x_2$$

We can also write it in a vector form, as a dot product between features and weights.

$$a(x) = \sum_{i=1}^{n} w_i x_i = \overrightarrow{w}^T \overrightarrow{x}$$

Our problem becomes one of finding weights **w** for our model that generate good predictions

Measuring error

We need to measure how well predictions from the model a(x) fit our examples y.

For this purpose we define a measure of error between predictions and labels, which we call an error function, or loss function. For regression, it is common to use mean squared errors as this metric, defined below.

$$L(\theta) = \frac{1}{2} \sum_{i=1}^{m} (a_w(x(i)) - y(i))^2$$

How does it learn

For simple regression models we can often produce a solution that minimises our error through exact methods.

However, we'll focus on learning techniques that iteratively find better estimations of the model, converging into good approximations of these models

This iterative technique is called **Gradient Descent**

Gradient Descent

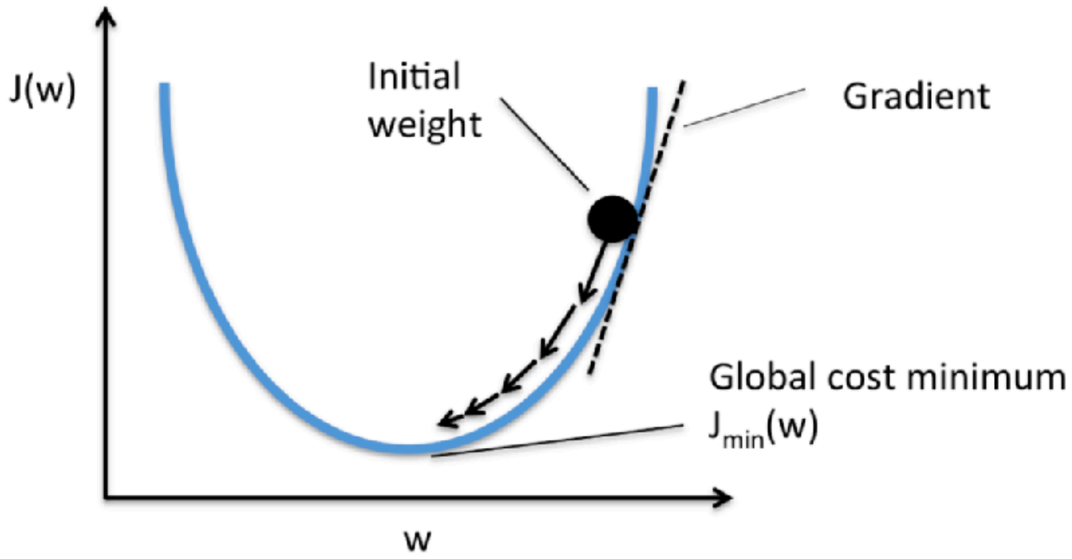
Gradient Descent makes use of the loss function, moving into the direction of error minimisation. For this purpose, it makes use of the derivative of the loss function.

The updates are done as follows:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} L(\theta)$$

For our cost function:

$$\frac{\partial}{\partial w_j} L(w) = (a_\theta(\vec{x}) - y) x_j$$



Gradient Descent

$$\vec{w} = \vec{w}^0$$

for **k** in **n_iter**:

$$\Delta L(w) = X^T(X\vec{w} - y)$$

$$\vec{w} := \vec{w} - \alpha \Delta L(\theta)$$

X - matrix of example features [n_samples, n_features]

y - vector of example labels [n_samples, 1]

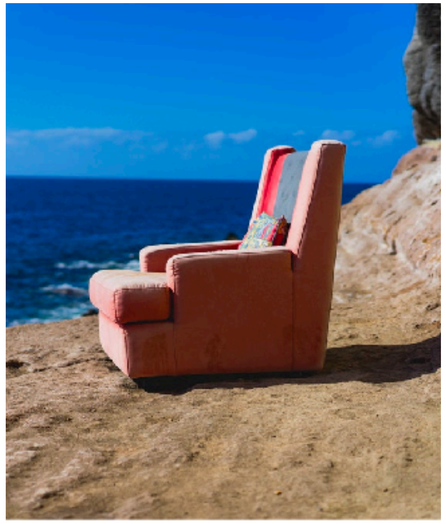
w - vector of weights [n_features, 1]

α - learning rate

Simple tasks - classification

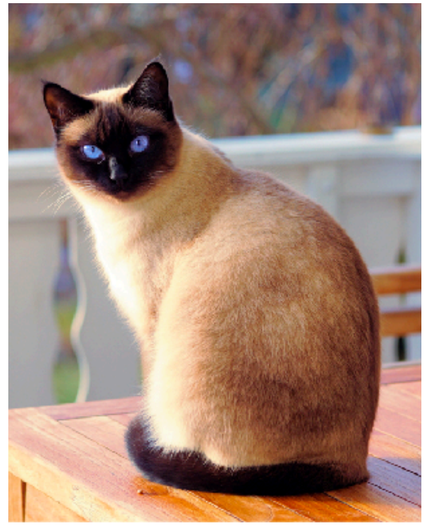
Now, let’s say that instead of predicting some value according to variables you collected, you want to split data into classes, for example identifying if a picture has a dog or not.

label: dog



label: not a dog

label: dog

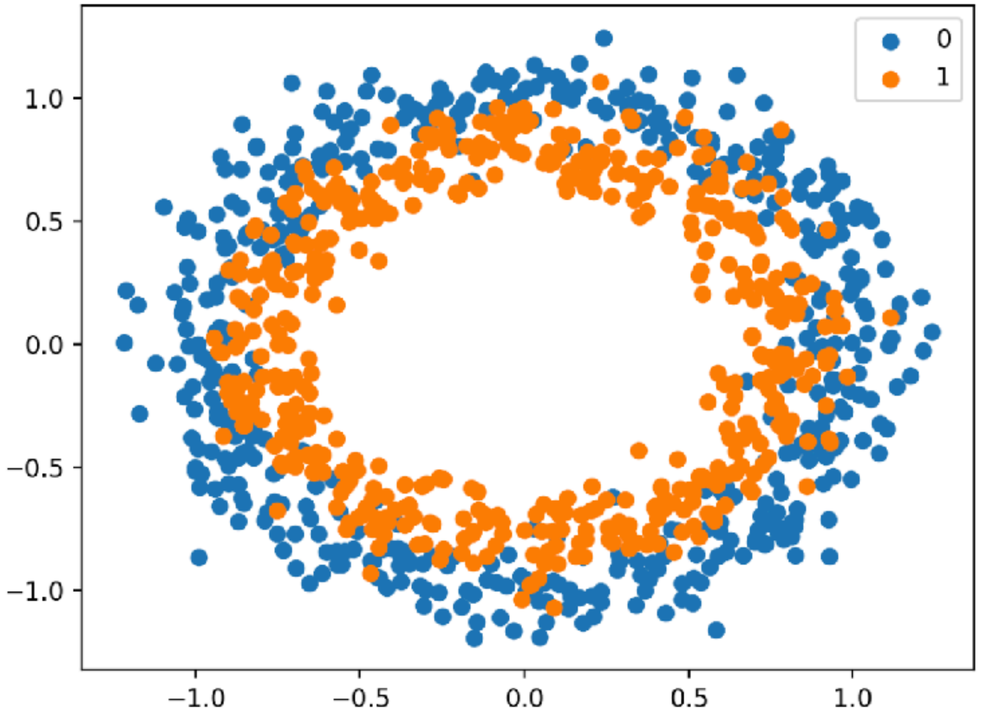


label: not a dog

label: not a dog

Simple tasks - classification

Or we want to do something slightly simpler, like finding a model to classify the data in the following picture



source: https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/

Simple tasks - classification

- Both of these are **classification tasks**, in particular binary classification tasks, since we only have two classes. In these, what we want to predict is usually a class from a discrete set of options.

- In binary classification, you can tweak the problem as that of finding the probability of the example belonging to the default class

$$P(y = 1 \mid x; w) = a_w(x)$$

$$P(y = 0 \mid x; w) = 1 - a_w(x)$$

A linear model for classification

We still would like to use a linear model for this particular use case, however, if you define a(x) as follows:

$$a_w(x) = w_0 + w_1 x_1 + w_2 x_2$$

$$a(x) = \sum_{i=1}^{n} w_i x_i = \overrightarrow{w}^T \overrightarrow{x}$$

The output value does not translate to a probability. We thus use the **logistic function** - a function that will transform our output into a value that can be interpreted as a probability

$$a_\theta(x) = \sigma(\overrightarrow{w}^T \overrightarrow{x})$$

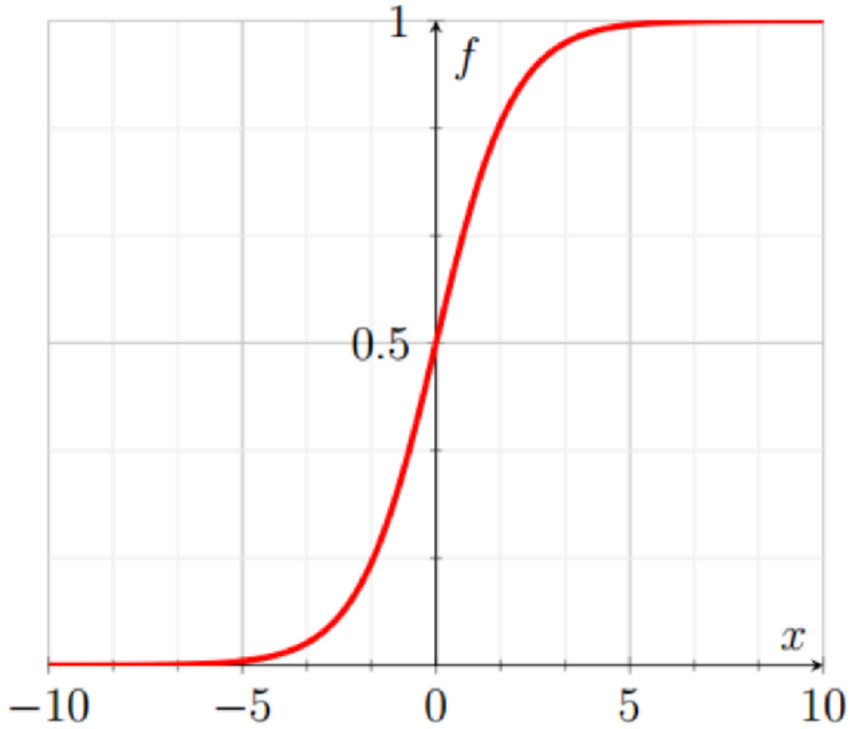
Our problem is still to find weights θ for our model

Logistic function

The logistic function is defined as follows

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Which, as you can see, it transforms any input into a value between 0 and 1



Measuring error

For this particular task, we'll measure error through cross entropy. This is a measure of how well two distributions fit. It relies on that by comparing predictions with the true labels in the following way for a given sample i:

$$l(x_i, y_i, w) = -\left[y_i \cdot logP(y_i = 1 \mid x_i, w) + (1 - y_i) \cdot log(1 - P(y_i = 1 \mid x_i, w))\right]$$

And for many samples, we just average over the metric:

$$L(X, \overrightarrow{y}, w) = \frac{1}{\ell} \sum_{i=1}^{\ell} l(x_i, y_i, w)$$

Gradient Descent

Once again, we'll apply the gradient descent. When we find the derivative of the cross entropy with respect to the weights, we end up with a similar function to the one in regression.

$$\frac{\partial}{\partial \theta_j} L(\theta) = (a_\theta(\vec{x}) - y)x_j$$

The main difference being that a(x) contains the sigmoid calculation. However, the algorithm is still pretty similar

Gradient Descent

$$\vec{w} = \vec{w}^0$$

for **k** in **n_iter**:

$$\Delta L(w) = X^T(\sigma(X\vec{w}) - y)$$

$$\vec{w} := \vec{w} - \alpha \Delta L(\theta)$$

X - matrix of example features [n_samples, n_features]

y - vector of example labels [n_samples, 1]

w - vector of weights [n_features, 1]

α - learning rate

The gradient descent method is quite useful, and it has many variations that improve over the initial algorithm. However, we don’t cover those.

We will, however, throughout the exercises, cover the concept of mini-batching, since it is one important aspect of Gradient Descent’s implementation and performance.

We’ll apply these concepts now in a few exercises

Practice time!

https://github.com/CatarinaSilva/meetup-deeplearning