LT1 - Intro

This course is about the architectures and algorithms behind neural computation, where to use them and their relationships to the human brain.

The recommended reading is as follows:

- **Deep Learning, Goodfellow et al.**, MIT Press, 2016 (online, school library)
- **Pattern Recognition and Machine Learning, Bishop**, Springer, 2007 (main library)

We will be using the following tools for the course:

- Python
- Jupyter notebooks
- Tensorflow

We will also be working through neural network training from first principles, so this may be maths heavy.

Course Details

- 100% Exam
- Optional (recommended) exercises
- p.k.lehre@cs.bham.ac.uk
- Office hours Tues 12-13, CS207
- Questions should be posed on Canvas

Tasks

ML is all about using experience to improve performance in **tasks**. Tasks are classic ML problems such as the following:

- Classification
- Regression
- Transcription
- Machine Translation etc.

Classification

Constructing a function that classifies objects into classes based on their features. Or alternatively, constructing a function which given features, returns the probability of each class.

Regression

Predicting a numerical value given some input.

$$f : R^N -> R \tag{1}$$

e.g. Prediction of car value based on mileage.

Transcription

Produce text from unstructured data.

e.g. Optical Character Recognition, Speech Recognition

Machine Translation

Translation from source language to target language. (example).

Synthesis and Sampling

Generation of new examples, similar to those in training data. This is useful in environments where it is expensive to generate content e.g. video games.

e.g. Generating art with Adversarial networks

Performance Measures, P

A performance measure P quantifies the ability of an algorithm to solve a task T. This measure depends on the task, and can be difficult to determine.

Classification

Accuracy - Proportion of correctly classified examples.

We then evaluate performance on unseen data by splitting it into two sets:

- **Training set** - Used to train a classified
- **Test set** - Used for performance evaluation

Experience, E

- ML often describes nature as an unknown probability distribution: $DoverR^d$.
- Our experience of nature are samples from the distribution: $X_1, ..., X_n\ D$
- This experience might also be called a *dataset*

Supervised Learning

The distribution D is over some set XxY where

- X is a set of **features** e.g. pictures
- Y is a set of **classes** e.g. cats, dogs

During training the algorithm is given **labelled examples**, a sequence of samples from the distribution which include both features and classes.

$$(x^{\{1\}}, y^{\{1\}}), ..., (x^{\{n\}}, y^{\{n\}}) \sim D \tag{2}$$

The goal is to predict the class given only the features i.e. *to learn / approximate a conditional probability distribution*

$$P(y|x) = \frac{^{Pr}(X, Y) \sim_D (X = x \cap Y = y)}{^{Pr}(X, Y) \sim_D (X = x)} \tag{3}$$

Given that we observed the features x, what is the probability that the corresponding class is y?

Unsupervised Learning

Now we just have a probability distribution over a set X, we observe some dataset:

- $x^{(1)}, ..., x^{(n)} \sim D$

The goal of unsupervised learning is to learn something about the distribution of the data, an example we have already been introduced to is K-means clustering, for grouping data into sets based on their features.

$$p(x) = Pr_{X \sim D}(X = x) \tag{4}$$

Or informally, what is the probability of observing the feature x in a random sample?

Reinforcement Learning

In RL the algorithm interacts with the **environment** through a sequence of **actions**, the experience of the algorithm depends both on the state of the environment and the actions.

For each action the algorithm receives a reward, which again depends on the state of the environment and the action taken. The objective is to **maximise the cumulative reward**.