Feed-forward neural networks

Sven Laur
Ways to view neural networks

- Artificial neural network models brain
  - Spike models, large scale simulations
- Artificial neural network is an easy way to repackage statistical methods
  - Neural networks solve every problem
- Artificial neural networks are nature-inspired tools for defining regression and classification models.
  - Sometimes brain is statistically sensible.
Two types of networks

- Feed-forward network have no information propagation cycles. Network has **no memory**.

- Recurrent network has many information propagation cycles. Network has **memory**.
Description of a single neuron

Output is computed in two stages:

- First inputs are aggregated
  \[ a = w_1 x_1 + \cdots + w_n x_n \]
- Then the value is mapped to output
  \[ y = f(a) = f(w_1 x_1 + \cdots + w_n x_n) \]
- Values \( w_1, \ldots, w_n \) are weights
- Activation function \( f(\cdot) \) determines main properties of the neuron.
Linear neuron and regression

Linear neuron does not change the aggregated value:

\[ y = w_1 x_1 + \cdots w_n x_n \]

- Now if we use minimal square error

\[ \text{MSE} = \frac{1}{m} \cdot \sum_{i=1}^{m} (y_i - f(x_i))^2 \]

with stohhastic gradient descent, we get ordinary least square regression implemented as perceptron learning.

- Recycling of statistical methods!
Linear neuron and classification

Using linear neuron for predicting classification labels is not good, as linear function is potentially unbounded.

Quick fix. Lets clip the linear function into range $[0, 1]$

$$f(a) = \begin{cases} 
1, & \text{if } a \geq 1, \\
 a, & \text{if } 0 < a < 1, \\
 0, & \text{if } a \leq 0,
\end{cases}$$

Rather good but introduces two angle points!

Not good if we want to compute gradients.
Using linear neuron for predicting classification labels is not good, as linear function is potentially unbounded.

**Next fix.** Let's use sigmoid function for clipping

\[ f(a) = \frac{1}{1 + \exp(-a)} \]

Sigmoid function is commonly used to model saturation effects nature. It is near linear in the centre.
Sigmoid and logistic regression

Standard mean square error can be used for doing linear classification with a sigmoid neuron.

Relative entropy is more common alternative

\[
RE = -\sum_{i=1}^{m} \ln[f(\boldsymbol{x}_i)^{y_i}(1 - f(\boldsymbol{x}_i))^{1-y_i}]
\]

Now if we compute the gradient with respect to weights then we can apply gradient descent.

This method is known as logistic regression.

- Recycling of statistical methods!
Pathological activation functions

Not all bounded functions are good for activation functions. A classical example of an ill-suited function is

\[ f(x) = \sin(wx) \]

This function can pass through any set of target points.

- Training errors is ideally zero.
- Its predictions on unseen points are arbitrary.
How much can neuron learn?

Consider the following game

- Choose $m$ data points $x_1, \ldots, x_m$ as you wish
- Train a neuron on concrete labelling
  \[ y_1, \ldots, y_m \in \{0, 1\} \]
- Count labelings neuron can perfectly reproduce

**Growth function** $\Phi(m)$ is the maximal number of valid labelings neuron can reproduce.

**Capacity of a neuron** is the maximal number $m$ such that it can reproduce all labelings:

\[
\max \left\{ m : \Phi(m) = 2^m \right\}
\]
Capacity and learning

**Observation.** We need at least $m$ data points to train a neuron with $m$ bit capacity.

- The same applies for larger networks as well
- Neurons with infinite capacity are not good.
- This is a very rough bound
- More advanced bounds do exist
- All of them are very pessimistic
Multi-layer neural network

- Map input to complex concepts
- Combine these concepts linearly
- Linearly recognisable concepts are common in the nature
- Multilayer networks can simulate decision trees
- Problem: we do not know how to define these concepts
Back-propagation

We need an efficient way to take derivatives

- Chain rule is solves all theoretical problems
- Derivatives contain many reoccurring parts
- We should cache repeating values
- This reverses information flow in the network
- Gradient evaluation is linear in the size of the net
Beyond back-propagation

Gradient descent has slow convergence rate

Newton method is has quadratic convergence rate but is much slower to evaluate (quadratic vs linear).

Hence, many trade-offs are used
- Diagonal approximation of Hessian
- Outer approximation (Levenberg-Marquart)
- Fast multiplication by Hessian

You do not implement them unless you have to!
Leverage of points revisited

A small error in a point with big leverage can make linear regression function arbitrary large

- What if we know that $-1 \leq f(x_1, x_2) \leq 1$?

Good layout

Bad layout
Two common sanity checks

If the bound holds for rectangular area $[-1, 1] \times [-1, 1]$ then we should solve the following task instead:

$$\frac{1}{m} \cdot \sum_{i=1}^{m} (y_i - f(x_i))^2 \rightarrow \min$$

s.t $|w_1| + |w_2| \leq 1$

If the bound holds for unit circle $x_1^2 + x_2^2 \leq 1$ then we should solve the following task instead:

$$\frac{1}{m} \cdot \sum_{i=1}^{m} (y_i - f(x_i))^2 \rightarrow \min$$

s.t $w_1^2 + w_2^2 \leq 1$
Lagrange’ trick

If we want to minimise $f(x)$ such that $g(x) \leq c$ for a non-negative function $g(\cdot)$, then there exists $\lambda \geq 0$ such that the solution of the original problem is a minimum for a modified function

$$f_*(x) = f(x) + \lambda g(x)$$

**Consequences**

- We can use penalty term for rectangular area
- We can use penalty term for circular area
How to approximate functions?

We can approximate functions with step functions

- This leads to decision trees

We can use saw-tooth like blocks for approximation

- There are several angle points

We can use Gaussian peaks for approximation

\[ f(x) \approx \sum_{i=1}^{k} w_i \cdot \exp \left( -\frac{1}{2\sigma_i^2} (x - t_i)^2 \right) \]
Radial basis unit

Approximation with Gaussian peaks fits the structure of multilayer neural network if we consider gaussian activation function

\[ f(a) = \exp\left(-a^2\right) \]

However, one usually talks about radial basis unit

\[ F(x) = \exp\left(-\frac{\|x - t\|^2}{2\sigma^2}\right) \]

Such unit is determined by the centre \( t \) and width \( \sigma \).
How to choose locations?

- Throw enough centres on the data range with properly width and let optimisation do its job.

- Use some intelligent pre-processing steps to find right places for the centres. Use clustering.

- Optimise RBF parameters during the learning procedure, i.e., optimise locations and widths.
Cross validation and RBF

The number of RB units and their widths depend strongly on the number of training points.

Consequently, 10-fold cross validation is not good for tuning RBF parameters. Use leave-one-out instead.