Summary

Luca Bracone

March 24, 2020

1 Basic Machine Learning

1.1 Neural Networks

Definition 1.1 (Perceptron)**.** A *Perceptron* is a function $p : \mathbb{R}^n \to \mathbb{R}$ given by

$$p(x_1, \dots, x_n) = g\left(w_0 + \sum_{i=1}^{n} w_i x_i\right)$$

Where $w_1, \dots, w_n \in \mathbb{R}$ are called the *weights* of p, w_0 is called the *bias*, and $g : \mathbb{R} \to \mathbb{R}$ is any non-linear function called the *activation*. Most of the time we use

$$g(x) = \frac{e^x}{e^x + 1}$$

the sigmoid function.

Definition 1.2 (Artificial Neural Network)**.** A (forward-feeding) *Artificial Neural Network* (ANN) is a function $T : \mathbb{R}^n \to \mathbb{R}^m$ that can be decomposed as such

$$T = l_k \circ \cdots \circ l_1$$

Where the $l_i : \mathbb{R}^{n_j} \to \mathbb{R}^{n_{j+1}}$ are called *layers* and are just vectors of perceptrons

$$l_i(\vec{x}) = (p_{i,1}(\vec{x}), \dots, p_{i,n_j}(\vec{x}))$$

. In the context of neural networks perceptrons are also called nodes.

For some ANN, T and some input $(x_1, ..., x_n)$, we have a desired output $(y_1, ..., y_n)$ we would like T to match. Denote $(\hat{y}_1, ..., \hat{y}_m)$ the output T actually produced.

Definition 1.3 (Loss Function)**.** The *loss* (or sometimes also called *cost*) of T for this input is a function that measures the "distance" between the produced and desired output. For most basic applications, the loss is defined as the l_2 norm on $\mathbb{R}^n$

$$\mathcal{L}(W; x) = \sum_{i=1}^{m} ||y_i - \hat{y}_i||_2^2$$

Definition 1.4 (Empirical Loss)**.** For a set of inputs $X = \{(x_1^1, \dots, x_n^1), \dots, (x_1^m, \dots, x_n^m)\}$, and T a NN, the *Empirical Loss* is defined as

$$J(W) = \frac{1}{|X|} \sum_{\vec{x} \in X} \mathcal{L}(W; x)$$

the average loss over X.

1.2 Gradient Descent

Any neural network is defined by the weights of its perceptrons. Therefore, in order to have the neural network behave as desired (namely being as close as possible to the desired outputs) one has to find an efficient way to tweak the weights such that the empirical loss becomes as low as possible, when tested on some new inputs. Applying this algorithm repeatedly is called training, and allows us to find a local minimum of the loss function.

1.2.1 Gradient Descent

The algorithm of gradient descent involves taking a step in the "right" direction until we reach a stopping condition, most of the time it is when the steps we are taking are too short. Consider $\frac{\partial J}{\partial w_i}$ where w_i is any weight in T, this expression represents how big of a change in J we'll obtain by moving w_i. For each weight w_i in T, we denote the vector $\left(\frac{\partial J}{\partial w_i}\right)_{i \in I}$ as ∇J. The gist of the algorithm revolves around updating the weights $W \leftarrow W - \eta \nabla J$. Where η is a constant that has to be well chosen. If too small, gradient descent stops when it meets the slightest uphill, and thus misses a better minimum which might be close by. If too big, gradient descent can only take giant steps and diverges to infinity.

In practice, calculating the derivative to some particular weight is quite simple, consider the following neural net:

$$\begin{tikzcd} x \arrow[r, "w_1"] & z_1 \arrow[r, "w_2"] & \hat{y} \arrow[r] & J(W) \end{tikzcd}$$

It's just an application of the chain rule:

$$\frac{\partial J}{\partial w_2} = \frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial w_2}$$
$$\frac{\partial J}{\partial w_1} = \frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_1} \frac{\partial z_1}{\partial w_1}$$

Since calculating ∇J is computationally intensive, consider taking a random subset of your data, and applying gradient descent to it instead. The convergence is going to be more erratic, depending on the size of the subset chosen.

1.3 Overfitting

When training our neural network, it might happen that it essentially "learns the dataset by heart" making it unable to properly tackle on never seen before data points. There are a few methods to avoid such a problem:

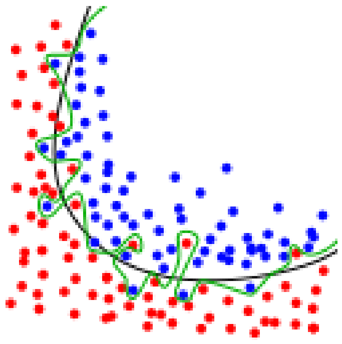


Figure 1: A parabola+noise is being misunderstood as a complicated winding line [1]

Dropout involves randomly setting some nodes to zero during training. This makes the network not rely too much on some path. *early stopping* is the process by which after each application of

[1]Courtesy of wikimedia foundation https://commons.wikimedia.org/wiki/File:Overfitting.svg

gradient descent, one tests the network. If it performs worse than it did previously, stop the training process.

2 Simplicial Complexes

Definition 2.1 (Convex Hull). Let $u_0, ..., u_k \in \mathbb{R}^n$ their *convex hull* is

$$\text{conv}\{u_0, ..., u_k\} = \left\{ x = \sum_{i=0}^{n} \lambda_i u_i \ \text{ s.t. } \sum_{i=0}^{n} \lambda_i = 1; \ \lambda_i > 0 \ \forall i \right\} \tag{1}$$

Definition 2.2 (Affinitely Independent, k-Simplex). Let $u_0, ..., u_k \in \mathbb{R}^n$. They are said to be *affinitely independent* if any point in their convex hull, when written as in (1) is uniquely written as such. In that case instead of convex hull, we speak of *k-simplex.*

Definition 2.3 (Faces and Cofaces). Let $\{u_{i_0}, ..., u_{i_m}\} \subseteq \{u_0, ..., u_k\}$ we say that $\tau = \text{conv}\{u_{i_1}, ..., u_{i_m}\}$ is a *face* of $\sigma = \text{conv}\{u_1, .., u_k\}$ which we write $\tau < \sigma$. Equivalently σ is a *co-face* of τ.

Definition 2.4 (Simplicial Complex). A *simplicial complex* K is a collection of simplices such that:

(S1) $\forall \sigma \in K, \ \tau < \sigma \implies \tau \in K$.

(S2) $\forall \sigma, \tau \in K, \ \sigma \cap \tau$ is either empty or a face.

The *dimension* of K is $\dim K = \max_{\sigma \in K} \dim \sigma$ and the dimension of a k-simplex is k. The *underlying space* of K, $|K|$ is the topological space $\bigcup_{\sigma \in K} \sigma$ with the subspace topology.

Definition 2.5 (Triangulation). Let X be a topological space and K a simplicial complex, if there exists a homeomorphism $\phi : X \to |K|$ we say that X is triangulable. In that case the couple (K, ϕ) is called a triangulation of X

Definition 2.6 (sub-Complex). A subset $L \subseteq K$ is a *subcomplex* of K, if it is itself a complex. It is called full if it contains all vertices of K.

Example 2.1 (Skeleton). The *j-skeleton* of a simplicial complex K is $K^{(j)} = \{\sigma \in K \text{ s.t. } \dim \sigma \leq j\} \subseteq K$. In particular the 0-skeleton is called vert(K) and it consists of the vertices of K

Definition 2.7 (Star and Link). Let K be a simplicial complex and pick $\sigma \in K$. Its *star*, $st\sigma = \{\tau \in K \text{ s.t. } \tau < \sigma\}$. unfortunately it's not a sub-complex of K because it's not closed under taking faces. For that we define the *closed star* $\overline{st}\sigma$ the smallest sub-complex that contains the star. Similarly, the *link* of a simplex $lk\sigma = \{\tau \in \overline{st}\sigma \text{ s.t. } \tau \cap \sigma = \emptyset\}$ the set of simplices in the closed star that don't touch σ

Definition 2.8 (Simplicial Maps and Homeomorphisms). Let K, L be two simplicial complexes and $\phi : \text{vert}K \to \text{vert}L$ a map such that vertices of every simplex in K gets mapped to a vertex of a simplex in L, in that case ϕ is called a *vertex map.*The map $f : |K| \to |L|$ defined by $\sum_{i=0}^{n} \lambda_i u_i \mapsto \sum_{i=0}^{n} \lambda_i \phi(u_i)$ is called a *simplicial map.* If ϕ is bijective and ϕ^{-1} is also a vertex map, we call it instead *simplicial homeomorphism.*

3 Simplicial Homology

Definition 3.1 (p-Chain). Let K be a simplicial complex. A *p-chains* is a formal sum of p-simplices of K over a field:

$$c = \sum_{i=1}^{n} a_i \sigma_i$$

For the time being, the field in question is $\mathbb{F}_2$. Componentwise addition: $c + d = \sum_{i=1}^{n}(c_i + d_i)\sigma_i$ makes the set of p-chains into a commutative group which we note as $(C_p, +)$.

Definition 3.2 (Boundary)**.** the *Boundary* of a p-simplex is a map $\partial_p : C_p \to C_{p-1}$ which for a p-simplex returns the sum of the $(p-1)$-faces:

$$\partial_p\sigma = \sum_{j=1}^{n}[u_1, \dots, \hat{u_j}, \dots, u_p]$$

where $[u_1 \dots u_p]$ is the p-simplex given by the convex hull of the $\{u_1, \dots, u_p\}$ and $\hat{u}_j$ means we omit the j-th term from that notation. This definition extends by linearity to the p-chains:

$$\partial_p c = \sum_{i=1}^{n} a_i \ \partial_p\sigma_i$$

Turns out $\partial_p : C_p \to C_{p-1}$ is a group homomorphism. for a simplicial complex K we define its chain complex:

$$\mathcal{C}_K : \ \cdots \longrightarrow C_p \xrightarrow{\partial_p} C_{p-1} \xrightarrow{\partial_{p-1}} \cdots \xrightarrow{\partial_1} C_0$$

Fundamental Lemma of Homology. *For all p and $d \in C_{p+1}$, we have $\partial_p\partial_{p+1}d = 0$*

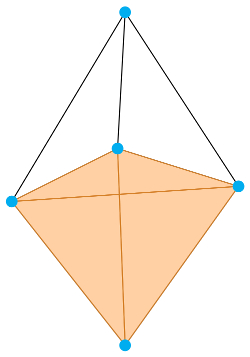
Whenever we have an interesting homomorphism it is natural to want to know more about its kernel and image, in this case they have special names:

Definition 3.3 (p-Boundaries and p-Cycles)**.** *p-cycles* are the elements of $\ker \partial_p = \{c \in C_p \text{ s.t. } \partial_p c = 0\} = Z_p$
p-boundaries are the elements of $\operatorname{Im} \partial_{p+1} = \{c \in C_p \text{ s.t. } \exists d \in C_{p+1},\ c = \partial_{p+1}d\} = B_p$

Remark 3.1. Whenever it is clear, we favor the notation ∂ over ∂_p, omitting the p. Furthermore, note that if $c \in B_p$ then $\partial c = \partial\partial d = 0$. Thus we notice that $B_p \subseteq Z_p$. But in general $B_p \neq Z_p$ in particular we are interested in the p-cycles that are not p-boundaries.

Definition 3.4 (Homology Groups and Betti numbers)**.** For a complex K its p-th *homology group* is the quotient $H_p = Z_p/B_p$.
The p-th *Betti number* is β_p the number of generators of H_p.

Example 3.1. Consider the simplex drawn in picture 3.1 on page 5, it's the union of two tetrahedrons except that for the upper one we removed its volume as well as all its faces, while for the bottom one we remove its volume. Let's calculate its homology groups.
First notice that since there is only one connected component, $H_0 = 0$. Indeed, the 0-cycles are the collections of vertices whose cardinality is even. Furthermore, any two vertices v, w can be joined by walking along edges $e_1, \dots, e_n$. In mathematical terms this means that $\partial c = \sum_{i=1}^{n} \partial e_i = v, w$. This means that every 0-cycle is also a 0-boundary which implies $Z_0 = B_0$ which implies $H_0 = 0$.
Now, for H_1; we notice that the 1-chain made up of the 1-simplices that form a face of one of the "empty triangles" are cycles, but that they are not the boundary of any 2-chain. Thus for each empty face, we have a generator of H_1. There are no other generators because any other 1-cycle can be molded into one that we have already found because all the other faces are full. So, H_1 is the free group generated by $\{f, g, h\}$ where f, g, h are the 1-chains that each correspond to one of the empty traingles in the upper tetrahedron.
Lastly, for H_2 there is only one 2-cycle that is not a 2-boundary, which is the "shell" of the bottom tetrahedron, because we specifically made it empty. Therefore, H_2 is the free group generated by the 2-chain that corresponds to that shell. If that volume were to be full, H_2 would instead be trivial.
The group H_3 is trivial because there are no 3-chains.
We see that the Betti numbers are $\beta_0 = 0$, $\beta_1 = 3$, $\beta_2 = 1$, $\beta_n = 0$ for all $n \geq 3$



Definition 3.5 (Induced Map)**.** Consider two complexes K, L and a simplicial map $f : K \to L$. We have a way to transform f into a map $f_\sharp : C_p(K) \to C_p(L)$, in this way

$$c = \sum_{i=1}^{n} a_i \sigma_i \ \mapsto \ \sum_{i=1}^{??} a_i \tau_i$$

Where $\tau_i = f(\sigma_i)$ if it has dimension p or $\tau_i = 0$ otherwise. We call $f_\sharp$ the *induced map* of f.