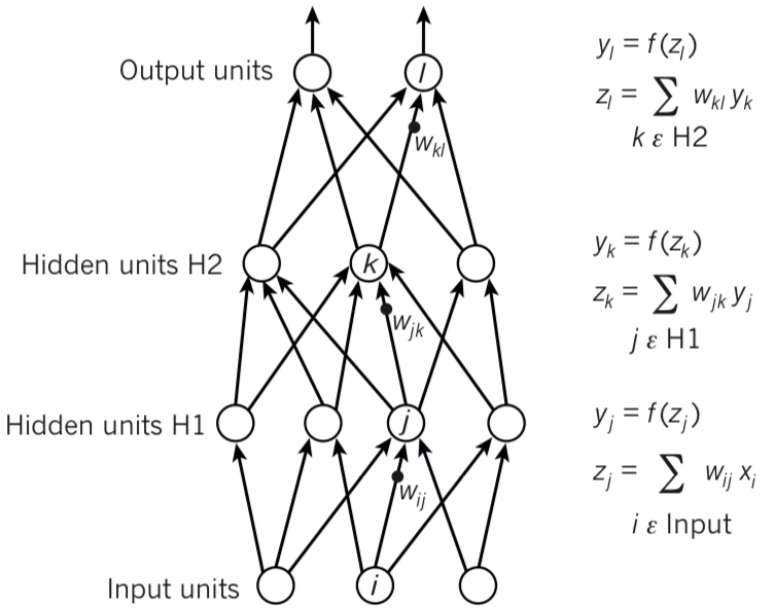

Figure 1: An example of neural network structure map. In this particular case there are three input variables, two hidden layers consisting of four and three neurons respectively and two output variables. Note that in standard nnet package there is only one hidden layer. Source **?**

1 Nnet

Neural networks were first developed in the middle of the twentieth century, had great importance for a few decades and then overshadowed by other algorithms such as support vector machines in thelater decades of the last century until they were discovered to be great in performance when paired with backpropagation especially for unsupervised learning. The idea behind this category of algorithms is that most real life problems are highly non linear, and therefore standard models such as linear regression will not be suited for approximation of reality. The way this class of algorithms work is calculating numerous weighted linear combinations of the input variables and applying a non linear function to it. This procedure is repeated multiple times depending on the design of the network, each of these basic operation is called a neuron. The group of neurons on a same level is called hidden layer, there can be multiple hidden layers each performed taking the outputs of the previous one. Neural networks not only depends on the number of neurons for each hidden layer, number of layers and definition of input variables for each neuron. In fact they are defined also by which nonlinear function is applied in each neuron: the traditionally used are sigmoid functions such as hyperbolic tangent and logistic function

$$f_1(t) := \tanh(t) \quad f_2(t) := \frac{1}{1+e^{-t}}$$

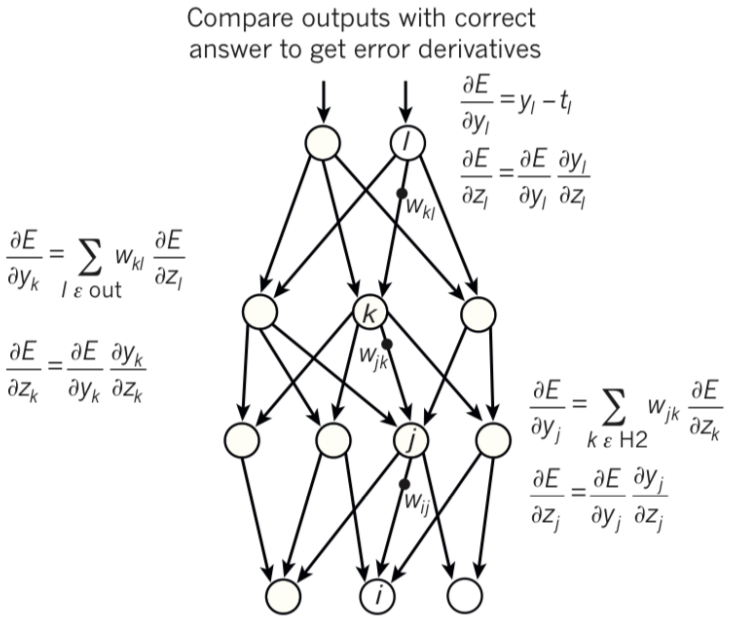

Figure 2: Diagram on how the backpropagation procedure is carried out. Starting from the error $y_l - t_l$, the chain rule of derivatives is used to explicit a variation in the error as function of variations in the input variables. Note that the error is taken as the differentiation of a cost function: in this case $\frac{1}{2}(y_l - t_l)^2$

.

The reason for the application of such functions for each elementary step is an attempt to linearise the boundary of highly nonlinear sets. Backpropagation is one of the milestones of the development of neural networks and plays a fundamental role especially for deep networks, which consist of multiple hidden layers. This process aims to find out the most appropriate weights $w_{i,j}$ for the construction of each neuron in the net (see Figure 1). To start with an objective function, generally the loss function, is differentiated and used as in Figure 2 together with the chain rule to find partial derivatives of the error function with respect to the weights $w_{i,j}$ for each stage and using gradient descent method with learning rate η to define new weights

$$\hat{w}_{i,j} := w_{i,j} - \eta \frac{\partial E}{\partial w_{i,j}}$$

One might think that local minima might be a problem, luckily *"Regardless of the initial conditions, the system nearly always reaches solutions of very similar quality. Recent theoretical and empirical results strongly suggest that local minima are not a serious issue in general."* (C.f. **?**). Once the weights and structure of the network are determined it is a mere substitution that will produce the results for each different input.

2 Random Forest

Definition 2.1. *A tree-structured classifier $h_{\theta_k}(\boldsymbol{x})$ is a generalised step function $h_{\theta_k} : \boldsymbol{X} \to F \subset \mathbb{R}$ where $\boldsymbol{X}$ is the data set and F is a finite cardinality set. This function has constant values on a partition of the data set consisting of hyper-rectangles.*

Note that θ_k indicates how to create such a partition by determining

- predictors to split on
- values of the splits
- depth of the tree

The concept of generalization error for a random forest was first introduced in **?**.

Definition 2.2. *A random forest is a classifier consisting of a collection of tree-structured classifiers $\{h(\boldsymbol{x}, \theta_k) \mid k \in \overline{n}\}$ where the θ_k are i.i.d. random vectors and each tree casts a unit vote for the most popular class at input $\boldsymbol{x}$.*

Since the results we are going to prove hold for any type of ensemble, not only of tree-structured classifiers, $h(\mathbf{x}, \theta_k)$ can be replace with a generic classifier $h_k(\mathbf{x})$ and viceversa.

Definition 2.3. *Given $\{h_k(\boldsymbol{x}) \mid k \in \overline{K}\}$, the training set drawn at random from the distribution $\boldsymbol{X}$ and its realisation Y (through $h(\cdot)$), define the margin function as*

$$mg(\boldsymbol{X}, Y) = \sum_{k=1}^{K} \frac{I(h_k(\boldsymbol{X}) = Y)}{K} - \max_{j \neq Y} \sum_{k=1}^{K} \frac{I(h_k(\boldsymbol{X}) = j)}{K}$$

where $I(\cdot)$ is the indicator function.

The margin measures the extent to which the average number of votes at $\mathbf{X}, Y$ for the right class exceeds the average vote for any other class. The larger the margin, the more confidence in the classification.
The generalization error is given by

$$\mathbb{P}(mg(\mathbf{X}, Y) < 0)$$

where the probability is over the $(\mathbf{X}, Y)$ space. For a large number of trees, it follows from the Strong Law of Large Numbers and the tree structure that:

Theorem 2.1. *As the number of trees K increases, almost surely the generalization error converges to*

$$\mathbb{P}_{\boldsymbol{X},Y}(\mathbb{P}_\theta(h(\boldsymbol{X},\theta)=Y)-\max_{j\neq Y}\mathbb{P}_\theta(h(\boldsymbol{X},\theta)=j)<0)$$

which assures the method does not overfit for big ensembles of trees.

Proof. By definition of almost sure convergence, it suffices to show that there is a set of probability zero C on the sequence space $\{\theta_k \mid k \in K\}$ such that outside of C, $\forall x$,

$$\sum_{k=1}^{K}\frac{I(h(\mathbf{x},\theta_k)=j)}{K}\to\mathbb{P}_\theta(h(\mathbf{x},\theta)=j)$$

For a fixed training set and fixed θ , $\{x \mid h(x,\theta)=j\}$ is a union of hyper-rectangles. This follows from the definition of decision tree. Moreover, $\forall\ h(x,\theta)\ \exists!T<\infty$ number of such unions of hyper-rectangles, denoted by $S_1,...,S_T$. Define $\phi(\theta)=t$ iff $\{x \mid h(x,\theta)=j\}=S_t$. Let K_t be the number of times that $\phi(\theta_k)=t$ in the first K trials. Then

$$\sum_{k=1}^{K}\frac{I(h(\mathbf{x},\theta_k)=j)}{K}=\sum_{t}\frac{K_tI(x\in S_t)}{K}$$

By the Law of Large Numbers,

$$K_t=\sum_{k=1}^{K}\frac{I(\phi(\theta_k)=t)}{K}$$

converges almost surely to $\mathbb{P}_\theta(\phi(\theta)=t)$. Taking unions of all the sets on which convergence does not occur for some value of k gives a set C of zero probability such that outside of C,

$$\sum_{k=1}^{K}\frac{I(h(\mathbf{x},\theta_k)=j)}{K}\to\sum_{t}\mathbb{P}_\theta(\phi(\theta)=t)I(x\in S_t)$$

The right hand side is $\mathbb{P}_\theta(h(\mathbf{x},\theta)=j)$ □