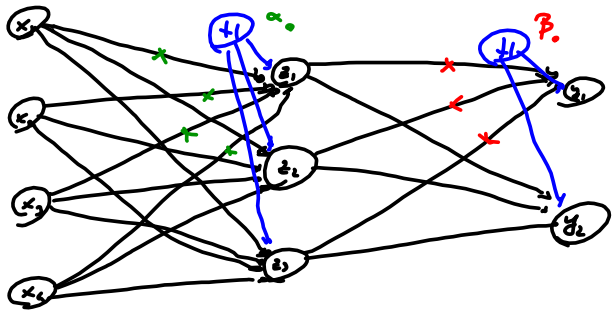
Neural networks (an introduction)



We want to relate p input variables x to the q output variables y through r latent variables z

input variables — p

hidden layer of latent variables — r

output variables — q

Essentially a neural network is a two-stage regression scheme, generally involving nonlinear effects

$$z_j = f_0\left(\alpha_0 + \sum_{h \to j} \alpha_{hj} x_h\right) \qquad j = 1, \dots, r$$

(α_{hj}: the edge between x_h and z_j)

$$y_k = f_1\left(\beta_0 + \sum_{j \to k} \beta_{jk} z_j\right) \qquad k = 1, \dots, q$$

where $\alpha_0, \beta_0, \alpha_{hj}, \beta_{jk}$ are parameters to be estimates, while f_0 and f_1 are defined functions (activation function), usually

$$f_0(v) = \frac{1}{1+e^{-v}}$$ sigmoid

$$f_0(v) = \tanh(v) = \frac{e^v - e^{-v}}{e^v + e^{-v}}$$ hyperbolic tangent

$$f_0(v) = \begin{cases} 0 & \text{for } v < 0 \\ v & \text{for } v \geq 0 \end{cases}$$ REctified Linear Unit (RELU)

$$f_0(v) = \begin{cases} 0.01\, v & \text{for } v < 0 \\ v & \text{for } v \geq 0 \end{cases}$$ Leaky RELU

While for regression, $f_1(v) = v$. we go back to linear regression

$f_0(v)$ and $f_1(v)$ cannot really be both linear, otherwise

Important property: neural networks with linear outcome can approximate any continuous function in a compact set, by increasing the number of units in the hidden layer

Extensions:
- use more than one hidden layer
  - use in practice
  - better to have multiple layers with few units than few layers with a lot of units.
  - remove edges, not necessary a complete graph.

We need to find the values of (r), α and β
no easy solution / accepted criterion → in practice try and fail

Once r is fixed, we can estimate α and β by minimizing the deviance

$$D = \sum_{i=1}^{n} \| y_i - f(x_i) \|^2$$

where y_i is the q-dimensional vector of response for observation i, and $f(x_i)$ is a q-dimensional vector which elements are

$$f_k(x) = f_1\left(\beta_0 + \sum_{j \to k} \beta_{jk} \underbrace{f_0\left(\alpha_0 + \sum_{h \to j} \alpha_{hj} x_h \right)}_{z_j} \right) \qquad k = 1, \dots, q$$

More elaborate versions of neural networks actually use penalized versions of D

$$D_p = D + \lambda J(\alpha, \beta)$$

where λ is the tuning parameter and $J(\alpha, \beta)$ the penalty, usually

$$J(\alpha, \beta) = \int \sum_{h,k} \frac{\partial y^2}{\partial^2 x_h} dx \approx \frac{1}{n} \sum_{i=1}^{n} \sum_{h,k} \frac{\partial y^2}{\partial x_h \partial x_h}$$

or

$$J(\alpha, \beta) = \|\alpha\|^2 + \|\beta\|^2$$ ← weight decay (shrinkage toward 0)

Choice of λ: not use Cross-Validation
default: value between 10^{-4} and 10^{-2}

to make sense this kind of penalty, we need to scale the variables

The minimization of D_p is done numerically → back-propagation algorithm

there are usually a lot of local minima => bad results with CV
=> repeat the procedure starting from different points.

Neural networks for classification

Straightforward extension: the activation function $f_k(v)$ must have $(0;1)$ as codomain

↳ softmax $$f_{1K}(v) = \frac{\exp\{\beta_0 + \sum_{j \to K} \beta_{jK} z_j\}}{\sum_{r=0}^{K-1} \exp\{\beta_0 + \sum_{j \to r} \beta_{jr} z_j\}}$$ $f(v) = v$ regression

- Obviously, the deviance is not anymore the Gaussian, but the entropy (binomial)
- the default choice of λ, is between 10^{-3} and 10^{-1}

Advantages of nn

- flexibility (related to the concept of universal approximator)
- compactness of representation
- sequential update

Disadvantages

- arbitrariness (how to choose r and λ, e.g.)
- instability (local minima)
- no inferential tools
- major problems with interpretability (especially with several layers)

Support vector machines $\in \mathbb{R}^{p-1}$ (classification)

We want to find that hyperplane (line in $\mathbb{R}^2$) among those which separate sets of points of different classes, that maximizes the distance from the closest representative of each class (m).

In general, an hyperplane in $\mathbb{R}^p$ is defined as

$$a + b^T x = 0 \qquad x \in \mathbb{R}^p$$

and therefore identified by a and b, where $a \in \mathbb{R}$, $b \in \mathbb{R}^p$

Properties

- at each point of the hyperplane, $b^T x = -a$
- if x' and x'' are two points of the hyperplane, $b^T(x' - x'') = 0$
- therefore b is orthogonal to the hyperplane, with $\hat{b} = \frac{b}{\|b\|}$ the unit-norm vector
- the signed distance from a point x to the hyperplane

$$\hat{b}^T (x - x_0) = \frac{1}{\|b\|}(b^T x + a)$$

where x_0 is the projection of x on the hyperplane.

$a + b^T x' = 0$, $a + b^T x'' = 0$ / $b^T(x' - x'') = 0$

With $K = 2$, $y \in \{-1, 1\}$, $\beta_0 + x^T \beta = 0$ being the general separating hyperplane and w.l.g. $\|\beta\| = 1$

for a fixed choice of the hyperplane, an unit $(\tilde{x}, y)$ is classified

correctly: $\tilde{y}(\beta_0 + \tilde{x}^T \beta) > 0$

$1 \times 1 > 0$

$(-1) \times (-1) > 0$

incorrectly: $\tilde{y}(\beta_0 + \tilde{x}^T \beta) < 0$

$1 \times (-1) < 0$

$(-1) \times 1 < 0$

then our optimization problem is

$$\max_{\beta_0, \beta} m \quad \text{subject to} \quad \begin{cases} \|\beta\| = 1 \\ y_i(\beta_0 + \tilde{x}_i^T \beta) \geq m \end{cases} \qquad i = 1, \dots, n$$

where $2m$ is called the margin.

We can rewrite the optimization problem as

$$\max_{\beta_0, \beta} m \quad \text{subject to} \quad y_i(\beta_0 + \tilde{x}_i^T \beta) \geq m \|\beta\|$$

Since the multiplication for a constant does not change the constrain, we can set $\|\beta\| = \frac{1}{m}$

$$\min_{\beta_0, \beta} \frac{1}{2}\|\beta\|^2 \quad \text{subject to} \quad y_i(\beta_0 + \tilde{x}_i^T \beta) \geq 1$$

↳ same to minimize $\sqrt{\beta_1^2 + \dots + \beta_p^2}$ or $\frac{1}{2}(\beta_1^2 + \dots + \beta_p^2)$

The optimization problem is reduced to a quadratic minimization with a linear constrain (easily solved by standard software methods)

→ maximal margin classifier

- cases in which there is an hyperplane perfectly seperating the classes are rare
  - deal with more realistic situations;
  - from maximal margin classifier to support vector machine

The idea is to introduce some auxiliary non-negative variables $\xi_1, \dots, \xi_n$ which express how far on the "wrong" side the points are from the margin (if the observation is in the "right" side or on the margin, $\xi_i = 0$)

The constrain we had $y_i(\beta_0 + \tilde{x}_i^T\beta) \geq m$ is substituted with

$$y_i(\beta_0 + \tilde{x}_i^T\beta) \geq m(1 - \xi_i)$$

and following the same procedure as before

$$\min_{\beta_0, \beta} \frac{1}{2}\|\beta\|^2 + \gamma \sum_{i=1}^{n} \xi_i \quad \text{subject to} \quad \begin{cases} y_i(\beta_0 + \tilde{x}_i^T\beta) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{cases} \quad \forall i = 1, \dots, n$$

where γ is a regularization parameter which assigns the cost of violations of the barriers. It can be shown that the solution of the minimization problem is

$$\hat{\beta} = \sum_{i=1}^{n} a_i y_i \hat{x}_i$$

where a_i is non-zero only for the observations which satisfy $*$ (support vectors)

Note: as often, it is useful to consider transformation of x

$$h(x) = (h_1(x), \dots, h_q(x))^T \qquad x \in \mathbb{R}^p$$

where q is not necessarily equal to p.