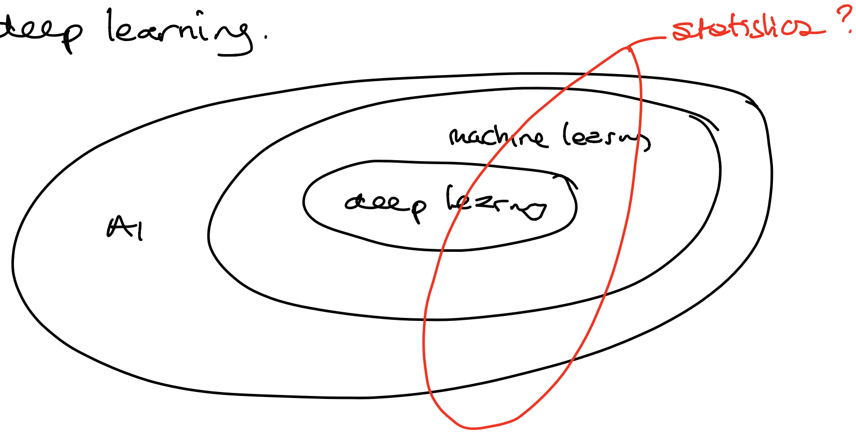
M11: Neural networks 25.03.2019 TMA4268

AIM: Understand the basic building blocks of (artificial) neural networks - and help the democratization of deep learning.



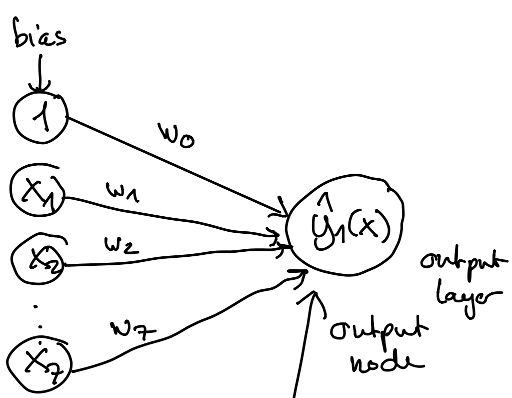
MULTIPLE LINEAR REGRESSION

Ⓢ statistics
ⓃⓃ neural networks

MODEL

Ⓢ $Y_i = x_i^T \beta + \varepsilon_i$, ε i.i.d $N(0, \sigma^2)$

ⓃⓃ



input layer

input nodes

first we sum, then apply ϕ_0

activation function in output layer

$$\boxed{\hat{y}_1(x) = \phi_0\left(w_0 + w_1 \cdot x_1 + \dots + w_7 \cdot x_7\right)}$$

output

$\phi_0(a) = a$

$\left[\hat{y}_1(x_i) = \phi_0(w_0 + w_1 x_{i1} + \dots + w_7 \cdot x_{i7})\right]$

METHOD

(S) $\underset{\beta}{\text{minimize}} \sum_{i=1}^{n} (y_i - x_i^T \beta)^2 \Rightarrow \hat{\beta} = (X^T X)^{-1} X^T Y$

(NN) $$J(w) = \frac{1}{n} \sum_{i=1}^{n} (\underset{\text{obs}}{y_i} - \underset{\text{predicted}}{\hat{y}_1(x_i)})^2$$

$J(w)$ ↑ loss function

Gradient-based optimization

First: Assume we only have one w to estimate.

$J(w)$ = function of w

$J'(w) = \frac{dJ(w)}{dw}$ gives the slope of $J(w)$ in w

We may reduce $J(w)$ by moving in the direction of $-J'(w)$

⇒ do that and move to some (local) minimum

Let $w^{(0)}$ be our initial guess of the w that minimizes $J(w)$,

Calculate $J'(w^{(0)})$ and we move to $w^{(1)} = w^{(0)} - \lambda J'(w^{(0)})$

↑ step size "learning rate"

But, we have more than one w, here (p+1) w's

$$\nabla_w J(w) = \begin{bmatrix} \frac{\partial J}{\partial w_0} \\ \frac{\partial J}{\partial w_1} \\ \vdots \\ \frac{\partial J}{\partial w_p} \end{bmatrix} (w) = \text{gradient of } J \text{ at } w$$

↑ vector containing the partial derivatives

To minimize J we find the direction in which J decreases the fastest → this is in the direction of the negative of the gradient

→ we decrease J by moving in the direction of the negative of the gradient. We are at $w^{(t)}$ and we move

$$w^{(t+1)} = w^{(t)} - \lambda \cdot \left(\nabla_w \mathcal{J}(w^{(t)})\right)$$

can be computed efficiently

backpropagation

the method of steepest descent

learning rate
can be found by line search
or just set to some small value

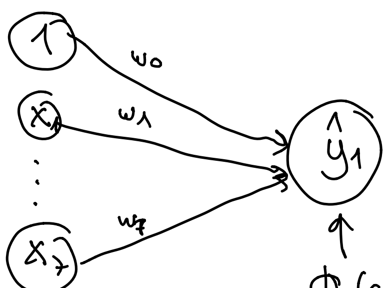
LOGISTIC REGRESSION

MODEL:

(S) $$Y_i = \begin{cases} 1 & p_i \\ 0 & 1-p_i \end{cases} \quad \text{where } p_i = \frac{1}{1+\exp\left(-(\beta_0 + \beta_1 x_{i1} + \dots + \beta_r x_{ir})\right)}$$

(NN) $$\hat{y}_1(x) = \phi_0 (w_0 + w_1 x_1 + \dots + w_r x_r) \qquad \phi_0(a) = \frac{1}{1+\exp(-a)}$$

sigmoid



$$\phi_0(a) = \frac{1}{1+\exp(-a)}$$

METHOD:

(S) maximum likelihood : $\ln(L(\beta)) = \sum_{i=1}^{n} \left(y_i \cdot \ln p_i + (1-y_i)\ln(1-p_i)\right)$

This time no closed form solution → use gradient based method – with information of curvature – to find β.

(NN) Loss function $\propto$ negative of loglikelihood

$$J(w) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \cdot \ln\left(\hat{y}_1(x_i)\right) + (1-y_i)\ln\left(1-\hat{y}_1(x_i)\right)\right]$$

binary cross-entropy loss

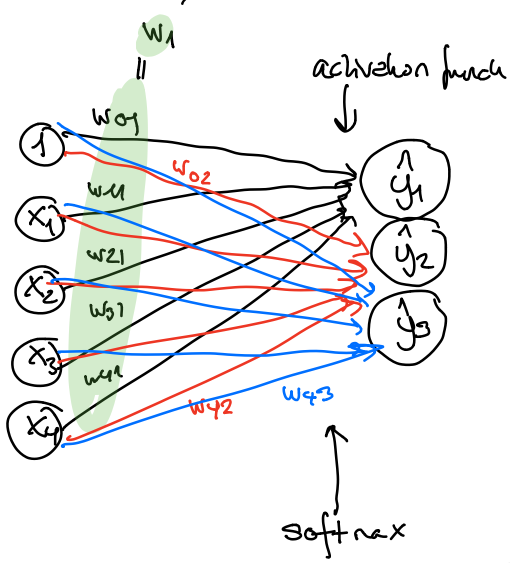
Find w using gradient descent!

MULTICLASS REGRESSION

(S) Generalization of logistic regression to C classes

Iris: 3 species, 4 covariates + 1 intercept: $5 \cdot 2 = 10$ parameters

(NN)



"one-hot" coding of the response y

1	0	0
0	1	0
0	0	1
class 1	2	3

We need to estimate 15 parameters

$$\hat{y}_c = \frac{\exp(x^T w_c)}{\sum_{s=1}^{C} \exp(x^T w_s)}$$

METHOD:

(S) maximize loglikelihood using Fisher scoring

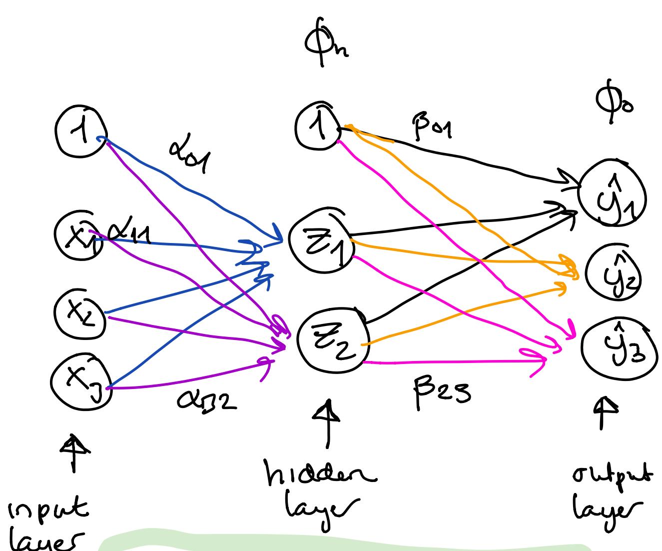
$$\sum_{i=1}^{n} \sum_{c=1}^{C} y_{ic} \ln(p_{ic})$$

(NN) $$J(w) = -\frac{1}{n} \sum_{i=1}^{n} \frac{1}{C} \sum_{c=1}^{C} y_{ic} \cdot \ln(\hat{y}_c(x_i))$$

categorical cross entropy

Minimize $J(w)$ with gradient descent.

FEEDFORWARD NETWORKS



3 inputs
2 hidden nodes
3 output nodes

$$\begin{bmatrix} p=3 \\ M=2 \\ C=3 \end{bmatrix}$$

$$z_m(x) = \phi_h\left(\alpha_{0m} + \sum_{j=1}^{p} \alpha_{jm} \cdot x_j\right)$$

$$\hat{y}_c(x) = \phi_o\left(\beta_{0c} + \sum_{m=1}^{M} \beta_{mc} z_m(x)\right)$$

fully connected (dense) network

$$\boxed{\alpha_{jm} \quad j = 0, 1, \dots, p = 3 \qquad \beta_{mc} \quad m = 1, \dots, M = 2}$$

← parameters { α from input to hidden; β from hidden to output }

$$\hat{y}_c(x) = \phi_o\left(\beta_{0c} + \sum_{m=1}^{M} \beta_{mc} \cdot \phi_h\left(\alpha_{0m} + \sum_{j=1}^{p} \alpha_{jm} \cdot x_j\right)\right)$$