DAT200

Sammendrag av klassifikasjonsmodeller.

Perceptron

η: learning rate. How big the weight updates should be.

Classes:

$\boxed{-1 \quad 1}$

Threshold value:

$$\boxed{z = w_0x_0 + w_1x_1 + \ldots + w_mx_m \qquad w_0 = -\theta,\ x_0 = 1}$$

Threshold function:

$$\boxed{\phi(z) = \begin{cases} 1 & z \geq 0 \\ -1 & otherwise. \end{cases}}$$

$\phi(z)$ was originally 1 for $z \geq \theta$ (therefore $w_0 = -\theta$)

Weight update:

$$\boxed{w_j = w_j + \Delta w_j}$$

Weight change:

$$\boxed{\Delta w_j = \eta(y^i - \hat{y}^i)x_j^i}$$

Adaline

η: learning rate. How big the weight updates should be.

Classes:

$\boxed{-1 \quad 1}$

Threshold value:

$$\boxed{z = w_0x_0 + w_1x_1 + \ldots + w_mx_m \qquad w_0 = -\theta,\ x_0 = 1}$$

Activation function:

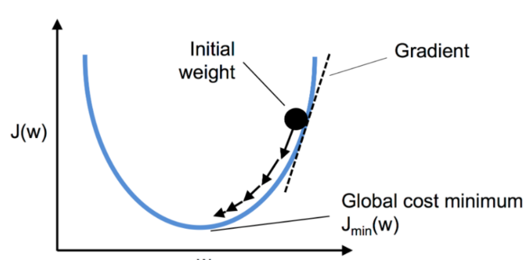
$$\boxed{\phi(z) = \phi(w^Tx) = w^Tx \text{ (linear activation function)}}$$

Threshold function:

$$\boxed{\varphi(z) = \begin{cases} 1 & \phi(z) \geq 0 \\ -1 & otherwise. \end{cases}}$$

Cost function:

$$\boxed{J(w) = \frac{1}{2}\sum_i \left(y^i - \phi(z^i)\right)^2}$$



Weight change:

$$\boxed{\Delta w = -\eta \nabla J(w)} \qquad \boxed{\Delta w_j = \eta \sum_i (y^i - \phi(z^i))x_j^i}$$

(where $\nabla J = -\sum_i (y^i - \phi(z^i))x_j^i$)

Perceptron vs. Adaline

Similarities:
→ Binary classification.
→ Linear decision boundary.
→ Threshold function ($\phi(z)$).
Differences:
→ Perceptron uses a step function ($\phi(z)$), Adaline uses a linear activation function ($\phi(z)$).
→ Perceptron compares true class labels to predicted labels, Adaline compares true class labels to continuous output from $\phi(z)$.
→ Perceptron updates weights immediately after misclassification, Adaline updates all weights at the end of each iteration.

Logistic regression

η: learning rate. How big the weight updates should be.

Classes:

$$\boxed{0 \quad 1}$$

Threshold value:

$$\boxed{z = w_0x_0 + w_1x_1 + \ldots + w_mx_m \qquad w_0 = -\theta,\ x_0 = 1}$$

Activation function:

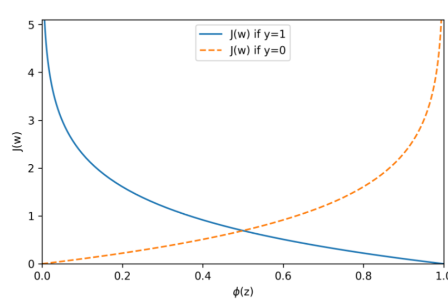
$$\boxed{\phi(z) = \frac{1}{1+e^{-z}} \text{ (sigmoid activation function)}}$$

Threshold function:

$$\boxed{\varphi(z) = \begin{cases} 1 & \phi(z) \geq 0.5 \\ 0 & otherwise. \end{cases}}$$

Cost function:

$$\boxed{J(w) = \sum_{i=1}^{n} \left[-y^i log(\phi(z^i)) - (1-y^i) log(1-\phi(z^i))\right]}$$



For $y^i = 0$:

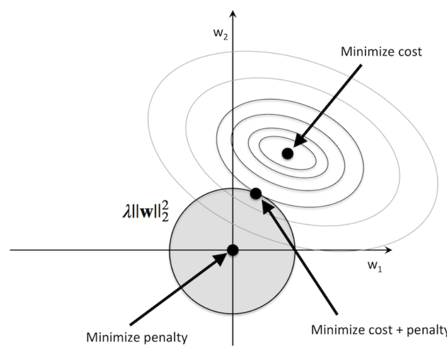
$$\boxed{J(w) = -log(1-y_{pred}) = -log(1-\phi(z^i))}$$

> For $y^i = 1$:
>
> $\boxed{J(w) = -log(y_{pred}) = -log(\phi(z^i))}$
>
> Weight change:
>
> $\boxed{\Delta w = -\eta \nabla J(w)}$

Regularisation

C: regularisation strenght. How greatly to punish large weights.

> Cost function:
>
> $\boxed{J(w) + \frac{\lambda}{2}||w||^2 = J(w) + \frac{1}{C}||w||^2 = J(w) + \frac{1}{C}\sum w_j^2}$



Support vector machines (SVM)

C: error penalisation. How greatly to punish misclassifications.

> Margin:
>
> $\boxed{\frac{w^T(x_{pos} - x_{neg})}{||w||} = \frac{2}{||w||}}$
>
> Goal: minimise $\frac{2}{||w||}$

Radial Basis Function Kernel SVM (RBF Kernel-SVM)

γ: penalise misclassifications. ($||x^i - x^j||^2$ is the (Euclidean) distance between two points.)

> Kernel function:
>
> $\boxed{\kappa(x^i, x^j) = exp\left[-\frac{||x^i - x^j||^2}{2\sigma^2}\right] = exp[-\gamma||x^i - x^j||^2]}$

Generally

Hyperparameter: large values = overfitting. (Goal: Penalises error)
Cost function: find (global) minimum. (Goal: Minimise cost)