TDA231
Multiclass Regression and Introduction to Neural Networks

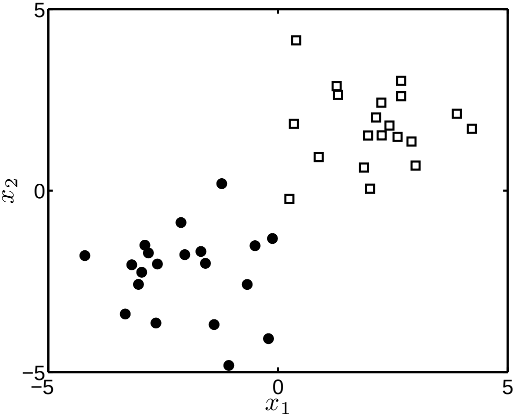
Devdatt Dubhashi

dubhashi@chalmers.se

Dept. of Computer Science and Engg.
Chalmers University

February 19, 2016

Some data

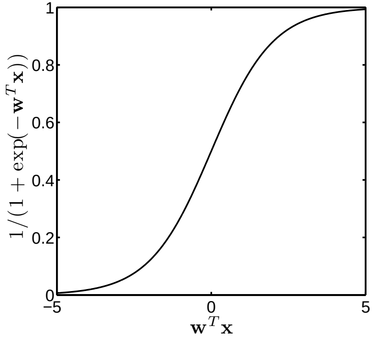


$\sigma(\cdot)$

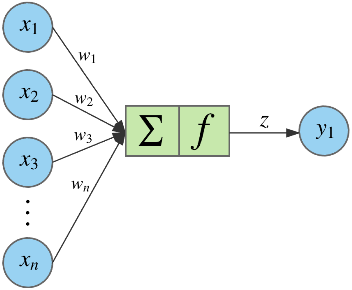
- For logistic regression (binary), we use the sigmoid function:

$$P(T = 1|\mathbf{x}, \mathbf{w}) = \sigma(\mathbf{w}^\mathsf{T}\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^\mathsf{T}\mathbf{x})}$$

$$P(T = 0|\mathbf{x}, \mathbf{w}) = 1 - \sigma(\mathbf{w}^\mathsf{T}\mathbf{x}) = \frac{\exp(-\mathbf{w}^\mathsf{T}\mathbf{x})}{1 + \exp(-\mathbf{w}^\mathsf{T}\mathbf{x})}$$



Perceptron



Likelihood

$$
\begin{aligned}
p(\mathbf{t}|\mathbf{X}, \mathbf{w}) &= \prod_{n=1}^{N} p(t_n|\mathbf{x}_n, \mathbf{w}) \\
&= \prod_{t_n=1} p(t_n|\mathbf{x}_n, \mathbf{w}) \prod_{t_n=0} p(t_n|\mathbf{x}_n, \mathbf{w}) \\
&= \prod_{t_n=1} \sigma(\mathbf{w}^T\mathbf{x}_n) \prod_{t_n=0} (1 - \sigma(\mathbf{w}^T\mathbf{x}_n))
\end{aligned}
$$

Cross Entropy

$$\begin{aligned}\mathcal{L}(\mathbf{t}|\mathbf{X}, \mathbf{w}) &= -\sum_{t_n=1} \log \sigma(\mathbf{w}^\top \mathbf{x}_n) - \sum_{t_n=0} \log(1 - \sigma(\mathbf{w}^\top \mathbf{x}_n)) \\ &= -\sum_{n=1}^{N} t_n \log \sigma(\mathbf{w}^\top \mathbf{x}_n) + (1 - t_n) \log(1 - \sigma(\mathbf{w}^\top \mathbf{x}_n))\end{aligned}$$

Gradient of Cross Entropy

$$\frac{\partial \mathcal{L}}{\partial w_j} = -\sum_{n=1}^{N} [t_n - \sigma(\mathbf{w}^T \mathbf{x}_n)] \mathbf{x}_{n,j}$$

Gradient Descent:

$$w_j \leftarrow w_j - \eta \frac{\partial \mathcal{L}}{\partial w_j}$$

Multiclass Classification

Data in K classes

$$(\mathbf{x}_1, t_1), \cdots (\mathbf{x}_N, t_N),$$

where each $t_n \in \{1 \cdots K\}$

One hot representation

Each label $t_n \in \{1 \cdots K\}$ can be represented as a 0/1 K-vector, with

$$t_{n,k} = \begin{cases} 1, \text{if } t_n = k \\ 0, \text{otherwise} \end{cases}$$

Softmax Regression

$$P(T = k|\mathbf{x}, \mathbf{w}) = \frac{\exp(-\mathbf{w}^{k,T}\mathbf{x})}{\sum_{\ell=1}^{K} \exp(-\mathbf{w}^{\ell,T}\mathbf{x})}$$

That is, we have K parameter vectors $\mathbf{w}^1, \cdots, \mathbf{w}^K$ with $\mathbf{w}^k$ used to predict $t_{n,k}$

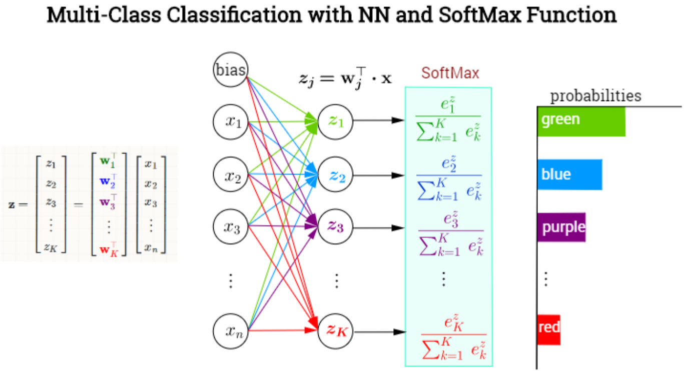
Cross Entropy: Multiple Classes

$$\mathcal{L} = \sum_{n=1}^{N} \sum_{k=1}^{K} t_{n,k} \log \frac{\exp(-\mathbf{w}^{k,T}\mathbf{x}_n)}{\sum_{\ell=1}^{K} \exp(-\mathbf{w}^{\ell,T}\mathbf{x}_n)}$$

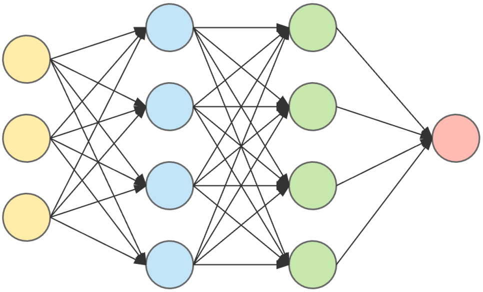
Gradient

$$\frac{\partial \mathcal{L}}{\partial w_j^k} = -\sum_{n=1}^{N} \left[t_{n,k} - \frac{\exp(-\mathbf{w}^{k,T}\mathbf{x}_n)}{\sum_{\ell=1}^{K} \exp(-\mathbf{w}^{\ell,T}\mathbf{x}_n)} \right] \mathbf{x}_{n,j}$$

Logistic to Softmax



Feed Forward Neural Networks

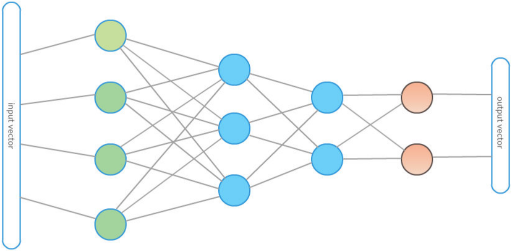


Each hidden layer node is computed from the previous layer:

$$\mathbf{h}_\ell = \sigma(\mathbf{W}_\ell \mathbf{h}_{\ell-1} + \mathbf{b})$$

The weights $\mathbf{W}_\ell$ are the parameters of the model.

BackPropagation



How to compute gradient of loss wrt parameters now??