Softmax

1. Relationship to logistic regression

Softmax is binary logistic regression's generalisation to multiple classes.

2. Formulation

Define mapping function:

$z = f(\mathbf{x}, \mathbf{w}) = \mathbf{w}_k^T \mathbf{x}$

The function $h_{\mathbf{w}}(\mathbf{x}) = p(y = C_k \mid \mathbf{x}; \mathbf{w}) = \frac{\exp(\mathbf{w}_k^T \mathbf{x})}{\sum_{k=1}^{K} \exp(\mathbf{w}_k^T \mathbf{x})} = \frac{\exp(z_j)}{\sum_{k=1}^{K} \exp(z_k)}$ is called softmax function: It takes a vector of an arbitrary real-valued scores (in z) and sqaushes it to a vector of values between zero and one.

From **Information theory** view:

The cross-entropy between a "true" distribution p and an estimated distribution q is defined as:

$H(p, q) = -\sum_x p(x) \log q(x)$

The softmax classifier is hence minimising the cross-entropy between the estimated class probabilities:

$q = \frac{\exp(z_j)}{\sum_{k=1}^{K} \exp(z_k)}$

and the "true" distribution, which is the distribution where all probability mass is on the correct class (i.e. p=[0,1,0,....])

From **probabilistic** interpretation:

The softmax function can be interpreted as the normalised probability assigned to the correct class C_k given the features $\mathbf{x}$ and parameterised by $\mathbf{w}$. Scores inside the output vector $\exp(z_j)$ as the unormalised log probabilities. Exponentiating these quantities give the probability and the division performs the normalisation so that the probabilities sum to one.

We are therefore minimising the negative log likelihood of the correct class, which can be interpreted as performing MLE. A nice feature of this view is that we can now also interpret the regularisation term $R(\mathbf{w})$ in the full loss functions as coming from a Gaussian prior over the weight matrix $\mathbf{w}$, where instead of MLE we are performing the **Maximum a posteriori (MAP)** estimation.

3. Gradient Descent

The formula of softmax loss:

$J(\mathbf{w}) = -[\sum_{n=1}^{N} \sum_{k=1}^{K} y_{nk} \log(h_{\mathbf{w}}(\mathbf{x}^n))]$

The gradient descent without regularisation term:

$w_{t+1} \leftarrow w_t - \eta(\frac{1}{N} \sum_{n=1}^{N} (h_w(x^n) - y^n)x^n)$

The gradient descent with regularisation term:

$w_{t+1} \leftarrow w_t - \eta(\frac{1}{N} \sum_{n=1}^{N} (h_w(x^n) - y^n)x^n + \lambda w_t)$

4. Practical issues: Numeric stability

The intermediate term $\exp(z_j)$ and $\sum_k \exp(z_k)$ may be very large due to the exponentials. Dividing large numbers can be numerically unstable.

Solution: normalisation trick!

$$\frac{\exp(z_j)}{\sum_k \exp(z_k)} = \frac{C \exp(z_j)}{C \sum_k \exp(z_k)} = \frac{\exp(z_j + \log C)}{\sum_k \exp(z_k + \log C)}$$

We are free to choose the value of C. This will not change any of the results. A common choice for C is to set $\log C = -\max_j z_j$