# Machine learning notes: week 5

Lars Yencken

## Notes: neural network training

### Cost function

- We have $L$ layers in the network, each with $s_l$ units
    - E.g. $s_L$ is the size of the output layer
- Our hypothesis: $h_\Theta(x) \in \mathcal{R}^K$
- Cost function $J(\Theta)$ is the sum of two components:
    - $\sum_k$ of the logistic regression cost over outputs $y_k$
    - Regularization term: $\frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_l+1} (\Theta_{j,i}^{(l)})^2$, ignoring the bias terms $\Theta_{i,0}^{(l)}$
- To minimise it, we need to calculate $\frac{\delta}{\delta \Theta_{i,j}^{(l)}} J(\Theta)$

### Backpropagation

- $\delta_j^{(l)}$ represents the error of node $j$ in layer $l$
    - $\delta_j^{(4)} = a_j^{(4)} - y_j = (h_\Theta(x))_j - y_j$
    - There is no $d^{(1)}$ term for the input layer
- $\frac{\delta}{\delta \Theta_{i,j}^{(l)}} J(\Theta) = a_j^{(l)} d_i^{(l+1)}$
- See notes for direct algorithm

### Implementation: unrolling

- `fminunc` and other optimising methods assume $\theta$ is a vector
    - This means we need to pack our matrices $\Theta^{(i)}$ into a vector
    - `thetaVec = [ Theta1(:); Theta2(:); Theta3(:)];`
    - Use `reshape()` to get the original structure back

## Gradient checking

- There's lots of ways to get the backpropagation algorithm wrong
- We can compare our gradient calculation with a numeric graident estimation
  - The estimation comes from $\frac{d}{d\Theta}J(\Theta) = \frac{J(\Theta+\epsilon)-J(\Theta-\epsilon)}{2\epsilon}$
- Do this on the unrolled version of $\Theta$ and compare to our more efficiently calculated gradient
  - The answers should be the same up to a few decimal places
  - Don't leave this running when training your network – it's too expensive!

## Initial values of $\Theta$

- We need to break symmetry, otherwise our network becomes degenerate
  - Think about it, the order of internal nodes is basically arbitrary
- Instead, pick a random number in $[-\epsilon, \epsilon]$ for each $\Theta_{i,j}^{(l)}$

## Big picture

- Picking an architecture (hidden layer? number of nodes?)
  - Input and output layer sizes are already fixed by your problem
  - More hidden units is better, but more expensive; up until $3-4\times$ your input feature space size is ok
- $J(\Theta)$ is no longer convex and can get stuck in local optima; not such a big problem in practice