6 Wrapping up Neural Networks

- Error function contains many local minima
- No guarantee of convergence
  - Not a "big" issue in practical deployments
- Improving backpropagation
  - Adding momentum
  - Using stochastic gradient descent
  - Train multiple times using different initializations

Adding momentum to the learning process refers to adding an "inertia" term which tries to keep the current value of a weight value similar to the one taken in the previous round.

7 Bias Variance Tradeoff

- Neural networks are *universal function approximators*
  - By making the model more complex (increasing number of hidden layers or m) one can lower the error
- Is the model with least training error the best model?
  - The simple answer is **no**!
  - Risk of overfitting (chasing the data)
  - Overfitting $\Leftarrow$ **High generalization error**

High Variance - Low Bias

- "Chases the data"
- Model parameters change significantly when the training data is changed, hence the term high variance.
- Very low training error

- Poor performance on unseen data

Low Variance - High Bias

- Less sensitive to training data
- Higher training error
- Better performance on unseen data

- General rule of thumb – If two models are giving similar training error, choose the **simpler** model
- What is simple for a neural network?
- Low weights in the weight matrices?
  - Why?
  - The simple answer to this is that if the weights in the weight vectors at each node are high, the resulting discriminating surface learnt by the neural network will be highly non-linear. If the weights are smaller, the surface will be smoother (and hence simpler).
- Penalize solutions in which the weights are high
- Can be done by introducing a penalty term in the objective function
  - **Regularization**

Regularization for Backpropagation

$$\widetilde{J} = J + \frac{\lambda}{2n}\left(\sum_{j=1}^{m}\sum_{i=1}^{d+1}(w_{ji}^{(1)})^2 + \sum_{l=1}^{k}\sum_{j=1}^{m+1}(w_{lj}^{(2)})^2\right)$$