Learning from Data: Layered Neural Networks 3

Amos Storkey, School of Informatics

November 9, 2005

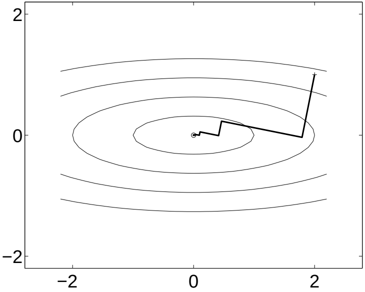
http://www.anc.ed.ac.uk/∼amos/lfd/

More on Optimisation

- Line Search
- Problems with gradient descent
- Second order methods
- Conjugate gradient
- Demos

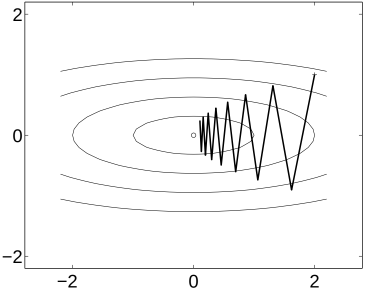
Line search

- Choose a search direction $\mathbf{v}$ starting from the current point $\boldsymbol{\theta}$
- Minimise $E(\boldsymbol{\theta} + \lambda \mathbf{v})$ with respect to λ (a one-dimensional minimisation)



$E = w_1^2 + 10 w_2^2$, using gradient as search direction

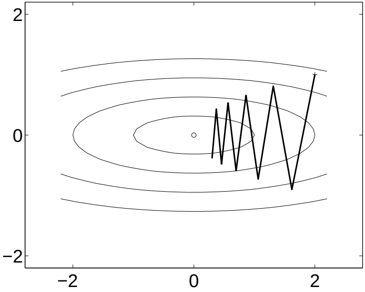
Problems with gradient descent



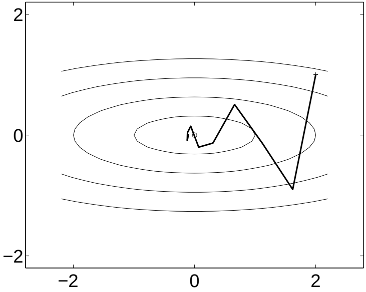
- Problem is *zig-zagging* in ravines
- Momentum (Mitchell §4.5.2.1)

$$\Delta\boldsymbol{\theta}(t+1) = -\eta\nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta}) + \alpha\Delta\boldsymbol{\theta}(t)$$

But We Need to Set “Fiddle Factors”



$\eta = 0.095 \; \alpha = 0.0$



$\eta = 0.095 \; \alpha = 0.05,$

Second Order Information

- Taylor expansion

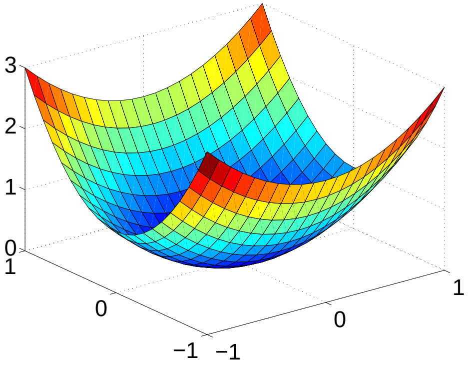
$$E(\boldsymbol{\theta}+\boldsymbol{\delta}) \simeq E(\boldsymbol{\theta}) + \boldsymbol{\delta}^T \nabla_{\boldsymbol{\theta}} E + \frac{1}{2}\boldsymbol{\delta}^T H \boldsymbol{\delta}$$

  where

$$H_{ij} = \frac{\partial^2 E}{\partial \theta_i \partial \theta_j}$$

- H is called the Hessian.
- If H is positive definite, this models the error surface as a quadratic bowl.

Quadratic Bowl



Direct Minimisation

- A quadratic function can be minimised directly using

$$\delta = -H^{-1}\nabla_{\theta}E$$

but this requires
  - Knowing/computing H, which has size $O(W^2)$
  - Inverting H, $O(W^3)$

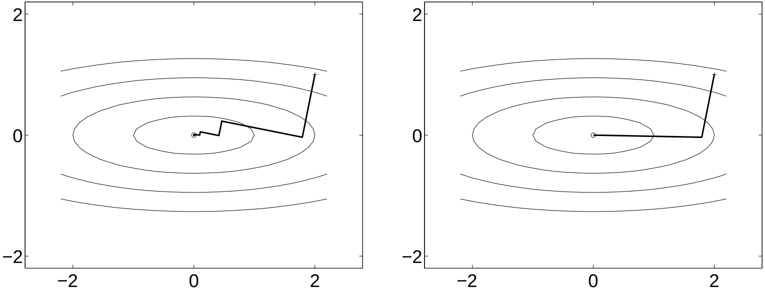
Conjugate Gradients

- The conjugate gradients algorithm minimises a quadratic form in D variables in D steps, without ever computing H or H^{-1} explicitly. This is very useful!
- It uses line search, but the directions chosen to go in are not usually the gradient:
- Conjugate directions

$$\mathbf{v}_i H \mathbf{v}_j = 0 \qquad i \neq j$$

Comparison

Gradient directions



Gradient Directions

Conjugate Directions

Batch vs online

- **Batch** learning: use all patterns in training set, and update weights after calculating

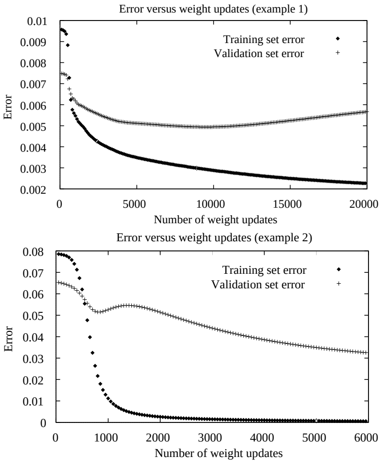
$$\frac{\partial E}{\partial \boldsymbol{\theta}} = \sum_{\mu} \frac{\partial E^{\mu}}{\partial \boldsymbol{\theta}}$$

- **On-line** learning: adapt weights after each pattern presentation, using $\frac{\partial E^{\mu}}{\partial \boldsymbol{\theta}}$
- **Batch** more powerful optimization methods
- **Batch** easier to analyze
- **On-line** more feasible for huge or continually growing datasets
- **On-line** may have ability to jump over local optima

Convergence of Backpropagation

- Dealing with local minima. Train multiple nets from different starting places, and then choose best (or combine in some way)
- Nature of Convergence
- Initialize weights near zero
- Therefore, initial networks are near-linear
- Increasingly non-linear functions possible as training progresses
- *Early stopping*: a heuristic regularisation technique.

Overfitting in Neural Networks



Optimization: Summary

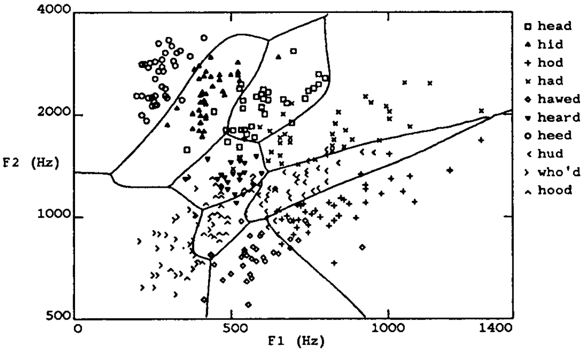
- Optimize over vector of all weights/biases in a network
- All methods considered find *local* optima
- Gradient descent is simple but slow
- In practice, second-order methods (*conjugate gradients*) are used for batch learning
- Overfitting can be a problem

Pause

Representation Power of ANNs

- Boolean functions:
  - Every boolean function can be represented by network with single hidden layer
  - but might require exponential (in number of inputs) hidden units
- Continuous functions:
  - Every bounded continuous function can be approximated with arbitrarily small error, by network with one hidden layer [Cybenko 1989; Hornik et al. 1989]
  - Any function can be approximated to arbitrary accuracy by a network with two hidden layers [Cybenko 1988].
  - Neural Networks are *universal approximators*.

Functional approximation



But, ...

- The fact that a function is representable does not tell us how many hidden units would be required for its approximation
- Nor does it tell us if it is *learnable* (a search problem)
- Nor does it say anything about how much training data would be needed to learn the function
- In fact universal approximation has only a limited benefit: need bias.

Hypothesis space and Inductive Bias for ANNs

- **Hypothesis space**: if there are $|\boldsymbol{\theta}|$ weights and biases

$$H = \left\{\boldsymbol{\theta} | \boldsymbol{\theta} \in \mathbb{R}^{|\boldsymbol{\theta}|}\right\}$$

- **Inductive Bias**: hard to characterize, depends on search procedure, regularisation and how weight space spans the space of representable functions
- Approximate statement: smooth interpolation between data points

Learning Hidden Layer Representations

- Backprop can develop intermediate representations of its inputs in the hidden layers
- These new features will capture properties of the input instances that are most relevant to learning the target function
- This ability to automatically discover useful hidden-layer representations is a key feature of ANN learning

Summary

- Neural networks are a powerful nonlinear modelling tool for classification and for regression.
- Rely on optimisation methods to find good models.
- Somewhat opaque assumptions.
- Local minimum problems.