Optimization

Some slides courtesy of: M. Davies, A. Storkey, S. Boyd, P. Patrinos

Optimization

- Why do we need it?
  - Demo: http://www.benfrederickson.com/numerical-optimization/
- How common it is?
  - http://online.stanford.edu/course/convex-optimization-winter-2014
  - MSc in optimization (Univ. of Edinburgh)

Optimization in Learning

Most learning algorithms involve optimization, e.g.

- Minimizing an error function (Multi-Layer Perceptrons)
- Maximizing a Likelihood (Nonlinear regression)
- Minimizing an expected loss function (Bayesian Decision Th'y)

Can I use direct optimization? ➔ Only for analytically solvable forms (e.g. quadratic)

In general analytical solutions **unavailable**.

Hence we need to use iterative schemes for optimization. e.g.

- Back-propagation (i.e. gradient descent)

Iterative optimization

We will now look into the field of numerical optimization in more detail:

*Problem: minimize the function **E(w)** with respect to **w***

The basis blocks of an iterative process in optimization:

- Currently at some position w_t.
- Choose a new position to go to w_{t+1}.
- Think of as choosing a direction to go in, and then a distance to go in that direction. Want to get as close to optimum.
- If we know the direction, but not sure how far, we can use line search.

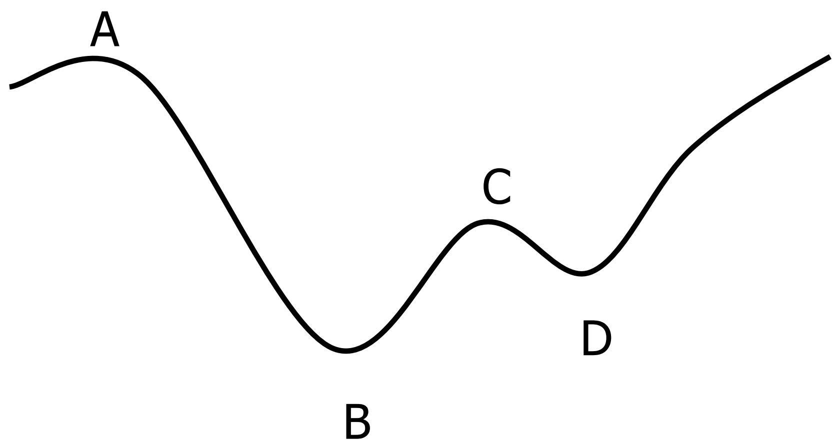
Maxima and minima

$A - \mathrm{D}$: stationary points of

$$E(w) \quad (i.e. \ \frac{\partial E}{\partial w} = 0)$$

D,B : minima

B : global minimum



A cost function may have multiple minima and other stationary points (including saddles). Recall minimum implies

1. gradient $\nabla E = 0$

2. curvature $\nabla^2 E$ is positive $(v^T \nabla^2 E v > 0$, for any $v)$.

In practice we generally have to make do with finding a <u>local</u> minimum.

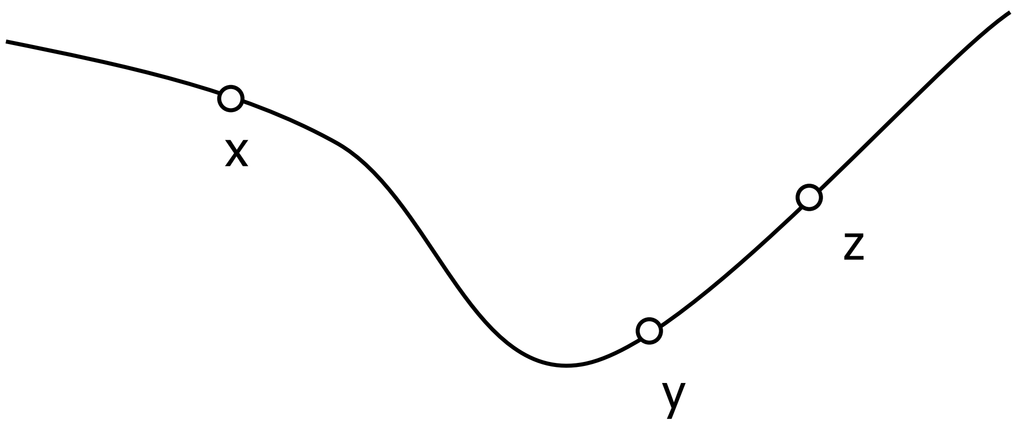
1-D Optimization

1-dimensional versus multi-dimensional

1-D and multi-D optimization are fundamentally different.

Let $x < y < z$ be three points on a 1-D cost function such that $E(y) <$ E(x)& E(z).

It follows that there must exist a minimum within the bracket (x,z).



No equivalent point-wise property exists in dimensions greater than one!

Bracket methods

Once we have a bracketed minimum we can proceed to find a new smaller bracket. There are various methods, e.g.

- Golden section search:

Start with the triple (a,b,c) then suppose we choose x $\in (b,c)$ and evaluate $E(x)$

- if $f(b) < f(x) \rightarrow$ new triple (a,b,x)
- otherwise new triple (b,x,c).

How to choose x (and from which interval)? New bracket length will either be:

$$c - b \text{ or } x - a$$

Hence choose x in largest interval.

either $x = a + \dfrac{3-\sqrt{5}}{2}(b-a)$ or $x = c + \dfrac{3-\sqrt{5}}{2}(b-c)$ (Golden mean)

convergence is linear (bracket guaranteed to be $<$ 0.618 smaller)

Multi-dimensional methods:
Local approximations

We can characterise a minimum locally by a quadratic approximation:

$$\boxed{E(w_0+\Delta w)\approx E(w_0)+\nabla E(w_0)^T\Delta w+\frac{1}{2}\Delta w^T\nabla^2 E(w_0)\Delta w+\ldots}$$

The gradient for E(w) can similarly be expressed as:

$$\nabla E(w_0+\Delta w)\approx \nabla E(w_0)+\nabla^2 E(w_0)\Delta w+\ldots$$

Suppose that w_0 is a minimum, by expanding around it:

$$E(w_0+\Delta w)\approx E(w_0)+0+\frac{1}{2}\Delta w^T\nabla^2 E(w_0)\Delta w+\ldots$$

no gradient

Hence at a minimum $\nabla^2 E(w_0)$ is positive definite $(v^T\nabla^2 E(w_0)v>0, \forall v)$.

$\rightarrow$ any perturbation *must* increase $E(w)$.

Multi-dimensional methods:
complexity and function evaluation

We will be looking at a number of different optimization techniques. We note the following costs:

$$\text{Evaluation of } E(w) \quad \rightarrow O(NW)$$

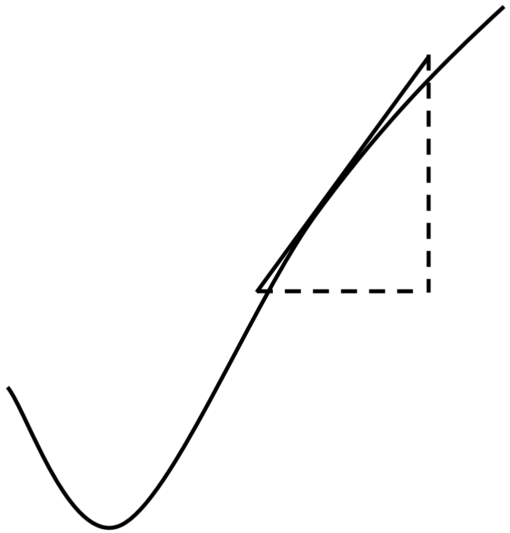
$$\text{Evaluation of } \nabla E(w) \quad \rightarrow O(NW)$$

$$\text{Evaluation of } \nabla^2 E(w) \quad \rightarrow O(NW^2)$$

$$\text{Inversion of } \nabla^2 E(w) \quad \rightarrow O(W^3)$$

Where *N* is the number of data observations and *W* is the number of weights.

Multi-dimensional methods:
Gradient descent with fixed step size



Recall gradient descent only uses linear approximation:

$$E(w_0 + \Delta w) \approx E(w_0) + \nabla E(w_0)^T \Delta w$$

For a fixed $|\Delta w|$ best approximate reduction in E(w) is:

$$w^{(k)} = w^{(k-1)} - \eta \nabla E(w^{(k-1)})$$

where η is a (small) step size.

Problem: no concept of a minimum.

Is there necessarily a good single value for η? **Answer:** NO!

Multi-dimensional methods:

Convergence in gradient descent

We first introduce a change in coordinates, using an eigenvalue decomposition

$$\nabla^2 E_0 = U \Lambda U^T, \Lambda - \text{diagonal}, \ U - \ \text{rotation}$$

So that $v = U^T w$ and:

$$E \approx E_0 + \left[\nabla E_0{}^T U^T \right] \Delta v + \frac{1}{2} \Delta v^T \left[U^T \nabla^2 E_0 U \right] \Delta v + \ldots$$

Now the contribution of each component of Δv is independent:

$$E = E_0 + \sum_k (b_k \Delta v_k + \frac{1}{2} \Lambda_{k,k} \Delta v_k^2)$$

(where $b = \nabla E_0{}^T U^T = (U \nabla E_0)^T$).

For each component the optimal $\eta = 1/\Lambda_{k,k}$, converging in a single step.

However gradient descent only allows us to set a single η.

Multi-dimensional methods:

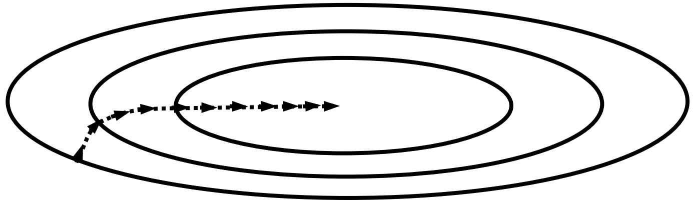
Effect of step-size in gradient descent

- If we choose it too small convergence is *very* slow.
- If we choose it too large the algorithm may become unstable (oscillate around) and never converge.

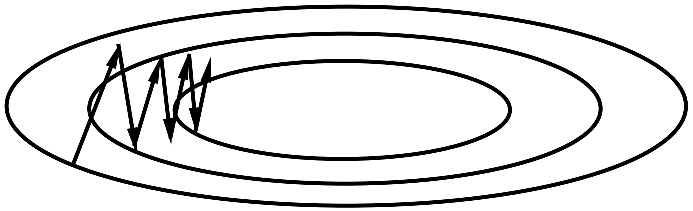
If all the eigenvalues, $\Lambda_{k,k}$ are similar then performance of gradient descent is okay.

If $\Lambda_{1,1} / \Lambda_{W,W} >> 1$, convergence will be bad!

η too small



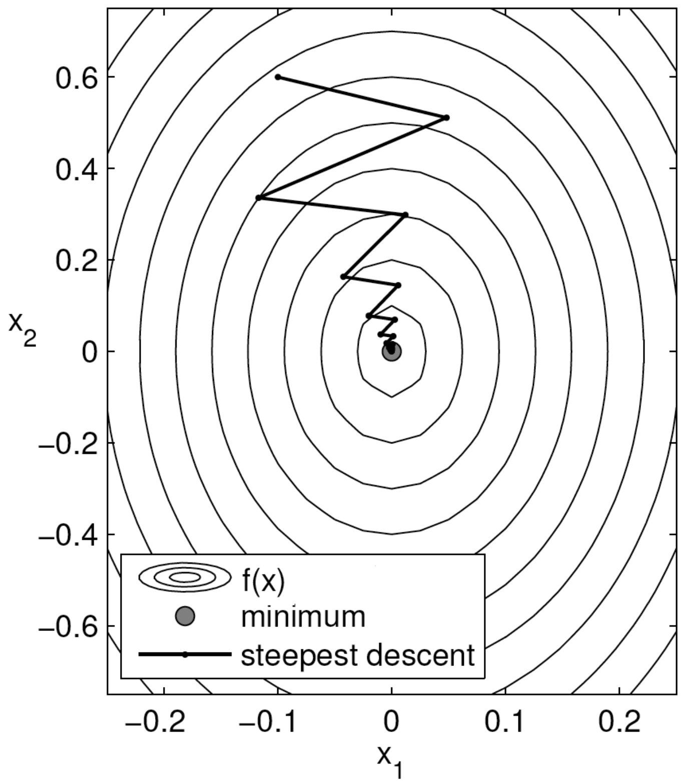
η too big



Multi-dimensional methods:

An example of gradient descent

- We start at $[-0.1\ 0.6]^T$



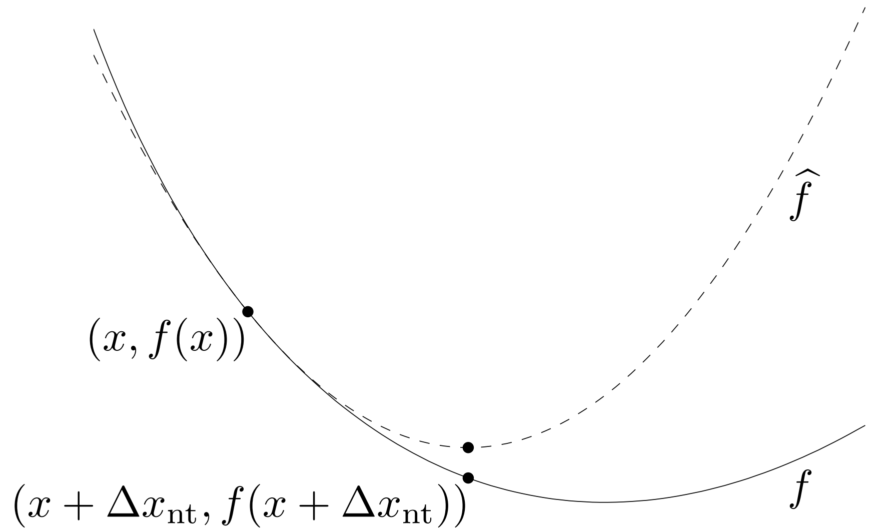
$$f(x_1, x_2) = 1 - e^{-(10x_1^2 + x_2^2)}.$$

Multi-dimensional methods:
Newton's method

Alternatively we can find the minimum of the quadratic approximation explicitly:

$$\Delta w = w^{(k+1)} - w^{(k)} = -\left[\nabla^2 E(w^{(k)})\right]^{-1} \nabla E(w^{(k)}) \Longrightarrow w^{(k+1)} = w^{(k)} - \left[\nabla^2 E(w^{(k)})\right]^{-1} \nabla E(w^{(k)})$$

This is Newton's method (will need to iterate since quadratic is an approximation)



To relate to variables in the figure:

$f = E$

$x = w$

$f^\wedge =$ approximation of f
(E in our case) by a quadratic

$\Delta x_{nt} = \Delta w$

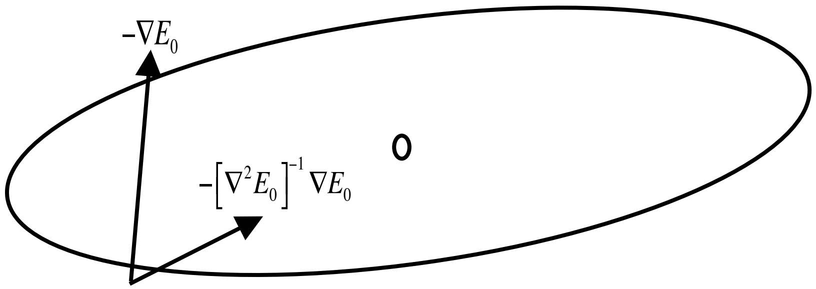
Figure 9.16 The function f (shown solid) and its second-order approximation $\widehat{f}$ at x (dashed). The Newton step Δx_{nt} is what must be added to x to give the minimizer of $\widehat{f}$.

Multi-dimensional methods:
Newton's method

Alternatively we can find the minimum of the quadratic approximation explicitly:

$$\Delta w = w^{(k+1)} - w^{(k)} = -\left[\nabla^2 E\left(w^{(k)}\right)\right]^{-1} \nabla E\left(w^{(k)}\right) ==> w^{(k+1)} = w^{(k)} - \left[\nabla^2 E\left(w^{(k)}\right)\right]^{-1} \nabla E\left(w^{(k)}\right)$$

This is Newton's method (will need to iterate since quadratic is an approximation)



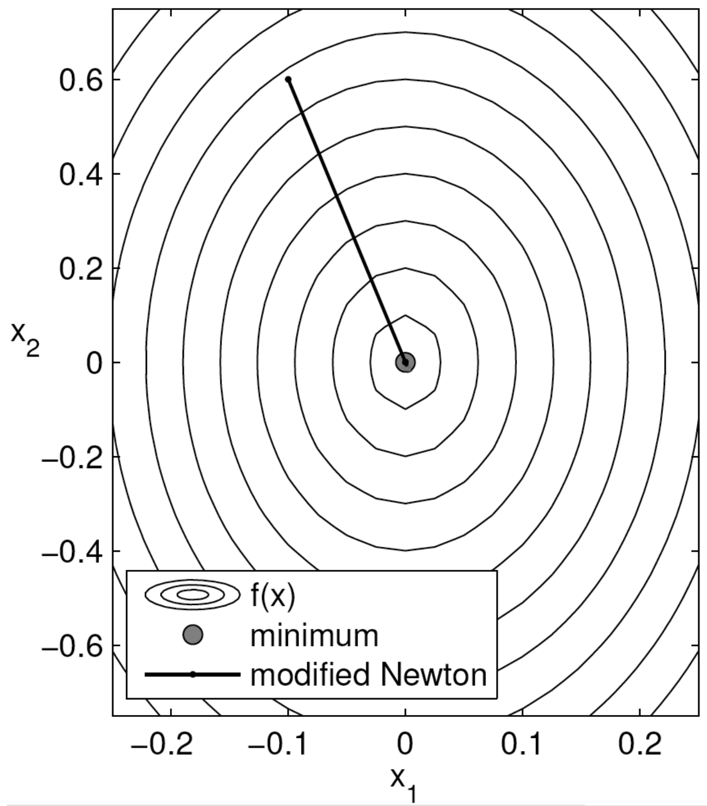
Gradient descent does not point directly towards minimum (in reality a stationary point), whereas the Newton direction does.

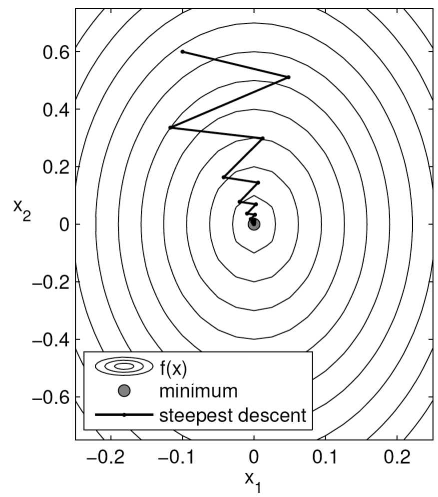
Both methods belong to a general iterative family with $\Delta w = -M^{-1} \nabla E_0$.

For gradient descent, $M = \eta^{-1} I$, for Newton $M = \nabla^2 E_0$

$$w^{(k)} = w^{(k-1)} - \eta \nabla E\left(w^{(k-1)}\right)$$

Multi-dimensional methods:
Newton's method





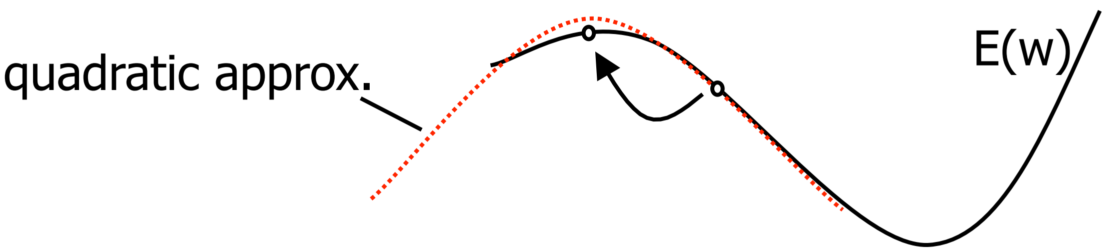
Compare this with the gradient decent result (few slides back is shown larger). The gradient points on a point of high gradient but not necessarily to a local stationary point. Thus, Newton methods have faster convergence but…they have problems. See next slide.

Multi-dimensional methods:
Problems with Newton Algorithm

There are a number of problems associated with a direct application of Newton's method in nonlinear networks

1. Evaluating and inverting Hessian is **expensive** (*Order*(NW^2) and *Order*(W^3))
2. Convergence issues:

   a) Newton is a zero-finding algorithm. May find a **maximum**.



   b) May go **unstable** - step size may take the *w* outside the range where the quadratic approximation is reasonable

Multi-dimensional methods:
regularizing the Newton method

To stabilize the Newton step we can restrict the region of search (and get good of both worlds [Newton and Gradient Descent]).

If M is symmetric +ve definite then $\Delta w = -M^{-1}\nabla E_0$ will point downhill.

Consider: $$\Delta w^{(k+1)} = -\left[\nabla^2 E(w^{(k)}) + \gamma I\right]^{-1} \nabla E(w^{(k)})$$

If $\gamma \to 0$ then $\mathrm{M} \to \nabla^2 E_0$ (Newton step)

If $\gamma \to \infty$ then $\mathrm{M} \to (1/\gamma)I$ (gradient descent, $\gamma = 1/\gamma$)

γ controls the size of the search region. We can choose γ adaptively.

e.g.

If $E(w^{(k)}) < E(w^{(k-1)})$ then

$\to \gamma = \gamma \div 10$

else

$\to w^{(k)} = w^{(k-1)}$ and $\gamma = \gamma \times 10$

Multi-dimensional methods:
Levenberg Marquardt method

We now specifically consider the sum-of-squared errors cost function.

$$E(w) = \frac{1}{2}\sum_n \left(f(x_n, w) - y_n \right)^2 = \frac{1}{2}\sum_n e_n^2$$

This term might make Hessian not +ve definite so we ignore it

With derivatives:

$$\nabla E(w) = \frac{1}{2}\sum_n \frac{\partial e_n^2}{\partial w} = \sum_n \frac{\partial f(x_n, w)}{\partial w} e_n$$

and

$$\nabla^2 E(w) = \frac{\partial}{\partial w}\left(\sum_n \frac{\partial f(x_n, w)}{\partial w} e_n \right) = \sum_n \left(\frac{\partial f(x_n, w)}{\partial w} \frac{\partial f(x_n, w)}{\partial w}^T + e_n \frac{\partial^2 f(x_n, w)}{\partial w^2} \right)$$

Ignoring second term Gives:

$$w^{(k+1)} = w^{(k)} - \left[\sum_n \frac{\partial f(x_n, w)}{\partial w} \frac{\partial f(x_n, w)}{\partial w}^T + \gamma I \right]^{-1} \sum_n \frac{\partial f(x_n, w)}{\partial w} e_n$$

Note $\frac{\partial f(x_n, w)}{\partial w}$ is just gradient (e.g. calculated using *backpropagation*).

Multiple dimensions and line searches

In gradient descent we progressed a small way down in one direction and then selected a new direction, but can we **instead also** decide how much to move down?

Alternative approach - continue along direction until a *line minimum* is found. We already know how to do this using bracketing methods!

Line searching:

1. Choose a direction d_1
2. Perform line minimisation on $E(w_1+\lambda d_1)$ *e.g.* using a bracketing method
3. Select new search direction and repeat.

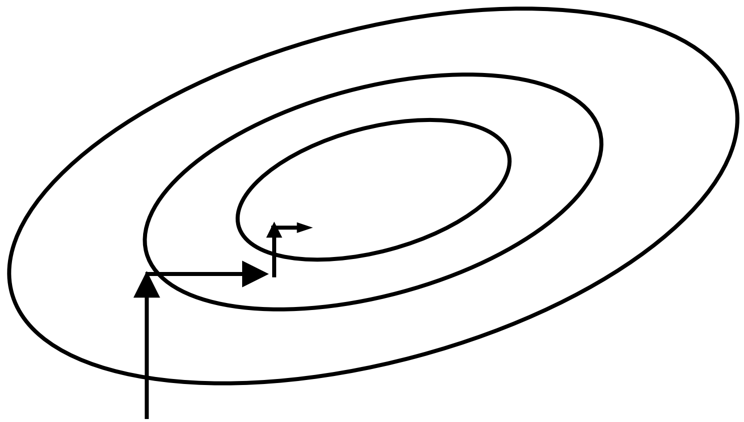
How do we select the new direction? *Gradient Descent?*

Line searches and gradient descent

A naive approach to line searching is to search along the steepest direction:

However each new search MUST be orthogonal to the last.

Why orthogonal?



line search minimum

$$\frac{\partial}{\partial \eta_k} E(w_{k+1}) = \nabla E(w_{k+1})^T \cdot \frac{\partial w_{k+1}}{\partial \eta_k} = \nabla E(w_{k+1})^T d_k$$

$$\rightarrow \quad \nabla E(w_{k+1})^T d_k = 0$$

new search direction, $d_{k+1} = -\nabla E(w_{k+1})^T$.

Hence d_k, d_{k+1} orthogonal

We are re-searching directions previously minimised (zigzag downhill)!

Conjugate directions I

Can we construct directions that preserve the previous minimization work?

Amazingly the answer is yes….

The idea of conjugate directions is to choose a direction $w = w_k + \lambda d_k$ such that:

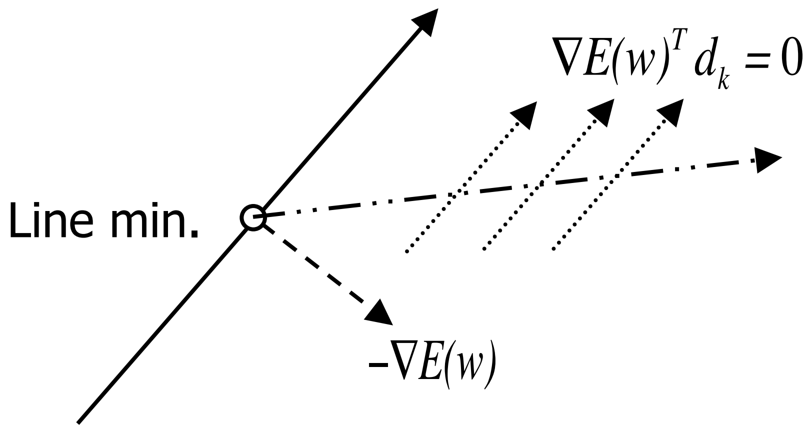
$$\nabla E(w_k + \lambda d_k)^T d_j = 0, \forall j < k$$

Which implies (with quadratic approx.)

$$(\nabla E(w_k) + \nabla^2 E(w_k)\lambda d_k)^T d_j = 0$$

And hence

$$d_k^T \nabla^2 E(w_k) d_j = 0$$



Conjugate gradient algorithm I

Suppose we have a quadratic function:

$$E(w) = E_0 + b^T w + \frac{1}{2} w^T H w$$

Starting at w_i and searching in direction d_i the line minimum is:

$$w_{i+1} = w_i + \alpha_i d_i$$

denoting the gradient at w_i as $g_i = \nabla E(w_i) = b + H w_i$ we can solve for α_i

$$g_{i+1}^T d_i = (b + H(w_i + \alpha_i d_i))^T d_i = g_i^T d_i + \alpha_i d_i^T H d_i = 0$$

Hence:

$$\alpha_i = -\frac{g_i^T d_i}{d_i^T H d_i}$$

Conjugate gradient algorithm II

We now choose a d_{i+1} that is conjugate to d_i we will try a modified gradient:

$$d_{i+1} = -g_{i+1} + \beta_i d_i$$

for some β_i. Solving for conjugacy gives:

$$\left(-g_{i+1} + \beta_i d_i\right)^T H d_i = 0$$

Hence:

$$\beta_i = \frac{g_{i+1}^T H d_i}{d_i^T H d_i}$$

In fact this choice of direction is conjugate with all d_j, $j < i$.

Finally we can write:

$$\beta_i = \frac{g_{i+1}^T \left(\alpha_i H d_i\right)}{d_i^T \left(\alpha_i H d_i\right)} = \frac{g_{i+1}^T \left(g_{i+1} - g_i\right)}{d_i^T \left(g_{i+1} - g_i\right)}$$

since $\quad g_{j+1} - g_j = H(w_{j+1} - w_j) = \alpha_j H d_j \quad$ (no need to use H)

Conjugate gradient algorithm summary

0. Choose initial weight w_1 and search direction $d_1 = -\nabla E(w_1)$

... at step j

1. Find line minimum for $E(w_j + \alpha_j d_j)$, setting $w_{j+1} = w_j + \alpha_j d_j$

$($if not at minimum$)$

2. Evaluate new gradient, $g_{j+1} = \nabla E(w_{j+1})$

3. Calculate new search direction, $d_{j+1} = -g_{j+1} + \beta_j d_j$,

using the formula: $\beta_j = \dfrac{g_{i+1}^T \left(g_{i+1} - g_i\right)}{d_i^T \left(g_{i+1} - g_i\right)}$

4. repeat from step 3 $($or after W steps begin again with step 2$)$

Note: the algorithm may also need to be iterated many more times $(c.f.$ Newton method$)$.
The search directions may deteriorate, therefore it is sensible to re-start the algorithm every W steps $($other strategies also exist$)$.