Introduction to advanced parameter optimization

So far:

- What is a neural network?
- Basic training algorithm:
  - Gradient descent
  - Backpropagation

Next: advanced training algorithms

Gradient descent algorithm

1. Choose an initial weight vector $\mathbf{w}_1$ and let $\mathbf{d}_1 = -\mathbf{g}_1$.
2. Let $\mathbf{w}_{j+1} = \mathbf{w}_j + \eta\mathbf{d}_j$.
3. Evaluate $\mathbf{g}_{j+1}$.
4. Let $\mathbf{d}_{j+1} = -\mathbf{g}_{j+1}$.
5. Let $j = j + 1$ and go to step 2.

Gradient descent review

Gradient descent:

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \Delta\mathbf{w}(t)$$

where,

$$\Delta\mathbf{w}(t) = -\eta\nabla E[\mathbf{w}(t)]$$

Two main problems:

- Slow convergence
- Trial-and-error selection of η

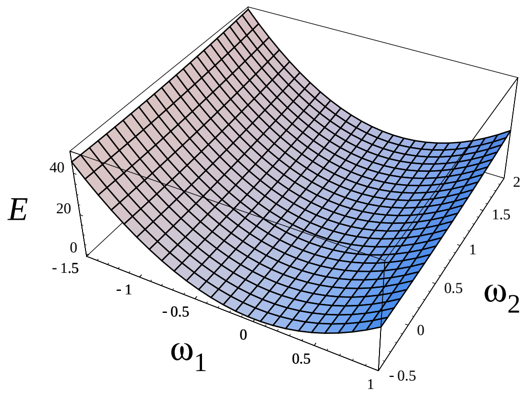
Goal: **cut number of epochs (training cycles) by orders of magnitude ... how?**

How to improve over gradient descent?

- Must understand convergence properties
- Use second-order information...

First case study

$E = 20\omega_1^2 + \omega_2^2$ (same as last time)



Will also look at simple nonquadratic error surfaces...

Why quadratic error surface?

Disadvantages:

- Too simple/too few parameters
- NN error surface not *globally* quadratic

Advantage:

- Easy to visualize
- NN error surfaces will be *locally* quadratic near a local minimum.

Taylor series expansion

Single dimension (from calculus):

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0) + f''(x_0)(x - x_0)^2$$

Multi-dimensional error surface:

$$E(\mathbf{w}) \approx E(\mathbf{w}_0) + (\mathbf{w} - \mathbf{w}_0)^T\mathbf{b} + \frac{1}{2}(\mathbf{w} - \mathbf{w}_0)^T H(\mathbf{w} - \mathbf{w}_0)$$

about some vector $\mathbf{w}_0$, where,

$\mathbf{b} = \nabla E(\mathbf{w}_0)$

$H = \nabla[\nabla E(\mathbf{w}_0)]$ **(*Hessian:* not just a German mercenary)**

Hessian matrix

Definition: The $W \times W$ *Hessian* matrix H of a W-dimensional function $E(\mathbf{w})$ is defined as,

$$H = \nabla[\nabla E(\mathbf{w})]$$

where,

$$\mathbf{w} = [\omega_1\ \omega_2\ \dots\ \omega_W]^T$$

Alternatively:

$$H_{(i,j)} = \frac{\partial^2 E}{\partial\omega_i \partial\omega_j}$$

Some linear algebra

Definition**: For a $W \times W$ square matrix H, the *eigenvalues* λ are the solution of,**

$$|\lambda I_W - H| = 0$$

Definition**: A square matrix H is *positive-definite*, if and only if all its eigenvalues λ_i are greater than zero. If a matrix is positive-definite, then,**

$$\mathbf{v}^T H \mathbf{v} > 0, \ \forall \mathbf{v} \neq 0 .$$

- Quadratic error surface: $H > 0$
- Arbitrary error surface: $H > 0$ near local minimum.

Gradient descent convergence rate

Near local minimum:

$$\lambda_{min} > 0 \ \text{(why?)}$$

Convergence governed by:

$$\left(\frac{\lambda_{min}}{\lambda_{max}}\right)$$

Learning rate bound:

$$0 < \eta < \frac{2}{\lambda_{max}}$$

Simple Hessian example

$$E = 20\omega_1^2 + \omega_2^2$$

First partial derivatives:

$$\frac{\partial E}{\partial \omega_1} = 40\omega_1 \qquad \frac{\partial E}{\partial \omega_2} = 2\omega_2$$

Second partial derivatives:

$$\frac{\partial^2 E}{\partial \omega_1^2} = 40 \qquad \frac{\partial^2 E}{\partial \omega_2^2} = 2 \qquad \frac{\partial^2 E}{\partial \omega_1 \partial \omega_2} = \frac{\partial^2 E}{\partial \omega_2 \partial \omega_1} = 0$$

Simple Hessian example (continued)

$$E = 20\omega_1^2 + \omega_2^2$$

Second partial derivatives:

$$\frac{\partial^2 E}{\partial \omega_1^2} = 40 \qquad \frac{\partial^2 E}{\partial \omega_2^2} = 2 \qquad \frac{\partial^2 E}{\partial \omega_1 \partial \omega_2} = \frac{\partial^2 E}{\partial \omega_2 \partial \omega_1} = 0$$

Hessian:

$$H = \begin{bmatrix} 40 & 0 \\ 0 & 2 \end{bmatrix}$$

Simple Hessian example (continued)

$$H = \begin{bmatrix} 40 & 0 \\ 0 & 2 \end{bmatrix}$$

What are the eigenvalues?

Computation of eigenvalues

$$|\lambda I_2 - H| = 0 \qquad \left| \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 40 & 0 \\ 0 & 2 \end{bmatrix} \right| = 0$$

$$\begin{vmatrix} \lambda - 40 & 0 \\ 0 & \lambda - 2 \end{vmatrix} = 0$$

$$(\lambda - 40)(\lambda - 2) = 0$$

$$\lambda_{min} = 2$$

$$\lambda_{max} = 40$$

Learning rate bounds

$$\lambda_{min} = 2$$

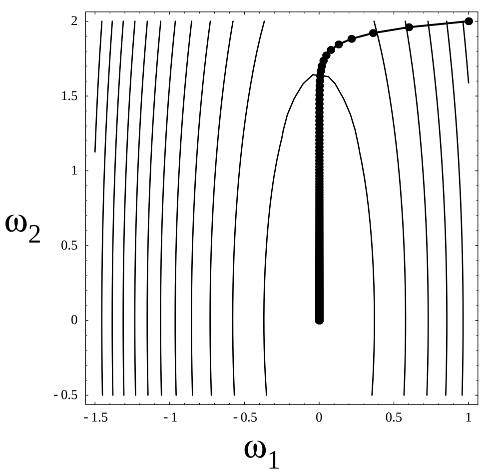
$$\lambda_{max} = 40$$

$$0 < \eta < \frac{2}{\lambda_{max}}$$

$$0 < \eta < \frac{2}{40} = 0.05 \text{ (same as fixed-point derivation)}$$

Convergence examples

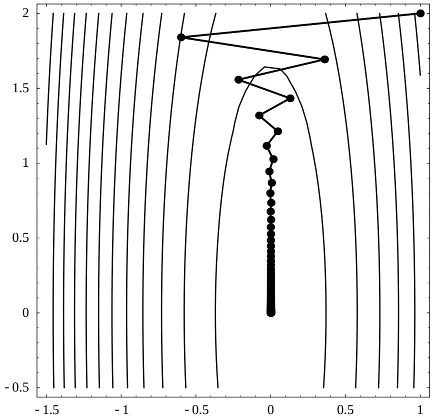
$$\eta = 0.01$$



719 steps

Convergence examples

$\eta = 0.04$



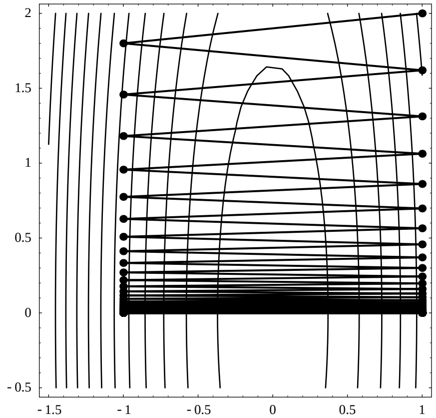
ω_2

ω_1

175 steps

Convergence examples

$\eta = 0.05$



ω_1

no convergence

Basic problem: "long valley with steep sides"

What characterizes a " long valley with steep sides?"

Length of contour lines proportional to:

$$\frac{1}{\sqrt{\lambda_1}} \text{ and } \frac{1}{\sqrt{\lambda_2}}$$

Small ratios:

$$\left(\frac{\lambda_{min}}{\lambda_{max}}\right)$$

So what can we do about this?

Solution

- Fixed learning rate is the problem
- Answer: different learning rates for each weight.

Key question: how to achieve automatically?

Heuristic extension: momentum μ

Gradient descent with momentum:

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \Delta\mathbf{w}(t)$$

$$\Delta\mathbf{w}(0) = -\eta\nabla E[\mathbf{w}(0)]$$

$$\Delta\mathbf{w}(t) = -\eta\nabla E[\mathbf{w}(t)] + \mu\Delta\mathbf{w}(t-1),\ t > 0,\ 0 \le \mu < 1$$

Notes:

- $\Delta\mathbf{w}(t)$ dependent on $\mathbf{w}(t)$ and $\mathbf{w}(t-1)$
- Ideally, high *effective* learning rate in shallow dimensions
- Little effect along steep dimensions

Gradient descent algorithm

1. Choose an initial weight vector $\mathbf{w}_1$ and let $\mathbf{d}_1 = -\mathbf{g}_1$.
2. Let $\mathbf{w}_{j+1} = \mathbf{w}_j + \eta\mathbf{d}_j$.
3. Evaluate $\mathbf{g}_{j+1}$.
4. Let $\mathbf{d}_{j+1} = -\mathbf{g}_{j+1}$.
5. Let $j = j+1$ and go to step 2.

Analyzing momentum term

Shallow regions: assume,

$$\nabla E(\mathbf{w}_t) \approx \nabla E(\mathbf{w}_0) = \mathbf{g},\ t \in \{1, 2, \ldots\}$$

Then:

$$\Delta\mathbf{w}(0) = -\eta\mathbf{g}$$

$$\Delta\mathbf{w}(1) \approx -\eta\mathbf{g} + \mu\Delta\mathbf{w}(0) = -\eta\mathbf{g}(1+\mu)$$

$$\Delta\mathbf{w}(2) \approx -\eta\mathbf{g} + \mu\Delta\mathbf{w}(1) = -\eta\mathbf{g} + \mu[-\eta\mathbf{g}(1+\mu)]$$

$$\Delta\mathbf{w}(2) \approx -\eta\mathbf{g}(1+\mu+\mu^2)$$

Analyzing momentum term

Assumption (shallow region):

$$\nabla E(\mathbf{w}_t) \approx \nabla E(\mathbf{w}_0) = \mathbf{g},\ t \in \{1, 2, \ldots\}$$

In general,

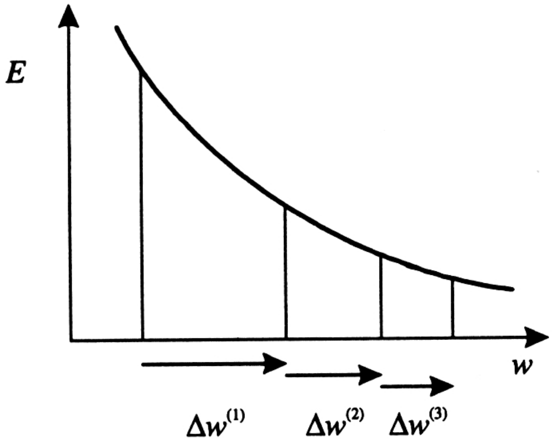
$$\Delta\mathbf{w}(t) \approx -\eta\mathbf{g}\left(\sum_{s=0}^{t}\mu^s\right) = -\eta\left(\frac{1-\mu^{t+1}}{1-\mu}\right)\mathbf{g}$$

In the limit:

$$\lim_{t\to\infty}\Delta\mathbf{w}(t) \approx \frac{-\eta}{(1-\mu)}\mathbf{g}$$

Analyzing momentum term

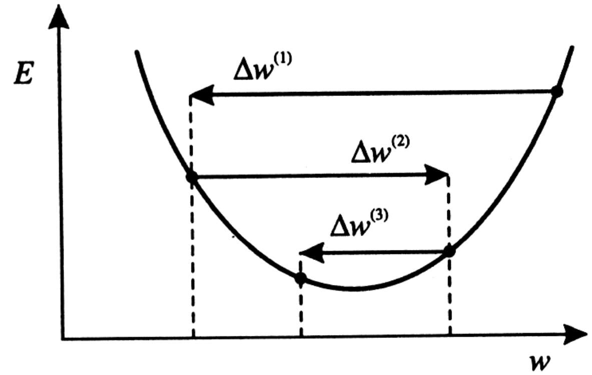
Effective learning rate (shallow regions): $\eta / (1 - \mu)$



Analyzing momentum term

Steep regions: oscillations

$$\nabla E[\mathbf{w}(t+1)] \approx -\nabla E[\mathbf{w}(t)]$$



Net effect (ideally): little

Momentum

Advantage:

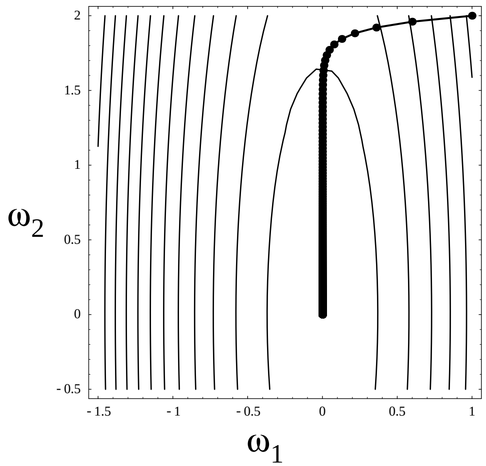
- Increase *effective* learning rate in shallow regions

Disadvantages:

- Yet another parameter to hand tune
- If not carefully chosen, μ can do more harm than good

Convergence examples

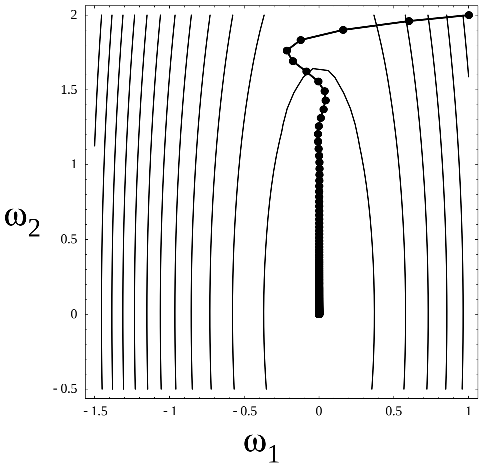
$\eta = 0.01, \mu = 0.0$



719 steps

Convergence examples

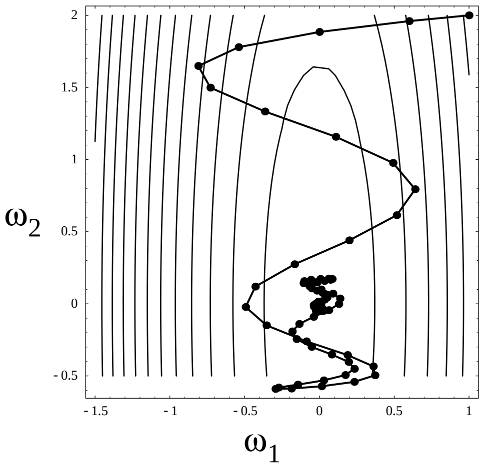
$\eta = 0.01, \mu = 0.5$



341 steps

Convergence examples

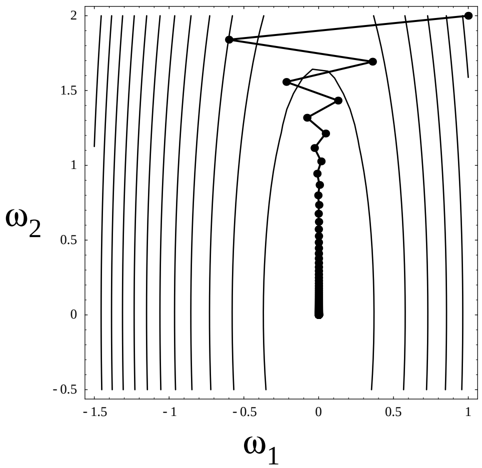
$\eta = 0.01, \mu = 0.9$



266 steps

Convergence examples

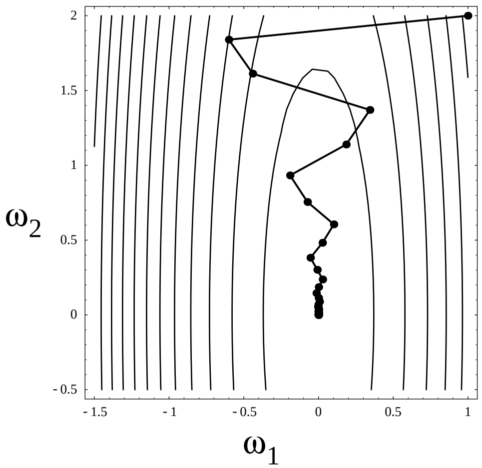
$\eta = 0.04, \mu = 0.0$



175 steps

Convergence examples

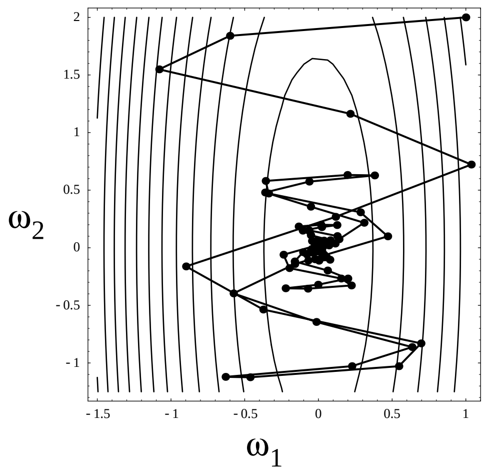
$\eta = 0.04, \mu = 0.5$



60 steps

Convergence examples

$\eta = 0.04, \mu = 0.9$



272 steps

Convergence examples: summary

	$\mu = 0.0$	$\mu = 0.5$	$\mu = 0.9$
$\eta = 0.01$	*719*	*341*	*266*
$\eta = 0.04$	*175*	*60*	*272*

Heuristic extensions to gradient descent

Momentum popular in neural network community.

Many other heuristic attempts (some examples):

- Adaptive learning rate (what should ρ and σ be?)

$$\eta_{new} = \begin{cases} \rho\eta_{old} & \Delta E < 0 \\ \sigma\eta_{old} & \Delta E > 0 \end{cases}$$

- $\eta_{max} = 2/\lambda_{max}$ (what's the problem here?)

Heuristic extensions to gradient descent

- Individual learning rates:

$$\Delta\eta_i = \gamma g_i^{(t)} g_i^{(t-1)} \text{ (problems?)}$$

- Quickprop: local independent quadratic assumption:

$$\Delta\omega_i^{(t+1)} = \frac{g_i^{(t)}}{g_i^{(t-1)} - g_i^{(t)}} \Delta\omega_i^{(t)} \text{ (problems?)}$$

Heuristic extensions to gradient descent

Problems:

- Additional hand-tuned parameters
- **Independence of weight assumptions**

More principled approach is desirable.

Steepest descent

Gradient descent:

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \Delta\mathbf{w}(t)$$

where,

$$\Delta\mathbf{w}(t) = -\eta\nabla E[\mathbf{w}(t)]$$

Question: why take all those little tiny steps?

Steepest descent: gradient descent with line minimization

1. Define search direction $\mathbf{d}(t)$:

$$\mathbf{d}(t) = -\nabla E[\mathbf{w}(t)]$$

2. Minimize:

$$E(\eta) \equiv E[\mathbf{w}(t) + \eta\mathbf{d}(t)]$$

such that:

$$E(\eta^*) \le E(\eta), \ \forall\eta$$

3. New update:

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \eta^*\mathbf{d}(t) \text{ (problems?)}$$

Steepest descent

Question: Do we need to compute $\partial E / \partial\eta$?

Answer: No. Use one-dimensional ***line search*, which requires only evaluation of $E(\eta)$.**

Line search: two steps

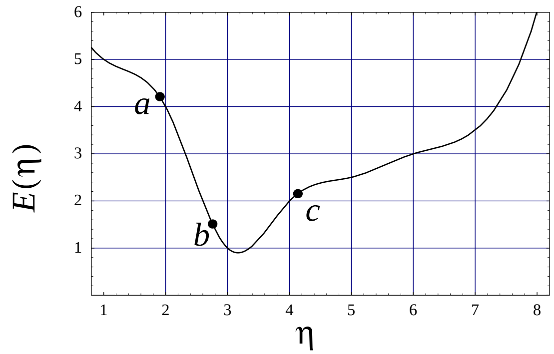
1. Bracket minimum
2. Line minimization

Line search: bracketing the minimum

Basic problem: need three values a, b, c **such that:**

$E(a) > E(b)$

$E(c) > E(b)$



Bracketing the minimum

1. Let $a = 0$. Let $b = \varepsilon$.

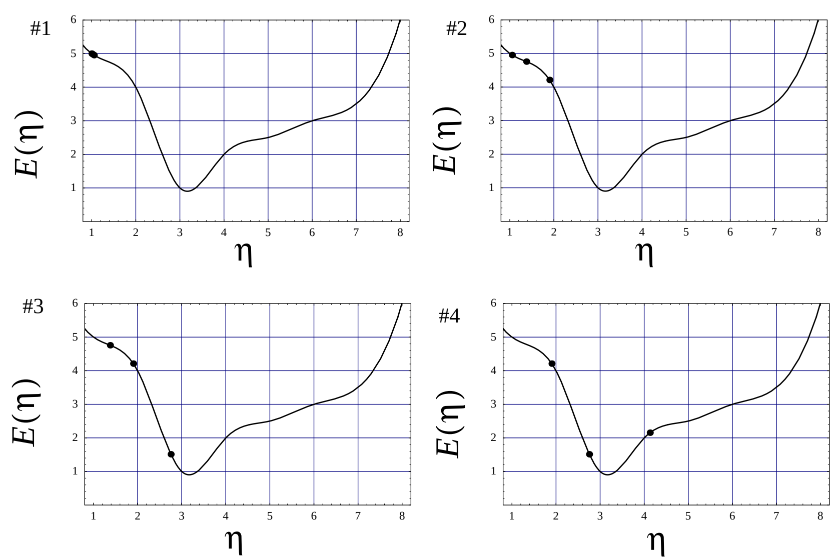
   Note: will satisfy $E(a) > E(b)$ (why?).

2. Let $c = k(b - a) + a$, where $k > 1$ (what should k be?).

3. If $E(c) > E(b)$, then done; else, let $a = b$ and $b = c$.
   Repeat step 2.

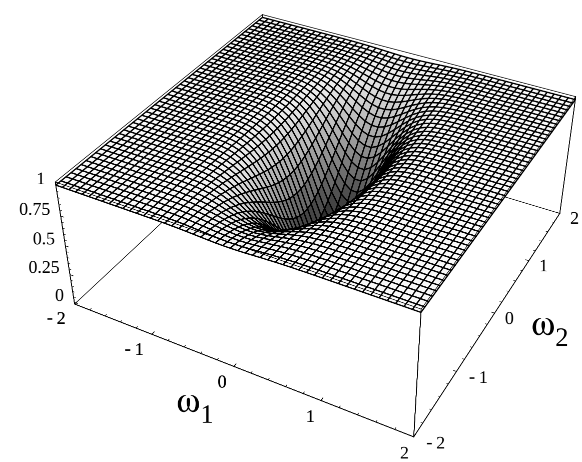
Note: one evaluation of E **per step.**

Bracketing example



Bracketing example: error surface

$$E(\omega_1, \omega_2) = 1 - \exp(-5\omega_1^2 - \omega_2^2)$$



What is $E(\eta)$?

Weights $(\omega_1, \omega_2) = (1, 2)$

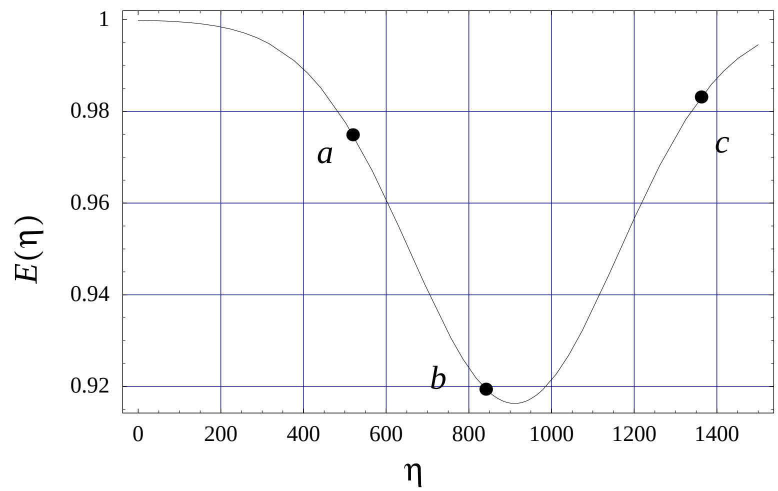
$$\nabla E(\omega_1, \omega_2) = [10\omega_1 \exp(-5\omega_1^2 - \omega_2^2), 2\omega_2] \exp(-5\omega_1^2 - \omega_2^2)$$

$$E(\eta) = 1 - \exp(-5(\omega_1 - 10\omega_1 \eta \exp(-5\omega_1^2 - \omega_2^2))^2 - (\omega_2 - 2\omega_2 \eta \exp(-5\omega_1^2 - \omega_2^2))^2)$$

At $(\omega_1, \omega_2) = (1, 2)$**:**

$$E(\eta) = 1 - \exp(-5(1 - 10\eta \exp(-9))^2 - (2 - 4\eta \exp(-9))^2)$$

Bracketing example: error surface



Line minimization

1. Pick a value of $\eta = x$ in larger interval: (a, b) or (b, c).
2. If (a, b) is larger interval, set new bracketing values to:

   $\{x, b, c\}$ if $E(x) > E(b)$, (set $a = x$), or

   $\{a, x, b\}$, if $E(b) > E(x)$, (set $c = b$ and $b = x$).

   Else, set new bracketing values to,

   $\{a, b, x\}$ if $E(x) > E(b)$, (set $c = x$), or

   $\{b, x, c\}$, if $E(b) > E(x)$, (set $a = b$ and $b = x$).
3. Iterate steps 1 and 2 until $(c - a) < \theta$.

Line minimization

What should the value of x be?

$x = 0.381966(c - b) + b$ [(b, c) is larger interval]

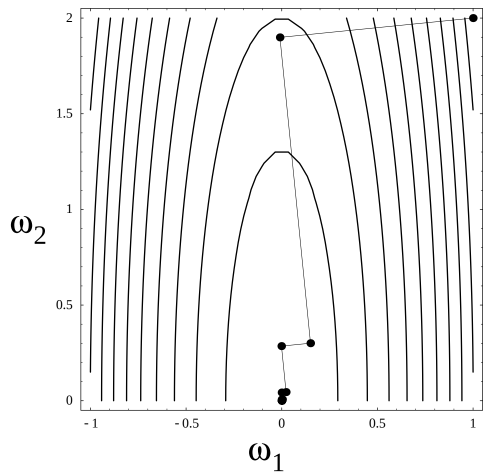
$x = b - 0.381966(b - a)$ [(a, b) is larger interval]

Rate of convergence proportional to:

$$\frac{1}{k} \approx 0.61803$$

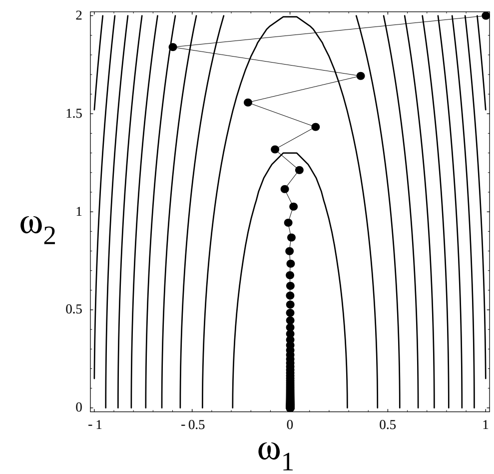
$$k = \frac{1 + \sqrt{5}}{2} \approx 1.61803 \text{ (golden mean)}$$

Examples: quadratic surface



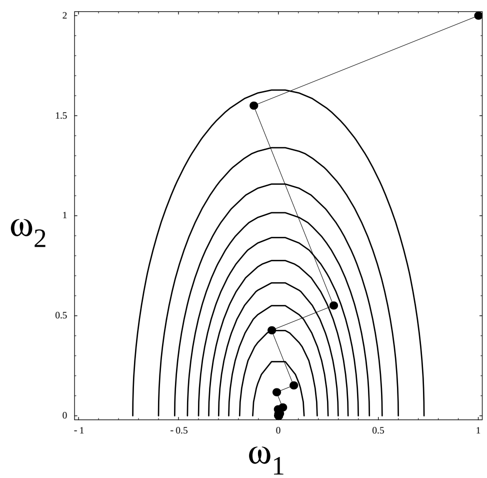
steepest descent
15 steps to convergence

Examples: quadratic surface (comparison)



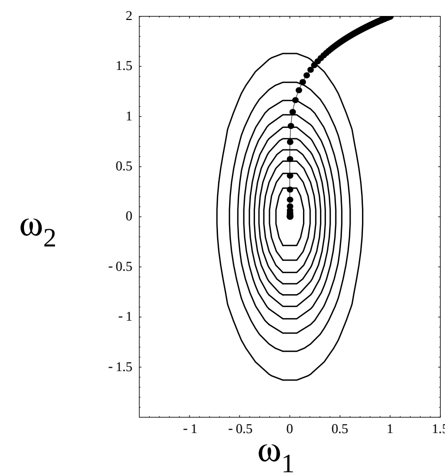
gradient descent $(\eta = 0.04)$
175 steps to convergence

Examples: nonquadratic surface



steepest descent
24 steps to convergence

Examples: nonquadratic surface



gradient descent $(\eta = 0.2)$
456 steps to convergence

Computational complexity

Steepest descent:

$$5NW + 10(2NW) = 25NW$$ computations/step (why?)

Gradient descent:

$5NW$ computations/step

Discussion

What's bad about steepest descent?

Answer: Orthogonality of consecutive steps.

- Why does this occur?
- Can we do better?