Numerical Optimization Techniques

Léon Bottou

NEC Labs America

COS 424 – 3/2/2010

Today's Agenda

Goals	Classification, clustering, regression, other.
Representation	Parametric vs. kernels vs. nonparametric Probabilistic vs. nonprobabilistic Linear vs. nonlinear Deep vs. shallow
Capacity Control	Explicit: architecture, feature selection Explicit: regularization, priors Implicit: approximate optimization Implicit: bayesian averaging, ensembles
Operational Considerations	Loss functions Budget constraints Online vs. offline
Computational Considerations	Exact algorithms for small datasets. Stochastic algorithms for big datasets. Parallel algorithms.

Introduction

General scheme

– Set a goal.
– Define a parametric model.
– Choose a suitable loss function.
– Choose suitable capacity control methods.
– Optimize average loss over the training set.

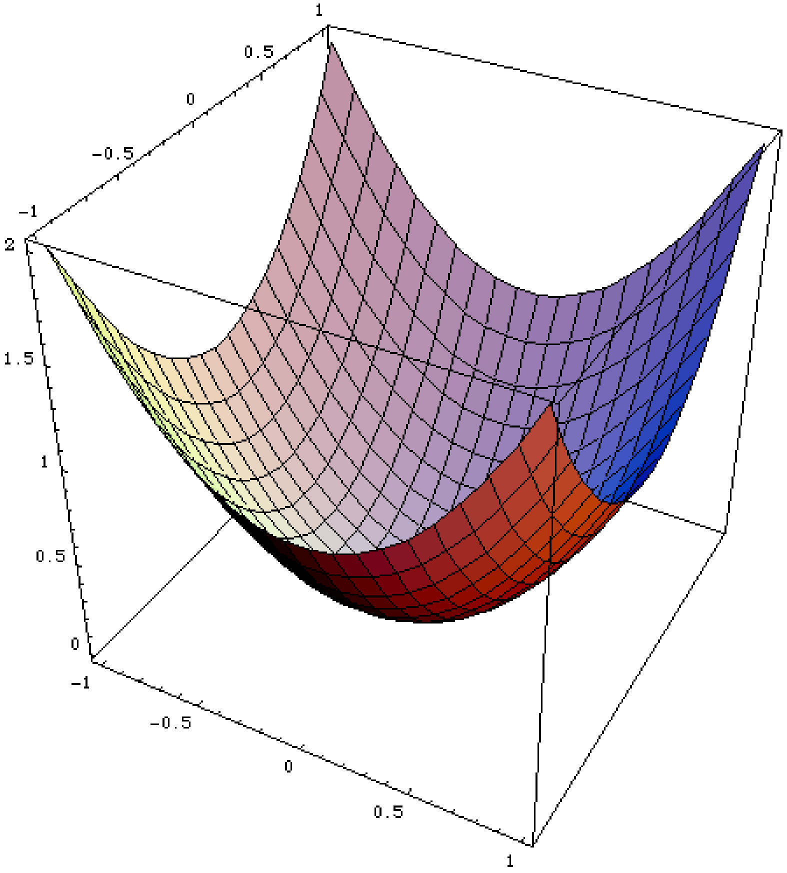
Optimization

– Sometimes analytic (e.g. linear model with squared loss.)
– Usually numerical (e.g. everything else.)

Summary

1. Convex vs. Nonconvex
2. Differentiable vs. Nondifferentiable
3. Constrained vs. Unconstrained
4. Line search
5. Gradient descent
6. Hessian matrix, etc.
7. Stochastic optimization

Convex



Definition

$\forall\, x, y,\ \forall\, 0 \leq \lambda \leq 1,$
$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y)$

Property

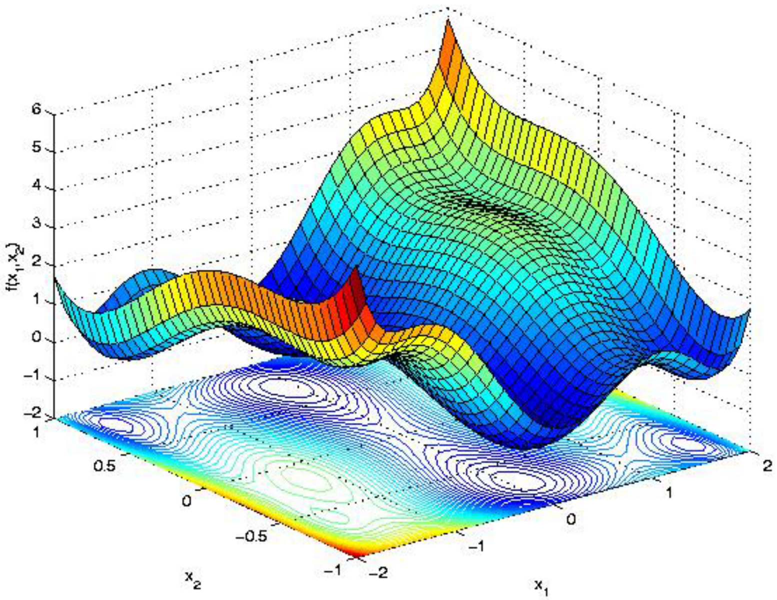
Any local minimum is a global minimum.

Conclusion

Optimization algorithms are easy to use.
They always return the same solution.

Example: Linear model with convex loss function.
– Curve fitting with mean squared error.
– Linear classification with log-loss or hinge loss.

Nonconvex



Landscape

– local minima, saddle points.

– plateaux, ravines, etc.

Optimization algorithms

– Usually find local minima.

– Good and bad local minima.

– Result depend on subtle details.

Examples

– Multilayer networks.

– Clustering algorithms.

– Learning features.

– Semi-supervised learning.

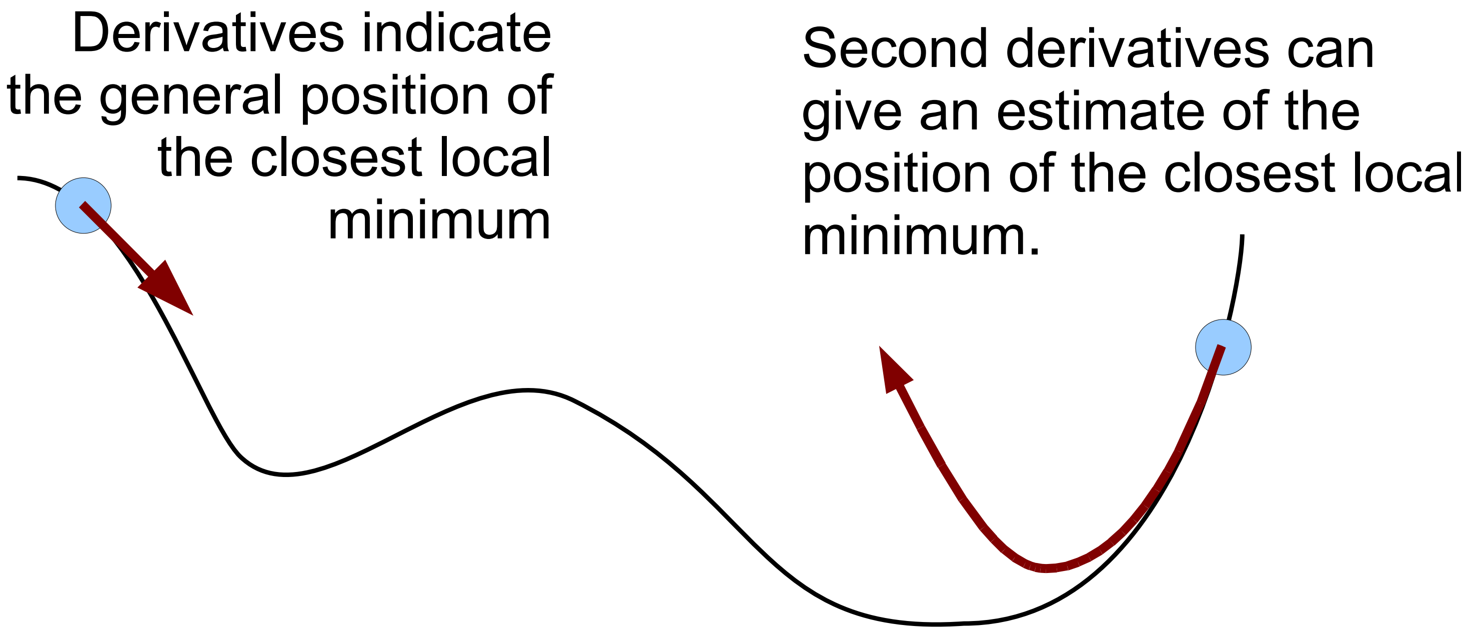
– Mixture models.

– Hidden Markov Models.

– Selecting features (some).

– Transfer learning.

Differentiable vs. Nondifferentiable



No such local cues without derivatives
– Derivatives may not exist.
– Derivatives may be too costly to compute.

Examples

– Log loss versus Hinge loss.

Constrained vs. Unconstrained

Compare

$$\min_w f(w) \quad \text{subject to} \quad w^2 < C$$

$$\min_w f(w) + \lambda w^2$$

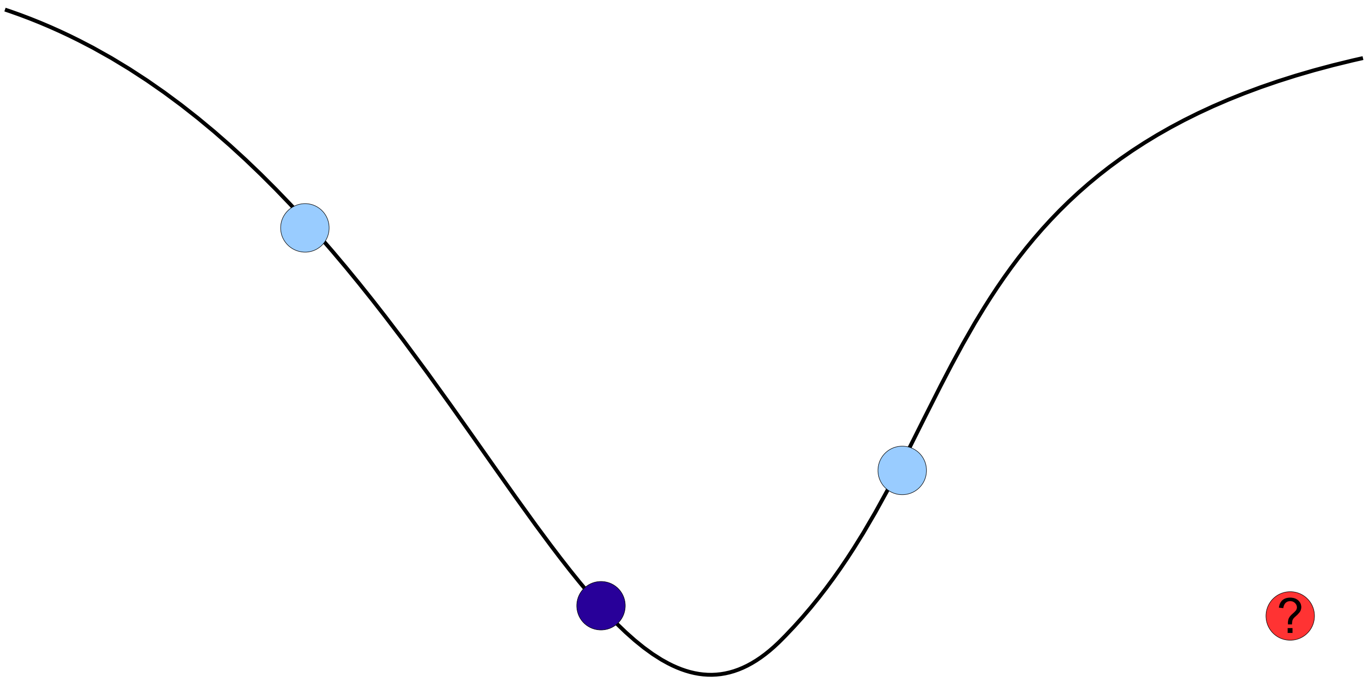
Constraints

– Adding constraints lead to very different algorithms.

Keywords

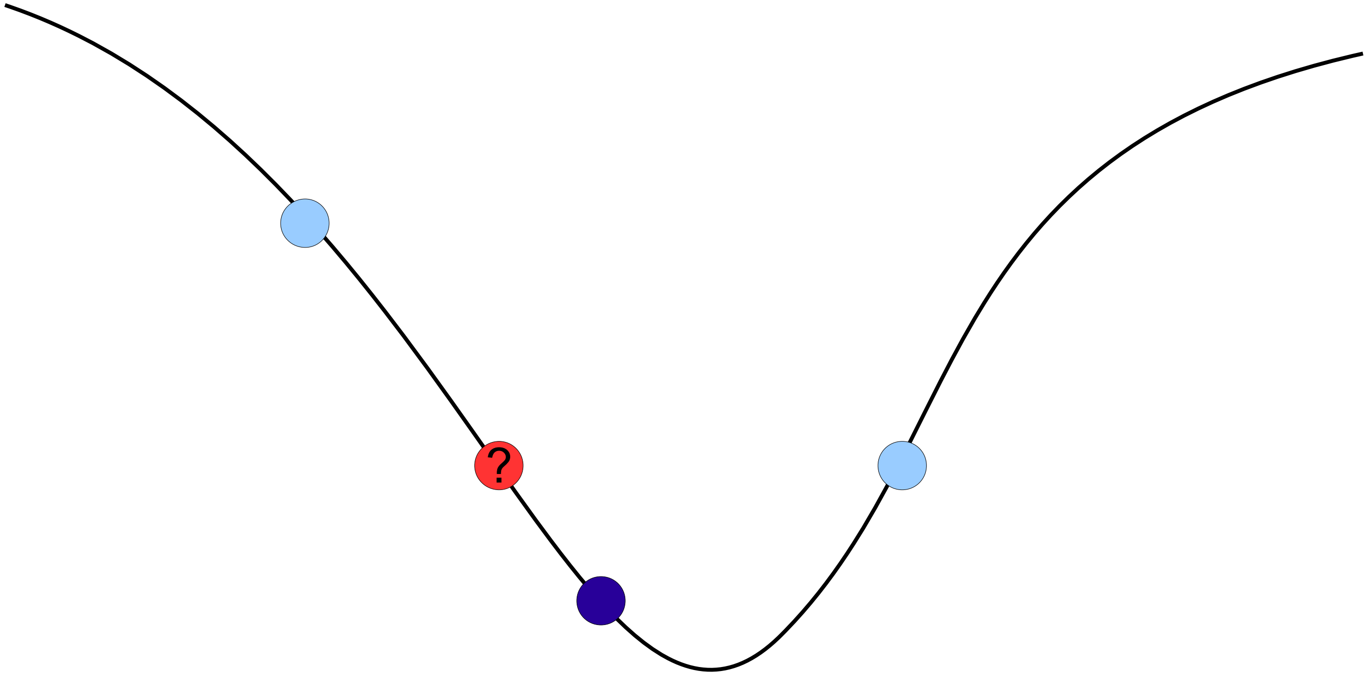
– Lagrange coefficients.
– Karush-Kuhn-Tucker theorem.
– Primal optimization, dual optimization.

Line search - Bracketing a minimum



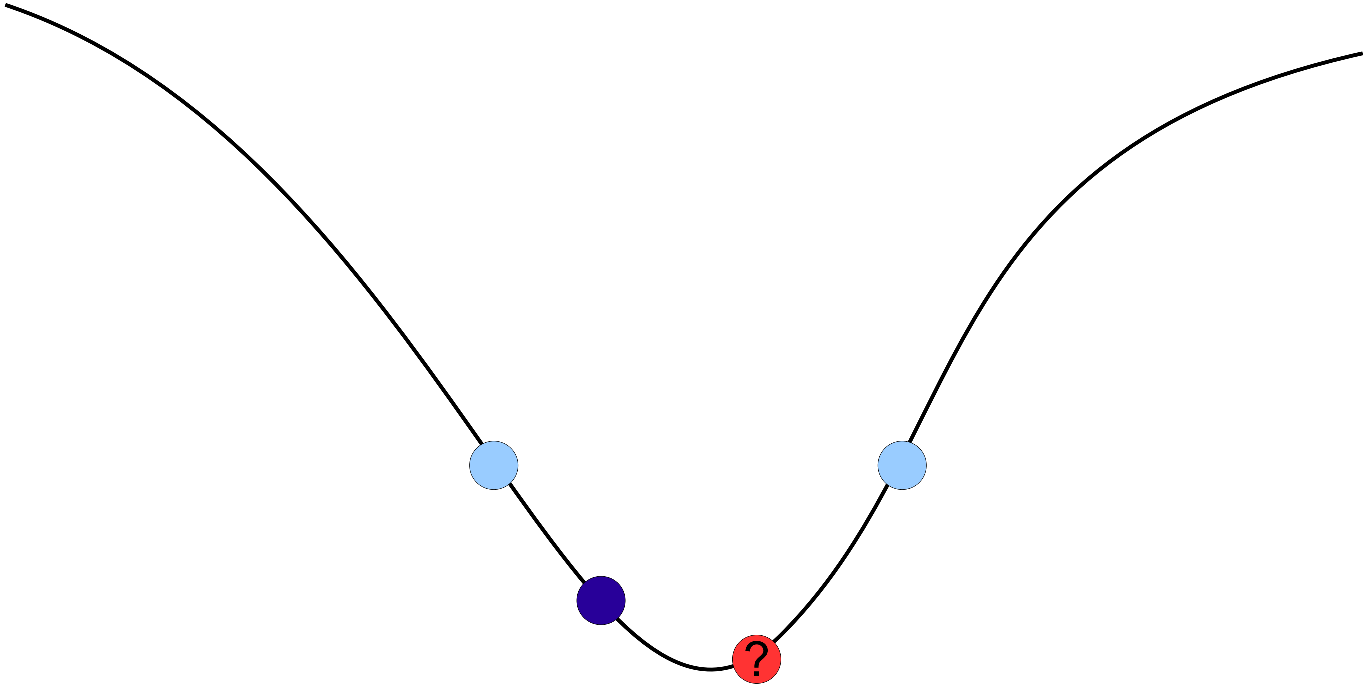
Three points $a < b < c$ such that $f(b) < f(a)$ and $f(b) < f(c)$.

Line search - Refining the bracket



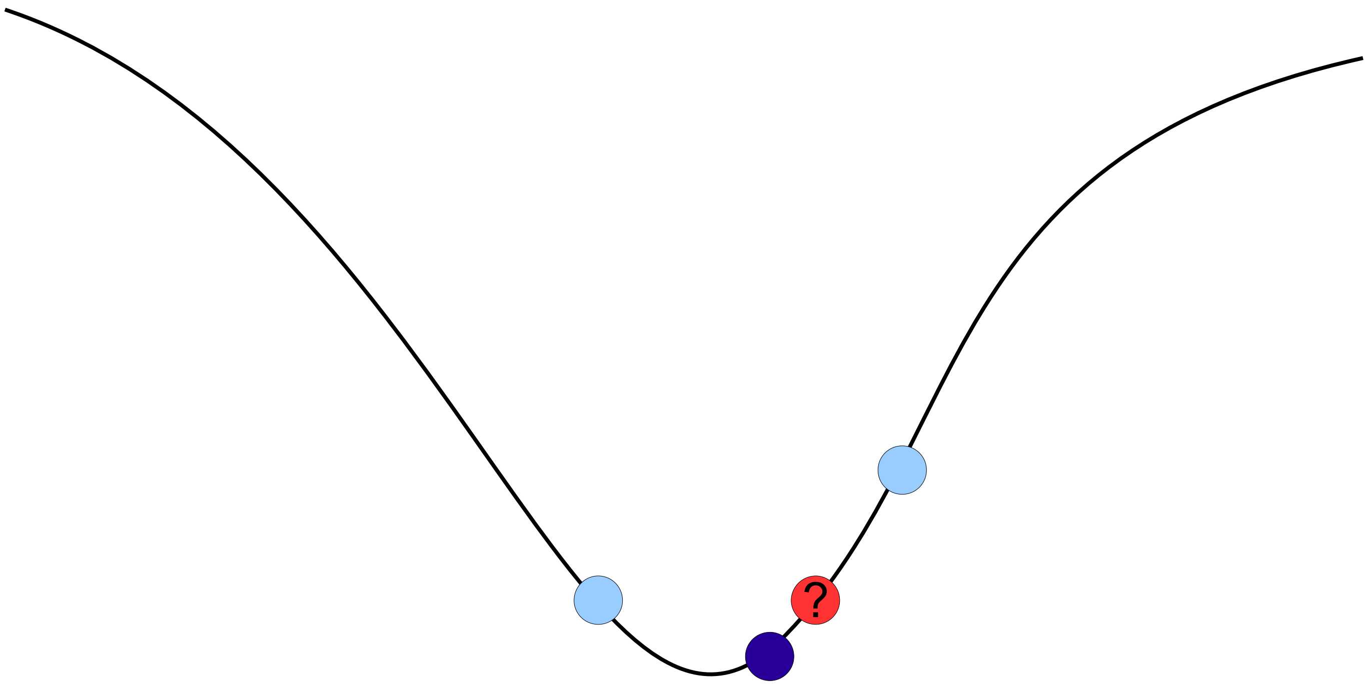
Split the largest half and compute $f(x)$.

Line search - Refining the bracket



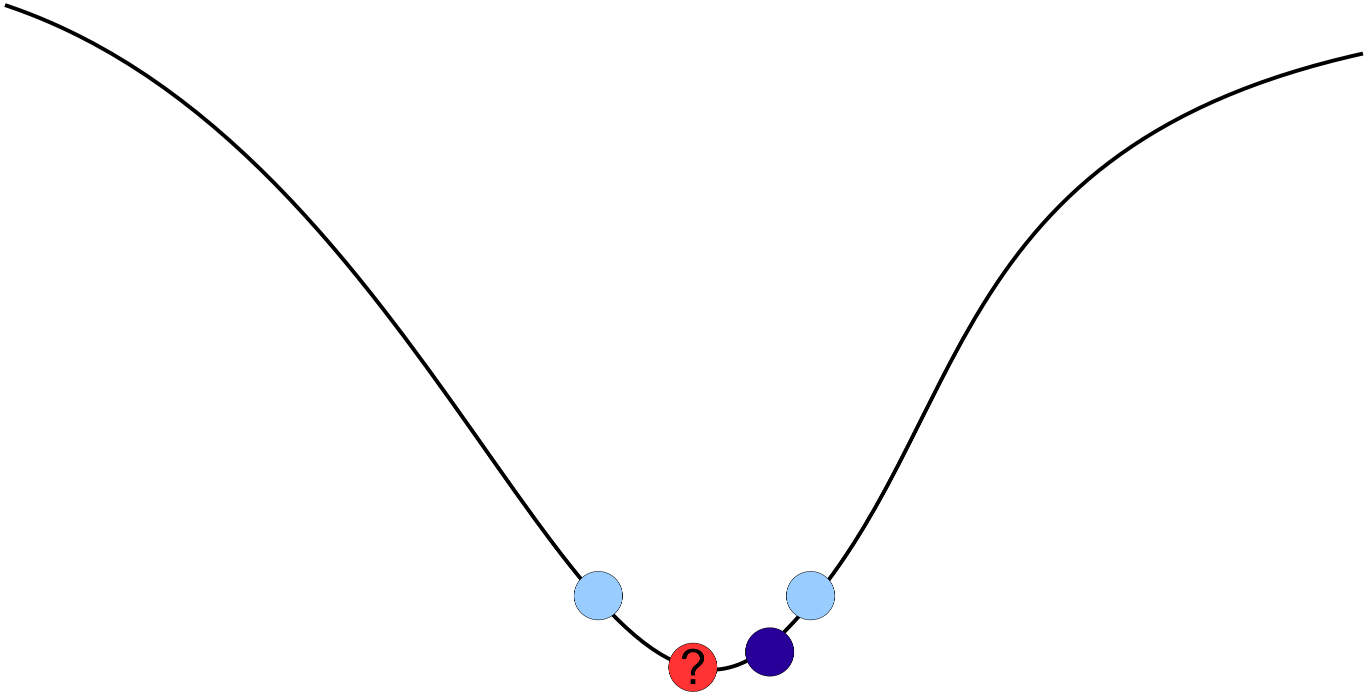
– Redefine $a < b < c$. Here $a \leftarrow x$.
– Split the largest half and compute $f(x)$.

Line search - Refining the bracket



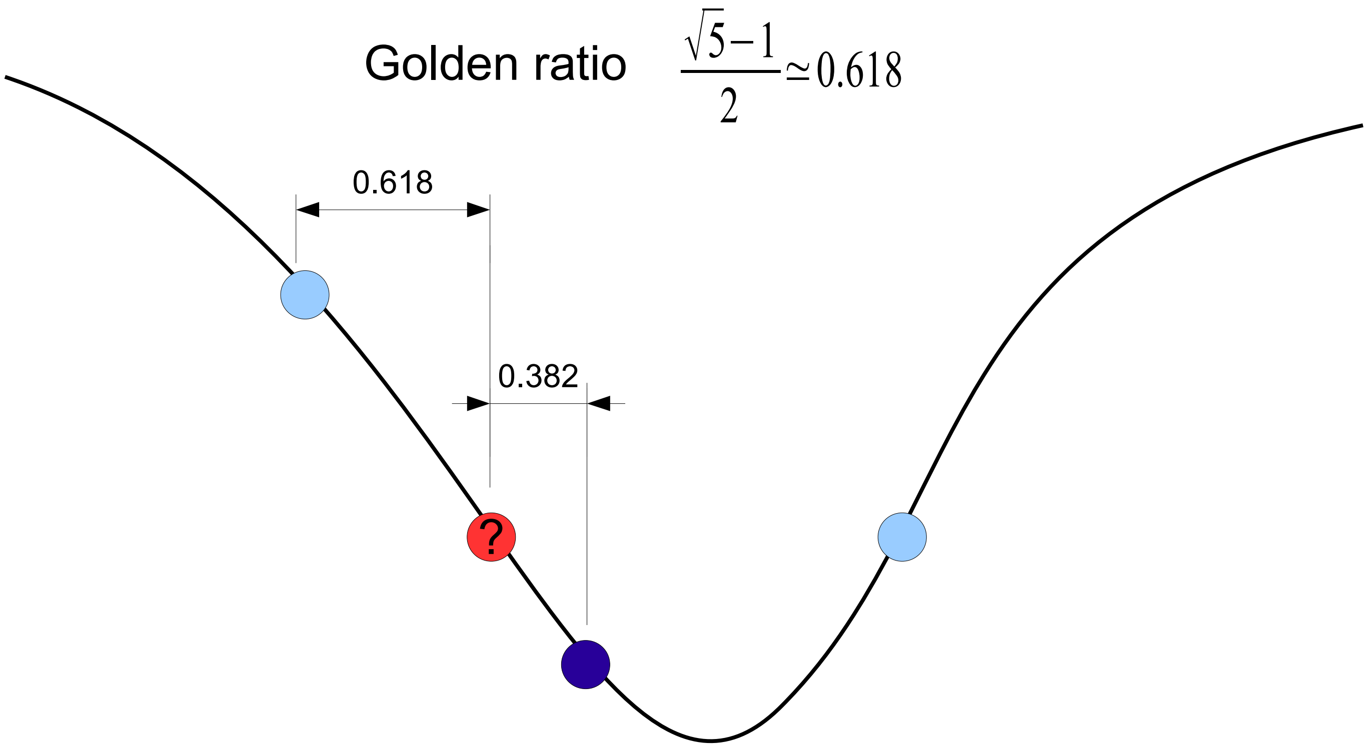
– Redefine $a < b < c$. Here $a \leftarrow b$, $b \leftarrow x$.
– Split the largest half and compute $f(x)$.

Line search - Refining the bracket



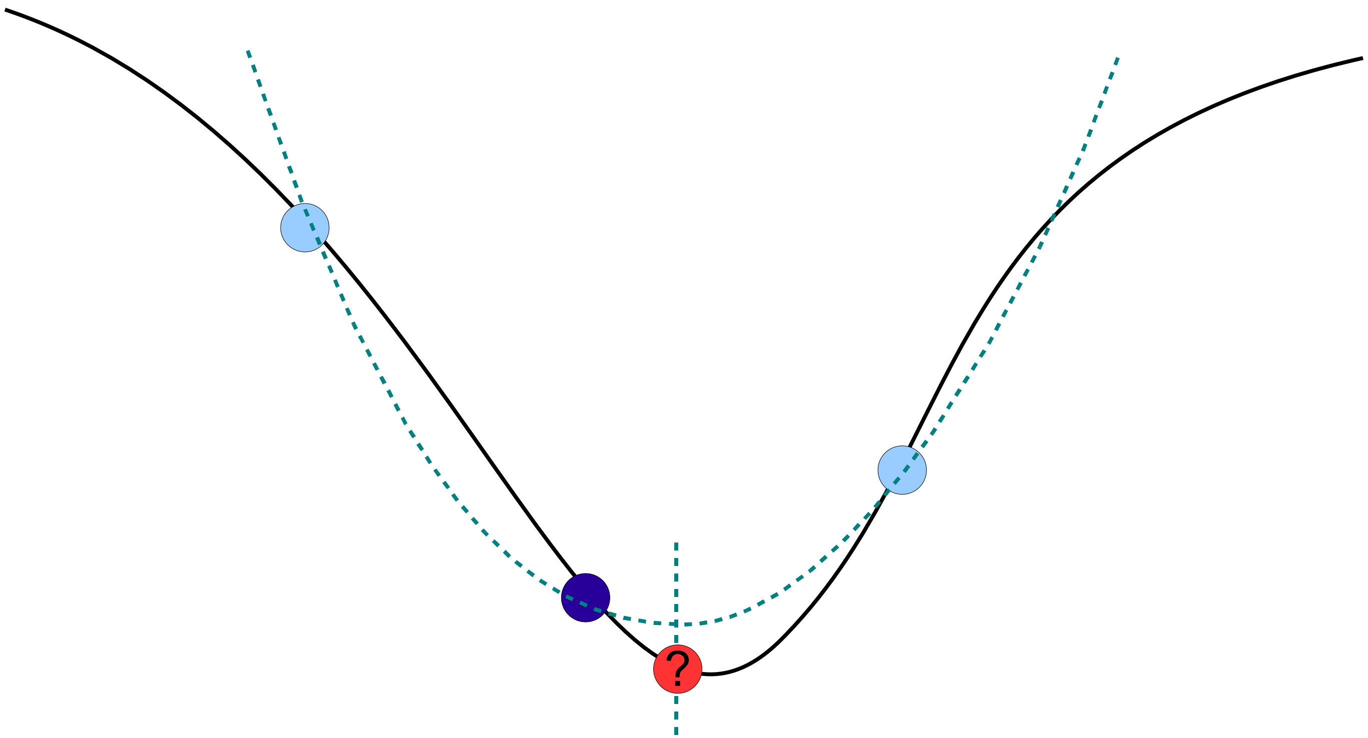
– Redefine $a < b < c$. Here $c \leftarrow x$.
– Split the largest half and compute $f(x)$.

Line search - Golden Section Algorithm



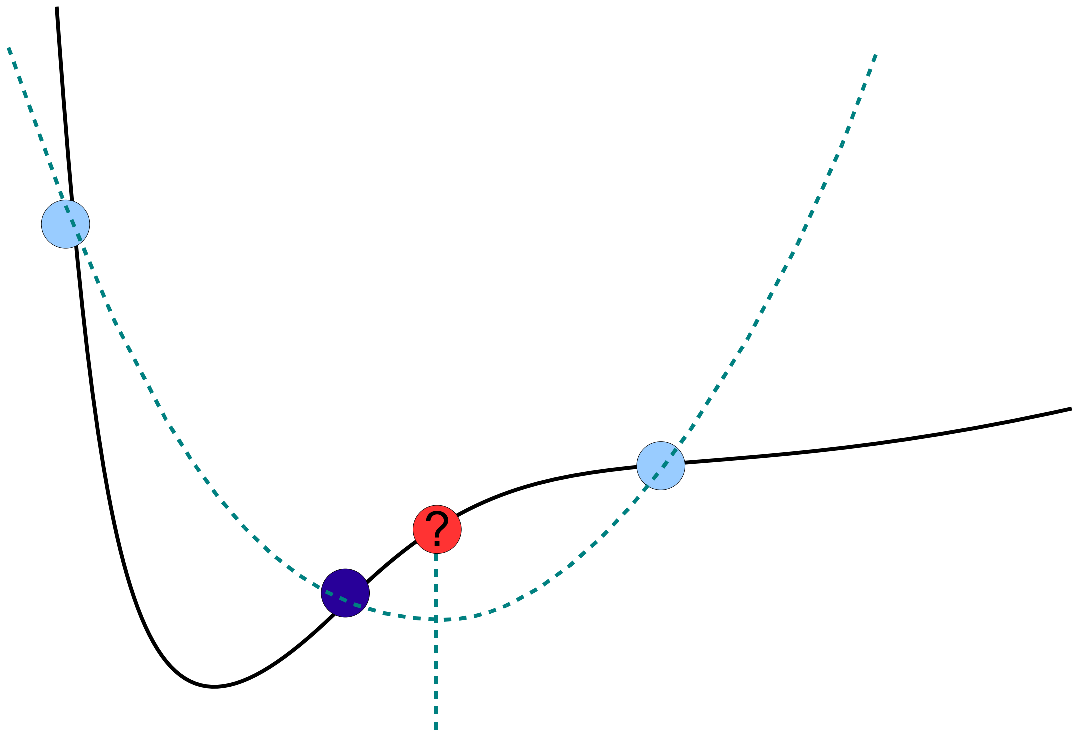
– Optimal improvement by splitting at the *golden ratio*.

Line search - Parabolic Interpolation



– Fitting a parabola can give much better guess.

Line search - Parabolic Interpolation



– Fitting a parabola sometimes gives much better guess.

Line search - Brent Algorithm

Brent Algorithm for line search

– Alternate golden section and parabolic interpolation.

– No more than twice slower than golden section.

– No more than twice slower than parabolic section.

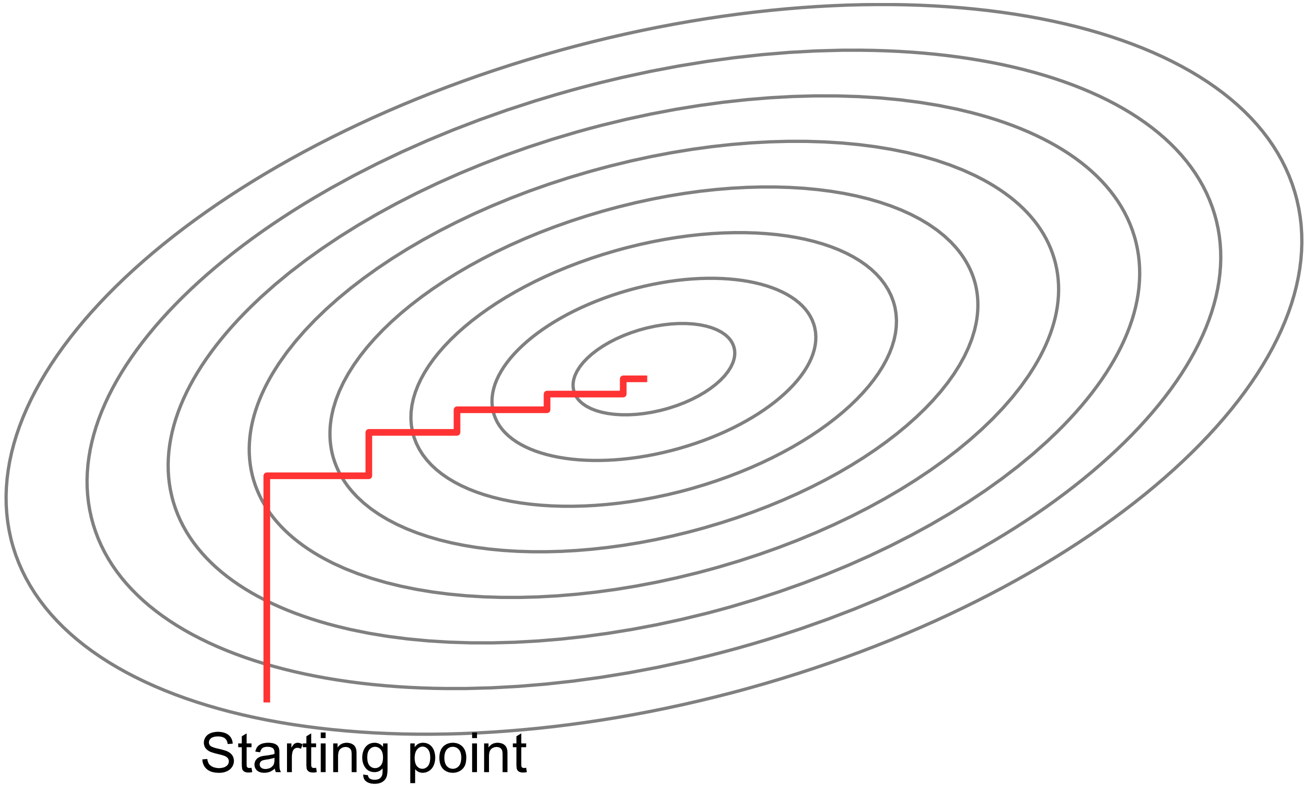
– In practice, almost as good as the best of the two.

Variants with derivatives

– Improvements if we can compute $f(x)$ and $f'(x)$ together.

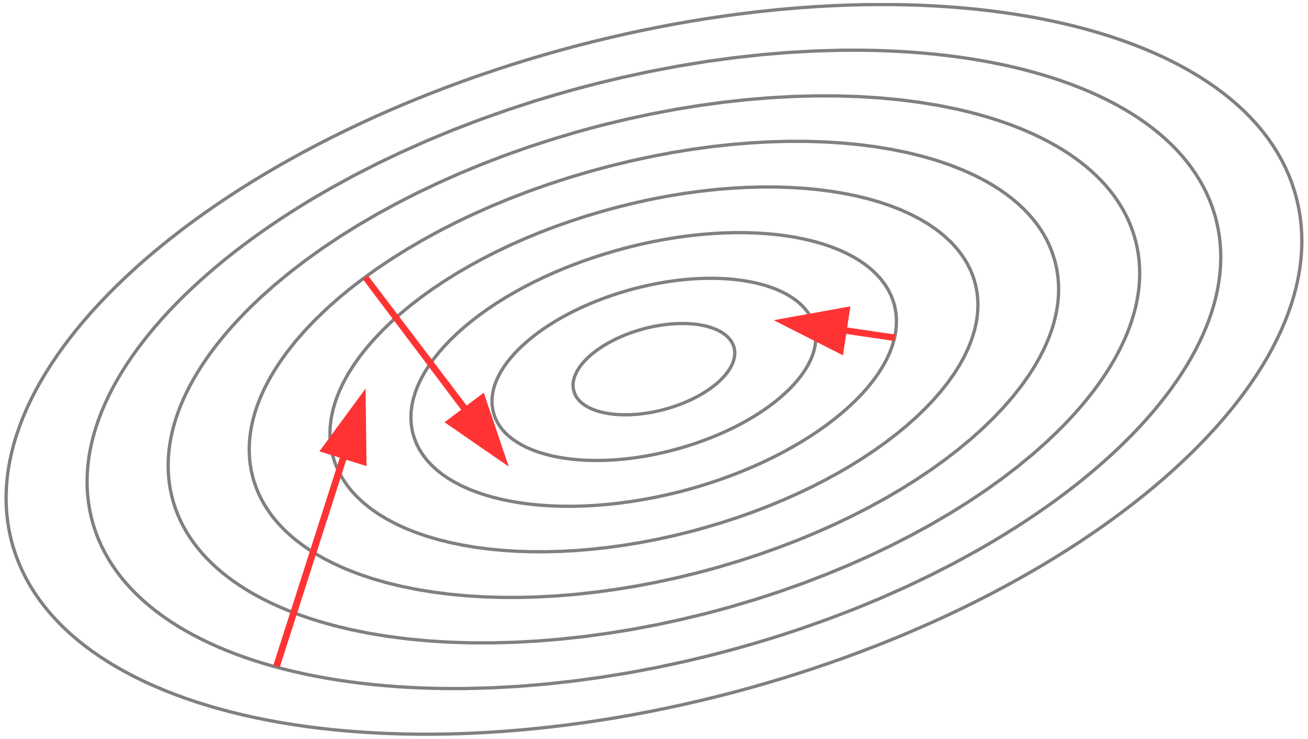
– Improvements if we can compute $f(x)$, $f'(x)$, $f''(x)$ together.

Coordinate Descent



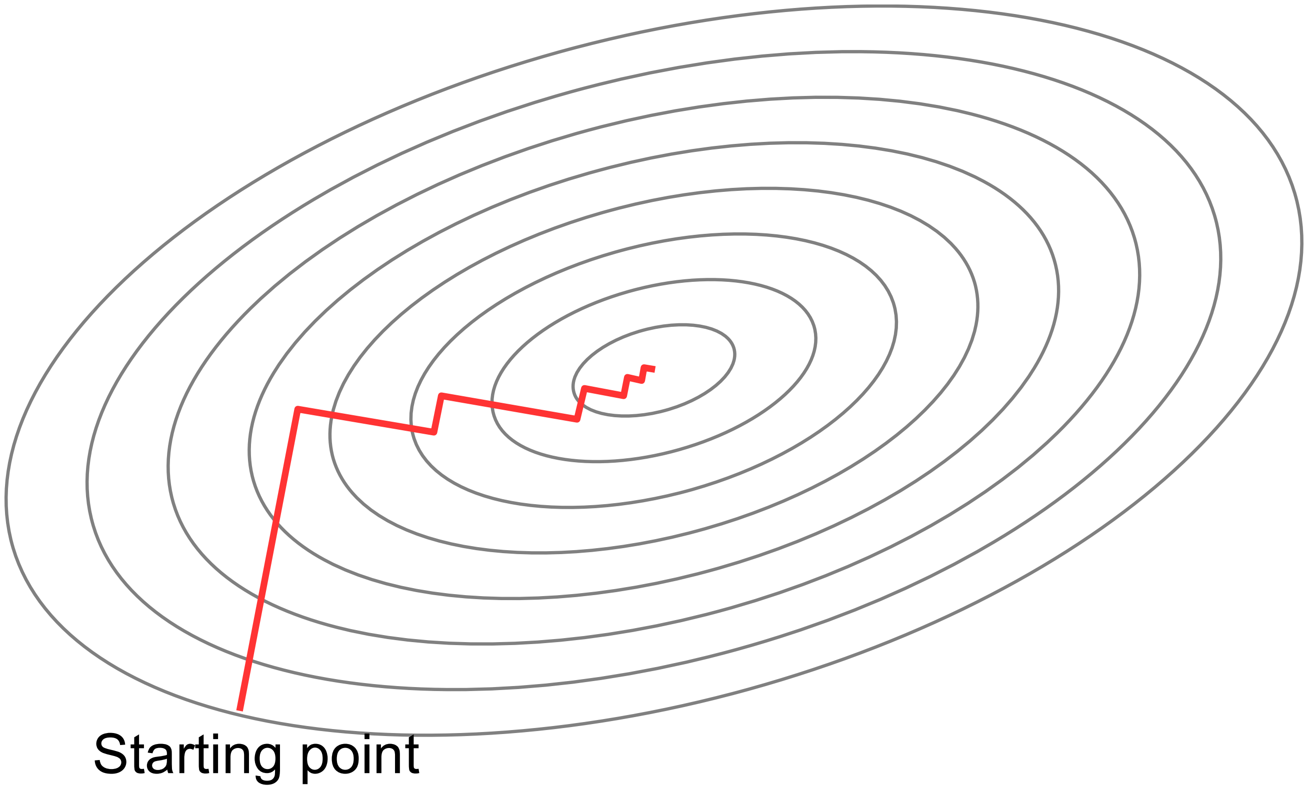
Perform successive line searches along the axes.
– Tends to zig-zag.

Gradient



The gradient $\frac{\partial f}{\partial w} = \left(\frac{\partial f}{\partial w_1}, \ldots, \frac{\partial f}{\partial w_d}\right)$ gives the steepest descent direction.

Steepest Descent



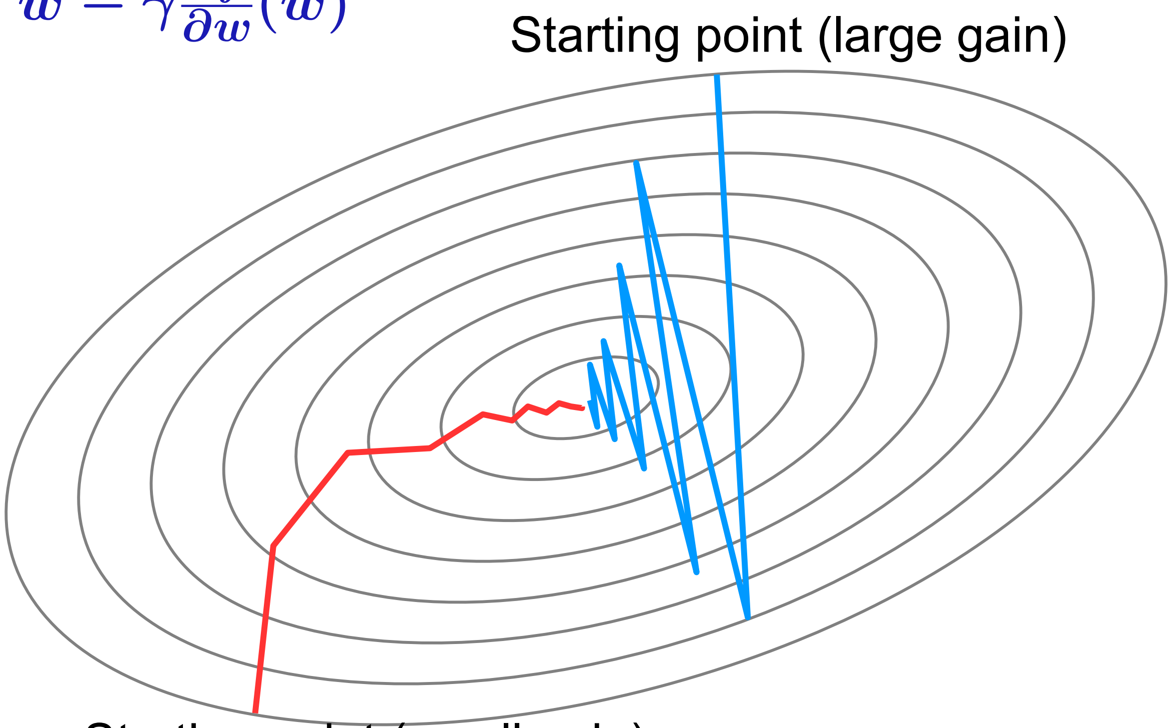
Perform successive line searches along the gradient direction.

– Beneficial if computing the gradients is cheap enough.

– Line searches can be expensive

Gradient Descent

Repeat $w \leftarrow w - \gamma \frac{\partial f}{\partial w}(w)$



– Merge gradient and line search.

– Large gain increase zig-zag tendencies, possibly divergent.

– High curvature direction limits gain size.

– Low curvature direction limits speed of approach.

Hessian matrix

Hessian matrix

$$H(w) = \begin{bmatrix} \frac{\partial^2 f}{\partial w_1\, \partial w_1} & \cdots & \frac{\partial^2 f}{\partial w_1\, \partial w_d} \\ \vdots & & \vdots \\ \frac{\partial^2 f}{\partial w_d\, \partial w_1} & \cdots & \frac{\partial^2 f}{\partial w_d\, \partial w_d} \end{bmatrix}$$

Curvature information

– Taylor expansion near the optimum w^*:

$$f(w) \approx f(w^*) + \frac{1}{2}(w - w^*)^\top H(w^*)\,(w - w^*)$$

– This paraboloid has ellipsoidal level curves.

– Principal axes are the eigenvectors of the Hessian.

– Ratio of curvatures = ratio of eigenvalues of the Hessian.

Newton method

Idea

Since Taylor says $\frac{\partial f}{\partial w}(w) \approx H(w)\,(w - w^*)$ then $w^* \approx w - H(w)^{-1}\frac{\partial f}{\partial w}(w)$.

Newton algorithm

$$\boldsymbol{w \leftarrow w - H(w)^{-1}\frac{\partial f}{\partial w}(w)}$$

– Succession of paraboloidal approximations.
– Exact when $f(w)$ is a paraboloid, e.g. linear model + squared loss.
– Very few iterations needed when $H(w)$ is definite positive!
– Beware when $H(w)$ is not definite positive.
– Computing and storing $H(w)^{-1}$ can be too costly.

Quasi-Newton methods

– Methods that avoid the drawbacks of Newton
– But behave like Newton during the final convergence.

Conjugate Gradient algorithm

Conjugate directions

– u, v conjugate $\iff u^\top H\, v = 0$. Non interacting directions.

Conjugate Gradient algorithm

– Compute $g_t = \frac{\partial f}{\partial w}(w_t)$.
– Determine a line search direction $d_t = g_t - \lambda d_{t-1}$
– Choose λ such that $d_t\, H\, d_{t-1} = 0$.
– Since $g_t - g_{t-1} \approx H\,(w_t - w_{t-1}) \propto H\, d_{t-1}$, this means $\lambda = \frac{g_t(g_t - g_{t-1})}{d_t(g_t - g_{t-1})}$.
– Perform a line search in direction d_t.
– Loop.

This is a fast and robust quasi-Newton algorithm.

A solution for all our learning problems?

Optimization vs. learning

Empirical cost

– Usually $f(w) = \frac{1}{n}\sum_{i=1}^{n} L(x_i, y_i, w)$
– The number n of training examples can be large (billions?)

Redundant examples

– Examples are redundant (otherwise there is nothing to learn.)
– Doubling the number of examples brings a little more information.
– Do we need it during the first optimization iterations?

Examples on-the-fly

– All examples may not be available simultaneously.
– Sometimes they come on the fly (e.g. web click stream.)
– In quantities that are too large to store or retrieve (e.g. click stream.)

Offline vs. Online

Minimize $C(w) = \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^{n} L(x_i, y_i, w)$.

Offline: process all examples together

– Example: minimization by gradient descent

$$\text{Repeat: } w \leftarrow w - \gamma \left(\lambda w + \frac{1}{n}\sum_{i=1}^{n} \frac{\partial L}{\partial w}(x_i, y_i, w) \right)$$

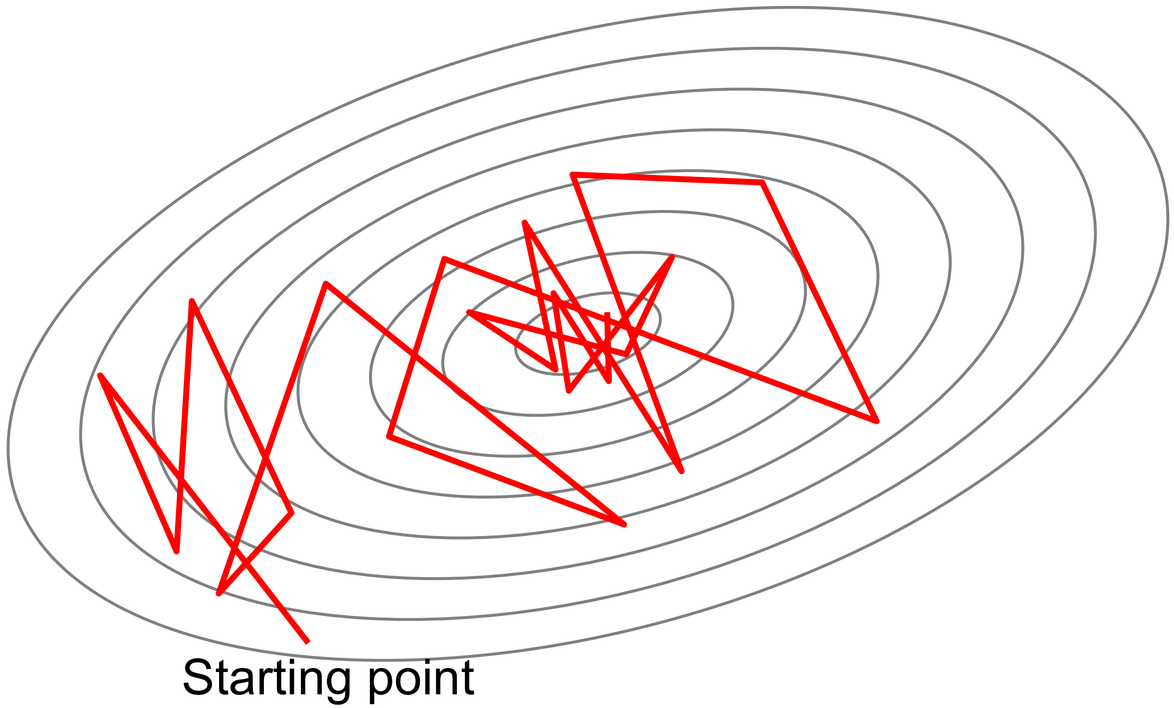
Offline: process examples one by one

– Example: minimization by stochastic gradient descent

Repeat: (a) Pick random example x_t, y_t

(b) $w \leftarrow w - \gamma_t \left(\lambda w + \frac{\partial L}{\partial w}(x_t, y_t, w) \right)$

Stochastic Gradient Descent



- Very noisy estimates of the gradient.
- Gain γ_t controls the size of the cloud.
- Decreasing gains $\gamma_t = \gamma_0(1 + \lambda\gamma_0 t)^{-1}$.
- Why is it attractive?

Stochastic Gradient Descent

Redundant examples

– Increase the computing cost of offline learning.
– Do not change the computing cost of online learning.

Imagine the dataset contains 10 copies of the same 100 examples.

- Offline Gradient Descent
  Computation is 10 times larger than necessary.

- Stochastic Gradient Descent
  No difference regardless of the number of copies.

Practical example

Document classification

- Similar to homework#2 but bigger.
- 781,264 training examples.
- 47,152 dimensions.

Linear classifier with Hinge Loss

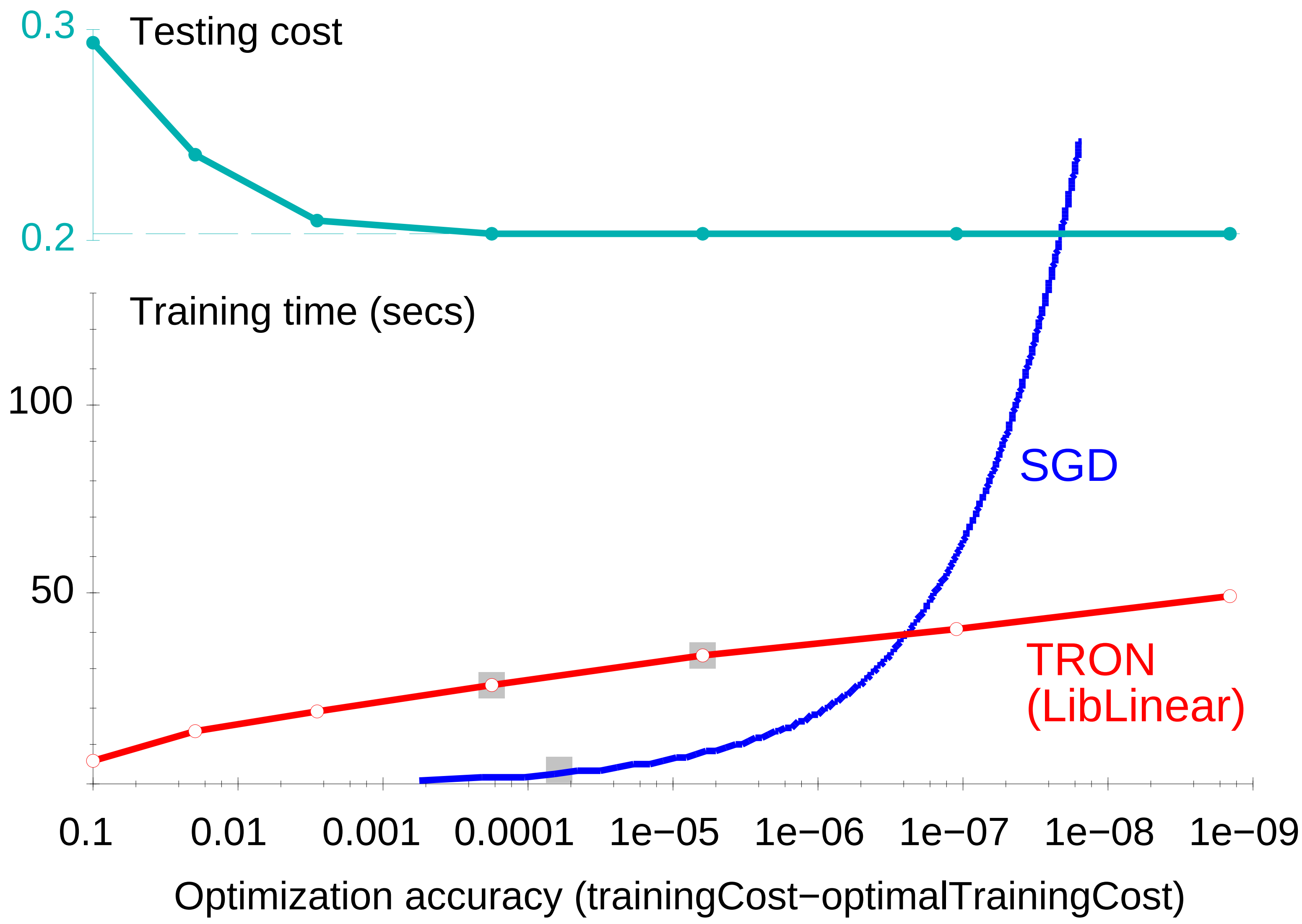
- Offline dual coordinate descent (svmlight): 6 hours.
- Offline primal bundle optimizer (svmperf): 66 seconds.
- Stochastic Gradient Descent: 1.4 seconds.

Linear classifier with Log Loss

- Offline truncated newton (tron): 44 seconds.
- Offline conjugate gradient descent: 40 seconds.
- Stochastic Gradient Descent: 2.3 seconds.

These are times to reach the same test set error.

The wall


0.3
Testing cost
0.2
Training time (secs)
100
50
SGD
TRON
(LibLinear)
0.1
0.01
0.001
0.0001
1e−05
1e−06
1e−07
1e−08
1e−09
Optimization accuracy (trainingCost−optimalTrainingCost)