Offline and Online Optimization in Machine Learning

Peter Sunehag

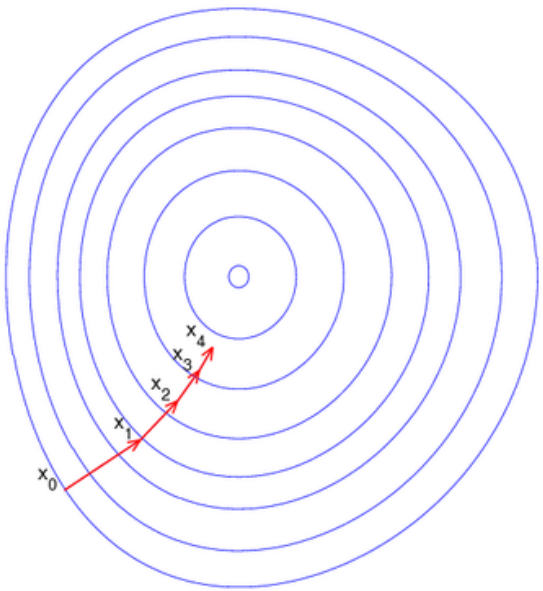
ANU

September 2010

Section

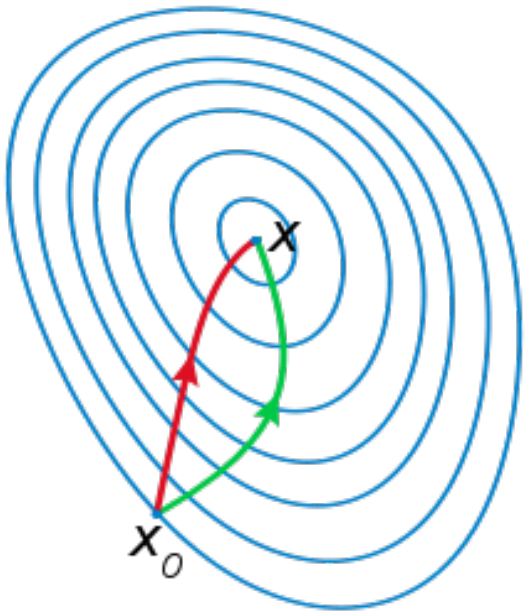
Steepest decent

- We want to find the minimum of a real valued function $f : \mathbb{R}^n \to \mathbb{R}$.
- We will primarily discuss optimization using gradients
- Gradient Decent is also called the steepest decent method



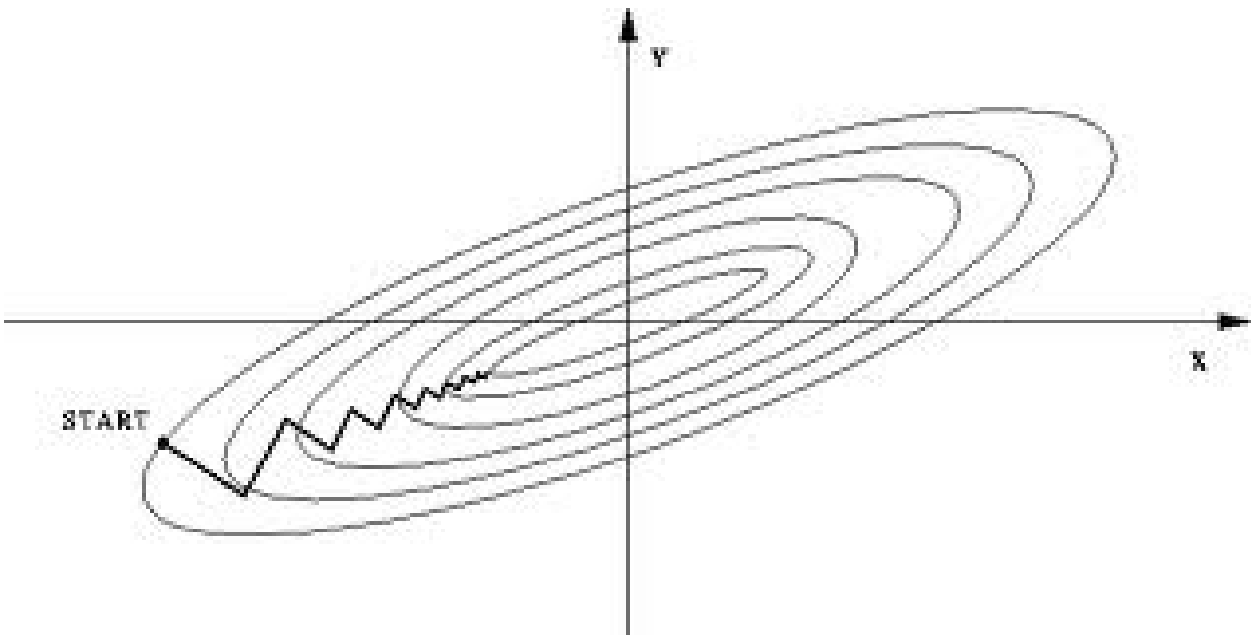
Problem: Curvature

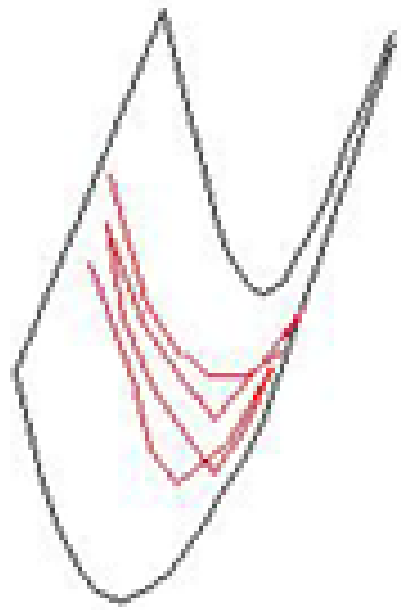
- If we the level sets are very curved (far from round), gradient decent might take a long route to the optimum
- Newton's method takes curvature into account and "corrects" for it. It relies on calculating inverse Hessian (second derivatives) of the objective function
- Quasi-Newton methods estimates the "correction" by looking at the path so far. Red Newton, Green Gradient Decent



Problem: Zig-Zag

- Gradient Descent has a tendency to Zig-Zag
- This is particularly problematic in valleys
- One solution: Introduce momentum
- Another solution: Conjugate gradients





Section

Optimization in Machine Learning

- Data is becoming "unlimited" in many areas.
- Computational resources are not
- Taking advantage of the data requires efficient optimization
- Machine Learning researchers have realized that we are in a different situation compared to the contexts in which the optimization (say convex optimization) field developed.
- Some of those methods and theories are still useful.
- Different ultimate goal.
- We optimize surrogates for what we really want.
- We also often have much higher dimensionality than other applications.

Optimization in Machine Learning

- Known training data $\mathcal{A}$, unknown test data $\mathcal{B}$
- We want optimal performance on the test data
- Alternatively we have streaming data (or pretend that we do).
- Given a loss function $L(w, z)$ (parameters $w \in W$, data sample(s) z), we want as low loss $\sum_{z \in \mathcal{B}} L(z, w)$ as possible on the test set.
- Since we do not have access to the test set, we minimize regularized empirical loss on the training set

$$\sum_{z \in \mathcal{A}} L(z, w) + \lambda R(w).$$

- Getting extremely close to the minimum of this objective might not increase performance on the test set.
- In Machine learning, the optimization methods of interest are those that fast get a good estimate w_k to the optimum w^*

Optimization in Machine Learning

- How large k to get $\epsilon_k = f(w_k) - f(w^*) \leq \epsilon$ for a given $\epsilon > 0$?
- Is this number like $1/\epsilon, 1/\epsilon^2, 1/\sqrt{\epsilon}, \log \frac{1}{\epsilon}$?
- Is ϵ_k like $1/k, 1/k^2$, etc
- How long does an iteration take?
- Quasi-Newton methods (BFGS, L-BFGS), are rapid when we are already close to the optimum but otherise not better than methods (gradient decent) with much cheaper iterations.
- We care the most about the early phase when we are not close to the optimum

Examples

- linear classifier (classes ± 1) $f(x) = \text{sgn}(w^T x + b)$.
- Training data $\{(x_i, y_i)\}$
- SVMs (hinge loss),
  $\min_w \sum_i \min(0, 1 - y_i(w^T x_i + b)) + \lambda \|w\|_2^2$
- Logistic multi-class: $\min_w \sum_i \log \frac{\exp w_{y_i}^T x_i}{\sum_l \exp w_l^T x_i} + \lambda \|w\|_2^2$
- Decision $l^* = \text{argmax}_l \{w_l^T x\}$
- Different regularizers, e.g. $\|x\|_1$ encourages sparsity.
- hinge loss and ℓ_1 regularization causes non-differentiability on null sets.

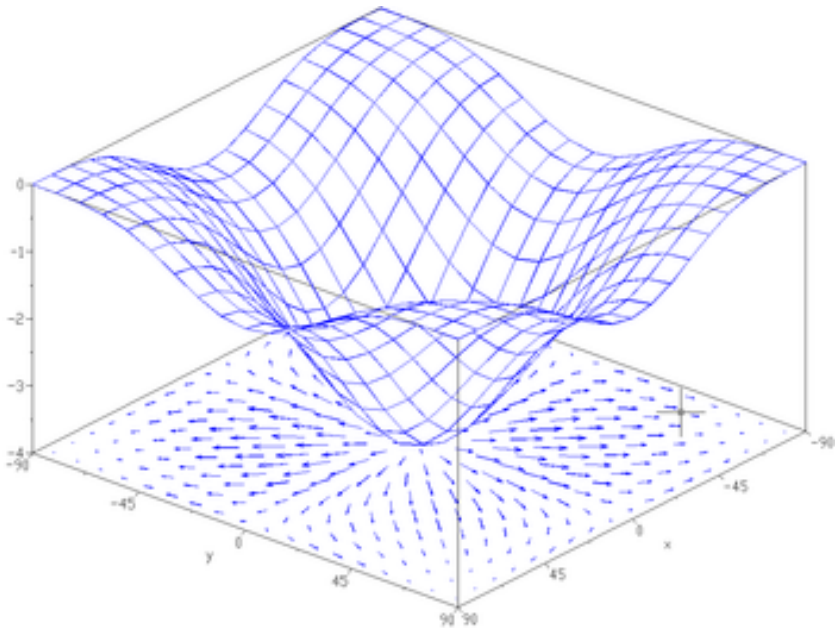
Section

Gradients

- We will make use of gradients with respect to a given coordinate system and its associated Euclidean geometry.
- If $f(x)$ is a real valued function that is differentiable at $x = (x_1, ..., x_d)$, then

$$\nabla_x f = (\frac{\partial f}{\partial x_1}, ..., \frac{\partial f}{\partial x_d})$$

- The gradient points in the direction of steepest ascent

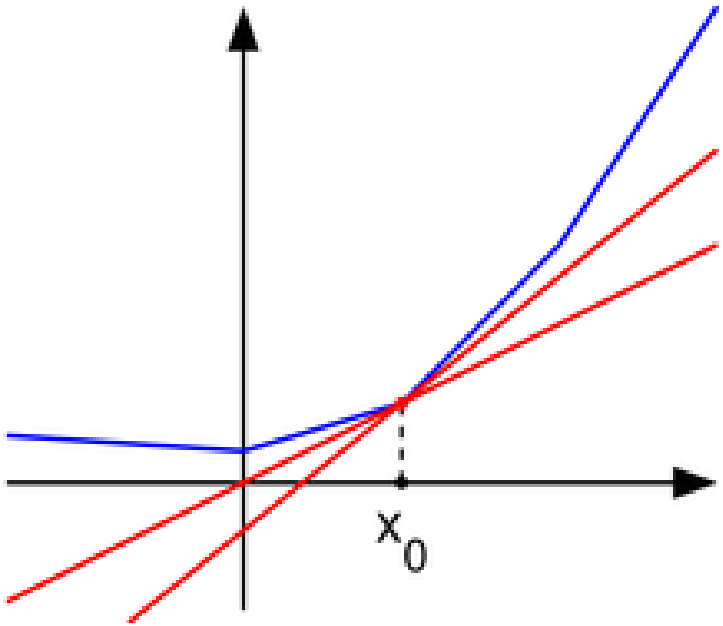


Subgradients

- If $f(x)$ is a convex real valued function, and if $g \in \mathbb{R}^d$ is such that

$$f(x) - f(y) \geq g^T(x - y)$$

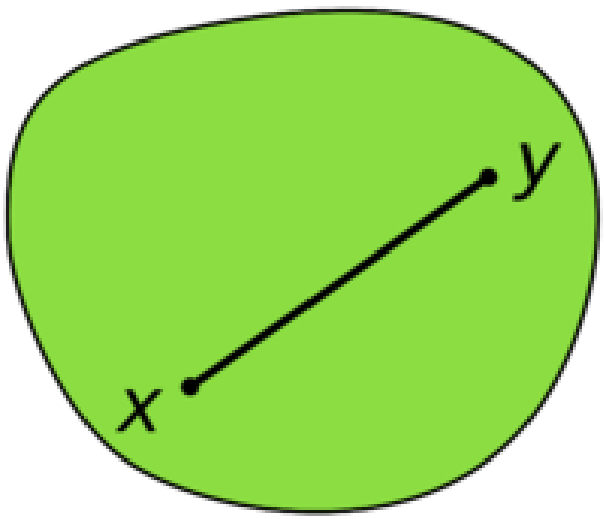
  for all y in some open ball around x, then g is in the subgradient set of f at x.
- If f is differentiable, then only $\nabla f(x)$ is in the subgradient set.
- A line with slope g can be drawn that only meet f at x

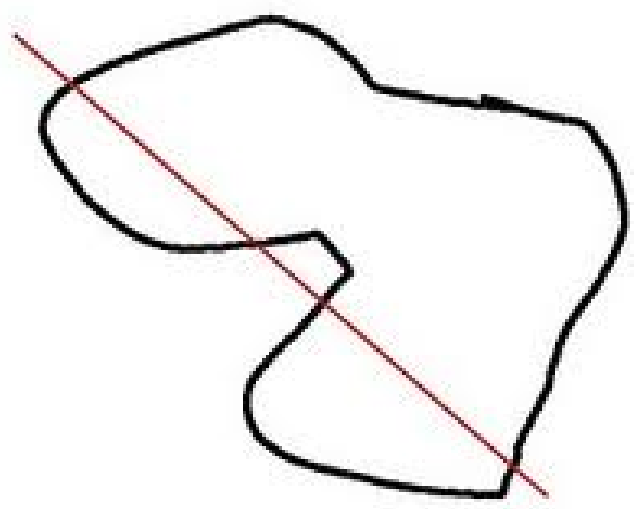


Convex set

- A set $X \subset \mathbb{R}^n$ is convex if for all $x, y \in X$ and $0 \leq \lambda \leq 1$,

$$(1 - \lambda)x + \lambda y \in X$$



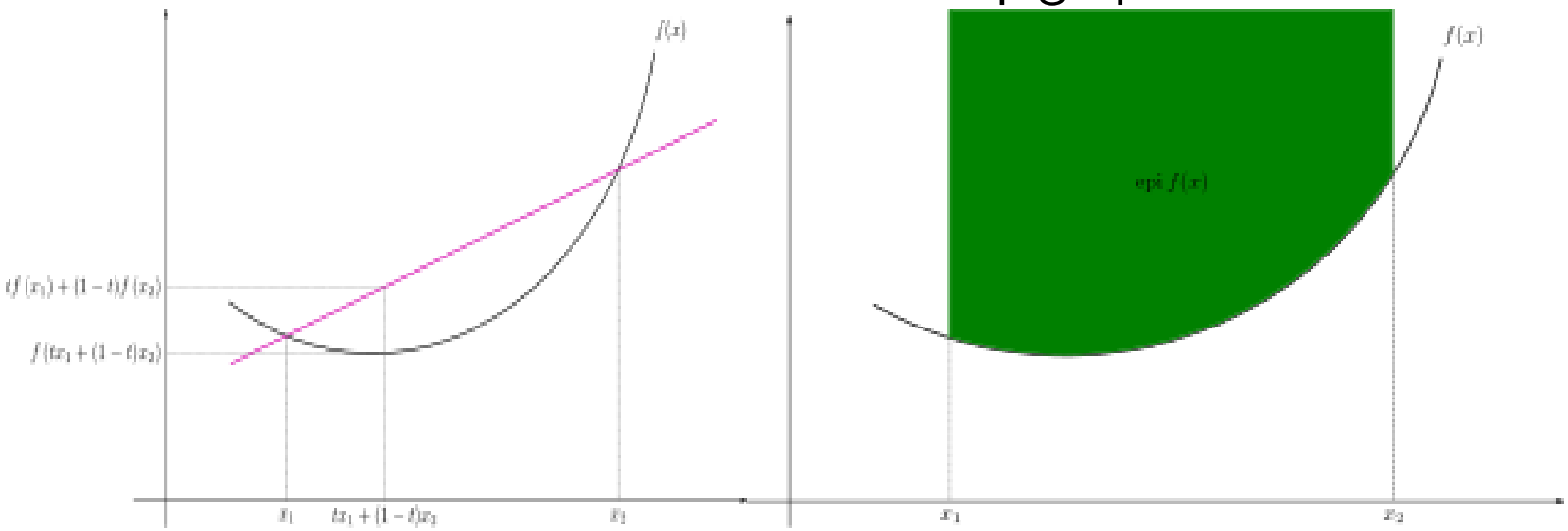


Convex function

- A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if $\forall \lambda \in [0, 1]$ and $x, y \in \mathbb{R}^n$,

$$f((1 - \lambda)x + \lambda y) \leq (1 - \lambda)f(x) + \lambda f(y).$$

- The set above the function's curve is called its epigraph
- A function is convex iff its epigraph is a convex set
- A convex function is called closed if the epigraph is closed



Strong Convexity

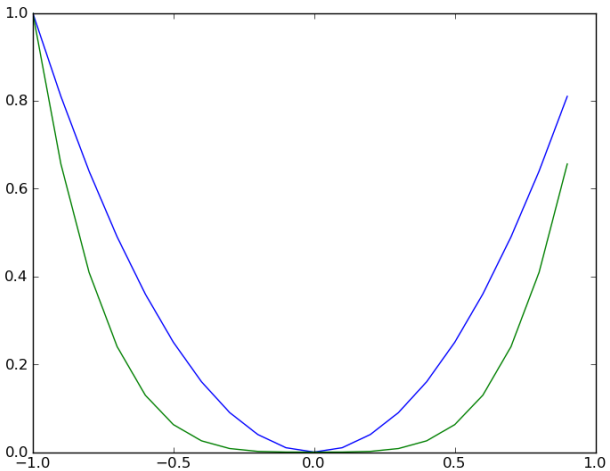
- A function $f : \mathbb{R}^n \to \mathbb{R}$ is ρ-strongly convex ($\rho > 0$) if $\forall \lambda \in [0, 1]$ and $x, y \in \mathbb{R}^n$,

$$f((1-\lambda)x + \lambda y) \leq (1-\lambda)f(x) + \lambda f(y) - \rho \frac{\lambda(1-\lambda)}{2} \|x - y\|_2^2$$

- or equivalently if $f(x) - (\rho/2)\|x\|_2^2$ is convex
- or equivalently

$$f(y) \geq f(x) + < \nabla f(x), y - x > + (\rho/2)\|x - y\|_2^2$$

- or equivalently (if twice diff) $\nabla^2 f(x) \geq \rho I$ (id matrix)
- Green ($y = x^4$) is too flat. Blue ($y = x^2$) is not too flat.

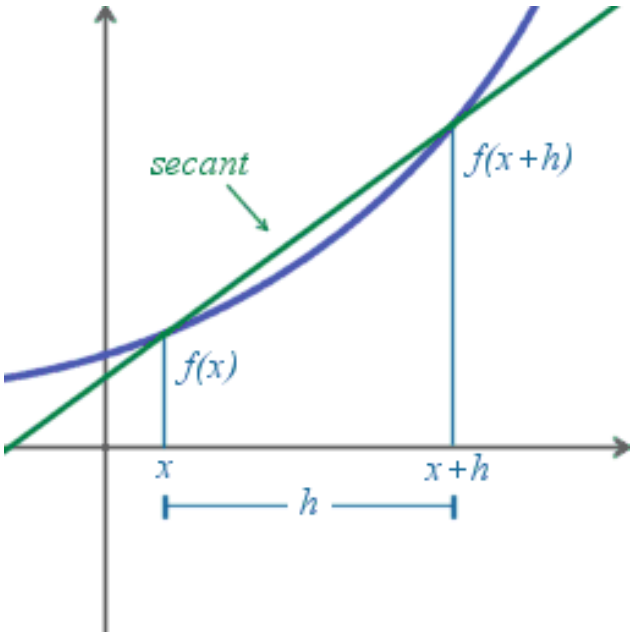


Lipschitz Continuity

- f is L-Lipschitz if

$$|f(x) - f(y)| \leq L\|x - y\|_2$$

- Lipschitz implies continuity but not differentiability
- If differentiable, then bounded (norm) gradients implies Lipschitz continuity.
- The secant slopes are bounded by L



Lipschitz Continuous Gradients

- If f is differentiable, then having L-Lipschitz gradients (L-LipG) means that

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2$$

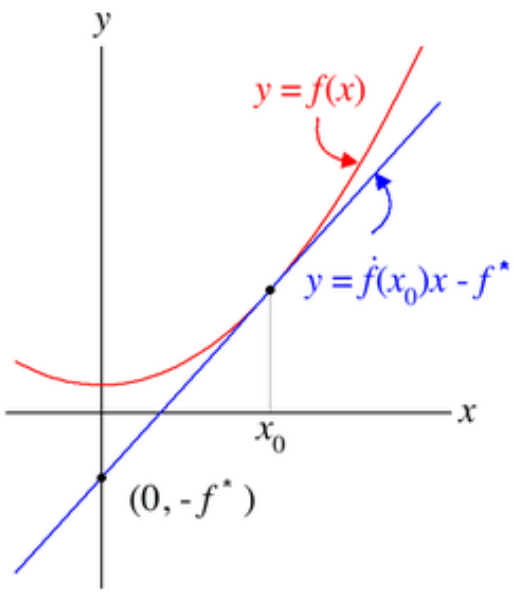
which is equivalent to

$$f(y) \leq f(x) + < \nabla f(x), y - x > + \frac{L}{2}\|x - y\|_2^2$$

which if f is twice differentiable is equivalent to $\nabla^2 f(x) \leq L \cdot I$.

Legendre transform

- Transformation between points and lines (set of tangents)
- One dimensional functions (generalizes to Fenchel transform)
- $f^*(p) = \max_x(px - f(x))$.
- If f differentiable, the maximizing x has the property that $f'(x) = p$.



Fenchel duality

- The fenchel dual of a function f is

$$f^*(s) = \sup_x < s, x > -f(x)$$

- If f is a closed convex function, then $f^{**} = f$.
- $f^*(0) = \max_x(-f(x)) = -\min_x f(x)$
-

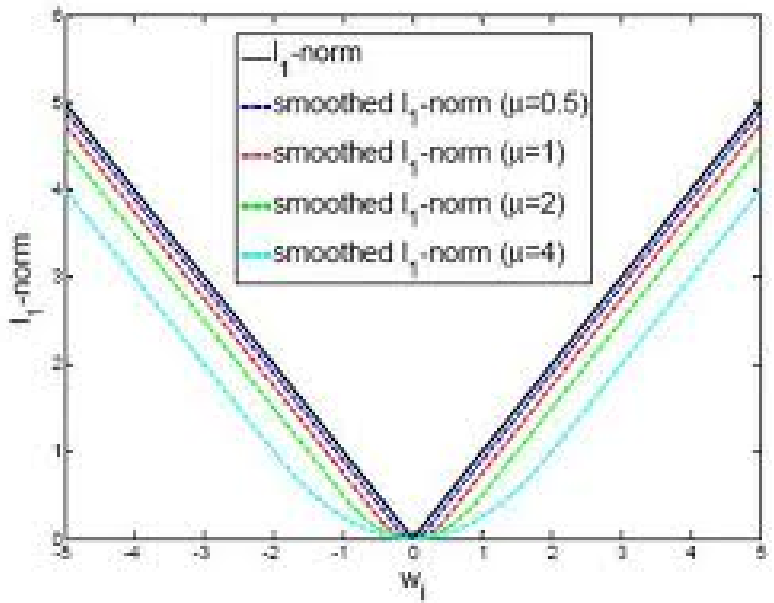
$$\nabla f^*(s) = \operatorname{argmax}_x < s, x > -f(x).$$

$$\nabla f(x) = \operatorname{argmax}_s < s, x > -f^*(s).$$

- $\nabla f^*(\nabla f(x)) = x$ and $\nabla f(\nabla f^*(s)) = s$

Duality between strong convexity and Lipschitz gradients

- f is ρ strongly convex iff f^* has $\frac{1}{\rho}$-Lip. gradients
- f^* has L-Lip. gradients iff f^* is $\frac{1}{L}$ strongly convex
- If a non-differentiable funtion f can be identified as g^* we can "smooth it out" by considering $(g + \delta\|\cdot\|_2^2)^*$ which is differentiable and has $\frac{1}{\delta}$-Lip gradients.
- $\delta = \frac{\mu}{2}$ below

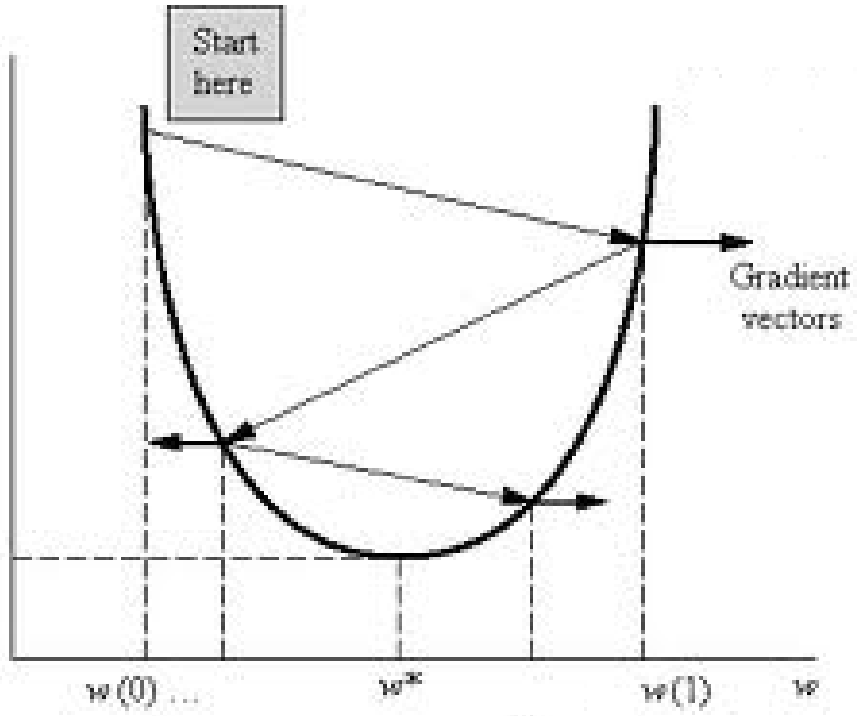


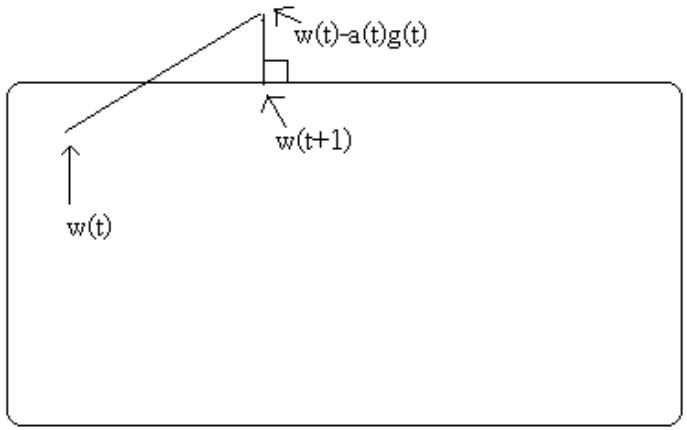
Section

(Sub)Gradient Methods

- Gradient descent: $w_{t+1} = w_t - a_t g_t$, g_t (sub)gradient of objective f at w_t.
- a_t can be decided by line search or by a predetermined scheme
- If we are constrained to a convex set X, then use

$$w_{t+1} = \Pi_X(w_t - a_t g_t) = \operatorname*{argmin}_{w \in X} \|w - (w_t - a_t g_t)\|_2$$





First and Second order information

- First order information: $\nabla f(w_t), t = 0, 1, 2, ..., n$
- Second order information is about $\nabla^2 f$ and is expensive to compute and store for high dimensional problems.
- Newton's method is a second order method where $w_{t+1} = w_t - (H_t)^{-1} g_t$.
- Quasi-Newton methods use first order information to try to approximate Newton's method.
- Conjugate gradient also use first order information

Lower Bounds

- There are lower bounds for how well a first order method, i.e. a method for which $w_{k+1} - w_k \in \text{Span}(\nabla f(x_0), ..., \nabla f(x_k))$ can do. They depend on the function class!
- Given w_0 (and L, ρ), the bound is saying that for any $\epsilon > 0$ there exists f in the class such that it takes at least n_ϵ steps for any first order method and a statement about n_ϵ is made.
- Differentiable functions with Lipschitz continuous gradients: Lower bound is $O(\sqrt{\frac{1}{\epsilon}})$
- Lipschitz continuous gradients and strong convexity: Lower bound is $O(\log \frac{1}{\epsilon})$.

Gradient Descent with Lipschitz gradients

- If we have L-Lipschitz continuous gradients, then define gradient decent by

$$w_{k+1} = w_k - \frac{1}{L}\nabla f(w_k).$$

- Monotone: $f(w_{k+1}) \leq f(w_k)$.
- If f is also ρ-strongly convex ($\rho > 0$), then

$$(f(w_k) - f(w^*)) \leq (1 - \frac{\rho}{L})^k (f(w_0) - f(w^*)).$$

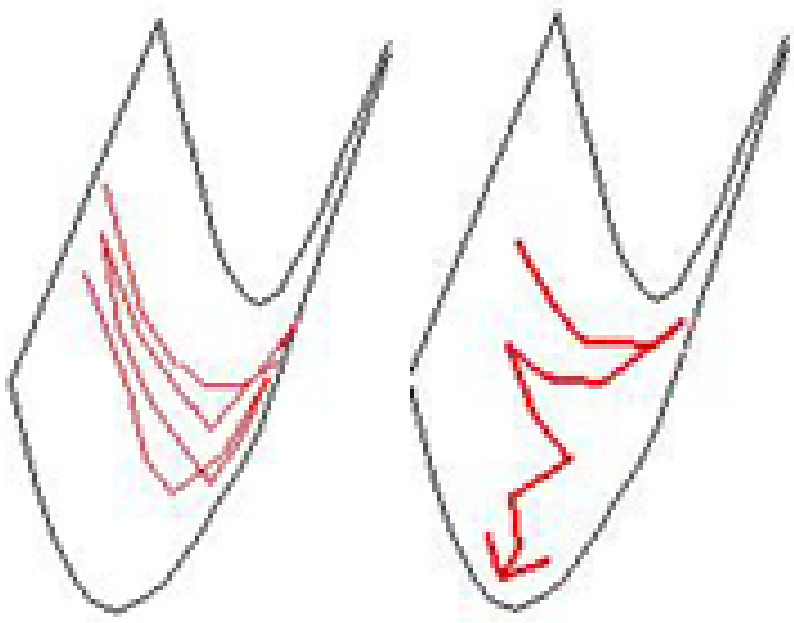
- This result translates into the optimal $O(\log \frac{1}{\epsilon})$ rate for first order methods

Gradient Descent without strong convexity

- Suppose that we do not have strong convexity ($\rho = 0$) but we have L-LipG.
- Then the gradient descent procedure $w_{k+1} = w_k - \frac{1}{L}\nabla f(w_k)$ has rate $O(\frac{1}{\epsilon})$. $(f(w_k) - f(w^*)) \leq \frac{L}{k}\|w^* - w_0\|^2$.
- The lower bound for this class is $O(\sqrt{\frac{1}{\epsilon}})$.
- There is a gap!
- The question arose if the bound was loose or if there is a better first order method

Nesterov's methods

- Nesterov answered the question in 1983 by constructing an "Optimal Gradient Method" with the rate $O(\sqrt{\frac{1}{\epsilon}})$ ($O(\sqrt{\frac{L}{\epsilon}})$ if we have not fixed L).
- Later variations (1988,2005) widened the applicability.
- Introduces a momentum term
- $w_{k+1} = y_k - \nabla f(y_k)$, $y_{k+1} = w_k + \alpha_k(w_k - w_{k-1})$
- $\alpha_k = \frac{\beta_k - 1}{\beta_{k+1}}$, $\beta_{k+1} = (1 + \sqrt{4\beta_k^2 + 1})/2$.
- Iteration cost similar to gradient descent



Non-differentiable objectives: Subgradient descent

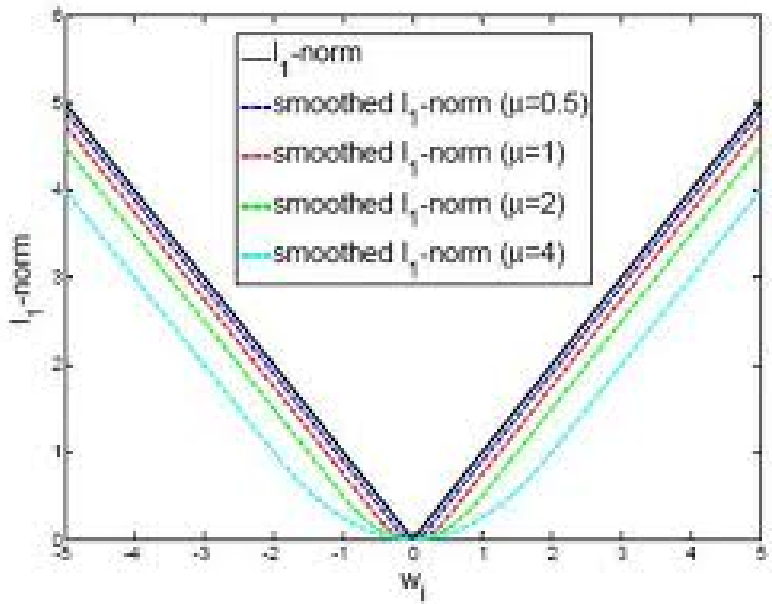
- Consider continuous but not necessarily differentiable objectives
- Given a subgradient g_k at w_k we can perform an update

$$w_{k+1} = w_k - \eta_k g_k.$$

- In the typical machine learning situation we only have non-differentiability on a set of Lebesgue measure 0 so g_k will actually often be a gradient
- However, the gradients will "jump" at the non-differentiability so we cannot have Lipschitz continuity of a (sub)gradient field in this situation.
- Subgradient descent has the bad rate $O(\frac{1}{\epsilon^2})$.

Nesterov's smoothing trick

- If we have a continuous non-differentiable function f that can be identified with g^*, then we can use the following trick.
- Given a desired accuracy $\epsilon > 0$, apply Nestov's method to $h = (g + \delta\|x\|_2^2)^*$ where $\delta = \frac{\epsilon}{2}$, but go to precision $\frac{\epsilon}{2}$.
- h has L-Lipschitz gradients where $L = \frac{1}{\delta}$
- $f(w) - h(w) \leq \delta = \frac{\epsilon}{2}$.
- Combined error at most ϵ.
- Since h has Lipschitz gradients it takes $O(\sqrt{\frac{L}{\epsilon}}) = O(\frac{1}{\epsilon})$.



Summary of offline gradient methods

- Normal gradient decent has optimal rate for strongly convex objectives with Lipschitz gradients. $O(\log \frac{1}{\epsilon})$
- Normal gradient decent is not optimal (its $O(\frac{1}{\epsilon})$) for convex but not strongly convex objectives. Nesterov's gradient method is optimal ($O(\sqrt{\frac{1}{\epsilon}})$).
- For non-differentiable objectives, subgradient decent is in general $O(\frac{1}{\epsilon^2})$.
- By utilizing special properties this can be improved. If the objective can be identified with the dual of another function, we can use a smoothing trick and get $O(\frac{1}{\epsilon})$.

Section

Online Optimization

- I will consider optimization problems of the form $C(w) = E_z L(z, w)$.
- If the underlying distribution for z is unknown, then we do not know C either.
- We observe a sequence $z_1, z_2, \ldots$ and we use that information to update parameters w_t.

Performance Measures

- There are different ways of measuring performance.
- $C(w_t)$, $\|w - w^*\|$ (if there exists a unique minimum w^*)
- $\sum_{i=1}^{t} L(z_i, w_i)$, $\sum_{i=1}^{t} L(z_i, w_i) - \inf_w \sum_{i=1}^{t} L(z_i, w)$
- Interesting function classes (C and L)
- E.g. $L(z, w) = \|w - z\|_2^2$.

Stochastic Gradient Decent

- When an empirical average $C_n(w) = \frac{1}{n}\sum_{i=1}^{n} L(z_i, w)$ is minimized, "Batch optimizers" have to calculate the entire sum for each evalutation of $C_n(w)$ and $\nabla_w C_n(w)$. (Gradient with respect to a chosen metric, e.g. standard Euclidean w.r.t. a given coordinate system)
- Stochastic gradient methods work with gradient estimates obtained from subsets of the training data.
- On large redundant data sets simple stochastic gradient descent (SGD) typically outperforms second order "batch" methods by orders of magnitude.
- $w_{t+1} = w_t - a_t Y_t$ where $E(Y_t) = \nabla_w C(w)$ and $a_t > 0$ defines SGD.
- Many different attempts at using second order information
- $w_{t+1} = w_t - a_t B_t Y_t$ where B_t is a positive scaling matrix.

Some Historical landmarks

- Robbins and Monro 1951 introduced one-dimensional SGD and proved a convergence theorem
- Blum 1954 generalizes this to the multivariate case in but with restrictive assumptions
- Robbins and Siegmund 1971 achieved a more general result for the multivariate case using super-martingale theory
- Lai and Robbins 1979 investigates one-dimensional "adaptive procedures" (step-sizes not predetermined) by introducing estimated curvature
- Wei 1987 does the multivariate case (proves asymptotic efficiency)
- More on second order methods Amari 1998, Murata 1998, Bottou and Lecun 2005
- Second order information only change the convergence by a constant factor, the rate remains the same.
- E. Hazan et al. 2006 shows logaritmic (instead of square root) regret for general online convex optimization with Newton step

Conditions for SGD convergence

- Cost function $C(w) = E_z L(z, w)$ is three times continuously differentiable and bounded from below (Can be replaced by other assumptions, e.g. bound on subgradient sets)
- $\sum_t a_t^2 < \infty$, $\sum_t a_t = \infty$
- $E_z(L(z, w)^2) \leq A + B\|w\|_2^2$
- $\inf_{\|w\|>D} w \cdot \nabla_w C(w) > 0$
- $\sum_{\|w\|<E} \|L(z, w)\| \leq K$
- Conclusion: $\|w_t - w^*\|_2^2 = O(1/t)$ a.s.
- Addition: Updates with scaling matrices in the updates converges a.s. if the eigenvalues are uniformly bounded from below by a strictly positive constant and from above by a finite constant (Sunehag et. al. 2009).

Step sizes

- The Robbins-Monro condition $\sum_t a_t^2 < \infty$, $\sum_t a_t = \infty$ is e.g. satisified by $a_t = 1/t$
- $\frac{b}{t+b}$ (or $\frac{1}{t+b}$) also works. Halfed after b steps. Initial search phase with rather flat schedule
- Step sizes that are not satisfying the Robbins-Monro conditions are also of interest
- Constant step sizes get us to a ball around the optimum whose radius depends on this constant. This is also good for tracking a changing environment
- Intermediary alternatives like $\frac{1}{\sqrt{t}}$.
- Various adaptive (data dependent) schemes have been tried, e.g. Stochastic Meta Decent by Schraudolph. Unclear verdict. Shows improvement in special cases. Causes extra computation.

Regret bounds

- Finite time analysis
- The regret is $\sum_{i=1}^{t} L(z_i, w_i) - \inf_w \sum_{i=1}^{t} L(z_i, w)$
- Under a strong convexity assumption one can prove that SGD with the right step sizes $\frac{1}{\rho t}$ has logarithmic regret which is the best possible. One has to know ρ. Depends on knowing ρ.
- Without strong convexity its $O(\sqrt{t}))$.
- Using Newton-steps, one can still have logarithmic regret.
- See later online learning lecture

Second order methods

- $w_t - w_{t-1} \approx H_{w_t}^{-1}(\nabla C(w_t) - \nabla C(w_{t-1}))$ (used by Quasi-Newton methods)
- Estimating full Hessians (various methods) gives cost order d^2 per step instead of d. Prohibitive for high dimensional problems.
- Low rank approximations (online LBFGS Schraudolph et. al. 2007) can be cost order kd where k is the memory length.
- Convergence speed at most improve my a constant factor. The cost increase by a constant factor.

Second order on the cheap

- Online LBFGS with memory 10 multiplies the cost per step with at least 11. Only worth it for really ill-conditioned problems
- Becker and Lecun estimated Diagonal Matrix for Neural Nets 1989.
- SGD-QN by Bordes, Bordes, Bottou, Gallinari wins wild track of the PASCAL Large Scale Learning Challenge 2008 by estimating Diagonal Scaling matrix (for SVMs) with LBFGS-inspired method

Support Vector Machines with SGD

- SVMs (hinge loss),
  $\min_w \frac{1}{m} \sum_{i=1}^{m} \min(0, 1 - y_i(w^T x_i)) + \lambda \|w\|_2^2$
- Instantaneous objective $\min(0, 1 - y_i(w_t^T x_i)) + \lambda \|w_t\|_2^2$ if we have drawn sample i for the current time t
- (Sub)Gradient: If correctly classified then $g_t = 2\lambda w_t$. If wrong, then $g_t = 2\lambda w_t - y_i x_i$.
- λ-strong convexity suggests step size $\frac{1}{\lambda t}$ or $\frac{1}{\lambda(t+t_0)}$

Support Vector Machines with Pegasos

- Note: Plugging in $w = 0$ makes the expression equal to 1. If we have w with $\|w\| \geq \frac{1}{\sqrt{\lambda}}$, then the expression is at least 1.
- Conclusion, we only have to consider $\|w\|_2 \leq \frac{1}{\sqrt{\lambda}}$.
- This is utilized by the Pegasos algorithm which alternates the gradient step above with a projection (scaling) onto this ball.
- Proven $O(\frac{1}{\epsilon})$.

Summary

- Simple methods with cheap updates can be the best idea for parameter estimation in machine learning
- This is particularly true in large scale learning
- With SGD, the time per iteration does not depend on data set size. Having more data just means that we do not rerun on old data.
- Faster progress is made when using "fresh data".
- More data, therefore, means LESS runtime (for SGD) until we reach the desired true accuracy.

Literature

- Numerical Optimization, Jorge Nocedal and Stephen J. Wright , Springer, August 2000
- Introductory Lectures on Convex Optimization: A Basic Course, Yurii Nesterov, Springer 2003
- Convex Analysis, R. Tyrrell Rockafellar, Princeton University Press 1996
- Convex Optimization, Stephen Boyd and Lieven Vandenberghe, Cambridge University Press 2004
- L. Bottou's SGD page: http://leon.bottou.org/research/stochastic