SVAN 2016 Mini Course: Stochastic Convex Optimization Methods in Machine Learning

Mark Schmidt

University of British Columbia, May 2016

www.cs.ubc.ca/~schmidtm/SVAN16

Some images from this lecture are taken from Google Image Search.

Last Time: Training vs. Testing

- In supervised learning we are given a training set X and y.
  - But what we care about is test error: are prediction accurate on new data?
- In order to say anything about new data, need assumptions:
  - IID assumption: training and test data drawn from same distribution.
- Often, we have an explicit test set to approximate test error.

Data: X, y, X_{test}, y_{test}

1. Train: $model = fit(X, y)$

2. Predict test set labels $\hat{y} = predict(model, X_{test})$

3. Evaluate $error = diff(\hat{y}, y_{test})$

- Golden rule: this test set cannot influence training in any way.
  - Otherwise, not valid approximation of test error.

Fundamental Trade-Off and Regularization

- Bias-variance and other learning theory results to trade-off:
  1. How small you can make the training error.

     vs.

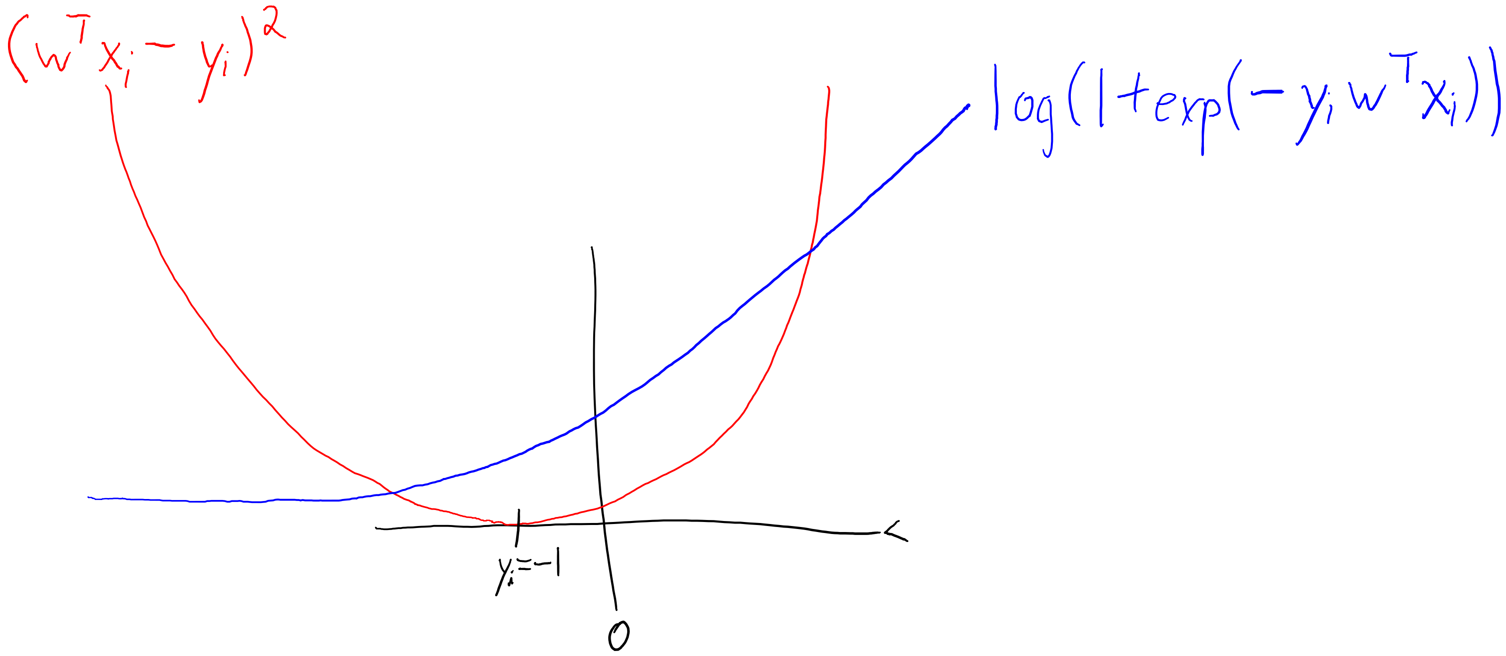
  2. How well training error approximates the test error.
- Simple models: high training error but don't overfit:
- Complex models: low training error but overfit.
- Regularization: reduces overfitting in complex models.
  - Common approach is L2-regularization:

$$\underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \frac{1}{2}\|Xw - y\|^2 + \frac{\lambda}{2}\|w\|^2$$

  - Increases training error, but typically decreases test error.
  - Increasing number of training examples 'n' has a similar effect on trade-off.

Last Time: Logistic Regression

- We considered binary labels y_i, and classifying with $\text{sign}(w^T x_i)$.
  - Squared error $(w^T x_i - y_i)^2$ is not ideal: penalizes model for "too right".
  - Minimizing number of errors is also not ideal: NP-hard.
  - Tractable upper bounds are hinge loss and logistic loss.



  - We also discussed defining losses with multiple classes (softmax loss).

Course Roadmap

- Part 1: Overview of Machine Learning
- Part 2: Large-scale machine learning.
  - How do we fit these models to huge datasets?
  - Why are SVMs/logistic easy while minimizing number of errors is hard?

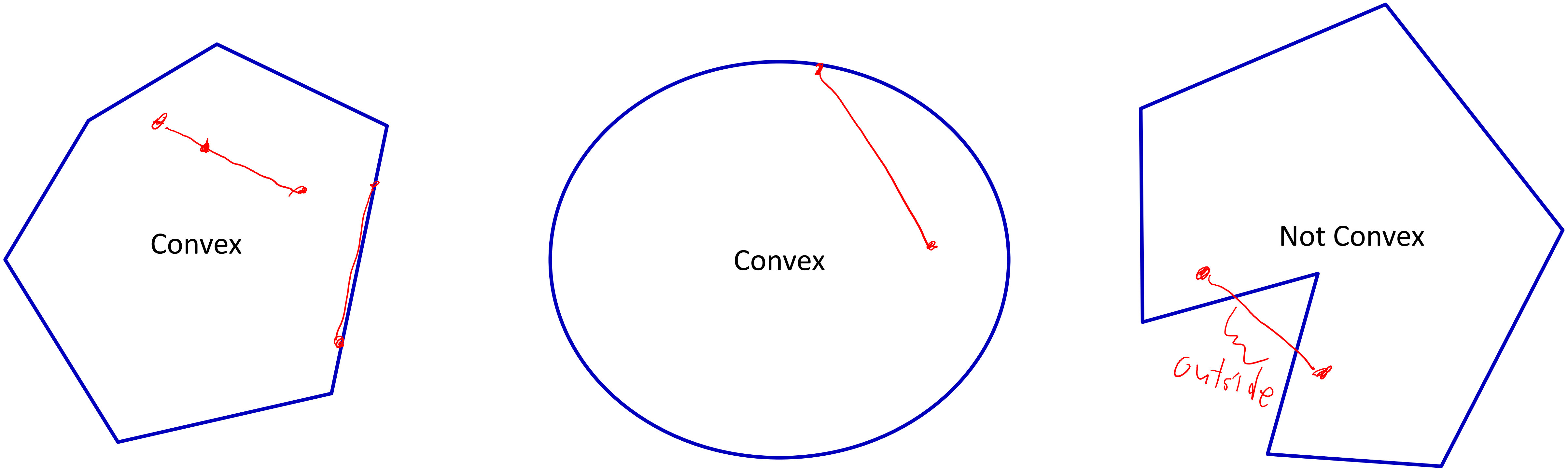
Convex Functions

- We are first going to discuss convex functions:
  - Minimizing convex functions is usually easy.
  - Minimizing non-convex functions is usually hard.
- The 'easy' problems we have discussed are convex:
  - Least squares, robust regression, logistic regression, support vector machines, multi-class logistic, brittle regression, Poisson regression.
  - All of the above with L2-regularization.
- The 'hard' problems we have discussed are non-convex:
  - 0-1 loss, "very robust" regression.

Convex Sets

- First we need to define a convex set:
  - A set is convex if the line between any two points stays in the set.

For all $x \in C$ and $y \in C$ we have $\theta x + (1-\theta) y \in C$ for $0 \leq \theta \leq 1$



Convex Sets

- Examples:

Real-space: $\mathbb{R}^d$

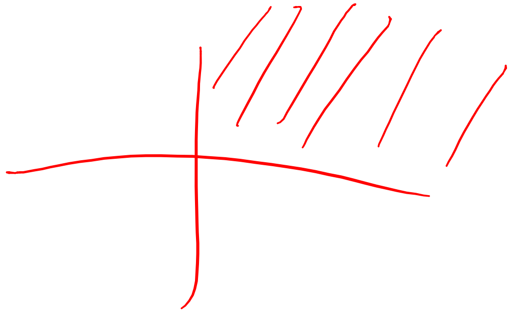
Positive orthant $\mathbb{R}^d_+$: $\{x \mid x \geq 0\}$

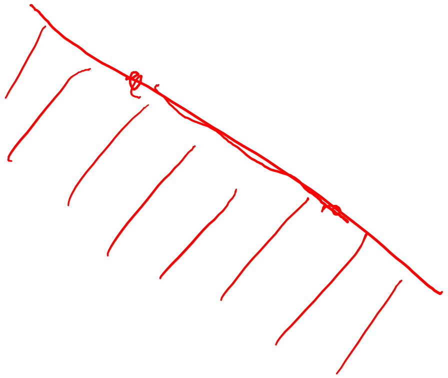
Hyper-plane: $\{x \mid a^T x = b\}$

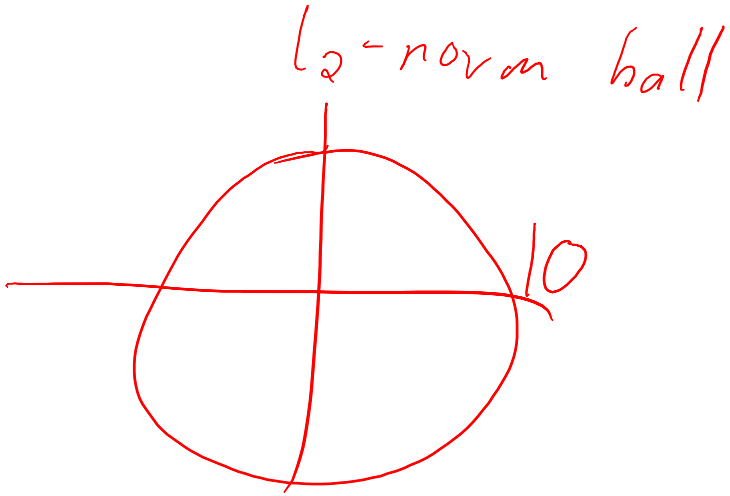
Half-space: $\{x \mid a^T x \leq b\}$

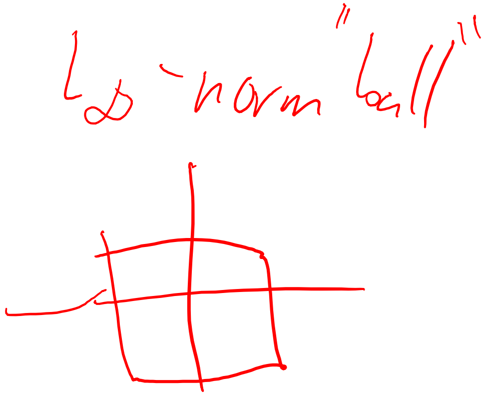
Norm-ball: $\{x \mid \|x\| \leq r\}$

Norm-cone: $\{(x, r) \mid \|x\| \leq r\}$









Showing a Set is Convex

How to prove a set is convex?

- One way: choose two generic x and y in the set, show that generic z between them is also in the set.
- Another way: show that set is intersection of sets that you know are convex.

E.g. if $C = \{x \mid a^T x = b\}$

then for $x \in C$ and $y \in C$ and $0 \le \theta \le 1$

we have
$$a^T(\theta x + (1-\theta)y) = \theta(a^T x) + (1-\theta)(a^T y) = \theta b + (1-\theta) b = b$$

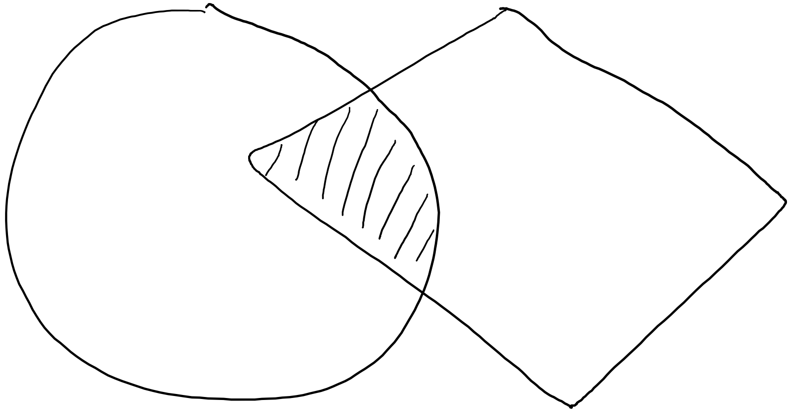
E.g., if $C = \{x \mid \|x\| \le 10\}$

then for $x \in C$ and $y \in C$ and $0 \le \theta \le 1$

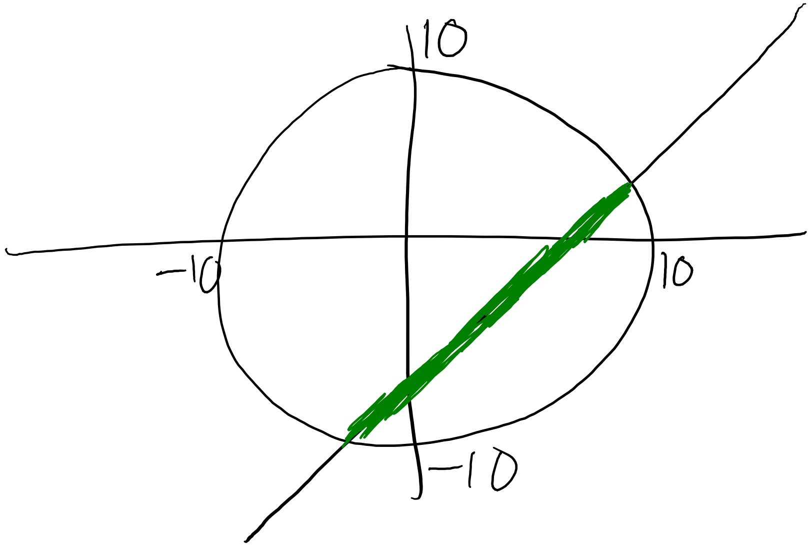
we have
$$\begin{aligned}
\|\theta x + (1-\theta) y\| &\le \|\theta x\| + \|(1-\theta) y\| && \text{(triangle inequality)} \\
&= |\theta| \cdot \|x\| + |1-\theta| \cdot \|y\| && \text{(homogeneity)} \\
&= \theta \|x\| + (1-\theta)\|y\| && \theta \ge 0 \\
&\le \theta \max\{\|x\|, \|y\|\} + (1-\theta)\max\{\|x\|, \|y\|\} && a \le \max\{a, b\} \\
&= \max\{\|x\|, \|y\|\} \le 10
\end{aligned}$$

Intersection of Convex Sets

- Intersection of convex sets is convex:



E.g., $\{x \mid a^T x = b\} \cap \{x \mid \|x\| \leq 10\}$ is a convex set.



For example, x satisfying linear program constraints are a convex set:

$$Ax \leq b$$

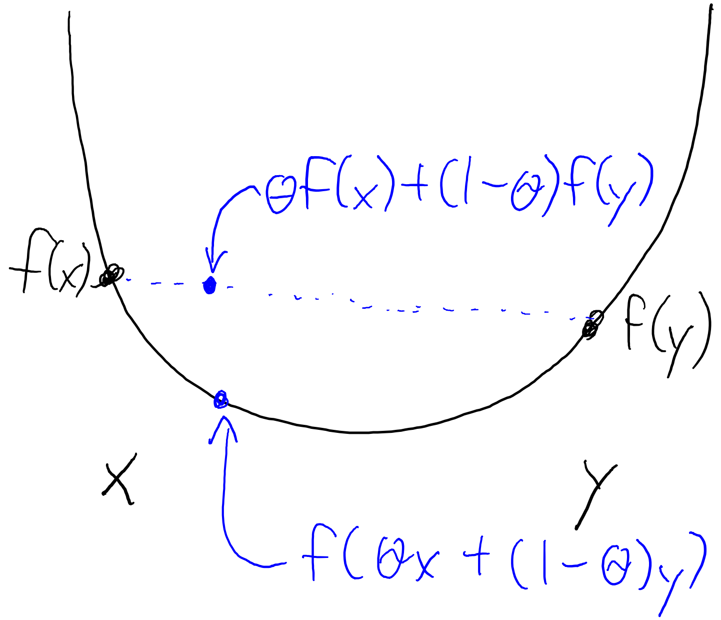
$$A_{eq} x = b_{eq}$$

$$LB \leq x \leq UB$$

Convex Functions

- A function ‘f’ is convex if:
  1. The domain of ‘f’ is a convex set.
  2. The function is always below ‘chord’ between two points.

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y) \quad \text{for all } x \in C, y \in C, \text{ and } 0 \leq \theta \leq 1$$



Implication: all local minima are global minima.

We can minimize a convex function by finding any stationary point.

Convex Functions

- Examples:

Quadratic functions: $f(x) = ax^2 + bx + c, \quad a > 0.$

Linear functions $f(x) = a^T x + b$

Exponential: $f(x) = \exp(ax)$

Negative logarithm: $f(x) = -\log(x)$

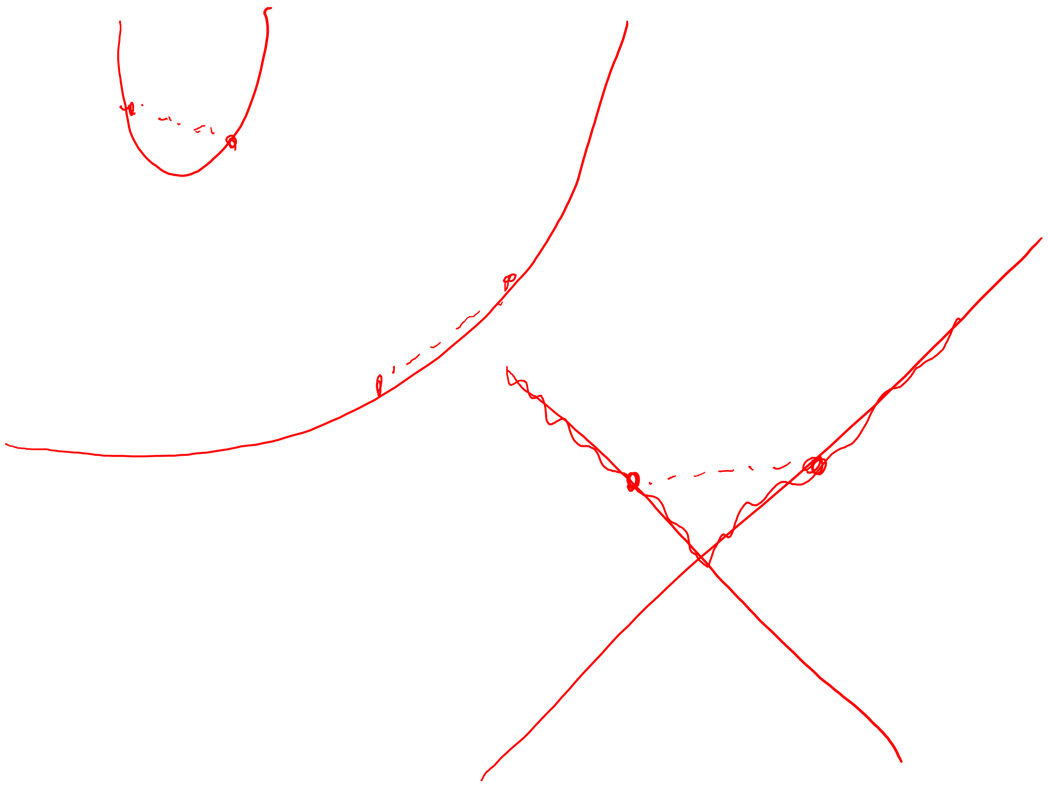
Absolute value: $f(x) = |x|$

Max function: $f(x) = \max_i \{x_i\}$

Negative entropy: $f(x) = x\log(x), \quad x > 0$

Logistic loss: $f(x) = \log(1 + \exp(-z))$

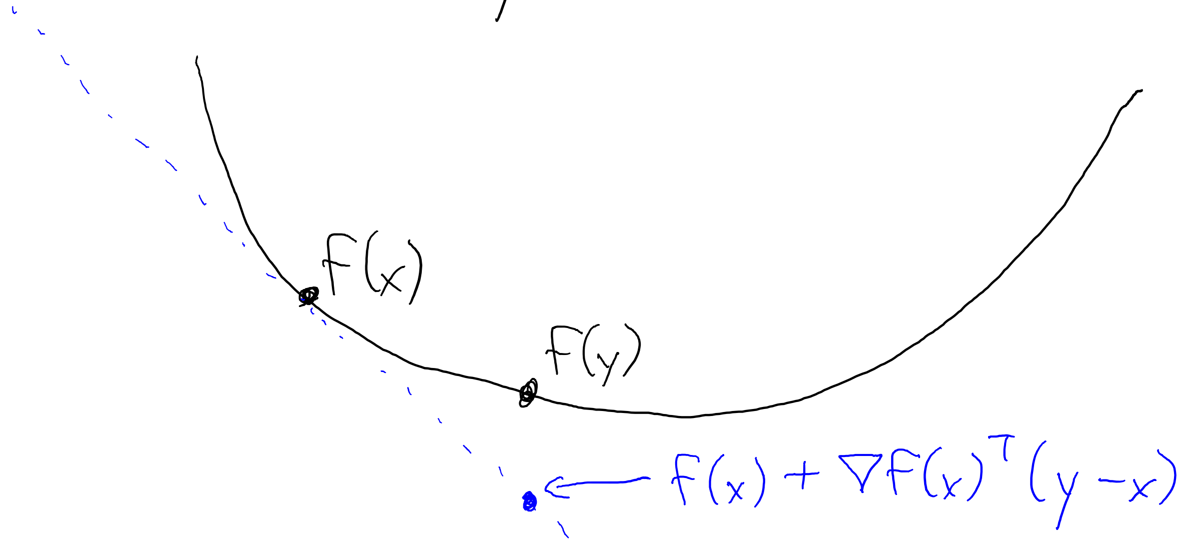
Log-sum-exp: $f(x) = \log\left(\sum_{i=1}^{d} \exp(x_i)\right)$



Differentiable Convex Functions

- A *differentiable* ‘f’ is convex iff ‘f’ is always above tangent:

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) \quad \text{for all } x \in C \text{ and } y \in C$$

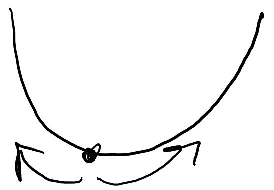


If $\nabla f(x) = 0$, this implies $f(y) \geq f(x)$ for <u>all</u> y so x is a global minimizer.

Twice-Differentiable Convex Functions

- A *twice-differentiable* 'f' is convex iff it's curved upwards everywhere.

For one-dimensional functions, reduces to $f''(x) \geq 0$.



– usually, this is the easiest way to show a function is convex.

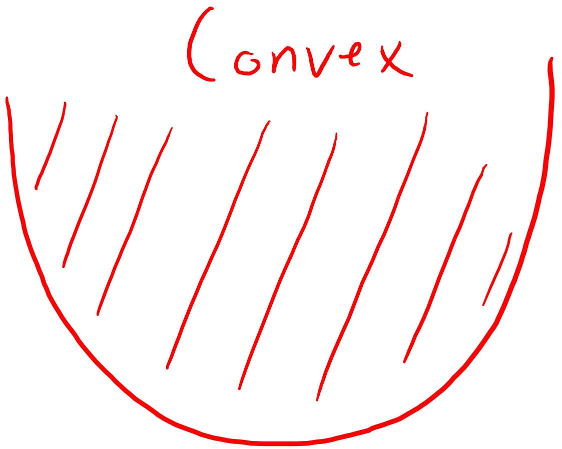
For multivariate functions, generalization is

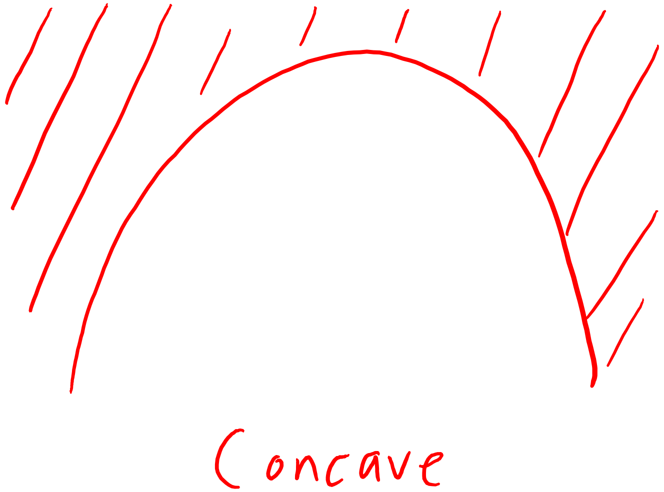
$$\nabla^2 f(x) \succeq 0 \quad \text{for all } x \in C.$$

$A \succeq 0$ means A is symmetric and positive semi-definite: $y^T A y \geq 0$ for all y

Concave Functions

- The negative of a convex function is a concave function:





Showing Functions are Convex

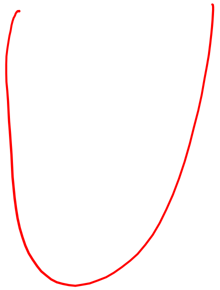
- Examples:

If $f(x) = x^2$

then $f'(x) = 2x$

and $f''(x) = 2$.

Since $2 \geq 0$ we've shown x^2 is convex.



If $f(x) = \frac{1}{2} x^T A x + b^T x + c$ with $A \succeq 0$

then $\nabla f(x) = Ax + b$

and $\nabla^2 f(x) = A$

Since $\nabla^2 f(x) \succeq 0$ we've shown $f(x)$ is convex.

Showing Functions are Convex

- Examples:

$$f(w) = \frac{1}{2}\|Xw - y\|^2$$
$$\nabla f(w) = X^T(Xw - y)$$
$$\nabla^2 f(w) = X^T X$$

Want to show that $\nabla^2 f(w) \succcurlyeq 0$, or equivalently $y^T \nabla^2 f(w) y \geq 0$.

We have $y^T \nabla^2 f(w) y = y^T X^T X y$

$$= (Xy)^T (Xy)$$
$$= \|Xy\|^2 \geq 0.$$

So least squares is convex and setting $\nabla f(w) = 0$ gives global minimum.

Strictly-Convex Functions

- A function is strictly-convex if these inequalities strictly hold:

$$f(\theta x + (1-\theta)y) < \theta f(x) + (1-\theta)f(y) \quad \text{for} \quad 0 < \theta < 1.$$

$$f(y) > f(x) + \nabla f(x)^T(y - x)$$

$$\nabla^2 f(x) \succ 0 \quad (y^T \nabla^2 f(x) y > 0 \text{ for all } y \neq 0)$$

- Strict convexity implies at most one global minimum:

Points 'x' and 'y' can't both be global minima if $x \neq y$, since this would imply $f(\theta x + (1-\theta)y)$ is below global min.

- This implies L2-regularized least squares has unique solution:

for $y \neq 0$

$$y^T \nabla^2 f(w) y = y^T (X^T X + \lambda I) y = y^T X^T X y + y^T (\lambda I) y = (Xy)^T (Xy) + \lambda y^T y = \|Xy\|^2 + \lambda \|y\|^2 > 0.$$

Operations that Preserve Convexity

- There are a few operations preserve convexity.
  - Often lets us avoid calculating Hessian.
  - Often lets us prove convexity of non-smooth functions.
- If f_1 and f_2 are convex, then convexity is preserved under:
  1. Weighted sums (non-negative coefficients):

     $f(x) = z_1 f_1(x) + z_2 f_2(x)$ is convex if $z_1 \geq 0$ and $z_2 \geq 0$
  2. Composition with affine function:

     $f(x) = f_1(Ax + b)$ is convex.
  3. Pointwise maximum:

     $f(x) = \max\{f_1(x), f_2(x)\}$ is convex.

Example: SVMs

$$f(x) = \underbrace{\sum_{i=1}^{n} \underbrace{\max\{\underbrace{0}_{\text{linear (convex)}}, \underbrace{1 - y_i w^T x_i}_{\text{linear (convex)}}\}}_{\text{max (convex) so convex}} + \frac{\lambda}{2}\|w\|^2}_{\text{non-negative sum of convex}}$$

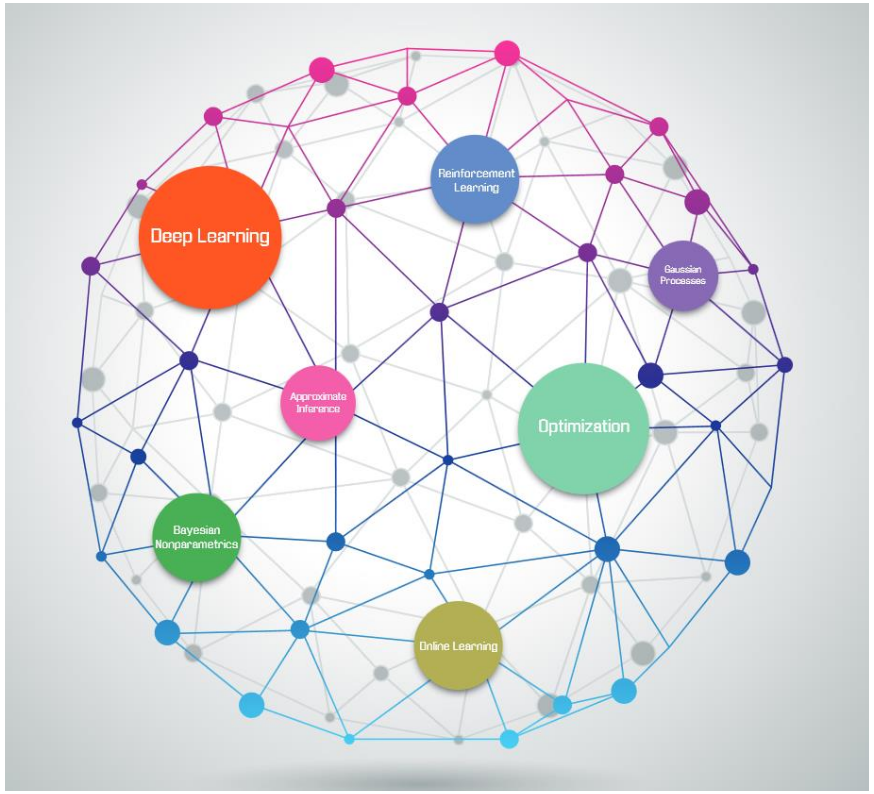
$\nabla^2\left[\frac{\lambda}{2}\|w\|^2\right] = \lambda I \succeq 0$

so convex

(pause)

Current Hot Topics in Machine Learning

- Graph of most common keywords among ICML papers last year:



- Why is there so much focus on deep learning and optimization?

Why Study Optimization in CPSC 540?

- In machine learning, training is typically written as optimization:
  - Numerically optimize parameters of model, given data.
- There are some exceptions:
  1. Counting- and distance-based methods (random forests, KNN).
     - See my undergraduate course
  2. Integration-based methods (Bayesian learning).
     - Covered after large-scale optimization in my grad course.

  Although you still need to tune parameters in those models.
- But why study optimization? Can't I just use Matlab functions?
  - '\', linprog, quadprog, fmincon, CVX,…

The Effect of Big Data and Big Models

- Datasets are getting huge, we might want to train on:
  - Entire medical image databases.
  - Every webpage on the internet.
  - Every product on Amazon.
  - Every rating on Netflix.
  - All flight data in history.
- With bigger datasets, we can build bigger models:
  - This is where deep learning comes in.
  - Complicated models can address complicated problems.
- Now optimization becomes a problem because of time/memory:
  - We can’ afford $O(d^2)$ memory, or an $O(d^2)$ operation.
  - Going through huge datasets 100s of times is too slow.
  - Evaluating huge models too many times is too slow.

Fitting Logistic Regression Models

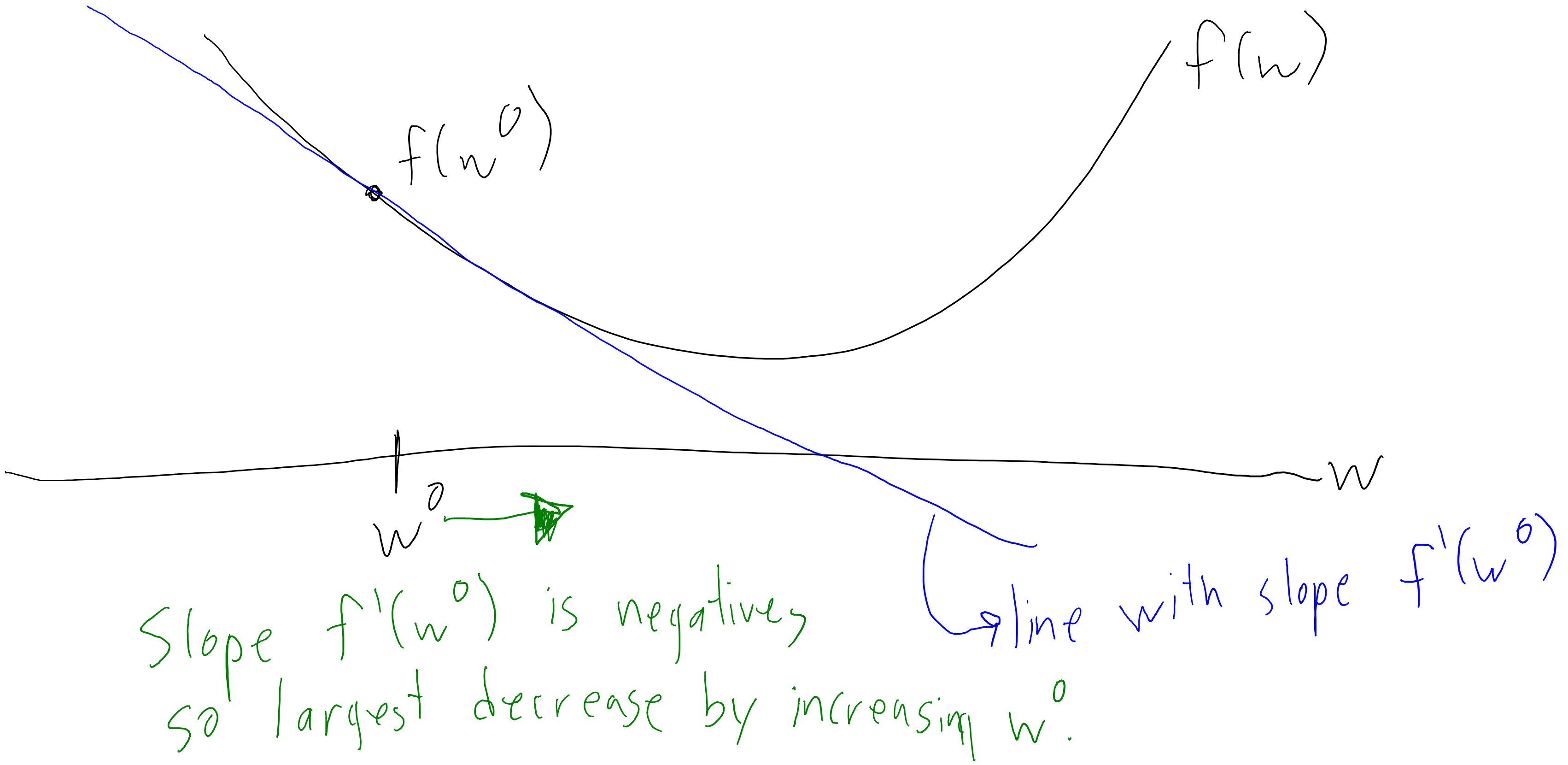
- Recall the L2-regularized logistic regression objective function:

$$\underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \sum_{i=1}^{d} \log(1 + \exp(-y_i w^T x_i)) + \frac{\lambda}{2}\|w\|^2$$

- This objective function is strictly-convex and differentiable.
- But we can't formulate as linear system or linear program.
- Nevertheless, we can efficiently solve this problem.
- There are many ways to do this, but we focus on gradient descent:
  – Iteration cost is linear in 'd' (not true of IRLS/Newton's method).
  – We can prove that we don't need too many iterations:
    - Number of iterations does not directly depend on 'd'.

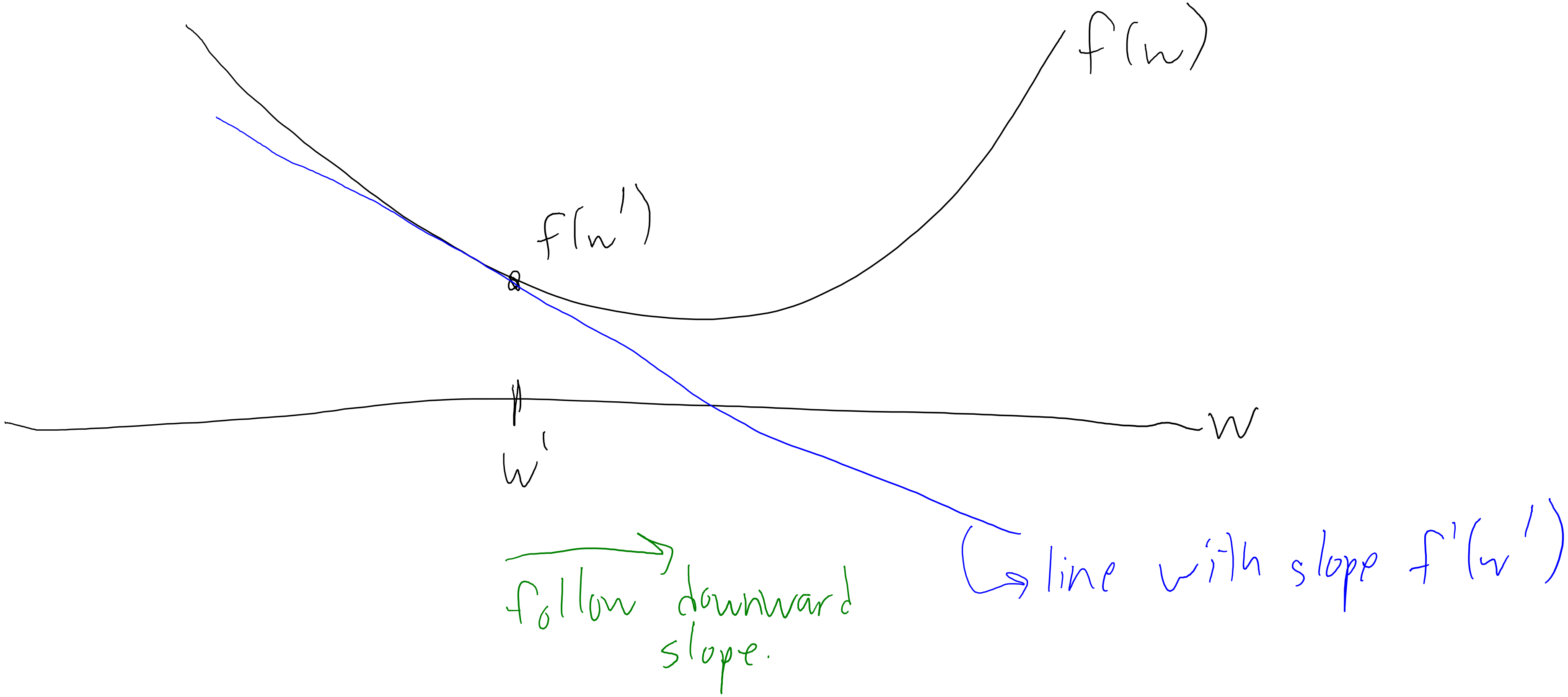
Gradient Descent

- Gradient descent is based on a simple observation:
  - Given parameters ‘w^0’, direction of largest decrease is $-\nabla f(w^0)$.



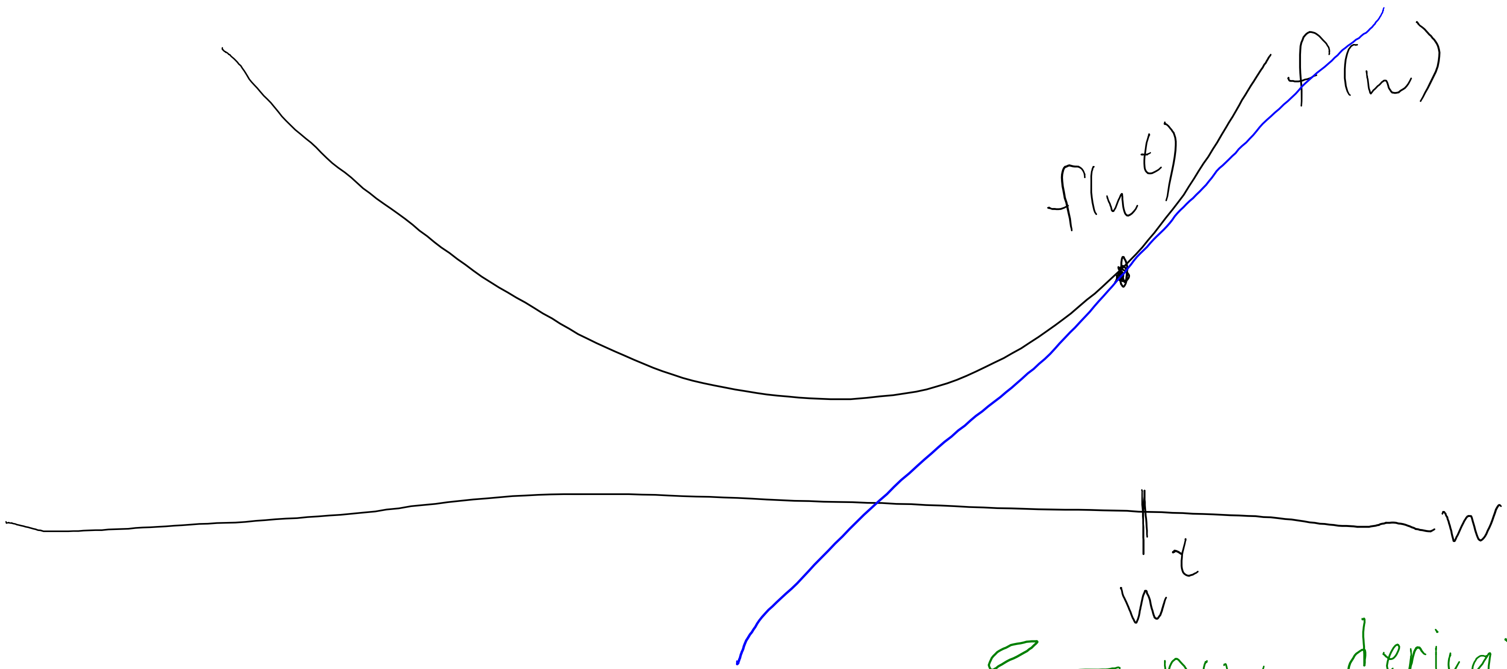
Gradient Descent

- Gradient descent is based on a simple observation:
  - Given parameters ‘w^0’, direction of largest decrease is $-\nabla f(w^0)$.



Gradient Descent

- Gradient descent is based on a simple observation:
  - Given parameters ‘w^0’, direction of largest decrease is $-\nabla f(w^0)$).



← now derivative is positive, so want to decrease w.

Gradient Descent

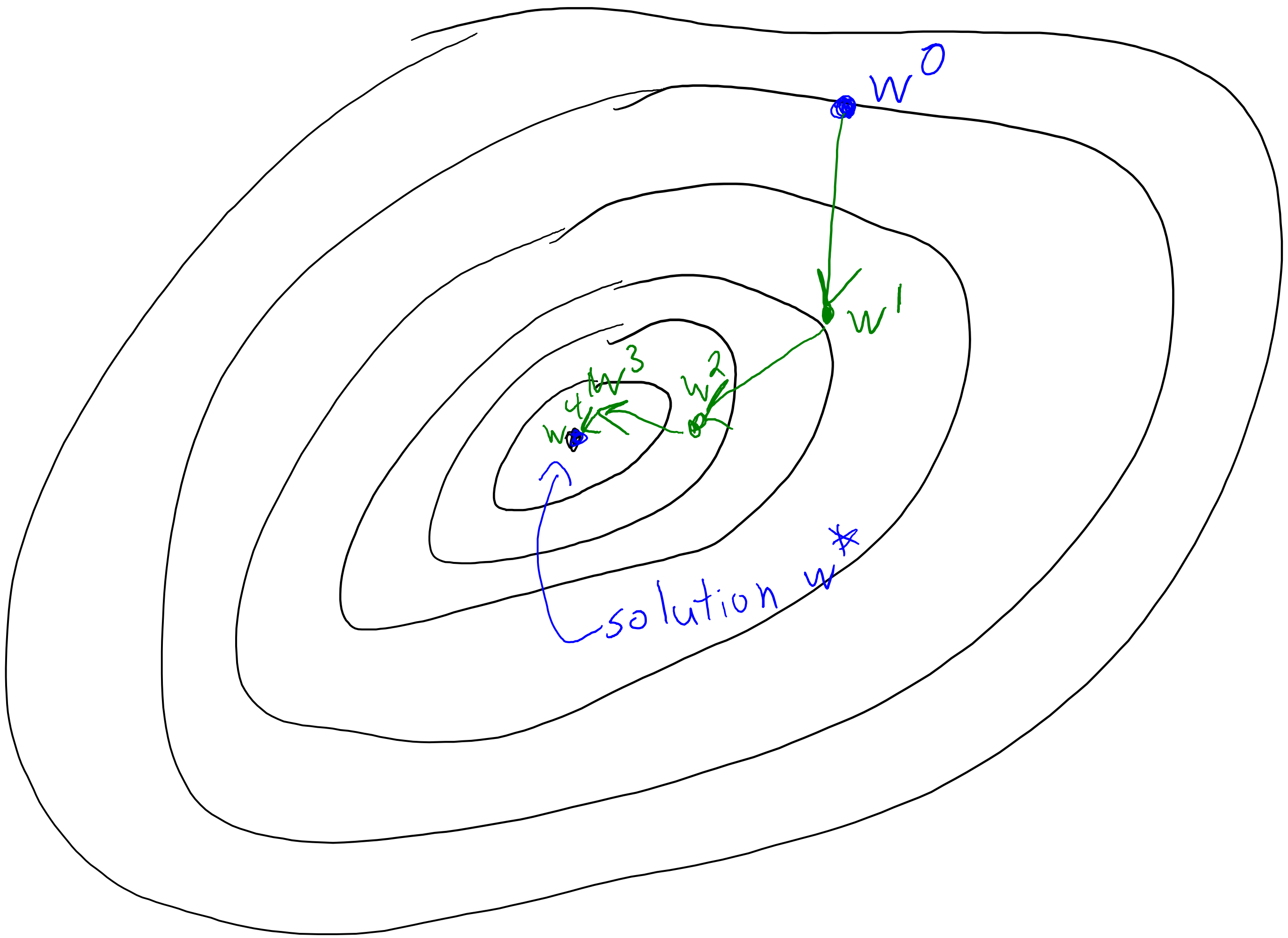
- Gradient descent is an iterative algorithm:
  - We start with some initial guess, w^0.
  - Generate new guess by moving in the negative gradient direction:

$$w^1 = w^0 - \alpha_0 \nabla f(w^0).$$

  (The scalar α_0 is the `step size'.)
  - Repeat to successively refine the guess: $w^{t+1} = w^t - \alpha_t \nabla f(w^t).$

    Generate $w^2, w^3, w^4, \ldots$
  - Stop if not making progress or $\|\nabla f(w^t)\| \leq \delta$ (some small number)

Gradient Descent in 2D

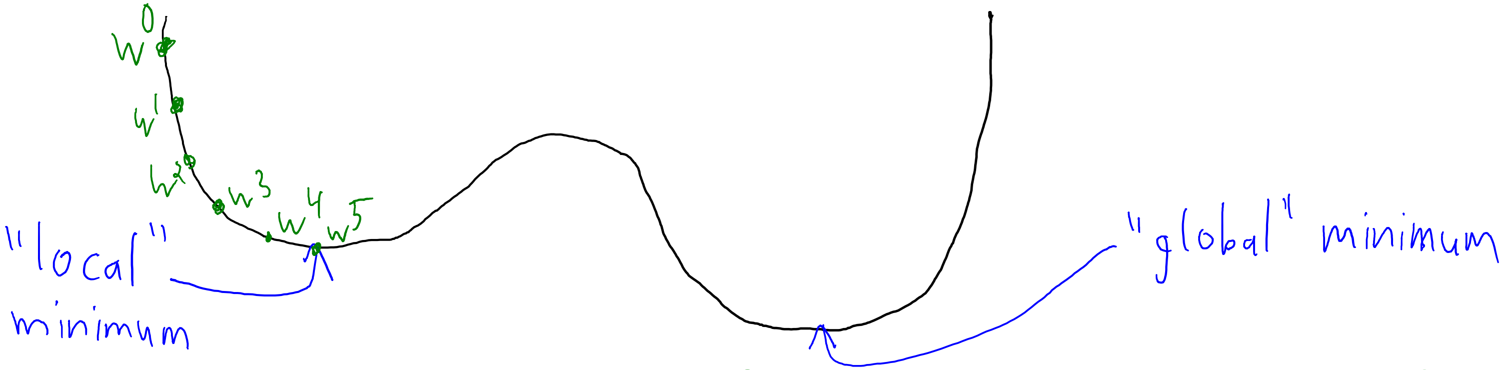


Gradient Descent

- If α_t is small enough and $\nabla f(w^t) \neq 0$, guaranteed to decrease 'f':

$$f(w^{t+1}) < f(w^t)$$

- Under weak conditions, procedure converges to a stationary point.



If 'f' is convex, converges to global minimum.

- Least squares via linear system vs. gradient descent:
  - Solving linear system cost $O(nd^2 + d^3)$.
  - Gradient descent costs $O(ndt)$ to run for 't' iterations.
    - Will be faster if t < d.

$O(nd)$

$X^T(Xw) + X^Ty$

Convergence Rate of Gradient Descent

- How many iterations do we need?
  - Let x^* be the optimal solution and ε be the accuracy we want.
  - What is the smallest number of iterations 't' such that:

$$f(x^t) - f(x^*) \leq \varepsilon$$

Notation:
- In optimization, we usually talk about optimizing x.

- To answer this question, need assumptions:
  - Let's assume $\underbrace{\mu I \preceq}_{\text{"strongly convex"}} \nabla^2 f(x) \underbrace{\preceq L I}_{\text{"strongly smooth"}}$ for all x and some $L < \infty$ and $\mu > 0$.

"strongly convex" $y^T \nabla^2 f(x) y \geq \mu \|y\|^2$

"strongly smooth" $y^T \nabla^2 f(x) y \leq L \|y\|^2$

Strongly-convex
$\Rightarrow$ strictly-convex
$\Rightarrow$ convex.

Example:
If $f(w) = \frac{1}{2}\|Xw - y\|^2 + \frac{\lambda}{2}\|w\|^2$, then $\mu = \min \operatorname{eig}(X^T X) + \lambda \geq \lambda$ and $L = \max \operatorname{eig}(X^T X) + \lambda$.

Bonus Slide: Constants for Least Squares

- Consider least squares: $f(x) = \frac{1}{2}\|Ax - b\|^2$

What are 'L' and 'μ' such that $\mu I \preceq \nabla^2 f(x) \preceq L I$?

Note that $\nabla^2 f(x) = A^T A$, and since it's symmetric we can use spectral decomposition:

$A^T A = \sum_{j=1}^{d} \lambda_j q_j q_j^T$ where $q_j^T q_j = 1$ and $q_i^T q_j = 0$ for $i \neq j$. (Assume $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$)

We can write any y as linear combination of orthogonal basis, $y = \alpha_1 q_1 + \alpha_2 q_2 + \cdots + \alpha_d q_d$.

So we have $y^T \nabla^2 f(x) y = y^T A^T A y = y^T \left(\sum_{j=1}^{d} \lambda_j q_j q_j^T\right) y = \sum_{j=1}^{d} \lambda_j \underbrace{y^T q_j}_{=\alpha_j} \underbrace{q_j^T y} = \sum_{j=1}^{d} \lambda_j \alpha_j^2$

Note that we can assume $\|y\| = 1$ or $y^T y = \sum_{j=1}^{d} \alpha_j^2 = 1$.

So $y^T \nabla^2 f(x) y$ is maximized when $\alpha_1^2 = 1$ and minimized when $\alpha_d^2 = 1$, giving $L = \lambda_1 = \max(\mathrm{eig}(A^T A))$ and $\mu = \lambda_n = \min(\mathrm{eig}(A^T A))$

Convergence Rate of Gradient Descent

- The gradient descent iteration:

$$x^{t+1} = x^t - \alpha_t \nabla f(x^t)$$

- Assumptions:
  - Function ‘f’ is L-strongly smooth and μ-strongly convex.
  - We set the step-size to $\alpha_t = 1/L$.
- Then gradient descent has a linear convergence rate:

$$f(x^t) - f(x^*) \leq O(\rho^t) \quad \text{for } \rho < 1.$$

  - It follows that we need $t = O(\log(1/\varepsilon))$ iterations.
    - This is good! We want ‘t’ to grow slowly in accuracy $1/\varepsilon$.
  - Also called ‘exponential’ convergence rate.

$f(x^t) - f(x^*) = \varepsilon \leq O(\rho^t)$

means $\varepsilon \leq c\rho^t$ for t large

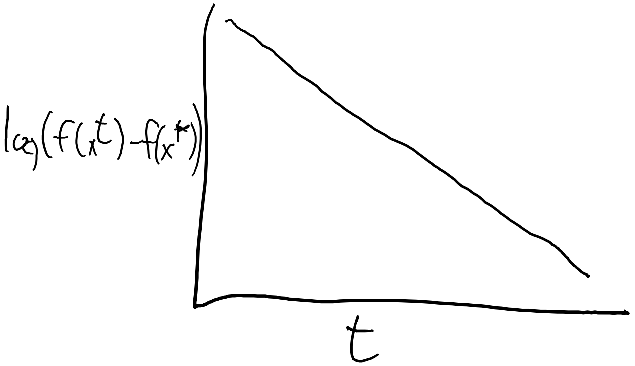
or $\log(\varepsilon) \leq \log(c\rho^t)$

$= \log(c) + t\log(\rho)$

or $t \geq \frac{\log(\varepsilon)}{\log(\rho)} - \text{const}$

or $t \geq O(\log(1/\varepsilon))$

(since $\rho < 1$)

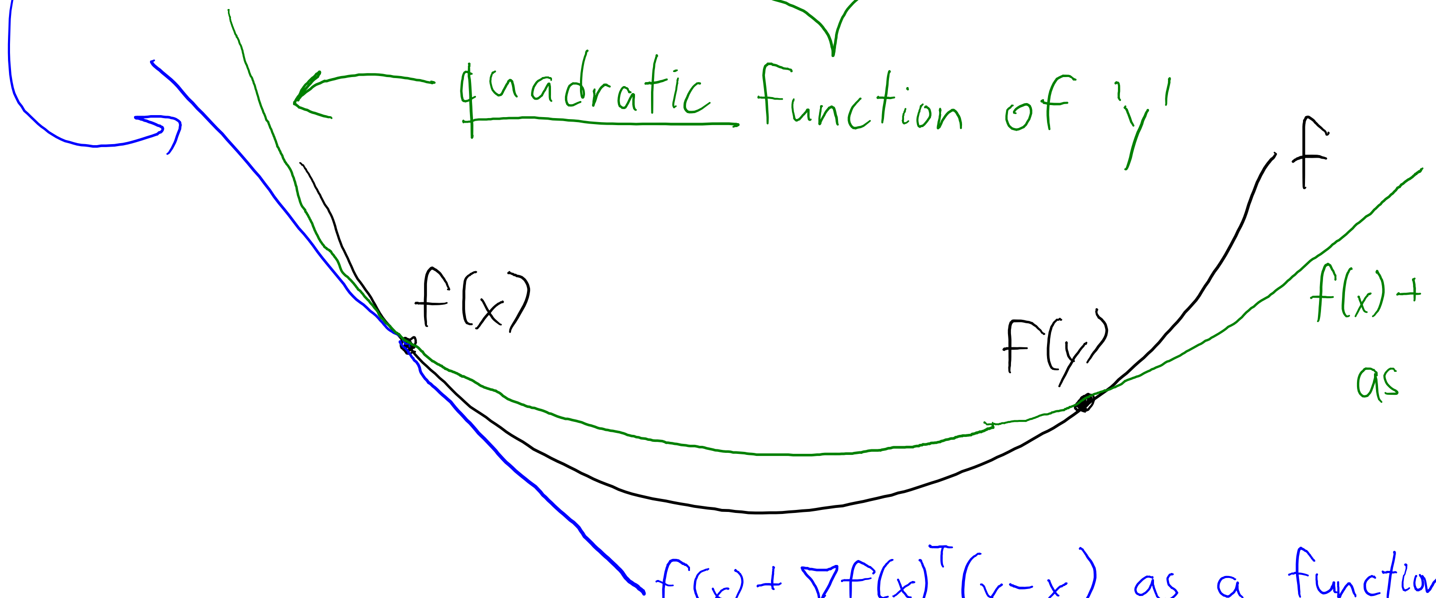


Convergence Rate of Gradient Descent

- One version of Taylor expansion:

$$f(y) = \underbrace{\underbrace{f(x) + \nabla f(x)^T(y-x)}_{\text{linearization of 'f' at 'x'}} + \frac{1}{2}(y-x)^T\nabla^2 f(z)(y-x)}_{\text{quadratic function of 'y'}}$$

for some z
for all x and y.



$f(x) + \nabla f(x)^T(y-x) + \frac{1}{2}(y-x)^T\nabla^2 f(z)(y-x)$ as function of y.

$f(x) + \nabla f(x)^T(y-x)$ as a function of 'y'

Using Strong-Smoothness

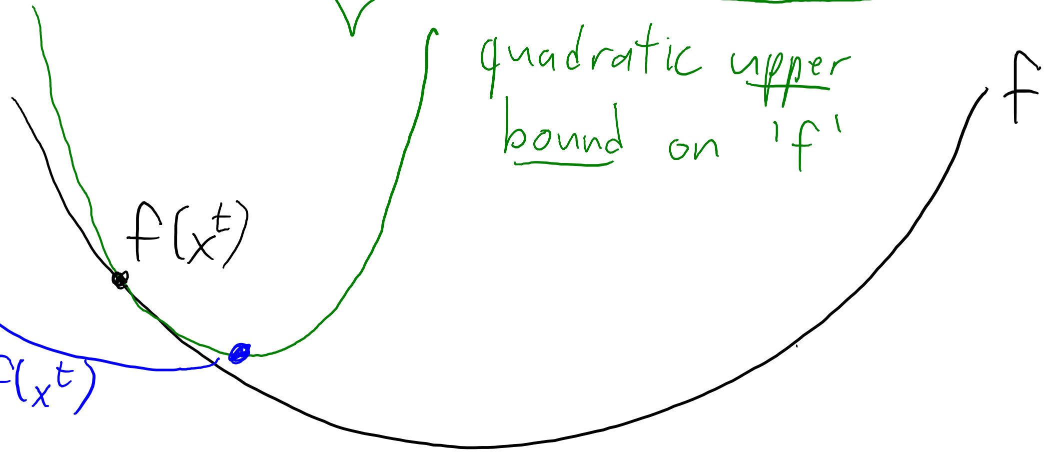
- One version of Taylor expansion:

$$f(y) = f(x) + \nabla f(x)^T(y-x) + \frac{1}{2}(y-x)^T\nabla^2 f(z)(y-x)$$ for <u>some</u> z

for <u>all</u> x and y.

From <u>strong-smoothness</u> we have: $v^T\nabla^2 f(z)v \leq L\|v\|^2$ for <u>any</u> z and v.

$$f(y) \leq \underbrace{f(x) + \nabla f(x)^T(y-x) + \frac{L}{2}\|y-x\|^2}$$ for <u>all</u> x and y.



If we set x^{t+1} to <u>minimize bound</u>, we get $x^{t+1} = x^t - \frac{1}{L}\nabla f(x^t)$

Let's find <u>min</u> of quadratic upper bound:

Let $q(y) = f(x) + \nabla f(x)^T(y-x) + \frac{L}{2}\|y-x\|^2$

$\nabla q(y) = 0 + \nabla f(x) - 0 + L(y-x)$

or

$y = x - \frac{1}{L}\nabla f(x)$.

Using Strong-Smoothness

- One version of Taylor expansion:

$$f(y) = f(x) + \nabla f(x)^T(y-x) + \frac{1}{2}(y-x)^T \nabla^2 f(z)(y-x)$$ for some z for all x and y.

From strong-smoothness we have: $v^T \nabla^2 f(z) v \leq L\|v\|^2$ for any z and v.

$$f(y) \leq f(x) + \nabla f(x)^T(y-x) + \frac{L}{2}\|y-x\|^2$$ for all x and y.

Set $x = x^t$ and $y = x^{t+1}$:

$$\begin{aligned} f(x^{t+1}) &\leq f(x^t) + \nabla f(x^t)^T(x^{t+1} - x^t) + \frac{L}{2}\|x^{t+1} - x^t\|^2 \\ &= f(x^t) + \nabla f(x^t)^T\left(-\frac{1}{L}\nabla f(x^t)\right) + \frac{L}{2}\left\|-\frac{1}{L}\nabla f(x^t)\right\|^2 \\ &= f(x^t) - \frac{1}{L}\nabla f(x^t)^T \nabla f(x^t) + \frac{1}{2L}\|\nabla f(x^t)\|^2 \\ &= f(x^t) - \frac{1}{2L}\|\nabla f(x^t)\|^2 \end{aligned}$$

$x^{t+1} = x^t - \frac{1}{L}\nabla f(x^t)$
(minimum of upper bound)

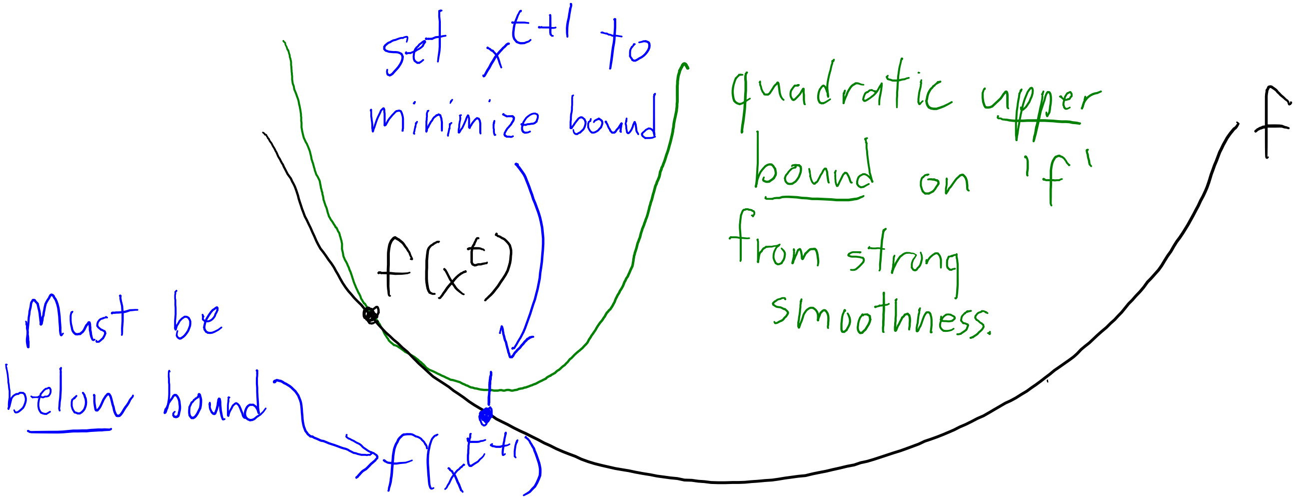
$\nabla f(x)^T \nabla f(x) = \|\nabla f(x)\|^2$

Using Strong-Smoothness

- We've derived a bound on guaranteed progress at iteration 't':

$$f(x^{t+1}) \leq f(x^t) - \frac{1}{2L}\|\nabla f(x^t)\|^2$$

  - If gradient is non-zero, guaranteed to decrease objective.
  - Amount we decrease grows with the size of the gradient.
  - Note: bound applies for any strongly-smooth function (e.g., non-convex)



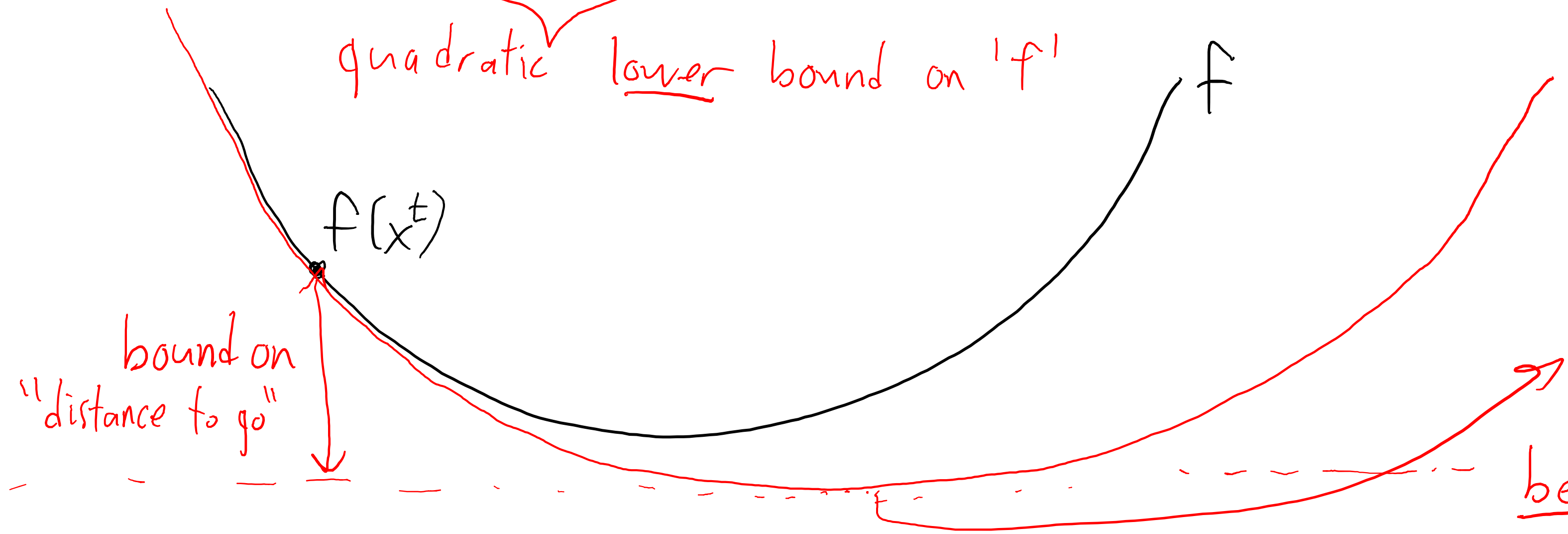
Using Strong-Convexity

- One version of Taylor expansion:

$$f(y) = f(x) + \nabla f(x)^T(y-x) + \frac{1}{2}(y-x)^T \nabla^2 f(z)(y-x)$$ for <u>some</u> z
for <u>all</u> x and y.

By <u>strong-convexity</u> we have $v^T \nabla^2 f(z) v \geq \mu$ for <u>all</u> v and z.

$$f(y) \geq \underbrace{f(x) + \nabla f(x)^T(y-x) + \frac{\mu}{2}\|y-x\|^2}_{\text{quadratic lower bound on 'f'}}$$



We know that $f(x^*)$ <u>cannot be</u> <u>below</u> minimum of bound.

Using Strong-Convexity

- One version of Taylor expansion:

$$f(y) = f(x) + \nabla f(x)^T(y-x) + \frac{1}{2}(y-x)^T \nabla^2 f(z)(y-x)$$ for <u>some</u> z

for <u>all</u> x and y.

By <u>strong-convexity</u> we have $v^T \nabla^2 f(z) v \geq \mu$ for <u>all</u> v and z.

$$f(y) \geq f(x) + \nabla f(x)^T(y-x) + \frac{\mu}{2}\|y-x\|^2$$

Minimize <u>both</u> sides with respect to y:

$$f(x^*) \geq f(x) - \frac{1}{2\mu}\|\nabla f(x)\|^2$$

Combining Strong-Smoothness and Convexity

- Our bound on guaranteed progress:

$$f(x^{t+1}) \leq f(x^t) - \frac{1}{2L}\|\nabla f(x^t)\|^2$$

- Our bound on 'distance to go':

$$f(x^*) \geq f(x^t) - \frac{1}{2\mu}\|\nabla f(x^t)\|^2 \iff -\frac{1}{2}\|\nabla f(x^t)\|^2 \leq -\mu(f(x^t) - f(x^*))$$

- Use 'distance to go' bound in guaranteed progress bound:

$$f(x^{t+1}) \leq f(x^t) - \frac{1}{L}(-\mu(f(x^t) - f(x^*)))$$

- Subtract $f(x^*)$ from both sides and simplify:

$$f(x^{t+1}) - f(x^*) \leq f(x^t) - f(x^*) - \frac{\mu}{L}(f(x^t) - f(x^*))$$

$$= \left(1 - \frac{\mu}{L}\right)[f(x^t) - f(x^*)]$$

Combining Strong-Smoothness and Convexity

- We've shown that:

$$f(x^t) - f(x^*) \leq \left(1 - \frac{\mu}{L}\right)\left[f(x^{t-1}) - f(x^*)\right]$$

- Applying this recursively:

$$\begin{aligned} f(x^t) - f(x^*) &\leq \left(1 - \frac{\mu}{L}\right)\left[\left(1 - \frac{\mu}{L}\right)\left[f(x^{t-2}) - f(x^*)\right]\right] \\ &= \left(1 - \frac{\mu}{L}\right)^2\left[f(x^{t-1}) - f(x^*)\right] \\ &= \left(1 - \frac{\mu}{L}\right)^3\left[f(x^{t-2}) - f(x^*)\right] \\ &\;\vdots \\ &= \underbrace{\left(1 - \frac{\mu}{L}\right)^t\left[f(x^0) - f(x^*)\right]}_{\rightarrow\, O(\rho^t)} \end{aligned}$$

- Since μ ≤ L, we've shown linear convergence rate.

Discussion of Linear Convergence Rate

- We've shown that gradient descent under certain settings has:

$$f(x^t) - f(x^*) \leq \left(1 - \frac{\mu}{L}\right)^t [f(x^0) - f(x^*)]$$

- The number L/μ is called the 'condition number' of 'f'.
- Connection to matrix condition number:
  - For least squares, condition number of 'f' is condition number of X^TX.
- This rate is dimension-independent:
  - It does not directly depend on dimensions 'd'.
  - In principle, applies to infinite-dimensional problems.
  - But, L may be larger (and μ smaller) in high-dimensional spaces.
- In practice, typically you don't have 'L'.
  - We'll get to practical issues later…

Weaker Assumptions for Linear Convergence

- We can get a linear convergence rate under weaker assumptions:
  - Proof works for any $\alpha < 2/L$.
    - Don't need 'L', just need step-size α small enough.
    - But optimal step-size in proof is $\alpha = 1/L$.
  - Proof works if you take the optimal step-size.

$$\alpha^* = \underset{\alpha > 0}{\text{argmin}} \{f(x^t + \alpha \nabla f(x^t))\} \Rightarrow f(x^t + \alpha^* \nabla f(x^t)) \leq f(x^t + \tfrac{1}{L} \nabla f(x^t))$$

    - You can compute this for quadratics: just minimizing a 1D quadratic.
  - Proof can be modified to work with approximation of 'L' or line-search.
    - What you typically do in practice.

Weaker Assumptions for Linear Convergence

- We can get a linear convergence rate under weaker assumptions:
  - Proof works for once-differentiable 'f' with L-Lipschitz continuous gradient:

Gradient does not change too quickly: $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$ for all x and y.

Since this implies: $f(y) \leq f(x) + \nabla f(x)^T(y - x) + \frac{L}{2}\|y - x\|^2$ for all y and x.

(see Nesterov's "Introductory Lectures on Convex Optimization")

  - This doesn't need to hold globally, proof works if we can show:

$$f(x^{t+1}) \leq f(x^t) + \nabla f(x^t)^T(x^{t+1} - x^t) - \frac{L}{2}\|x^{t+1} - x^t\|^2$$ for some L and all x^t and x^{t+1}

  - Basically, for differentiable functions this is a very weak assumption.

Weaker Assumptions for Linear Convergence

- We can get a linear convergence rate under weaker assumptions:
  - Strong-convexity is defined even for non-differentiable functions:

We say 'f' is μ-strongly convex if $f(x) - \frac{\mu}{2}\|x\|^2$ is a convex function of x.

  - For differentiable functions this is equivalent to:

$$f(y) \geq f(x) + \nabla f(x)^T(y - x) + \frac{\mu}{2}\|y - x\|^2 \quad \text{for } x \text{ and } y$$

  - This is still a strong assumption:
    - But note if 'f' is convex then 'f(x) + $(\lambda/2)||x||^2$ is λ-strongly convex.
  - What about non-convex functions?
    - Proof works if gradient grows faster than quadratic as you move away from solution.
    - Two phase analysis: prove that algorithm gets near minimum, then analyze local rate.
      - Convergence rate only applies for 't' large enough.

(pause)

Gradient Method: Practical Issues

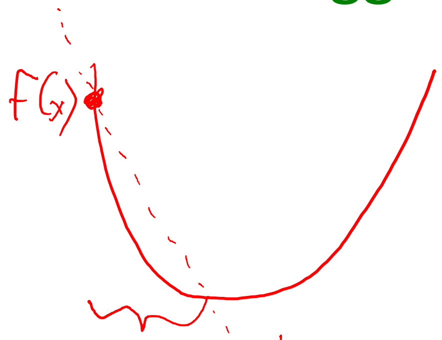
- In practice, you should never use $\alpha = 1/L$.
  - Often you don't know L.
  - Even if did, "local" L may be much smaller than "global" L: use bigger steps.
- Practical options:
  - Adaptive step-size:
    - Start with small 'L' (e.g., L = 1).
    - Double 'L' it if the guaranteed progress inequality from proof is not satisfied:

$$f\left(x^t - \tfrac{1}{L}\nabla f(x^t)\right) \leq f(x^t) + \nabla f(x^t)\left(\left(x^t - \tfrac{1}{L}\nabla f(x^t)\right) - x^t\right) + \tfrac{L}{2}\left\|\left(x^t - \tfrac{1}{L}\nabla f(x^t)\right) - x^t\right\|^2$$
$$= f(x^t) - \frac{1}{2L}\|\nabla f(x^t)\|^2$$

    - Use $\alpha_t = 1/L$.
    - Usually, end it up with much smaller 'L': bigger steps and faster progress.
    - With this strategy, step-size never increases.

Gradient Method: Practical Issues

- In practice, you should never use $\alpha = 1/L$.
  - Often you don't know L.
  - Even if did, "local" L may be much smaller than "global" L: use bigger steps.
- Practical options:
  - Armijo backtracking line-search:
    - On *each* iteration, start with large step-size α.
    - Decreasing α if Armijo condition is not satisfied:



$$f(x^{t+1}) \leq f(x_t) - \alpha\gamma\|\nabla f(x^t)\|^2 \quad \text{for some } \gamma \in (0, 1/2], \text{ usually } 10^{-4}$$

    - Works very well, particularly if you cleverly initialize/decrease α.
      - Fit linear regression to 'f' as α changes under (quadratic or cubic) basis, set α to minimum.
    - Even more fancy line-search: Wolfe conditions (makes sure α is not too small).

Gradient Method: Practical Issues

- Gradient descent codes requires you to write objective/gradient:

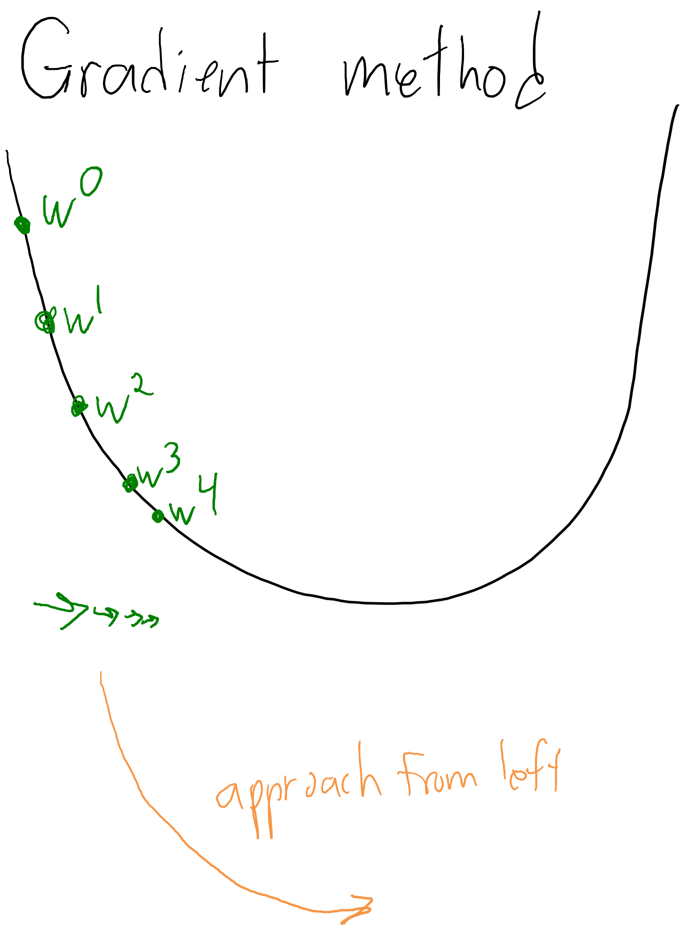
```
function [nll,g] = logisticGrad(w,X,y)
yXw = y.*(X*w);

% Function value
nll = sum(log(1+exp(-yXw)));

% Gradient
g = -X'*(y./(1+exp(yXw)));
end
```

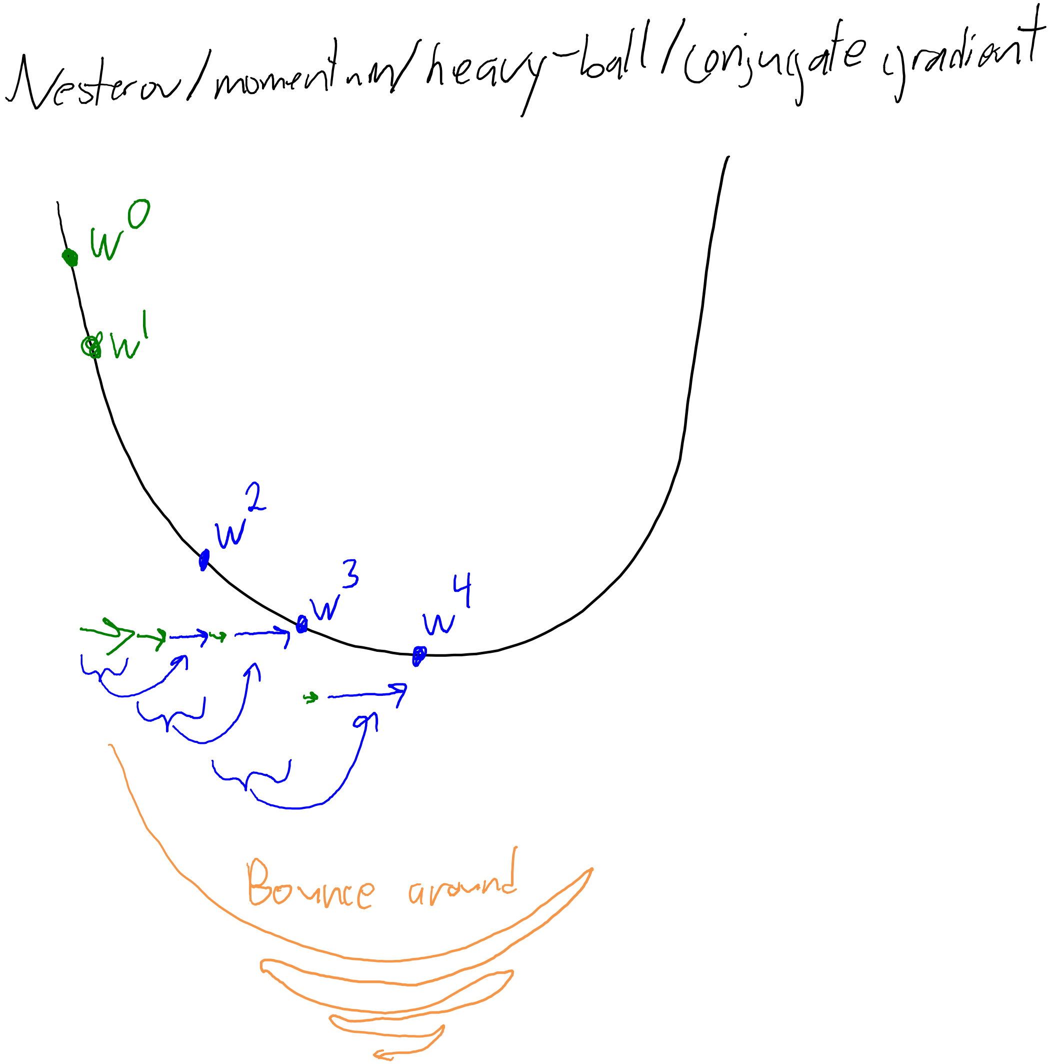
$$f(w) = \sum_{i=1}^{n} \log(1 + \exp(-y_i w^T x_i))$$

$$\nabla f(w) = \sum_{i=1}^{n} -\frac{y_i}{1+\exp(y_i w^T x_i)} x_i$$

- Make sure to check your derivative code:
  - Numerical approximation to partial derivative: $\nabla_i f(x) \approx \frac{f(x+\delta e_i) - f(x)}{\delta}$
  - Numerical approximation to direction derivative: $\nabla f(x)^T d \approx \frac{f(x+\delta d) - f(x)}{\delta}$

Nesterov's Method





Nesterov's Method

- Nesterov's accelerated gradient method (starting with $y^0 = x^0$):

$$x^{t+1} = y^t - \alpha_t \nabla f(y^t)$$
$$y^{t+1} = x^t + \beta_t (x^{t+1} - x^t)$$

$\alpha_t = \frac{1}{L}$

If 'f' is strongly-convex and ∇f is L-Lipschitz, improves from $O(\frac{L}{\mu}\log(\frac{1}{\varepsilon}))$ to $O(\sqrt{\frac{L}{\mu}}\log(\frac{1}{\varepsilon}))$

(close to optimal)

$\beta_t = \frac{1-\sqrt{\frac{\mu}{L}}}{1+\sqrt{\frac{\mu}{L}}}$

- Similar to heavy-ball/momentum method:

$$x^{t+1} = x^t - \alpha_t \nabla f(x^t) + \beta_t (x^t - x^{t-1})$$

  – Conjugate gradient: optimal α and β for strictly-convex quadratics.

Newton's Method

- Can be motivated as a quadratic approximation:

$$f(y) = f(x^t) + \nabla f(x^t)^T(y - x^t) + \frac{1}{2}(y - x^t)\nabla^2 f(z)(y - x^t)$$ for some z between y and x^t.

$$\approx f(x^t) + \nabla f(x^t)^T(y - x^t) + \frac{1}{2\alpha}(y - x^t)\nabla^2 f(x^t)(y - x^t)$$ (assuming $\nabla^2 f(x^t) \succ 0$)

- Newton's method is a second-order strategy (uses 2nd derivatives):

$$x^{t+1} = x^t - \alpha_t d^t$$ where d_t is the solution of $\nabla^2 f(x^t) d^t = \nabla f(x^t)$

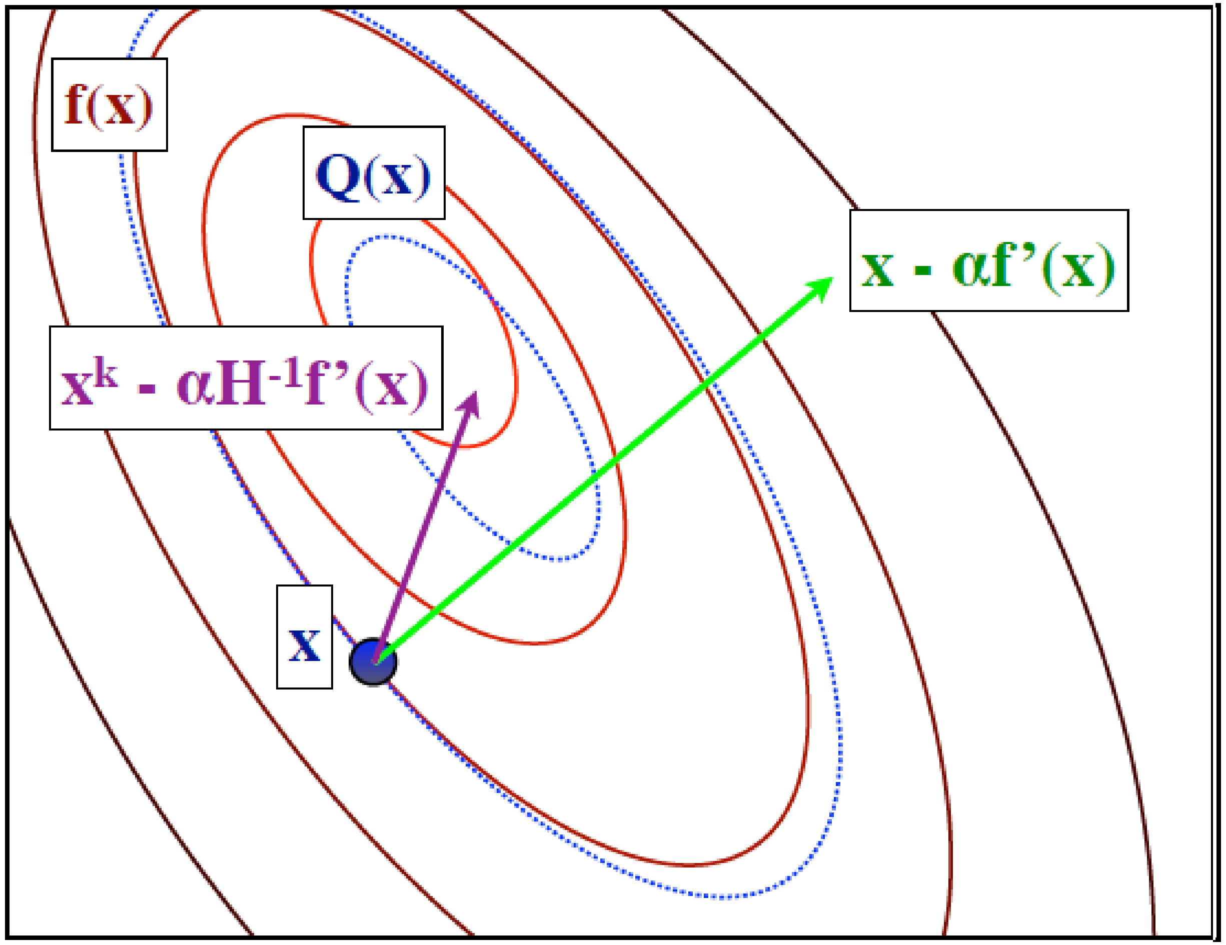
  – In stats, Newton's method applied to functions of form f(Ax) called "IRLS".

- Generalization of Armijo rule:

$$f(x^{t+1}) \leq f(x^t) - \alpha_t \gamma \nabla f(x^t)^T d^t$$

- Step-size α_t goes to 1 as we approach minimizer.

Newton’s Method



Convergence Rate of Newton's Method

If $\nabla^2 f(x)$ is Lipschitz-continuous and $\nabla^2 f(x^*) \succeq \mu I$ then for 't' large enough:

$$f(x^{t+1}) - f(x^*) \leq \underbrace{\rho_t}_{\text{progress changes at each iteration 't'}} [f(x^t) - f(x^*)] \quad \text{with} \quad \lim_{t \to \infty} \rho_t = 0.$$

- Local superlinear convergence: very fast, use it if you can.
- "Cubic regularization" of Newton's method gives global rates.
- But Newton's method is expensive if dimension 'd' is large:

Requires solution of $\underbrace{\nabla^2 f(x^t)}_{\text{'d' by 'd'}} d^t = \nabla f(x^t)$

Practical Approximations to Newton's Method

- Practical Newton-like methods:
  - Diagonal approximation: Approximate $\nabla^2 f(x^t)$ by diagonal H^t with elements $\nabla^2_{ii} f(x^t)$
  - Limited-memory quasi-Newton (L-BFGS): Diagonal plus low-rank Hessian approximation, chosen to satisfy "quasi-Newton" equations.
  - Barzilai-Borwein approximation: Approximate $\nabla^2 f(x^t)$ by identity matrix I, choose initial step-size α_t as least squares solution to quasi-Newton equations.
  - Hessian-free Newton: Apply gradient or conjugate gradient to approximately minimize quadratic approximation.

    Gradient requires $\nabla f(x^t) v$ but this can be cheaply approximated: $\nabla^2 f(x) d = \lim_{\delta \to 0} \frac{\nabla f(x + \delta d) - \nabla f(x)}{\delta}$
  - Non-linear conjugate gradient.

Practical Exercise and Homework?

- For practical experience with gradient/Nesterov/Newton methods:
  - http://www.cs.ubc.ca/~schmidtm/MLSS/differentiable.pdf
- Corresponding code is available here:
  - http://www.cs.ubc.ca/~schmidtm/MLSS
- Works through a Matlab implementation of:
  - Gradient descent (fixed step size)
  - Armijo line-search.
  - Hermite polynomial
  - Nesterov and Newton method.
  - Practical approximations of Newton's method.
- At the end, you will have a useful large-scale code.

Summary

- Convex functions: all stationary points are global minima.
- Showing functions are convex.
- Gradient descent finds stationary point of differentiable function.
- Rate of convergence of gradient descent is linear.
- Weaker assumptions for gradient descent:
  – L-Lipschitz gradient, weakening convexity, practical step sizes.
- Faster first-order methods like Nesterov's and Newton's method.

- Next time:
  – What if we don't know which features are relevant or which basis to use?