Optimization and Inference

COS 424/524, SML 302: Fundamentals of Machine Learning

Professor Engelhardt

COS424/524, SML 302

Lecture 13

Algorithms for fitting models to data

Most methods and models in machine learning require estimating model parameters θ from a data set $\mathcal{D}$.

We can frame model fitting as an *optimization problem*, where we optimize a *cost function* with respect to some *parameters* theta.

Sometimes (e.g., linear, ridge regression), we can compute optimal points of log likelihood cost functions analytically (closed-form solution)

Closed form solutions

For a point estimate of parameters, with a tractable closed-form solution available, we should use that first.

Often we cannot use closed-form solutions:

- a closed form solution may not be available (e.g., logistic regression)
- the problem is ill-conditioned
- the closed form solution is computationally intractable.

In these situations, we must rely on optimization approaches that *depend on the difficulty of the optimization problem*

Convexity and optimization

When our cost function is *strongly convex*, we have tools available from convex optimization that are tractable with some guarantees on behavior.

Example of cost functions that are strongly convex

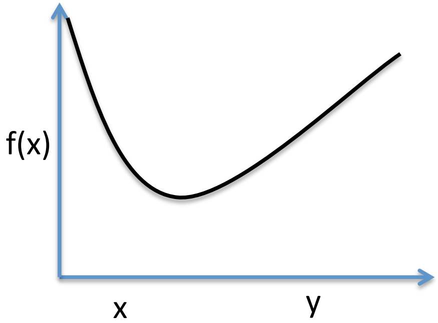
- Log likelihood for exponential family distributions
- Log likelihood for generalized linear models

Convex functions

Definition of a convex function

A function $f(\cdot)$ is *convex* if its domain is a convex set and if, for all x, y in the domain and θ with $0 \leq \theta \leq 1$ we have:

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

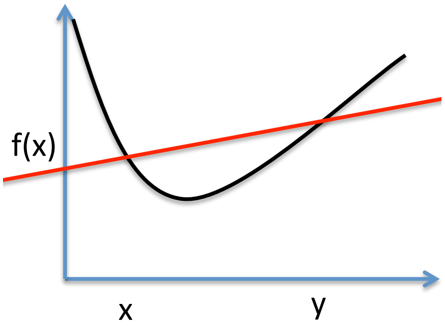


Convex functions

Definition of a convex function

A function $f(\cdot)$ is convex if its domain is a convex set and if, for all x, y in the domain and θ with $0 \leq \theta \leq 1$ we have:

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

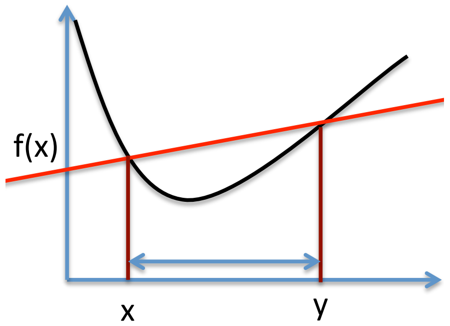


Convex functions

Definition of a convex function

A function $f(\cdot)$ is convex if its domain is a convex set and if, for all x, y in the domain and θ with $0 \leq \theta \leq 1$ we have:

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

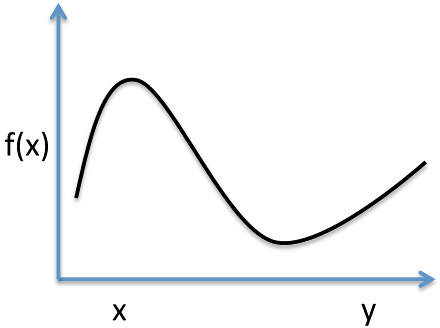


Non convex functions

Definition of a convex function

A function $f(\cdot)$ is convex if its domain is a convex set and if, for all x, y in the domain and θ with $0 \leq \theta \leq 1$ we have:

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

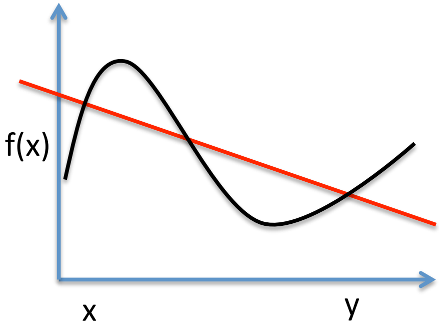


Non convex functions

Definition of a convex function

A function $f(\cdot)$ is convex if its domain is a convex set and if, for all x, y in the domain and θ with $0 \leq \theta \leq 1$ we have:

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

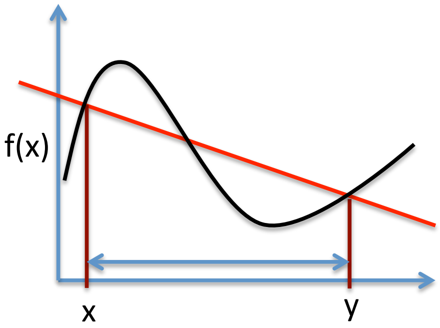


Non convex functions

Definition of a convex function

A function $f(\cdot)$ is convex if its domain is a convex set and if, for all x, y in the domain and θ with $0 \leq \theta \leq 1$ we have:

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

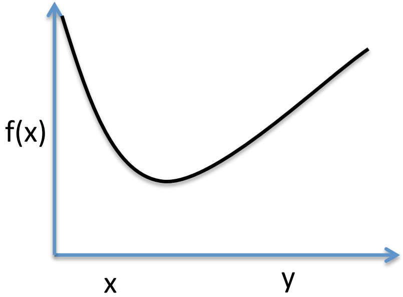


Convex functions: second order conditions

Second order conditions

A function $f(\cdot)$ is *convex* if and only if its domain is convex and

$$\nabla_x^2 f(x) \geq 0.$$



Exercise: prove the second order condition for a univariate function.

Convex functions

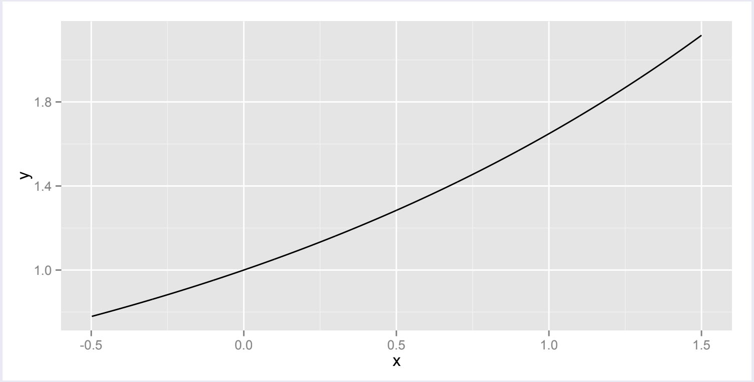
Examples of convex functions

- Exponential: $f(x) = e^{ax}$ for $a \in \Re$ (convex)
- Powers: $f(x) = x^a$, when $a = 2, 4$ (convex)
- Powers of absolute value: $f(x) = |x|^a$ for $a \geq 1$ (convex)
- Logarithm: $f(x) = \log x$ for $x > 0$ (concave)
- Negative entropy: $f(x) = x \log x$ for $x > 0$ (convex)
- Norms p: $f(x) = ||x||_p$ for all p-norms (convex)
- Max: $f(x) = \max(x)$ for $x \in \Re^p$ (convex)
- Log-sum-exp n: $f(x) = \log(e^{x_1} + \cdots + e^{x_n})$ (convex)
- log determinant n: $f(x) = log|X|$ where $|\cdot|$ is determinant function (concave)

Exercise: prove that these functions are convex/concave

Convex functions: exponential

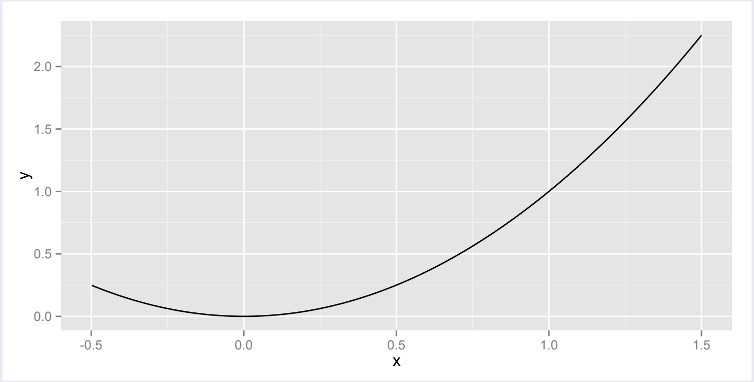
Examples of convex functions



Exponential: $f(x) = e^{ax}$ for $a = 2$ (convex)

Convex functions: powers

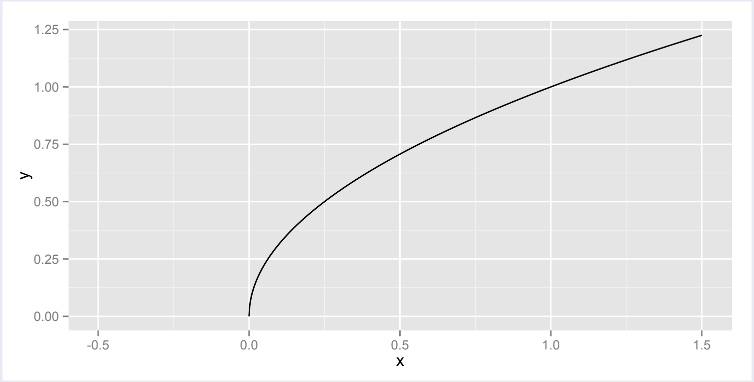
Examples of convex functions



Powers: $f(x) = x^a$, when $a = 2$ (convex)

Convex functions: powers

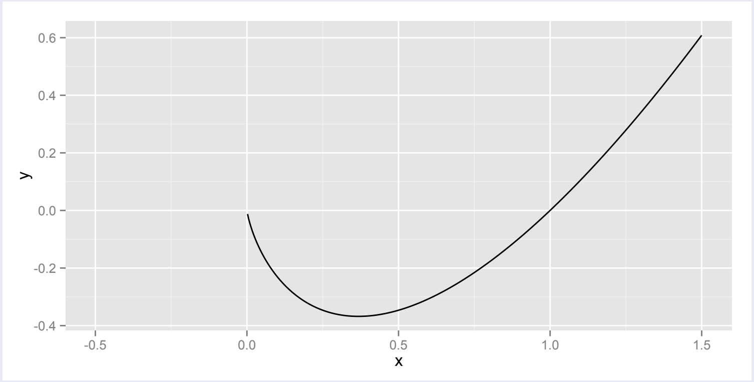
Examples of convex functions



Powers: $f(x) = x^a$, when $a = 0.5$ (concave)

Convex functions: Negative entropy

Examples of convex functions



Negative entropy: $f(x) = x \log x$ for $x > 0$ (convex)

Convex functions

Operations that preserve convexity

If function f is convex, then so is:

- Non-negative weighted sums: $f = w_1 f_1 + \cdots + w_m f_m$
- Affine mapping: $g(x) = f(Ax + b)$
- Point-wise maximum: $f(x) = \max\{f_1(x), \ldots, f_m(x)\}$
- Many types of function composition (see Equations 3.10, 3.11 in CO)

Exercise: prove that these operations preserve convexity

Strong convexity

When our cost function to optimize is *strongly convex*, we have a number of tools available from convex optimization that will allow us to solve these problems quickly and with some guarantees on their behavior.

Strongly convex functions

A function $f(\cdot)$ is strongly convex for x in a convex domain if there exists an $m > 0$ such that

$$\nabla^2 f(x) \geq m.$$

Is a linear function (e.g., $f(x) = x$) strongly convex? How about $f(x) = x^2$?

Example of cost functions that are strongly convex

- Log likelihood for exponential family distributions
- Log likelihood for generalized linear models

Implications of strong convexity

Strong convexity gives us a number of guarantees about a function:

- If $\nabla f(x)$ is small for some x, then x is nearly optimal
- The optimal point x^* of cost function f is unique
- If we knew m, we could determine a practical stopping criterion; m is rarely known.

Descent methods

Descent methods produce a minimizing sequence of $x^{(k)}$ for iterations $k = 1, 2, \ldots$ of the form:

$$x^{(k+1)} \leftarrow x^{(k)} + \rho^{(k)} \Delta x$$

where

- Δx is the *search direction*
- $\rho^{(k)}$ is the *step size*
- k is the iteration number.

Descent methods imply a minimization, and thus:

$$f(x^{(k+1)}) < f(x^{(k)})$$

Descent methods

General descent method

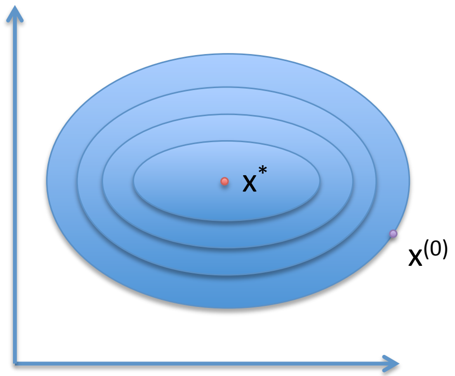
1. Given a starting point x, repeat until a stopping criterion is achieved:
    1. Determine a descent direction Δx
    2. Choose a step size $\rho > 0$
    3. Update: $x^{(k+1)} \leftarrow x^{(k)} + \rho^{(k)} \Delta x$

The stopping criterion is often of the form $||\nabla f(x)||_2 \leq \epsilon$, where $\epsilon > 0$ and small.

What assumption makes this reasonable?

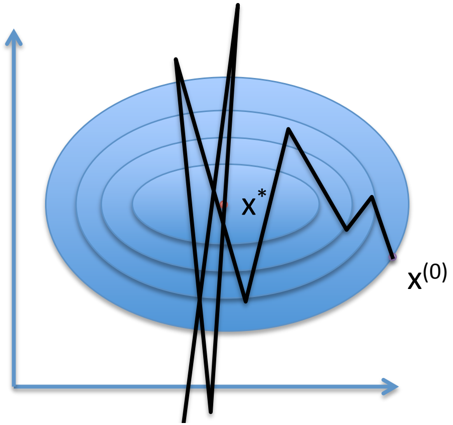
How to choose the step size?

One question we have avoided thusfar is how to set the step size.



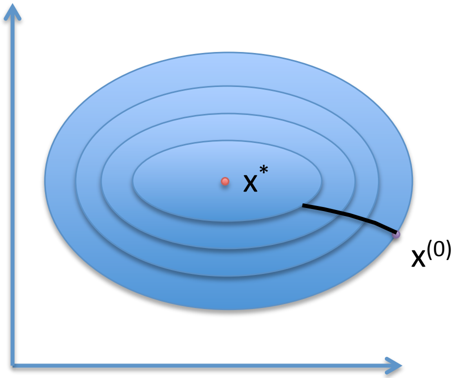
We are taking steps starting at $x^{(0)}$ towards x^*, which optimizes our cost function $f(x)$.

How to choose the step size?



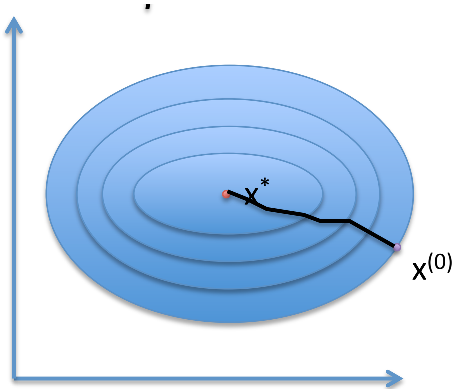
If the step size is too large, the sequence may diverge.

How to choose the step size?



If the step size is too small, the sequence may fail to converge in a reasonable number of iterations.

How to choose the step size?



A good step size will allow the sequence to converge in a reasonable number of iterations from many starting points in the function domain.

Step size: backtracking line search

To set a reasonable step size, use backtracking line search at each iteration

Backtracking line search

1. Fix a parameters $0 < \beta < 1$, $0 < \alpha < 0.5$ (often $\alpha = 0.5$)
2. Start with $\rho = 1$
3. While $f(x + \rho\Delta x) > f(x) + \alpha\rho\nabla f(x)^T \Delta x$:
    1. $\rho \leftarrow \beta\rho$

This is called *backtracking* line search because it starts with $t = 1$ and continues to scale the step size by β

- $f(x + \rho\Delta x)$ is the point that you step to
- $f(x) + \alpha\rho\nabla f(x)^T \Delta x$ is the linear extrapolation of $f(x)$, where α is fraction of decrease in $f(x)$ (predicted by linear extrapolation) that we would accept.

Gradient descent

In this descent framework, a natural choice for the step direction is the negative gradient:

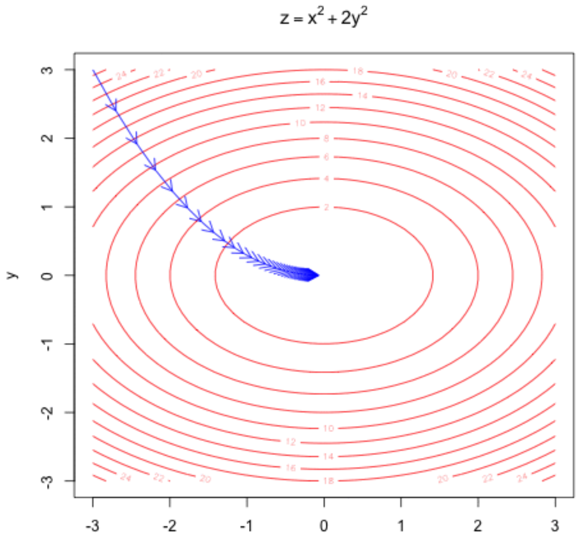
$$\Delta x = -\nabla f(x).$$

Then:

Gradient descent method

1. Given a starting point x, repeat until $||\nabla f(x)||_2 \leq \epsilon$:
    1. Set $\Delta x = -\nabla f(x)$
    2. Choose a step size $\rho > 0$ via backtracking line search
    3. Update: $x^{(k+1)} \leftarrow x^{(k)} + \rho^{(k)} \Delta x$

Demonstration of gradient descent



Gradient descent: Summary

- Gradient descent often has approximately linear convergence, meaning $f(x^{(k)}) - f(x^*)$ converges to zero as a geometric series
- Choice of backtracking parameters α, β has some effect on convergence
- For many high dimensional problems, convergence rates are prohibitively slow.

Stochastic gradient ascent

What if we have online data, such as emails, for which we cannot touch every sample at each iteration?

- Stochastic gradient ascent is critical to large scale optimization.
- The idea is to follow noisy realizations of the gradient with a decreasing step size.
- As long as the expectation of the noisy gradient is equal to the true gradient, stochastic gradient ascent will converge to a local optimum.

Scaling optimization

Why does this help us scale up optimization problems?

Many complex cost functions are sums of simpler cost functions.

Cost functions

For statistical models we have studied, data are IID, variables are observed.
Then, the log likelihood function, $\mathcal{L}(\beta; \mathcal{D})$, decomposes by sample,

$$\mathcal{L}(\beta; \mathcal{D}) = \sum_{i=1}^{n} \mathcal{L}_i(\beta; (y_i, \mathbf{x_i})).$$

Recall that β are the coefficients (parameters). We saw this with linear and logistic regression.

(In the next lecture, we will discuss inference in models with latent variables.)

Noisy gradient

Approximate the gradient by computing a noisy gradient from a subset of the data.

- First, choose x_t uniformly at random from the data set.
- Then, compute the noisy gradient as follows,

$$\hat{\nabla}_\beta \mathcal{L}(\beta) = n \nabla_\beta \mathcal{L}_i(\beta; (y_i, x_i)).$$

- Confirm that $\mathrm{E}\left[\hat{\nabla}_\beta\right] = \nabla_\beta \mathcal{L}(\beta)$.

Stochastic gradient ascent algorithm

Stochastic gradient ascent algorithm

1. Given a starting point $\beta^{(k)}$, repeat until $||\nabla f(x)||_2 \leq \epsilon$:
    1. Select a sample from the data i uniformly at random

$$(y_i, x_i) \sim \mathrm{Unif}((y_1, x_1), \ldots, (y_n, x_n))$$

    2. Compute a noisy gradient $\mathcal{L}_i(\beta; (y_i, \mathbf{x_i}))$

$$\hat{\nabla}_\beta \mathcal{L}(\beta^{(k)}; \mathcal{D}) = n \nabla_\beta \mathcal{L}_i(\beta^{(k)}; (y_i, x_i))$$

    3. Update the current estimate of the parameters:

$$\beta^{(k+1)} = \beta^{(k)} + \rho^{(k)} \hat{\nabla}_\beta \mathcal{L}(\beta^{(k)}; \mathcal{D})$$

With appropriate settings of the step size, stochastic gradient ascent converges to a local optimum of the likelihood.

Mini-batch stochastic gradient ascent

The noisy gradient need not be estimated from one random sample with large numbers of samples and slow convergence rates.

In practice, we can sample a *mini-batch* of data at each iteration, which is the largest subset that we can handle computationally.

If B is mini-batch size, then we scale noisy gradient by n/B to ensure that its expectation is equal to the true gradient.

Setting the step size for stochastic gradient descent

Line search and approximate line search are infeasible for stochastic approaches.

[Robbins & Monro 1951]: step size must satisfy these conditions

$$\begin{aligned} \sum_{k=1}^{\infty} \rho^{(k)} &= \infty \\ \sum_{k=1}^{\infty} (\rho^{(k)})^2 &< \infty \end{aligned}$$

Steps size: forgetting rates

One expression that satisfies the Robbins-Monro conditions is

$$\rho^{(k)} = (k + \tau)^{-\kappa}.$$

The *forgetting rate* $\kappa \in (0.5, 1]$ controls how quickly old information is lost.

The *delay* τ downweights early iterations.

Can SGD be parallelized?

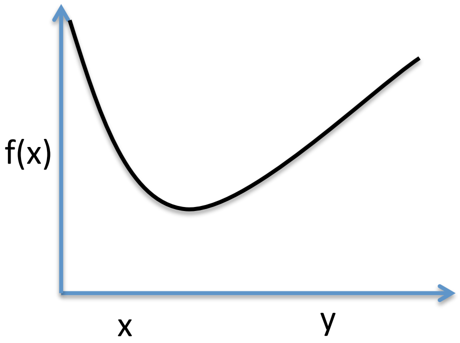
Newton-Raphson method

The Newton-Raphson method (or Newton's method) is similar to the gradient descent method, but in Newton-Raphson we use the function's Hessian (second derivative) to dictate the step size.

- We need to compute the inverse of the Hessian, which may (or may not) be computationally expensive to compute
- Newton-Raphson generally requires fewer iterations to converge than (stochastic) gradient ascent because of adaptive step size
- Newton-Raphson is often easier to make work, because step size does not need to be adjusted carefully.

Newton-Raphson intuition

- The intuition behind using inverse of second derivative of cost function as step size is that, near optimum, first derivative is small
- Thus, the closer we are to optimum, the smaller our gradient step.



Newton-Raphson algorithm

In the descent framework, we choose the following:

$$\Delta x = -\nabla^2 f(x)^{-1} \nabla f(x).$$

Because f is strongly convex, we know that $\nabla^2 f(x)^{-1} > 0$

Then:

Newton-Raphson method

1. Given a starting point x, repeat until $||\nabla f(x)||_2 \leq \epsilon$:
    1. Set $\Delta x = -\nabla^2 f(x)^{-1} \nabla f(x)$
    2. Update: $x^{(k+1)} \leftarrow x^{(k)} + \Delta x$

Can adapt the stopping criterion to include Hessian.

Example: Newton-Raphson for logistic regression

For logistic regression, cost function $f(\cdot)$ is log likelihood $\log \mathcal{L}(\beta; \mathcal{D})$.

We take the partial derivatives of $\log \mathcal{L}(\beta; \mathcal{D})$ with respect to β:

$$\begin{aligned}\frac{\partial^2 \log \mathcal{L}}{\partial \beta \; \partial \beta^T} &= \sum_{i=1}^{n} x_i \frac{\partial \mu_i}{\partial \beta} \\ &= \sum_{i=1}^{n} -x_i x_i^T (\mu_i(1-\mu_i)) = -\sum_{i=1}^{n} x_i^T S x_i \\ &\text{where } S = diag(\mu_i(1-\mu_i)).\end{aligned}$$

Recall that $\mu_i = \frac{1}{1+\exp\{-\beta^T \mathbf{x_i}\}}$

What is this term with respect to the Bernoulli distribution?

Example: Newton-Raphson for logistic regression

The Newton-Raphson update for logistic regression is:

$$\beta^{(k+1)} \quad \leftarrow \quad \beta^{(k)} + (x_i^T S_t x_i)^{-1} x_i^T (y_i - \mu_i),$$

which is called *iteratively reweighted least squares (IRLS)*.

Note $S_{i,i}$ is the variance term for our predicted response y_i:

- when our uncertainty is greater for our predictions (i.e., predictions are closer to 0.5), we will take smaller steps
- when we are more certain about our predictions (i.e., predictions are closer to 0 or 1), our step size will be larger

Newton-Raphson summary

- Convergence of Newton-Raphson is fast: quadratic near x^*
- Newton-Raphson scales well with sample size
- No parameters to tune
- Downside: Hessian may be expensive to compute

Non-convex optimization

What do we do if our cost function is non-convex?

- When possible, use *alternating minimization*, also known as *coordinate descent* or *expectation-maximization*
- Non-convex functions may be relaxed and approximated by convex functions
- Bayesian optimization methods
- Many other approaches, many of which are heuristic.

Alternating minimization

Often it is useful to iteratively minimize one variable and then another variable, while holding the alternative variable fixed.

Let's defer discussion of alternating minimization to the lectures on *expectation maximization*.

Convex relaxation

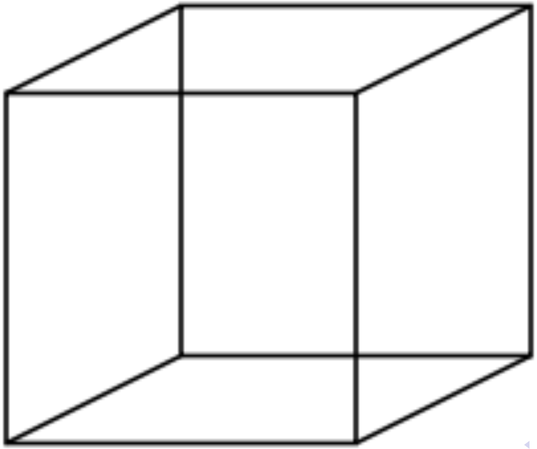
Recall that we have seen an example of a *convex relaxation* in this course.

- When performing sparse regression, our ideal goal was to use ℓ_0 regularization to select the smallest number of predictors possible.
- For p predictors, this goal required a search over a non-convex space of size 2^p (the corners of a p-hypercube)
- We relaxed this intractable problem to a *convex* problem by using the ℓ_1 norm, or the sums over the absolute values of the coefficients.
- This is equivalent to searching within the space $\Re^p$

Why is this approach not consistently useful in practice?

Convex relaxation

A solution in the relaxed space may not be able to be mapped back to the true space without loss of optimality.



Optimization Summary

- If the model can be fit with a closed form solution, do that.
- If the log likelihood is strongly convex, try gradient descent or Newton's method
- If the model is not strongly convex, try EM, convex relaxations, or Bayesian optimization
- We have only scratched the surface of optimization
- Important theme: trade-offs between model complexity, computational tractability, and guarantees on the solution

Additional Resources

- Stephen Boyd & Lieven Vandenberghe *Convex Optimization* (PDF online): much of the material from today's class was taken from this text, and Chapters 3 and 9 in particular.
- MLAPA: Chapter 8
- *Elements of Statistical Learning*: Chapter 10.10

- (videos) Stephen Boyd, Stanford: EE364a Lecture Videos
- Metacademy: *optimization problems*
- Metacademy: *convex optimization*