# CS-GY 6923: Lecture 6
# Gradient Descent + Stochastic Gradient Descent

NYU Tandon School of Engineering, Prof. Christopher Musco

**Goal**: Minimize generic differentiable loss function:

$$L(\beta) = -\sum_{i=1}^{n} y_i \log(h(\beta^T \mathbf{x}_i)) + (1 - y_i) \log(1 - h(\beta^T \mathbf{x}_i))$$

$$L(\beta) = \|\mathbf{X}\beta - \mathbf{y}\|_2^2$$

$$L(\beta) = \|\mathbf{X}\beta - \mathbf{y}\|_1 + \lambda\|\beta\|_2^2$$

I.e. find $\beta^* = \arg\min L(\beta)$.

**Gradient Descent:** Most common iterative method for solving this problem.

# FIRST ORDER OPTIMIZATION

Given a function $L$ to minimize, assume we have routines for computing:

- **Function oracle**: Evaluate $L(\beta)$ for any $\beta$.
- **Gradient oracle**: Evaluate $\nabla L(\beta)$ for any $\beta$

Gradient descent will use these routines in a black-box way to find the optimal $\beta^*$.

# GRADIENT DESCENT

**Basic Gradient descent algorithm:**

- Choose starting point $\beta^{(0)}$
- For $i = 0, \ldots, T$:
  - $\beta^{(i+1)} = \beta^{(i)} - \eta \nabla L(\beta^{(i)})$
- Return $\beta^{(t)}$.

$\eta$ is the step-size parameter or learning rate.

**We came to an important observations:**

1. For small enough $\eta$, we always have that $L(\beta^{(i+1)}) \leq L(\beta^{(i)})$.

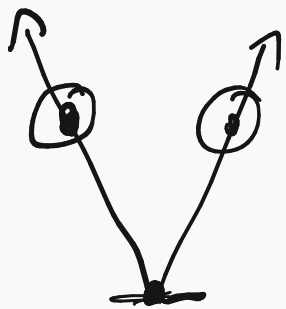
$$L(\beta + \underline{v}) - L(\beta) \approx \langle \nabla L(\beta), v \rangle. = \langle \nabla L(\beta), -\eta \nabla L(\beta) \rangle$$

want negative

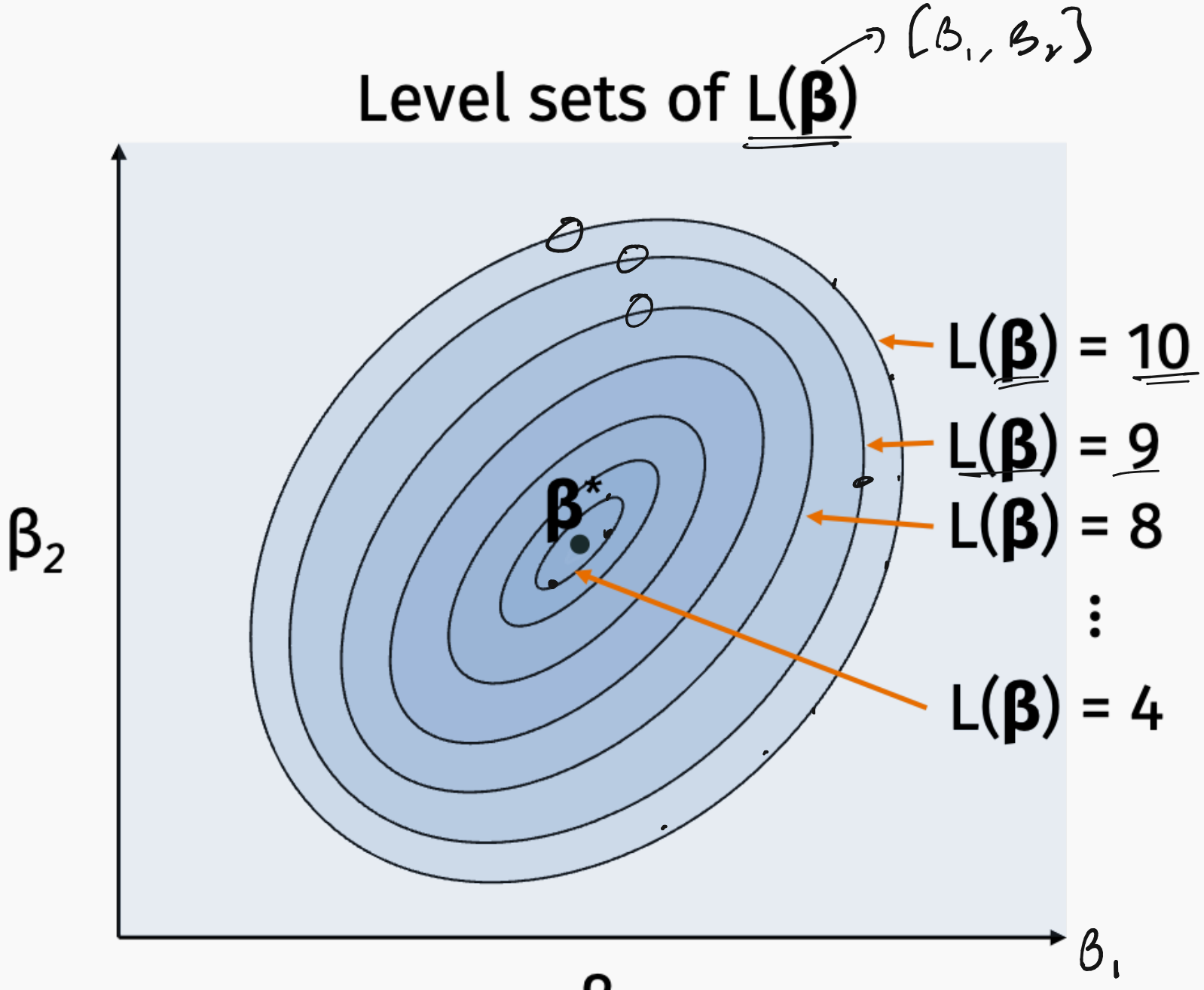
$$v = -\eta \nabla L(\beta) \qquad = -\eta \|\nabla L(\beta)\|_2^2$$

↓ negative number

**Conclusion:** Gradient descent always converges to a local minimum or stationary point of $L$. Typically to a local minimum.



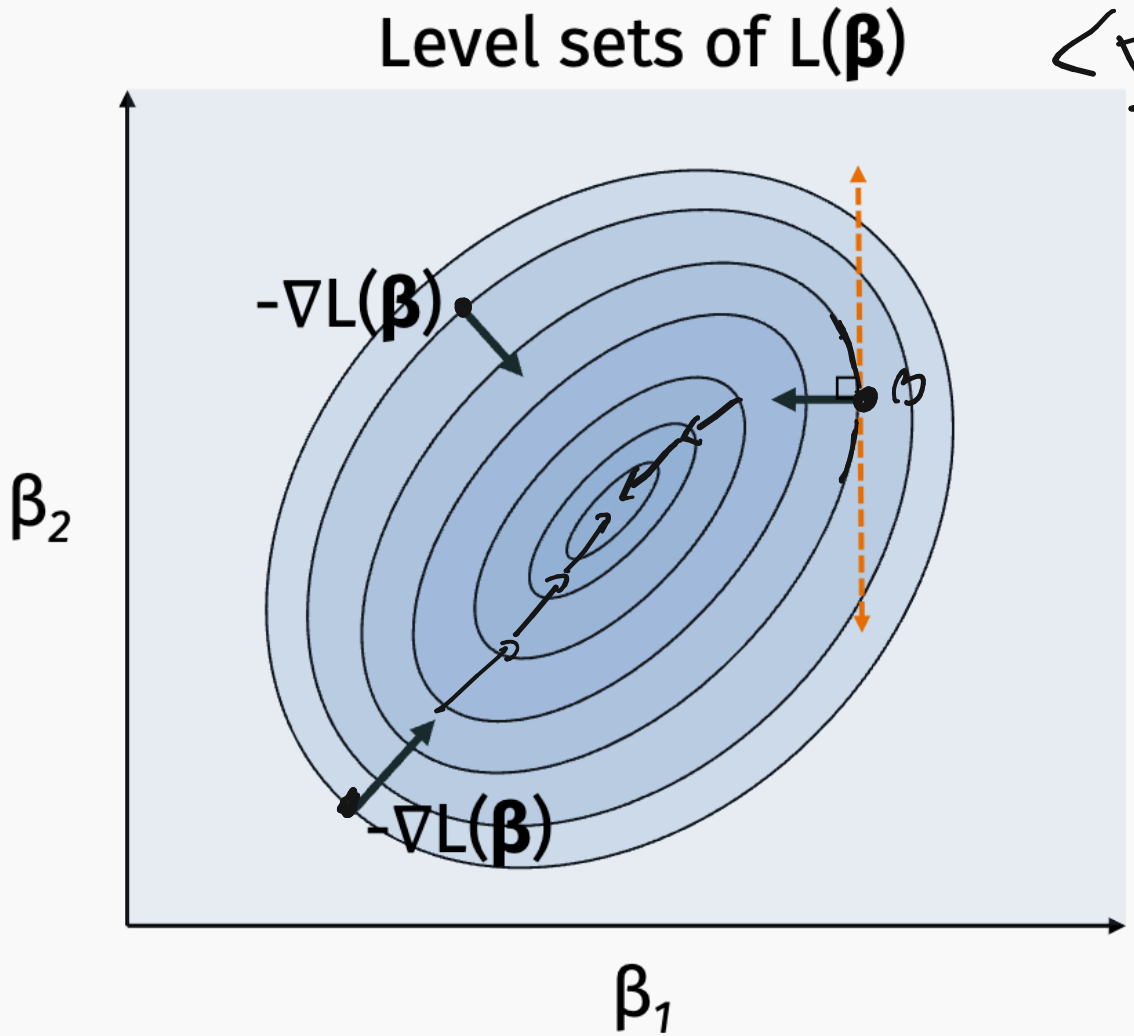
# VISUALIZING IN 2D



# STEEPEST DESCENT

**Claim (Gradient descent = Steepest descent)**

$$\frac{-\nabla L(\beta)}{\|\nabla L(\beta)\|_2} = \arg\min_{v, \|v\|_2 = 1} \langle \nabla L(\beta), v \rangle$$



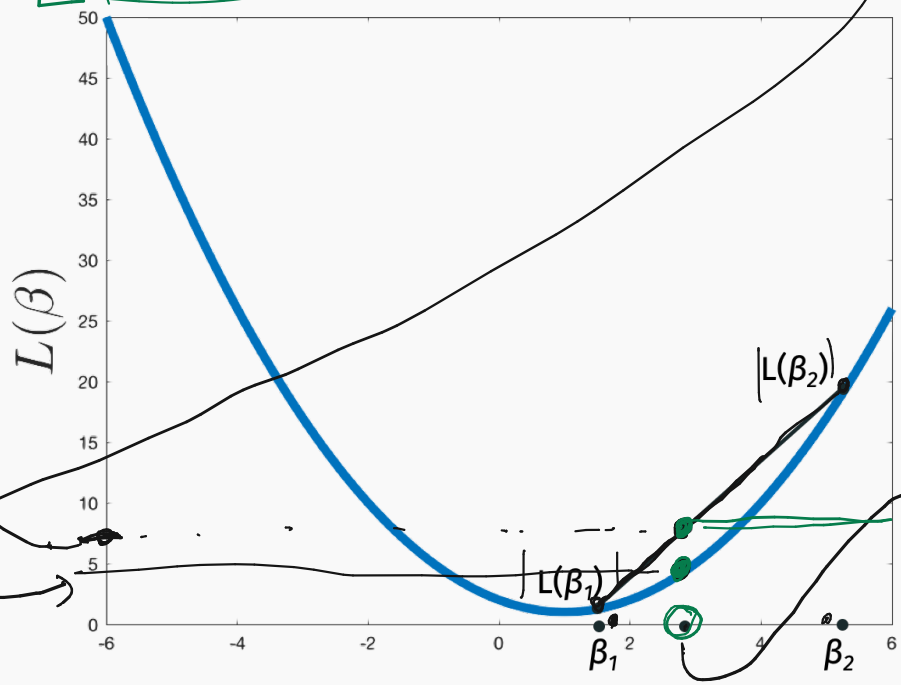

$\langle \nabla L(\beta), v \rangle = 0$

For a broad class of functions, GD converges to global minima.

**Definition (Convex)**

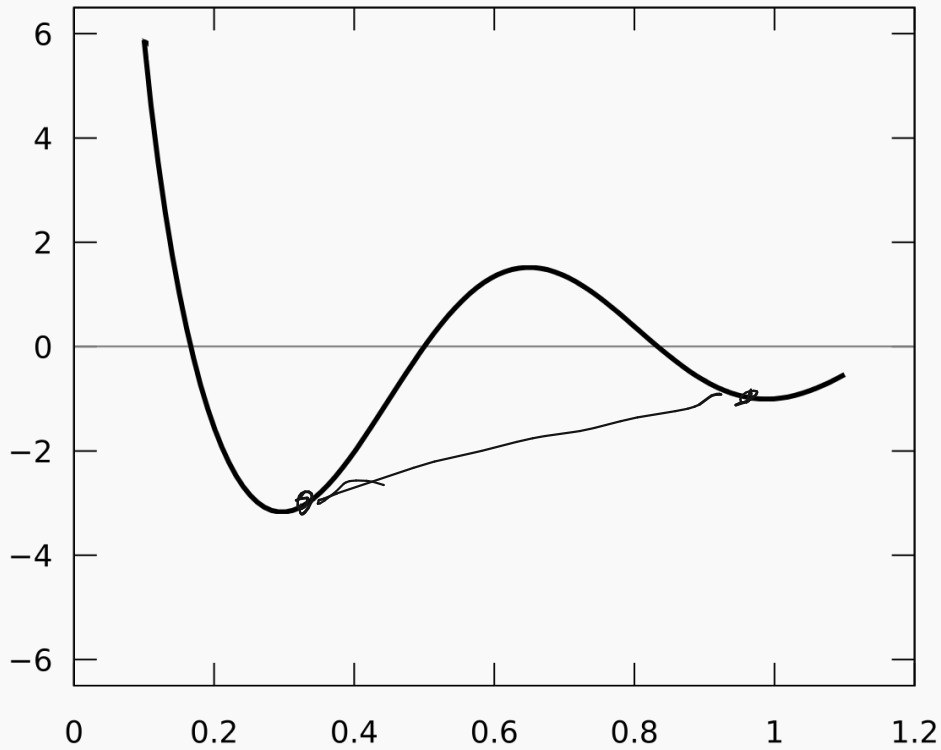
A function $L$ is convex iff for any $\beta_1, \beta_2, \lambda \in [0, 1]$:

$$(1-\lambda) \cdot L(\beta_1) + \lambda \cdot L(\beta_2) \geq L((1-\lambda) \cdot \beta_1 + \lambda \cdot \beta_2)$$



# CONVEX FUNCTION

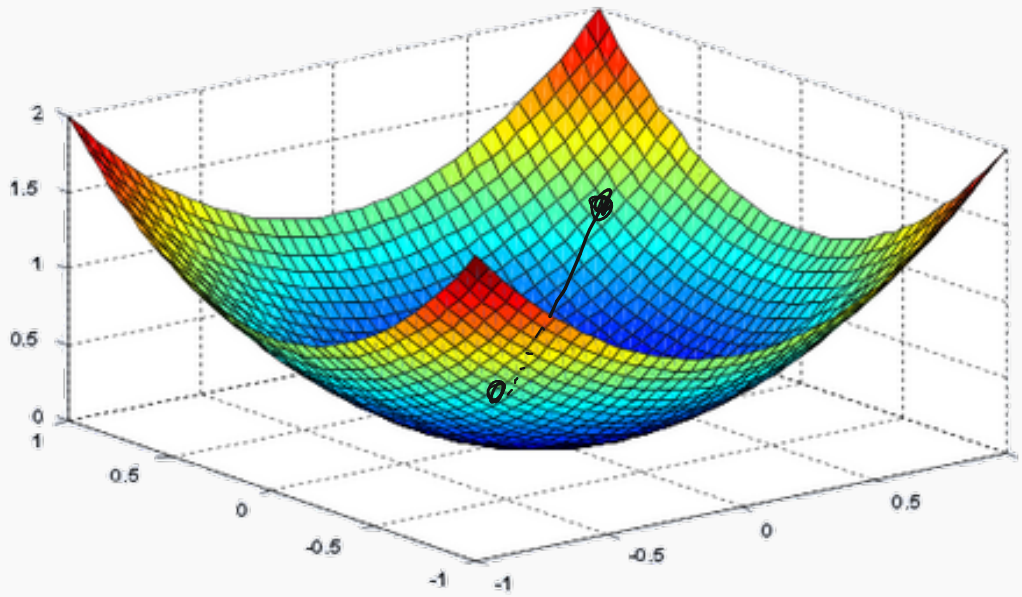
**In words:** A function is convex if a line between any two points on the function lies above the function. Captures the notion that a function looks like a bowl.




This function **is not** convex.

# CONVEX FUNCTION

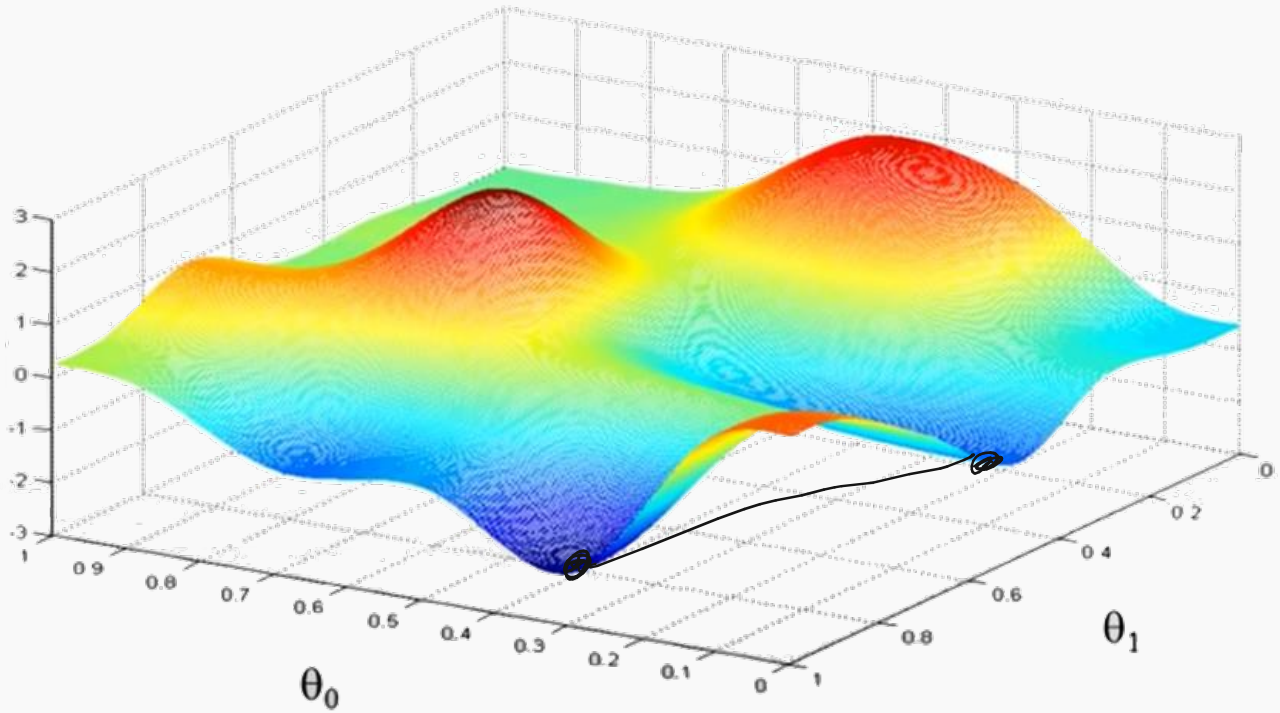
**In words:** A function is convex if a line between any two points on the function lies above the function. Captures the notion that a function looks like a bowl.




This function **is** convex.

# CONVEX FUNCTION

**In words:** A function is convex if a line between any two points on the function lies above the function. Captures the notion that a function looks like a bowl.




This function **is** not convex.

## What functions are convex?

- Least squares loss for linear regression.
- $\ell_1$ loss for linear regression.
- Either of these with and $\ell_1$ or $\ell_2$ regularization penalty.
- Logistic regression! Logistic regression with regularization.
- Many other models in machine leaning.

# CONVEXITY OF LEAST SQUARES REGRESSION LOSS

See notes from last week on proof that $L(\boldsymbol{\beta}) = \|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|_2^2$ is convex. ~~For now just consider $\lambda = \frac{1}{2}$ case.~~

Simpler problem: prove that $L(\beta) = \beta^2$ is convex.

$$\left((1-\lambda)\beta_1 + \lambda\beta_2\right)^2 \leq (1-\lambda)\beta_1^2 + \lambda\beta_2^2$$

# CONVERGENCE ANALYSIS FOR CONVEX FUNCTIONS

**Assume:**

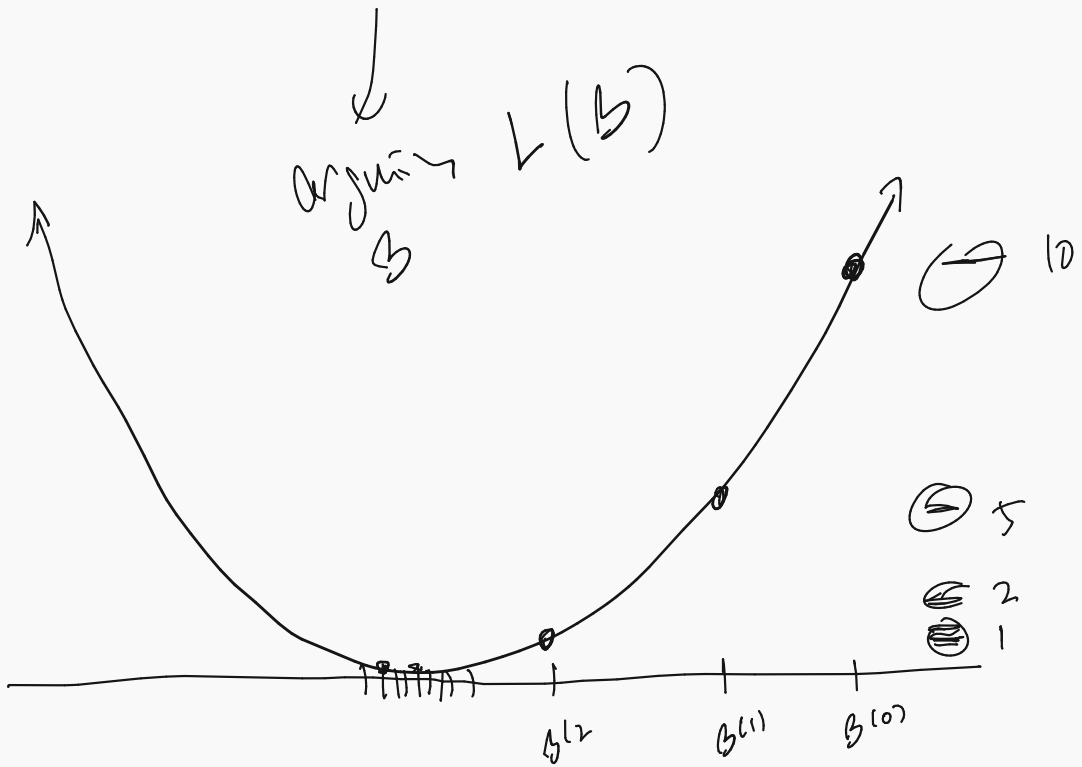
- $L$ is convex.
- Lipschitz function: for all $\boldsymbol{\beta}$, $\|\nabla L(\boldsymbol{\beta})\|_2 \leq G$.
- Starting radius: $\|\boldsymbol{\beta}^* - \boldsymbol{\beta}^{(0)}\|_2 \leq R$.

**Gradient descent:**

- Choose number of steps $T$.
- Starting point $\boldsymbol{\beta}^{(0)}$. E.g. $\boldsymbol{\beta}^{(0)} = \mathbf{0}$.
- $\eta = \frac{R}{G\sqrt{T}}$
- For $i = 0, \ldots, T$:
  - $\boldsymbol{\beta}^{(i+1)} = \boldsymbol{\beta}^{(i)} - \eta \nabla L(\boldsymbol{\beta}^{(i)})$
- Return $\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}^{(i)}} L(\boldsymbol{\beta})$.

**Claim (GD Convergence Bound)**

*If* $T \geq \frac{R^2 G^2}{\epsilon^2}$, *then* $L(\hat{\beta}) \leq L(\beta^*) + \epsilon$.



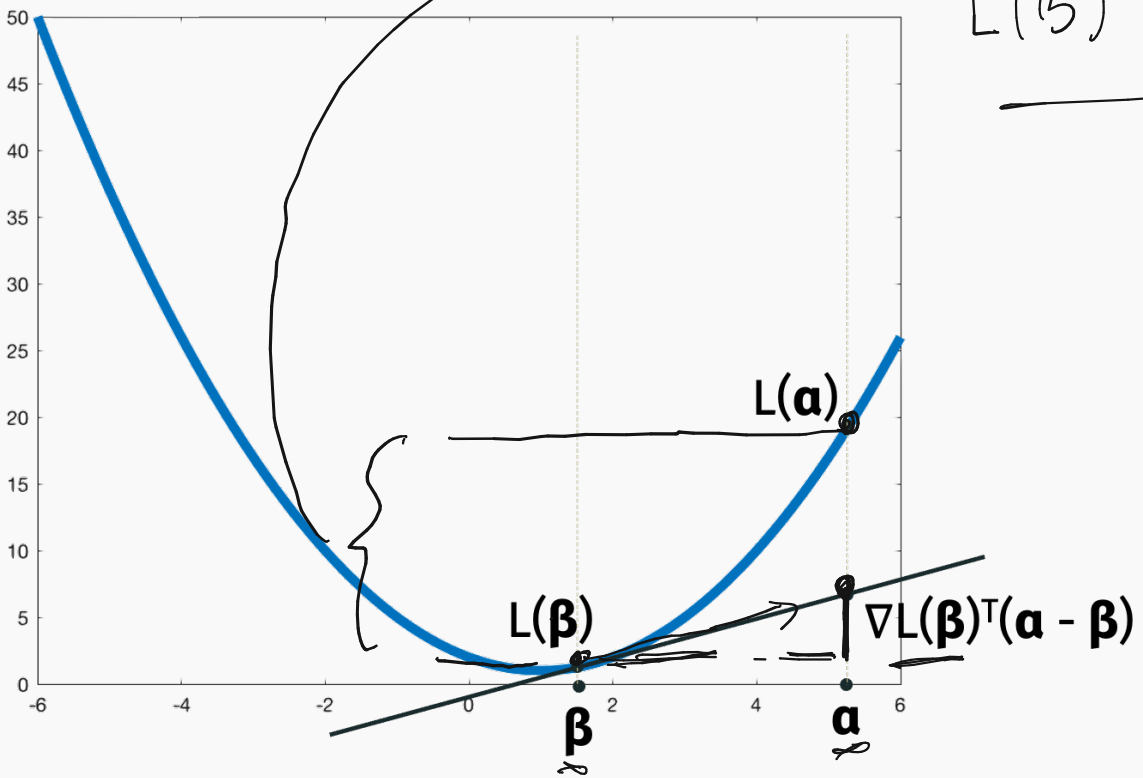

Proof is made tricky by the fact that $L(\beta^{(i)})$ does not improve monotonically. We can "overshoot" the minimum. This is why the step size needs to depend on $1/G$.

## Definition (Alternative Convexity Definition)

A function $L$ is convex if and only if for any $\beta, \alpha$:

$$L(\alpha) - L(\beta) \geq \nabla L(\beta)^T (\alpha - \beta)$$

$L(\beta) - L(\alpha) \leq \nabla L(\beta)^T (\beta - \alpha)$



## Claim (GD Convergence Bound)

*If* $T \geq \frac{R^2 G^2}{\epsilon^2}$ *and* $\eta = \frac{R}{G\sqrt{T}}$, *then* $L(\hat{\boldsymbol{\beta}}) \leq L(\boldsymbol{\beta}^*) + \epsilon$.

**Claim 1:** For all $i = 0, \ldots, T$,

$$L(\boldsymbol{\beta}^{(i)}) - L(\boldsymbol{\beta}^*) \leq \frac{\|\boldsymbol{\beta}^{(i)} - \boldsymbol{\beta}^*\|_2^2 - \|\boldsymbol{\beta}^{(i+1)} - \boldsymbol{\beta}^*\|_2^2}{2\eta} + \frac{\eta G^2}{2}$$

**Claim 1(a):** For all $i = 0, \ldots, T$,

$$\nabla L(\boldsymbol{\beta}^{(i)})^T (\boldsymbol{\beta}^{(i)} - \boldsymbol{\beta}^*) \leq \frac{\|\boldsymbol{\beta}^{(i)} - \boldsymbol{\beta}^*\|_2^2 - \|\boldsymbol{\beta}^{(i+1)} - \boldsymbol{\beta}^*\|_2^2}{2\eta} + \frac{\eta G^2}{2}$$

Claim 1 follows from Claim 1(a) by our new definition of convexity.

$$L(\beta^{(i)}) - L(\beta^*) \leq \nabla L(\beta^{(i)})^T (\beta^{(i)} - \beta^*)$$

$\nabla L(\beta^{(i)})$ ↓ $d$ dimensional vector

## Claim (GD Convergence Bound)

*If* $T \geq \frac{R^2 G^2}{\epsilon^2}$ *and* $\eta = \frac{R}{G\sqrt{T}}$, *then* $L(\hat{\beta}) \leq L(\beta^*) + \epsilon$.

**Claim 1(a):** For all $i = 0, \ldots, T$, [1]

$$\nabla L(\beta^{(i)})^T (\beta^{(i)} - \beta^*) \leq \frac{\|\beta^{(i)} - \beta^*\|_2^2 - \|\beta^{(i+1)} - \beta^*\|_2^2}{2\eta} + \frac{\eta G^2}{2}$$

$$\|\beta^{(i+1)} - \beta^*\|_2^2 = \|\beta^{(i)} - \eta \nabla L(\beta^{(i)}) - \beta^*\|_2^2 = \|(\beta^{(i)} - \beta^*) - \eta \nabla L(\beta^{(i)})\|_2^2$$

$$\|\beta^{(i+1)} - \beta^*\|_2^2 = \|\beta^{(i)} - \beta^*\|_2^2 - 2\eta \nabla L(\beta^{(i)})^T (\beta^{(i)} - \beta^*) + \|\eta \nabla L(\beta^{(i)})\|_2^2$$

$$\eta^2 \|\nabla L(\beta^{(i)})\|_2^2 \leq \eta^2 \cdot G^2$$

$$2\eta \nabla L(\beta^{(i)})^T (\beta^{(i)} - \beta^*) \leq \|\beta^{(i)} - \beta^*\|_2^2 - \|\beta^{(i+1)} - \beta^*\|_2^2 + \eta^2 G^2$$

[1] Recall that $\|\mathbf{x} - \mathbf{y}\|_2^2 = \|\mathbf{x}\|_2^2 - 2\mathbf{x}^T\mathbf{y} + \|\mathbf{y}\|_2^2$.

$(x-y)^T(x-y) = x^Tx - x^Ty - y^Tx + y^Ty$

## Claim (GD Convergence Bound)

*If* $T \geq \frac{R^2 G^2}{\epsilon^2}$ *and* $\eta = \frac{R}{G\sqrt{T}}$, *then* $L(\hat{\boldsymbol{\beta}}) \leq L(\boldsymbol{\beta}^*) + \epsilon$.

**Claim 1:** For all $i = 0, \ldots, T$,

$$L(\boldsymbol{\beta}^{(i)}) - L(\boldsymbol{\beta}^*) \leq \frac{\|\boldsymbol{\beta}^{(i)} - \boldsymbol{\beta}^*\|_2^2 - \|\boldsymbol{\beta}^{(i+1)} - \boldsymbol{\beta}^*\|_2^2}{2\eta} + \frac{\eta G^2}{2}$$

**Telescoping sum:**

$$\sum_{i=0}^{T-1} \left[ L(\boldsymbol{\beta}^{(i)}) - L(\boldsymbol{\beta}^*) \right] \leq \frac{\|\boldsymbol{\beta}^{(0)} - \boldsymbol{\beta}^*\|_2^2 - \|\boldsymbol{\beta}^{(1)} - \boldsymbol{\beta}^*\|_2^2}{2\eta} + \frac{\eta G^2}{2}$$

$$+ \frac{\|\boldsymbol{\beta}^{(1)} - \boldsymbol{\beta}^*\|_2^2 - \|\boldsymbol{\beta}^{(2)} - \boldsymbol{\beta}^*\|_2^2}{2\eta} + \frac{\eta G^2}{2}$$

$$+ \frac{\|\boldsymbol{\beta}^{(2)} - \boldsymbol{\beta}^*\|_2^2 - \|\boldsymbol{\beta}^{(3)} - \boldsymbol{\beta}^*\|_2^2}{2\eta} + \frac{\eta G^2}{2}$$

$$\vdots$$

$$+ \frac{\|\boldsymbol{\beta}^{(T-1)} - \boldsymbol{\beta}^*\|_2^2 - \|\boldsymbol{\beta}^{(T)} - \boldsymbol{\beta}^*\|_2^2}{2\eta} + \frac{\eta G^2}{2}$$

## Claim (GD Convergence Bound)

*If* $T \geq \frac{R^2 G^2}{\epsilon^2}$ *and* $\eta = \frac{R}{G\sqrt{T}}$, *then* $L(\hat{\boldsymbol{\beta}}) \leq L(\boldsymbol{\beta}^*) + \epsilon$.

Telescoping sum:

$$\sum_{i=0}^{T-1} \left[ L(\boldsymbol{\beta}^{(i)}) - L(\boldsymbol{\beta}^*) \right] \leq \frac{\|\boldsymbol{\beta}^{(0)} - \boldsymbol{\beta}^*\|_2^2 - \|\boldsymbol{\beta}^{(T)} - \boldsymbol{\beta}^*\|_2^2}{2\eta} + \frac{T\eta G^2}{2}$$

$$\frac{1}{T} \sum_{i=0}^{T-1} \left[ L(\boldsymbol{\beta}^{(i)}) - L(\boldsymbol{\beta}^*) \right] \leq \frac{R^2}{2T\eta} + \frac{\eta G^2}{2} \leq \epsilon$$

$\leq B^2$

$\leq \frac{B^2}{2\eta} + \frac{T\eta G^2}{2}$

$= \frac{B^2}{2T \, B/G\sqrt{T}} + \frac{\frac{B}{G\sqrt{T}} G^2}{2} = \frac{BG}{2\sqrt{2}} + \frac{BG}{2\sqrt{T}}$

$= \frac{BG}{\sqrt{T}} \leq \epsilon$

**Claim (GD Convergence Bound)**

*If* $T \geq \frac{R^2 G^2}{\epsilon^2}$ *and* $\eta = \frac{R}{G\sqrt{T}}$, *then* $L(\hat{\beta}) \leq L(\beta^*) + \epsilon$.

$\frac{1}{T}\sum_{i=0}^{T-1} L(\beta^{(i)}) \geq \frac{1}{T}\sum_{i=0}^{T-1} \min_{j \in 0,\ldots,T-1} L(\beta^{(j)})$

$= \min_j L(\beta^{(j)})$

**Final step:**

$$\frac{1}{T}\sum_{i=0}^{T-1}\left[L(\beta^{(i)}) - L(\beta^*)\right] \leq \epsilon$$

$$\left[\frac{1}{T}\sum_{i=0}^{T-1} L(\beta^{(i)})\right] - L(\beta^*) \leq \epsilon$$

We always have that $\min_i L(\beta^{(i)}) \leq \frac{1}{T}\sum_{i=0}^{T-1} L(\beta^{(i)})$, so this is what we return:

$$L(\hat{\beta}) = \min_{i \in 1,\ldots,T} L(\beta^{(i)}) \leq L(\beta^*) + \epsilon.$$

# Setting learning rate/step size

**Gradient descent algorithm for minimizing $L(\boldsymbol{\beta})$:**

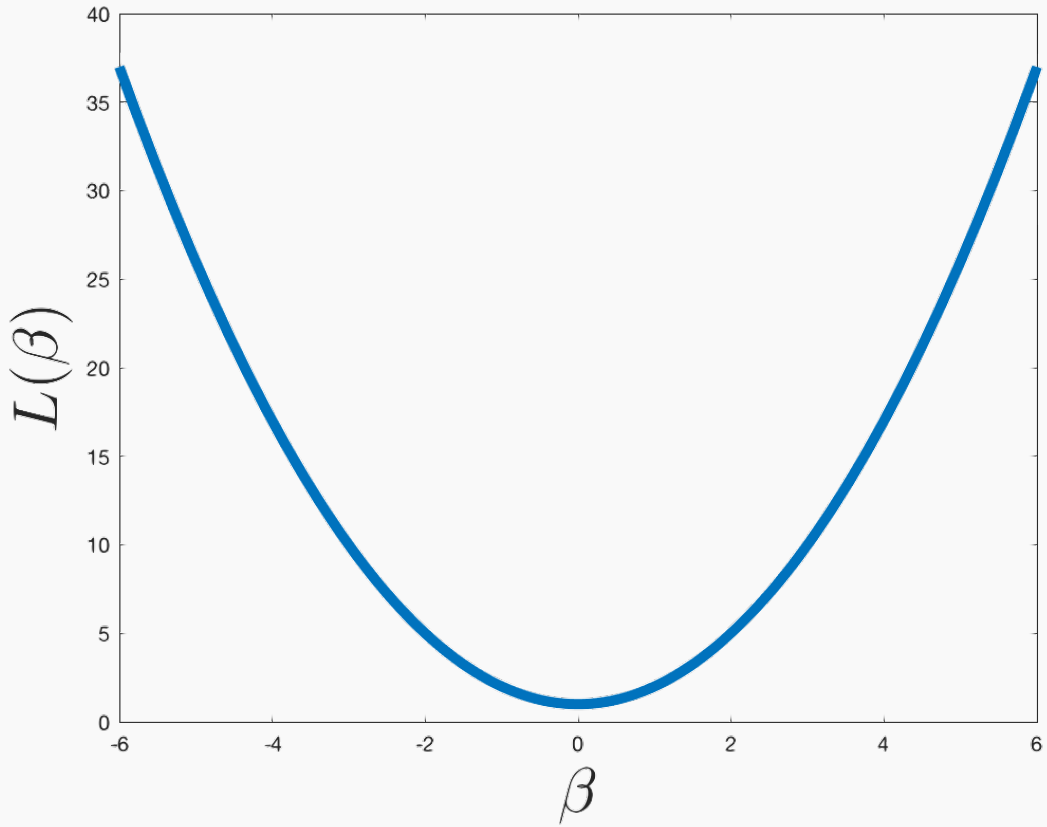
- Choose arbitrary starting point $\boldsymbol{\beta}^{(0)}$.
- For $i = 1, \ldots, T$:
  - $\boldsymbol{\beta}^{(i+1)} = \boldsymbol{\beta}^{(i)} - \eta \nabla L(\boldsymbol{\beta}^{(i)})$
- Return $\boldsymbol{\beta}^{(t)}$.

In practice we don't set the step-size/learning rate parameter $\eta = \frac{R}{G\sqrt{T}}$, since we typically don't know these parameters. The above analysis can also be loose for many functions.

$\eta$ needs to be chosen sufficiently small for gradient descent to converge, but too small will slow down the algorithm.

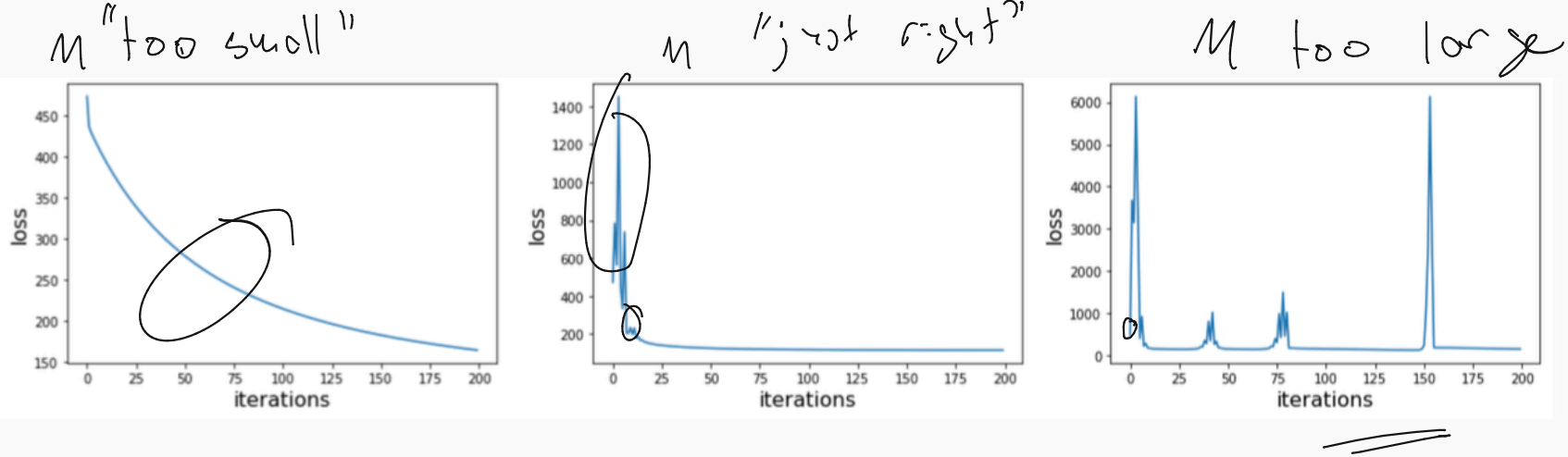
# LEARNING RATE

Precision in choosing the learning rate $\eta$ is not super important, but we do need to get it to the right order of magnitude.



# LEARNING RATE

"Overshooting" can be a problem if you choose the step-size too high.




Often a good idea to plot the entire optimization curve for diagnosing what's going on.

We will have a lab on gradient descent optimization after the midterm we're you'll get practice doing this.

# EXPONENTIAL GRID SEARCH

Just as in regularization, search over a grid of possible parameters:

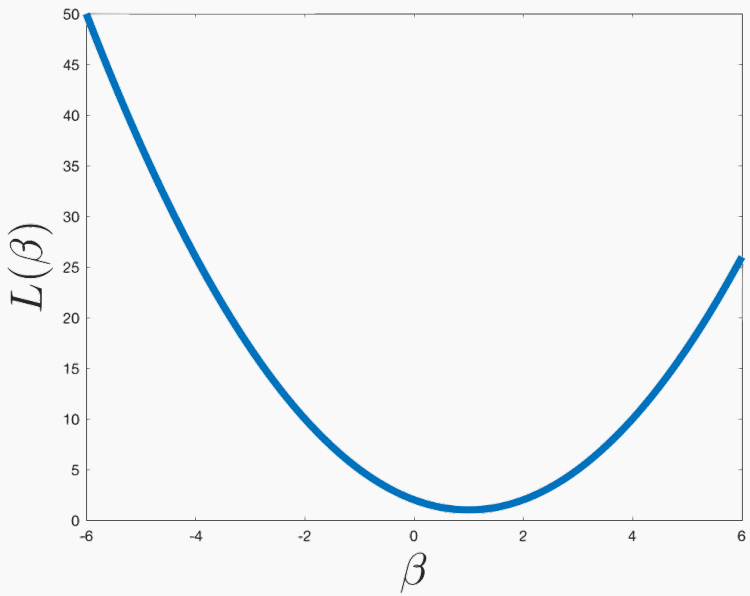
$$\eta = [2^{-5}, 2^{-4}, 2^{-3}, \ldots, 2^{9}, 2^{10}].$$

Or tune by hand based on the optimization curve.

# BACKTRACKING LINE SEARCH/ARMIJO RULE

**Recall**: If we set $\beta^{(i+1)} \leftarrow \beta^{(i)} - \eta \nabla L(\beta^{(i)})$ then:

$$L(\beta^{(i+1)}) \approx L(\beta^{(i)}) - \eta \left\langle \nabla L(\beta^{(i)}), \nabla L(\beta^{(i)}) \right\rangle$$
$$= L(\beta^{(i)}) - \eta \|\nabla L(\beta^{(i)})\|_2^2.$$




Approximation holds true for small $\eta$. If it holds, error monotonically decreases.

# Backtracking line search/Armijo rule

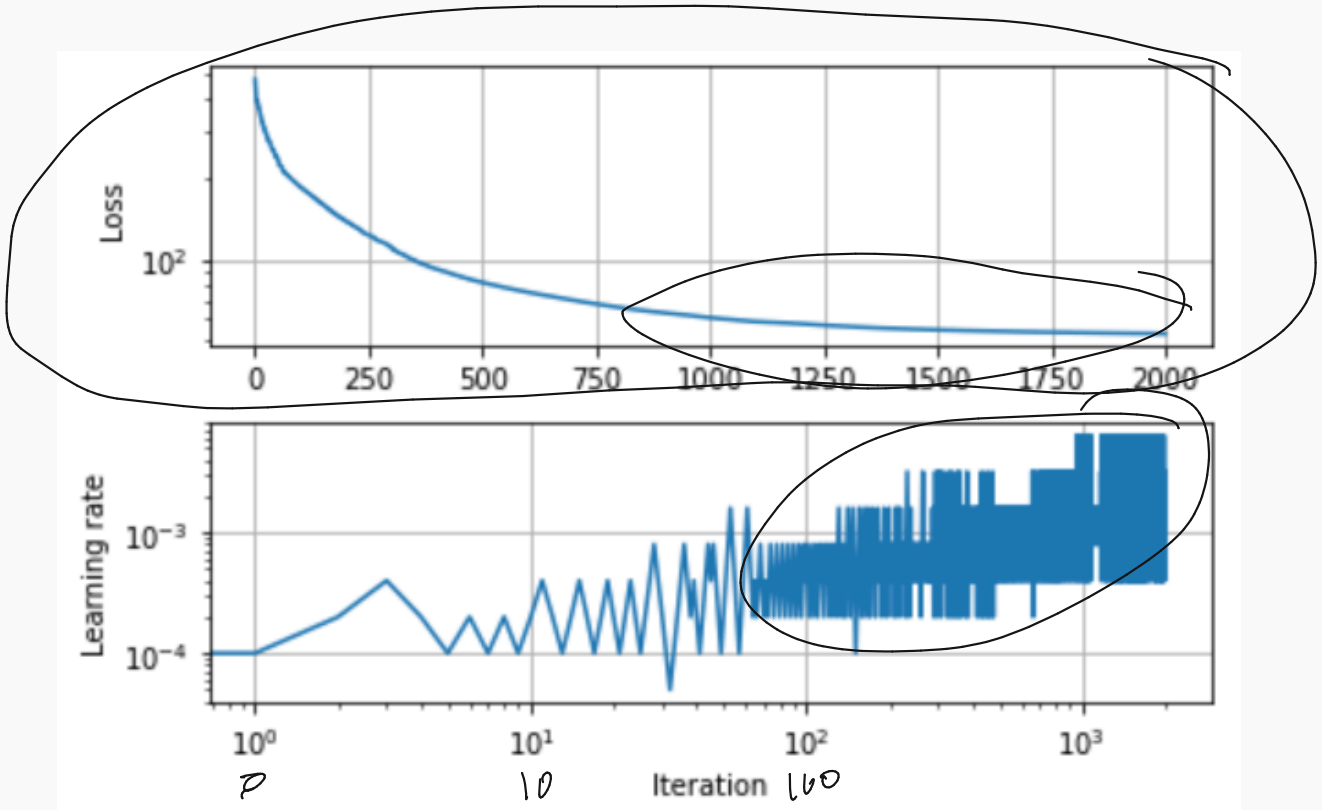
Gradient descent with backtracking line search:

- Choose arbitrary starting point $\beta$.
- Choose starting step size $\eta$.
- Choose $\tau, c < 1$ (typically both $c = 1/2$ and $\tau = 1/2$)
- For $i = 1, \ldots, T$:
  - $\beta^{(new)} = \beta - \eta \nabla L(\beta)$
  - If $L(\beta^{(new)}) \leq L(\beta) - c\eta \|\nabla L(\beta)\|_2^2$
    - $\beta \leftarrow \beta^{(new)}$
    - $\eta \leftarrow \tau^{-1}\eta$
  - Else
    - $\eta \leftarrow \tau\eta$

Always decreases objective value, works very well in practice.

# BACKTRACKING LINE SEARCH/ARMIJO RULE

Gradient descent with backtracking line search:



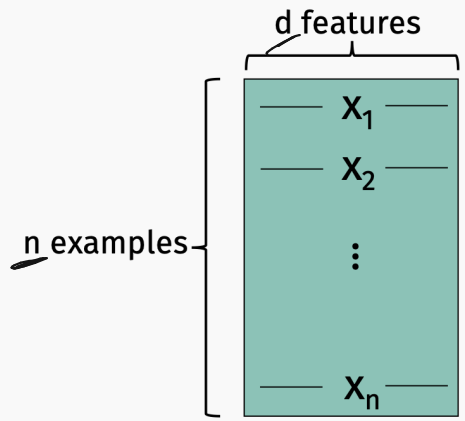

Always decreases objective value, works very well in practice.

# COMPLEXITY OF GRADIENT DESCENT

Complexity of computing the gradient will depend on you loss function.

**Example 1:** Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ be a data matrix.

$$L(\boldsymbol{\beta}) = \|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|_2^2 \qquad \nabla L(\boldsymbol{\beta}) = 2\mathbf{X}^T(\mathbf{X}\boldsymbol{\beta} - \mathbf{y}) = 0$$




- Runtime of closed form solution $\boldsymbol{\beta}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$:
- Runtime of one GD step:

Complexity of computing the gradient will depend on you loss function.

**Example 1:** Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ be a data matrix.

$$L(\boldsymbol{\beta}) = -\sum_{i=1}^{n} y_i \log(h(\boldsymbol{\beta}^T \mathbf{x}_i)) + (1 - y_i) \log(1 - h(\boldsymbol{\beta}^T \mathbf{x}_i))$$

$$\nabla L(\boldsymbol{\beta}) = \mathbf{X}^T (h(\mathbf{X}\boldsymbol{\beta}) - \mathbf{y})$$

- No closed form solution.
- Runtime of one GD step: $O(nd)$

# COMPLEXITY OF GRADIENT DESCENT

Frequently the complexity is $O(nd)$ if you have $n$ data-points and $d$ parameters in your model.

Not bad, but the dependence on $n$ can be a lot! $n$ might be on the order of thousands, or millions.

Stochastic Gradient Descent (SGD).

- Powerful randomized variant of gradient descent used to train machine learning models when $n$ is large and thus computing a full gradient is expensive.

**Applies to any loss with finite sum structure:**

$$L(\beta) = \sum_{j=1}^{n} \ell(\beta, x_j, y_j) = \sum_{j=1}^{n} L_j(\beta)$$

$$= \sum_{j=1}^{n} (x_j^T \beta - y_j)^2$$

# STOCHASTIC GRADIENT DESCENT

Let $L_j(\boldsymbol{\beta})$ denote $\ell(\boldsymbol{\beta}, \mathbf{x}_j, y_j)$.

$f = g + h$

$f' = g' + h'$

**Claim:** If $j \in 1, \ldots, n$ is chosen uniformly at random. Then:

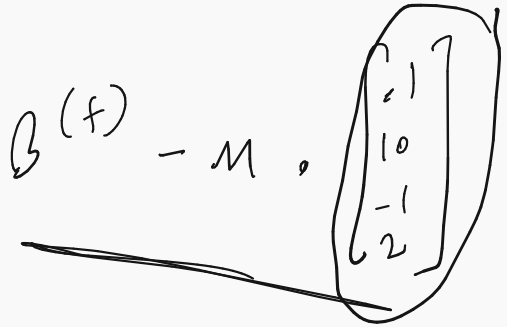
$$\mathbb{E}\left[n \cdot \nabla L_j(\boldsymbol{\beta})\right] = \nabla L(\boldsymbol{\beta}).$$

$$= \sum_{j=1}^{n} \Pr[\text{choose } j] \cdot n \nabla L_j(\beta)$$

$$= \sum_{j=1}^{n} \frac{1}{n} \cdot n \nabla L_j(\beta) = \sum_{j=1}^{n} \nabla L_j(\beta) = \nabla\left[\sum_{j=1}^{n} L_j(\beta)\right]$$

linearity of gradient

$\nabla L_j(\boldsymbol{\beta})$ is called a stochastic gradient.



SGD iteration:

- Initialize $\beta^{(0)}$.
- For $i = 0, \ldots, T-1$:
  - Choose $j$ uniformly at random.
  - Compute stochastic gradient $\mathbf{g} = \nabla L_j(\beta^{(i)})$.
  - Update $\beta^{(t+1)} = \beta^{(t)} - \eta \cdot n\mathbf{g}$

Move in direction of steepest descent in expectation.

Cost of computing $\mathbf{g}$ is independent of $n$!

$O(d)$

**Example:** Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ be a data matrix.

$$L(\boldsymbol{\beta}) = \|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|_2^2 = \sum_{j=1}^{n} (y_j - \boldsymbol{\beta}^T \mathbf{x}_j)^2$$

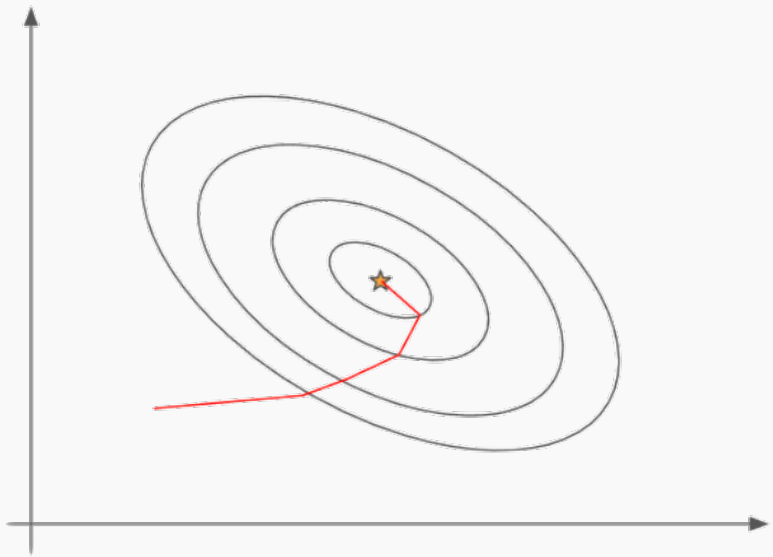
$L(\beta) = \sum_{j=1}^{n} L_j(\beta)$ where $L_j(\beta) = (y_j - \beta^T x_j)^2$ $O(d)$

$$\nabla L_j(\beta) = \begin{bmatrix} \partial L_j/\partial \beta_1 \\ \vdots \\ \partial L_j/\partial \beta_d \end{bmatrix} = \begin{bmatrix} 2(y_j - \beta^T x_j) \cdot (-1) \cdot x_j[1] \\ \vdots \\ 2(y_j - \beta^T x_j) \cdot (-1) \cdot x_j[d] \end{bmatrix} \quad + O(1) \cdot d$$

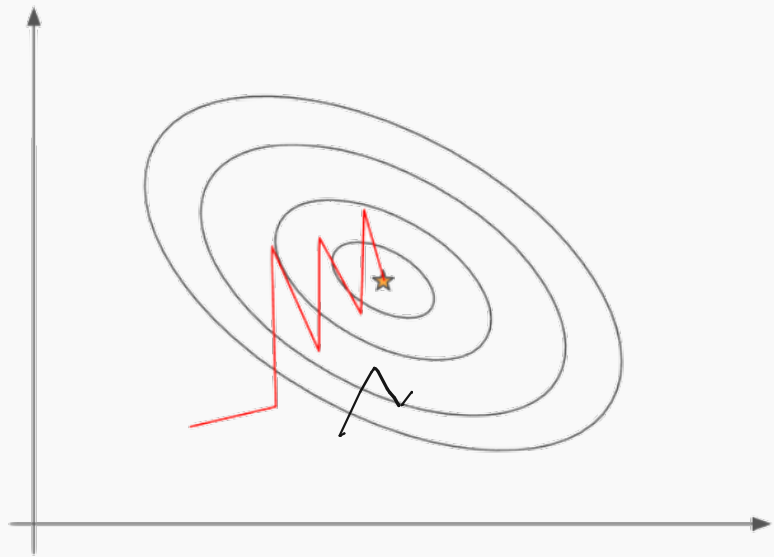
- Runtime of one SGD step: $O(d)$

**Gradient descent:** Fewer iterations to converge, higher cost per iteration.

**Stochastic Gradient descent:** More iterations to converge, lower cost per iteration.



Gradient Descent

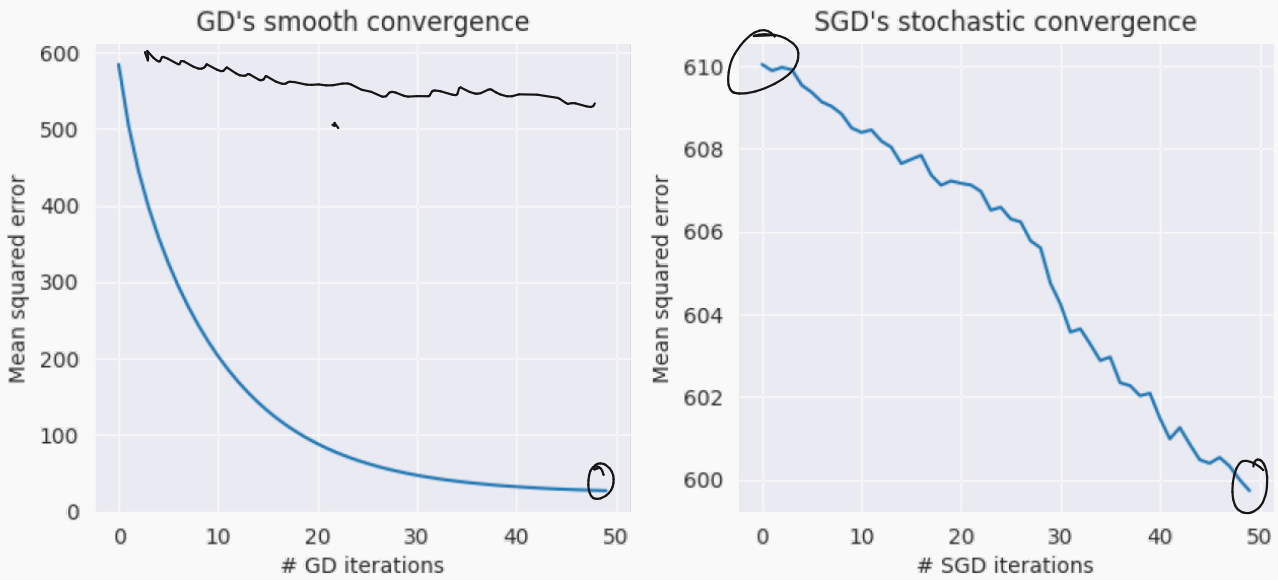


Stochastic Gradient Descent

# STOCHASTIC GRADIENT DESCENT

**Gradient descent:** Fewer iterations to converge, higher cost per iteration.

**Stochastic Gradient descent:** More iterations to converge, lower cost per iteration.



Typical implementation: **Shuffled Gradient Descent.**

Instead of choosing $j$ independently at random for each iteration, randomly permute (shuffle) data and set $j = 1, \ldots, n$. After every $n$ iterations, reshuffle data and repeat.

- Relatively similar convergence behavior to standard SGD.
- **Important term:** one epoch denotes one pass over all training examples: $j = 1, \ldots, j = n$.
- Convergence rates for training ML models are often discussed in terms of epochs instead of iterations.

# STOCHASTIC GRADIENT DESCENT IN PRACTICE

Practical Modification: **Mini-batch Gradient Descent.**

Observe that for any batch size $s$,

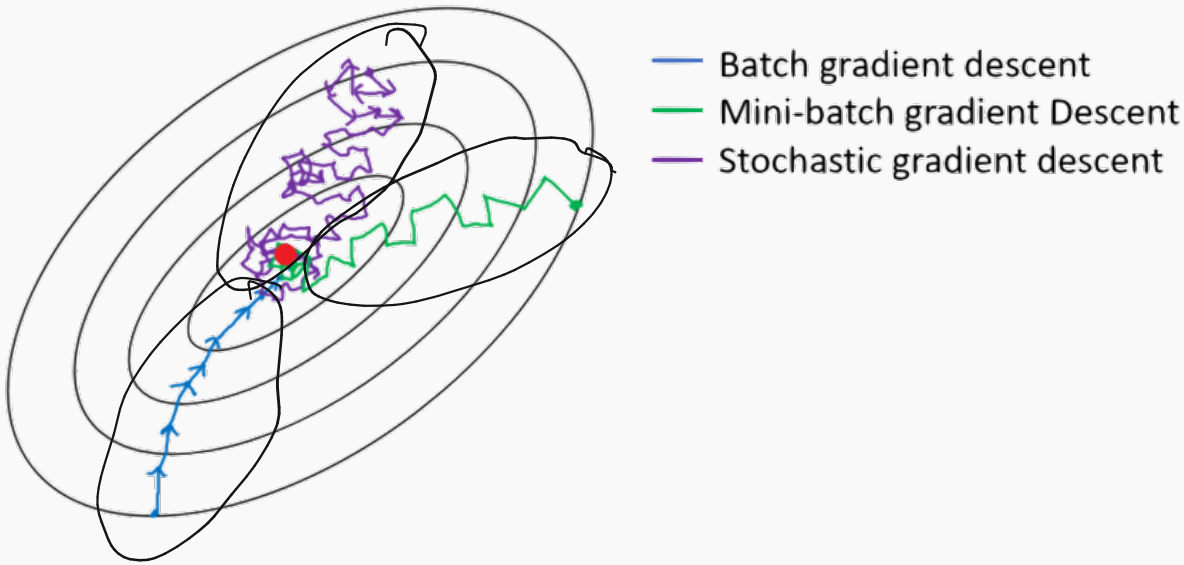
$$\mathbb{E}\left[\frac{n}{s}\sum_{i=1}^{s}\nabla L_{j_i}(\boldsymbol{\beta})\right] = \nabla L(\boldsymbol{\beta}).$$

if $j_1, \ldots, j_s$ are chosen independently and uniformly at random from $1, \ldots, n$.

Instead of computing a full stochastic gradient, compute the average gradient of a small random set (a mini-batch) of training data examples.

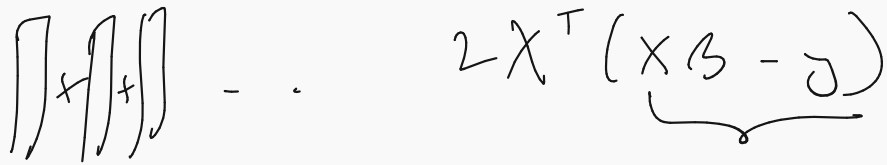
**Question:** Why might we want to do this?

# MINI-BATCH GRADIENT DESCENT

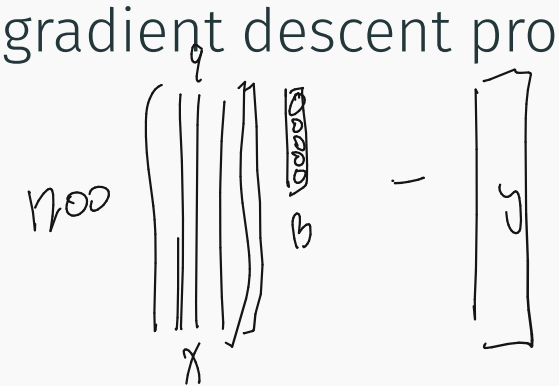


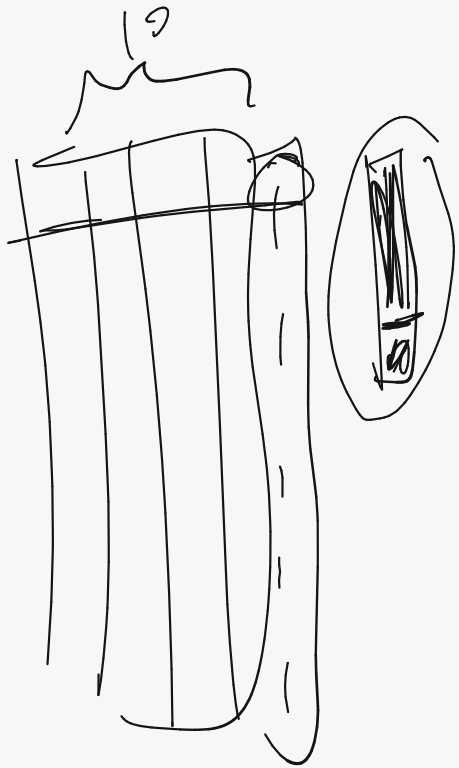

- Overall faster convergence (fewer iterations needed).

# MIDTERM



- 1 hour long, here in the classroom. We will have lecture after.
- You can bring in a single, 2-sided cheat sheet with terms, definitions, etc.
- Mix of short answer questions (true/false, matching, etc.) and questions similar to the homework but easier.
- Might need to write some easy pseudocode.
- Covers everything through today. Don't need to know gradient descent proof of convergence.






$$\begin{bmatrix} x_j & , 1 \end{bmatrix}^T \begin{bmatrix} w \\ b \end{bmatrix}$$