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

NYU Tandon School of Engineering, Prof. Christopher Musco

Goal: Minimize generic <u>differentiable</u> loss function:

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

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

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

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

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

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

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

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

Basic Gradient descent algorithm:

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

η is the step-size parameter or learning rate.

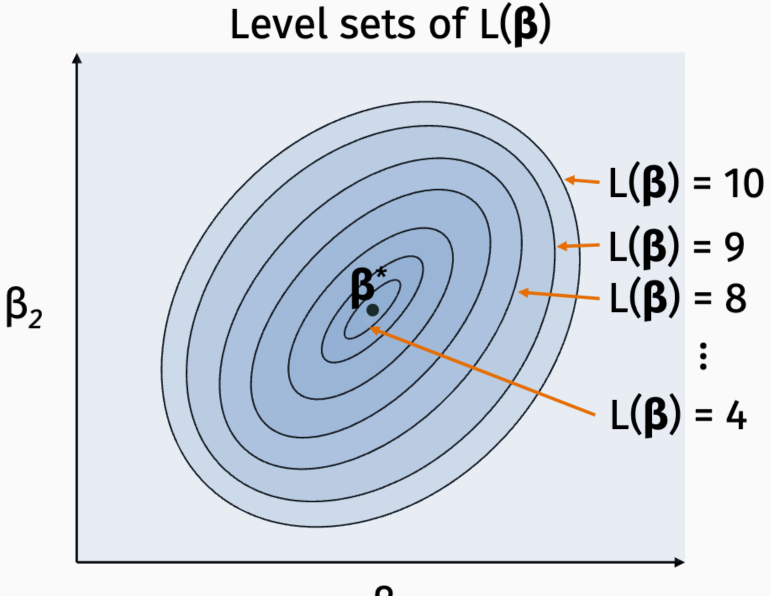
We came to an important observations:

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

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

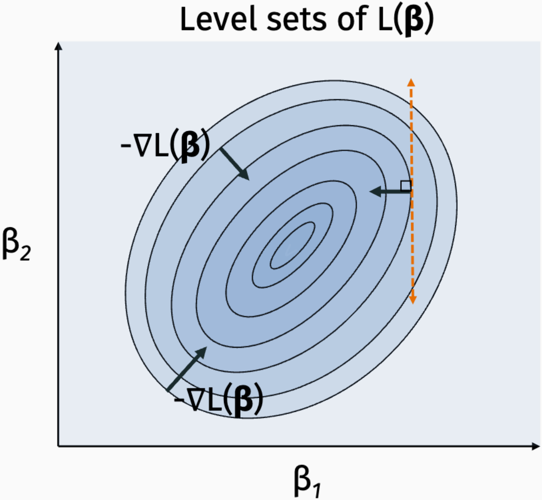
Conclusion: Gradient descent always converges to a local minimum or stationary point of L. Typically to a local minimum.

VISUALIZING IN 2D



Claim (Gradient descent = Steepest descent)

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

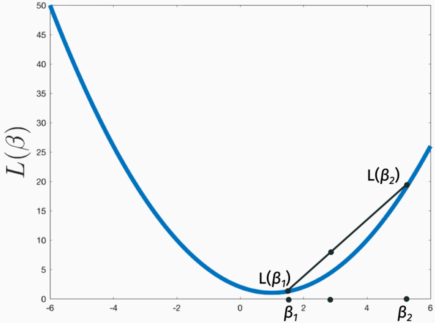


For a broad class of functions, GD converges to global minima.

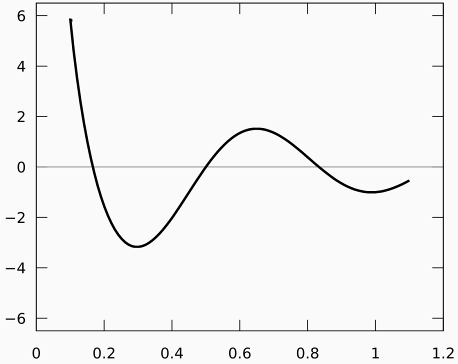
Definition (Convex)

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

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

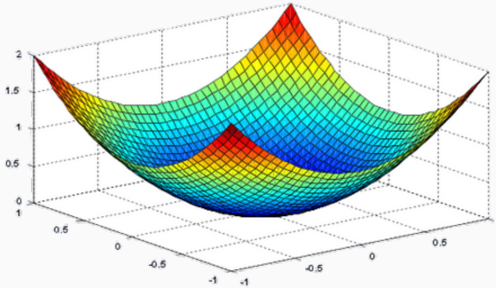


In words: A function is convex if a line between any two points on the function lies above the function. Captures the notion that a function looks like a bowl.



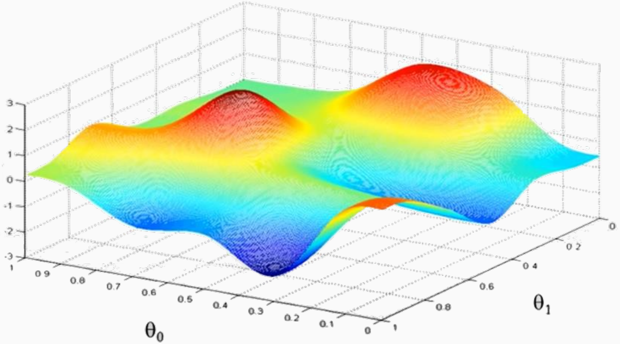
This function **is not** convex.

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.

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.

What functions are convex?

- Least squares loss for linear regression.
- ℓ_1 loss for linear regression.
- Either of these with and ℓ_1 or ℓ_2 regularization penalty.
- Logistic regression! Logistic regression with regularization.
- Many other models in machine leaning.

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.

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)

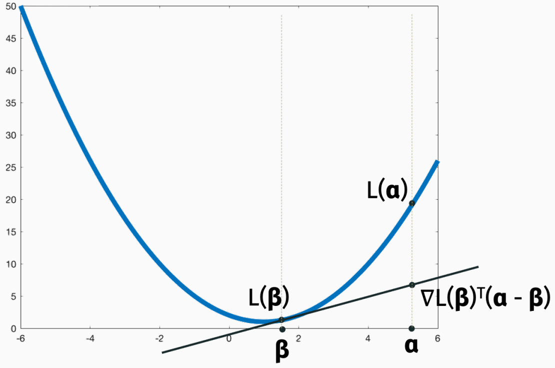
If $T \geq \frac{R^2G^2}{\epsilon^2}$, *then* $L(\hat{\beta}) \leq L(\beta^*) + \epsilon$.

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 $\boldsymbol{\beta}, \boldsymbol{\alpha}$:

$$f(\boldsymbol{\alpha}) - f(\boldsymbol{\beta}) \leq \nabla f(\boldsymbol{\beta})^T(\boldsymbol{\alpha} - \boldsymbol{\beta})$$



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.

Claim (GD Convergence Bound)

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

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

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

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

Claim (GD Convergence Bound)

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

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:

$$\begin{aligned}
\sum_{i=0}^{T-1} \left[L(\boldsymbol{\beta}^{(i)}) - L(\boldsymbol{\beta}^*)\right] \leq\ & \frac{\|\boldsymbol{\beta}^{(0)} - \boldsymbol{\beta}^*\|_2^2 - \|\boldsymbol{\beta}^{(1)} - \boldsymbol{\beta}^*\|_2^2}{2\eta} + \frac{\eta G^2}{2} \\
& + \frac{\|\boldsymbol{\beta}^{(1)} - \boldsymbol{\beta}^*\|_2^2 - \|\boldsymbol{\beta}^{(2)} - \boldsymbol{\beta}^*\|_2^2}{2\eta} + \frac{\eta G^2}{2} \\
& + \frac{\|\boldsymbol{\beta}^{(2)} - \boldsymbol{\beta}^*\|_2^2 - \|\boldsymbol{\beta}^{(3)} - \boldsymbol{\beta}^*\|_2^2}{2\eta} + \frac{\eta G^2}{2} \\
& \vdots \\
& + \frac{\|\boldsymbol{\beta}^{(T-1)} - \boldsymbol{\beta}^*\|_2^2 - \|\boldsymbol{\beta}^{(T)} - \boldsymbol{\beta}^*\|_2^2}{2\eta} + \frac{\eta G^2}{2}
\end{aligned}$$

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:

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

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

Claim (GD Convergence Bound)

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

Final step:

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

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

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

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

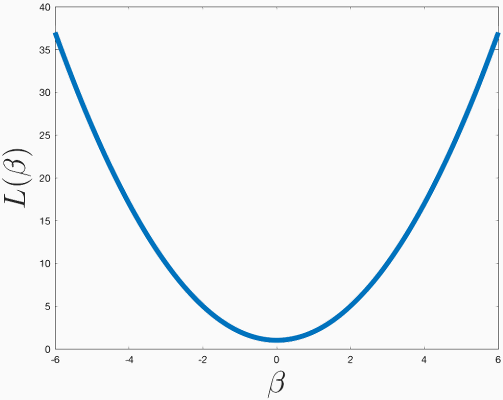
Gradient descent algorithm for minimizing $L(\boldsymbol{\beta})$:

- 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 $\boldsymbol{\beta}^{(t)}$.

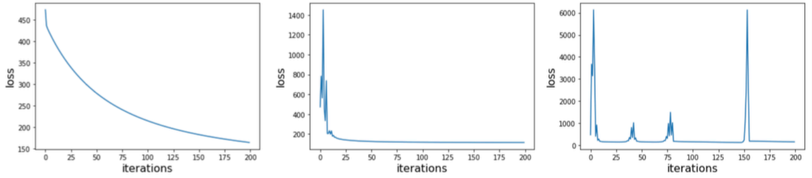
In practice we don't set the step-size/learning rate parameter $\eta = \frac{R}{G\sqrt{T}}$, since we typically don't know these parameters. The above analysis can also be loose for many functions.

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

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



"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.

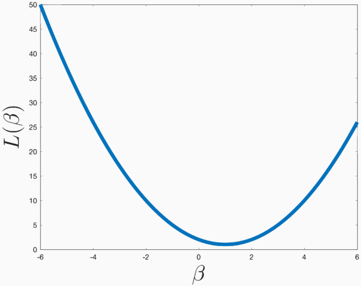
Just as in regularization, search over a grid of possible parameters:

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

Or tune by hand based on the optimization curve.

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 η. If it holds, error monotonically decreases.

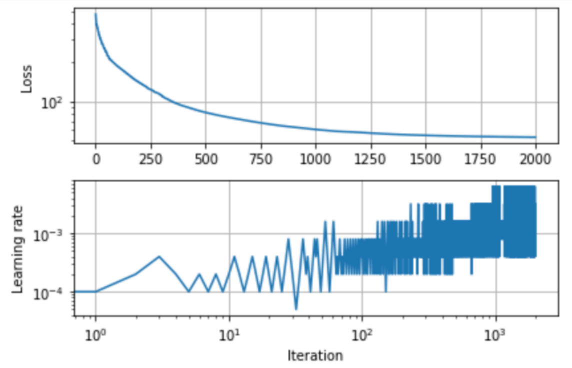
BACKTRACKING LINE SEARCH/ARMIJO RULE

Gradient descent with backtracking line search:

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

Always decreases objective value, works very well in practice.

Gradient descent with backtracking line search:

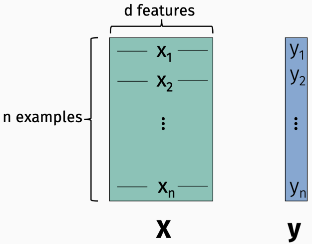


Always decreases objective value, works very well in practice.

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.

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



- **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.

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

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

- No closed form solution.
- Runtime of one GD step:

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:

$$L(\boldsymbol{\beta}) = \sum_{j=1}^{n} \ell(\boldsymbol{\beta}, \mathbf{x}_j, y_j)$$

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

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

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

$\nabla L_j(\boldsymbol{\beta})$ 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 g is independent of n!

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

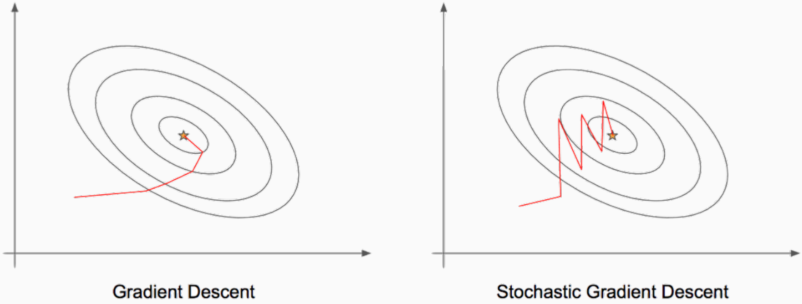
$$L(\boldsymbol{\beta}) = \|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|_2^2 = \sum_{j=1}^{n} (y_j - \boldsymbol{\beta}^T \mathbf{x}_j)^2$$

- Runtime of one SGD step:

STOCHASTIC GRADIENT DESCENT

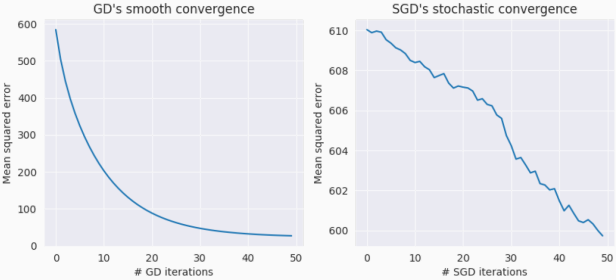
Gradient descent: Fewer iterations to converge, higher cost per iteration.

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



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.

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

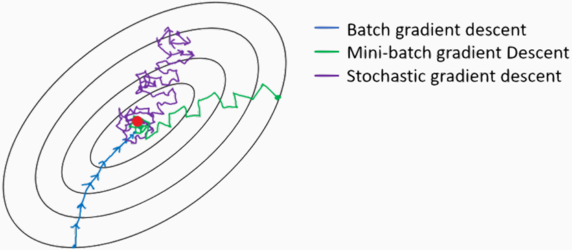
Observe that for any batch size s,

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

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?



- 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.