ESE 402/542 Recitation 8: Gradient Descent

Outline

1. Starting in 1-D
2. Introducing gradient descent
3. When is gradient descent guaranteed to work?
4. Examples in data science

Starting in 1-D

- Consider function $f(x) = x^2$. Given an arbitrary starting point x_0, want to be able to find the minimum (let's say we didn't know it was at $x = 0$).

Starting in 1-D

- Consider function $f(x) = x^2$. Given an arbitrary starting point x_0, want to be able to find the minimum (let's say we didn't know it was at $x = 0$).
- Propose algorithm: for $t = 0, \dots, N$, $x_{t+1} = x_t - \eta f'(x_t)$. $\eta > 0$ is step-size hyperparameter.

Starting in 1-D

- Consider function $f(x) = x^2$. Given an arbitrary starting point x_0, want to be able to find the minimum (let's say we didn't know it was at $x = 0$).
- Propose algorithm: for $t = 0, \ldots, N$, $x_{t+1} = x_t - \eta f'(x_t)$. $\eta > 0$ is step-size hyperparameter.
- Intuition: derivative shows rate of *growth* at a point. Algorithm is subtracting derivative from current point–go toward direction of decay.

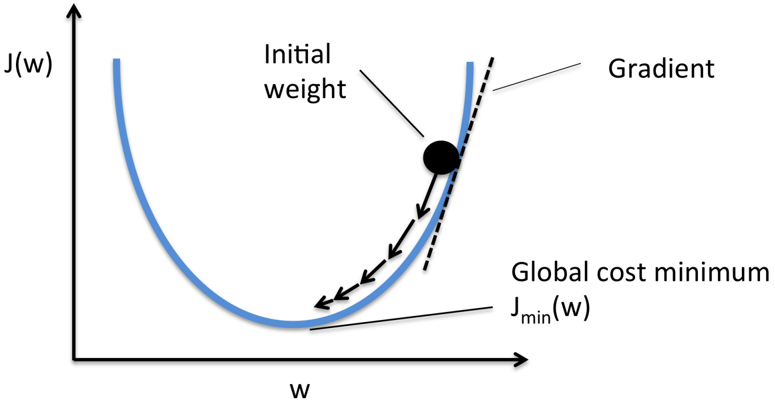
Starting in 1-D

- Consider function $f(x) = x^2$. Given an arbitrary starting point x_0, want to be able to find the minimum (let's say we didn't know it was at $x = 0$).
- Propose algorithm: for $t = 0, \ldots, N$, $x_{t+1} = x_t - \eta f'(x_t)$. $\eta > 0$ is step-size hyperparameter.
- Intuition: derivative shows rate of *growth* at a point. Algorithm is subtracting derivative from current point–go toward direction of decay.
- Example: $x_0 = 1$, $\eta = 0.1$. $x_1 = x_0 - 0.1 \cdot 2x_0 = 0.8$. $x_2 = x_1 - 0.2 \cdot x_1 = 0.64$ etc.

Starting in 1-D

- Consider function $f(x) = x^2$. Given an arbitrary starting point x_0, want to be able to find the minimum (let's say we didn't know it was at $x = 0$).
- Propose algorithm: for $t = 0, \ldots, N$, $x_{t+1} = x_t - \eta f'(x_t)$. $\eta > 0$ is step-size hyperparameter.
- Intuition: derivative shows rate of *growth* at a point. Algorithm is subtracting derivative from current point–go toward direction of decay.
- Example: $x_0 = 1$, $\eta = 0.1$. $x_1 = x_0 - 0.1 \cdot 2x_0 = 0.8$. $x_2 = x_1 - 0.2 \cdot x_1 = 0.64$ etc.
- Notice in general, we get $x_{t+1} = (1 - 2\eta)x_t$, which implies recursively $x_{t+1} = (1 - 2\eta)^{t+1}x_0$. Iterates decay exponentially toward 0.

Starting in 1-D



Introducing Gradient Descent

- Following intuition from previous part, let $f(\vec{x}) : \mathbb{R}^n \to \mathbb{R}$ be a scalar function that brings a vector in $\mathbb{R}^n$ to $\mathbb{R}$. Simple example extending 1-D case: $f(x) = x_1^2 + x_2^2 + \cdots + x_n^2$.

Introducing Gradient Descent

- Following intuition from previous part, let $f(\vec{x}) : \mathbb{R}^n \to \mathbb{R}$ be a scalar function that brings a vector in $\mathbb{R}^n$ to $\mathbb{R}$. Simple example extending 1-D case: $f(x) = x_1^2 + x_2^2 + \cdots + x_n^2$.
- What is gradient (at a point)? Simply vector in $\mathbb{R}^n$ whose entries are the partial derivatives evaluated at a point:

$$\nabla f(z) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(z) \\ \vdots \\ \frac{\partial f}{\partial x_n}(z) \end{bmatrix}.$$

Introducing Gradient Descent

- Following intuition from previous part, let $f(\vec{x}) : \mathbb{R}^n \to \mathbb{R}$ be a scalar function that brings a vector in $\mathbb{R}^n$ to $\mathbb{R}$. Simple example extending 1-D case: $f(x) = x_1^2 + x_2^2 + \cdots + x_n^2$.
- What is gradient (at a point)? Simply vector in $\mathbb{R}^n$ whose entries are the partial derivatives evaluated at a point:

$$\nabla f(z) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(z) \\ \vdots \\ \frac{\partial f}{\partial x_n}(z) \end{bmatrix}.$$

- This is straightforward extension of 1-D case: instead of one direction of growth, we take vector with directions of growth of every coordinate direction.

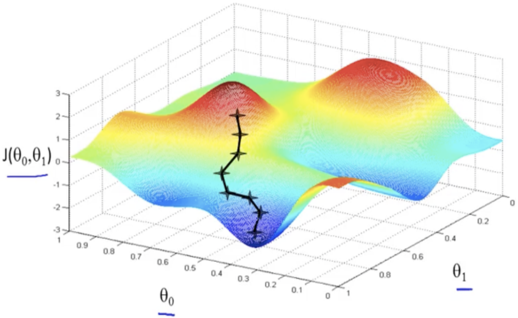
Introducing Gradient Descent

- Following intuition from previous part, let $f(\vec{x}) : \mathbb{R}^n \to \mathbb{R}$ be a scalar function that brings a vector in $\mathbb{R}^n$ to $\mathbb{R}$. Simple example extending 1-D case: $f(x) = x_1^2 + x_2^2 + \cdots + x_n^2$.
- What is gradient (at a point)? Simply vector in $\mathbb{R}^n$ whose entries are the partial derivatives evaluated at a point:

$$\nabla f(z) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(z) \\ \vdots \\ \frac{\partial f}{\partial x_n}(z) \end{bmatrix}.$$

- This is straightforward extension of 1-D case: instead of one direction of growth, we take vector with directions of growth of every coordinate direction.
- Base algorithm is the same: $t = 0, \ldots, N$, $x_{t+1} = x_t - \eta \nabla f(x_t)$. $\eta > 0$ is step-size hyperparameter.

Introducing Gradient Descent



When is GD guaranteed to work?

- Very important notion: convexity.

When is GD guaranteed to work?

- Very important notion: convexity.
- In general, function is convex if

$$f\left(\lambda x + (1-\lambda)y\right) \leq \lambda f(x) + (1-\lambda)f(y),$$

for all $x, y \in \mathrm{dom}(f)$.

When is GD guaranteed to work?

- Very important notion: convexity.
- In general, function is convex if

  $$f\left(\lambda x+(1-\lambda)y\right)\leq\lambda f(x)+(1-\lambda)f(y),$$

  for all $x, y \in \mathrm{dom}(f)$.
- In English: function value evaluated anywhere along the line between any two points x, y is always lower than the accordingly weighted function values at the end points.

When is GD guaranteed to work?

- Very important notion: convexity.
- In general, function is convex if

$$f\left(\lambda x + (1-\lambda)y\right) \leq \lambda f(x) + (1-\lambda)f(y),$$

  for all $x, y \in \mathrm{dom}(f)$.
- In English: function value evaluated anywhere along the line between any two points x, y is always lower than the accordingly weighted function values at the end points.
- Gradient naturally requires differentiability.

When is GD guaranteed to work?

- Very important notion: convexity.
- In general, function is convex if

  $$f\left(\lambda x+(1-\lambda)y\right)\leq \lambda f(x)+(1-\lambda)f(y),$$

  for all $x, y \in \mathrm{dom}(f)$.
- In English: function value evaluated anywhere along the line between any two points x, y is always lower than the accordingly weighted function values at the end points.
- Gradient naturally requires differentiability.
- Can show that for convex, differentiable functions, minimizer x^* satisfies $\nabla f(x^*) = \vec{0}$.

When is GD guaranteed to work?

Intuitively, gradient descent converges on continuously differentiable functions because it goes toward the minimizer x^*, and thus $\nabla f(x_t) \to_t \vec{0}$.

When is GD guaranteed to work?

Intuitively, gradient descent converges on continuously differentiable functions because it goes toward the minimizer x^*, and thus $\nabla f(x_t) \to_t \vec{0}$.

Every caveat is important to guarantee convergence of GD:

When is GD guaranteed to work?

Intuitively, gradient descent converges on continuously differentiable functions because it goes toward the minimizer x^*, and thus $\nabla f(x_t) \rightarrow_t \vec{0}$.

Every caveat is important to guarantee convergence of GD:

- Convex, but not continuously differentiable.

When is GD guaranteed to work?

Intuitively, gradient descent converges on continuously differentiable functions because it goes toward the minimizer x^*, and thus $\nabla f(x_t) \to_t \vec{0}$.

Every caveat is important to guarantee convergence of GD:

- Convex, but not continuously differentiable.
- Continuously differentiable, but not convex.

When is GD guaranteed to work?

Intuitively, gradient descent converges on continuously differentiable functions because it goes toward the minimizer x^*, and thus $\nabla f(x_t) \to_t \vec{0}$.

Every caveat is important to guarantee convergence of GD:

- Convex, but not continuously differentiable.
- Continuously differentiable, but not convex.
- Bad choice of step-size.

$$f(x) = x^2. \quad x_{t+1} = x_t - f'(x_t), \quad \eta = 1.$$

When is GD guaranteed to work?

That being said, variants of GD are used heavily in practice when aforementioned hypotheses are violated, e.g. neural net optimization. Deep theory to understand why this is OK; even if we don't have theory, GD is often a good starting algorithm.

Data Science Example 1: Least-Squares

- Recall that least-squares solves $\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - \boldsymbol{X}\beta\|_2^2$
- Objective $f(\boldsymbol{\beta}) = \|\boldsymbol{y} - \boldsymbol{X}\beta\|_2^2$ is convex in $\boldsymbol{\beta}$
- $\nabla_{\boldsymbol{\beta}} f(\boldsymbol{\beta}) = \nabla_{\boldsymbol{\beta}}(\boldsymbol{y} - \boldsymbol{X}\beta)^\top(\boldsymbol{y} - \boldsymbol{X}\beta) =$
  $\nabla_{\boldsymbol{\beta}}(\boldsymbol{y}^\top\boldsymbol{y} - 2\boldsymbol{\beta}^\top\boldsymbol{X}^\top\boldsymbol{y}^\top + \boldsymbol{\beta}^\top\boldsymbol{X}^\top\boldsymbol{X}\beta) = -2\boldsymbol{X}^\top\boldsymbol{y} + 2\boldsymbol{X}^\top\boldsymbol{X}\beta$
- Gradient Descent: iterate $\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \eta(-2\boldsymbol{X}^\top\boldsymbol{y} + 2\boldsymbol{X}^\top\boldsymbol{X}\beta)$
- If η small enough, will converge to $\boldsymbol{\beta}^*$

Data Science Example 1: Least-Squares

- Recall that least-squares solves $\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - \boldsymbol{X}\beta\|_2^2$
- Objective $f(\boldsymbol{\beta}) = \|\boldsymbol{y} - \boldsymbol{X}\beta\|_2^2$ is convex in $\boldsymbol{\beta}$
- $\nabla_{\boldsymbol{\beta}} f(\boldsymbol{\beta}) = \nabla_{\boldsymbol{\beta}}(\boldsymbol{y} - \boldsymbol{X}\beta)^\top(\boldsymbol{y} - \boldsymbol{X}\beta) = \nabla_{\boldsymbol{\beta}}(\boldsymbol{y}^\top\boldsymbol{y} - 2\boldsymbol{\beta}^\top\boldsymbol{X}^\top\boldsymbol{y}^\top + \boldsymbol{\beta}^\top\boldsymbol{X}^\top\boldsymbol{X}\beta) = -2\boldsymbol{X}^\top\boldsymbol{y} + 2\boldsymbol{X}^\top\boldsymbol{X}\beta$
- Gradient Descent: iterate $\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \eta(-2\boldsymbol{X}^\top\boldsymbol{y} + 2\boldsymbol{X}^\top\boldsymbol{X}\beta)$
- If η small enough, will converge to $\boldsymbol{\beta}^*$
- Known closed form solution: $\boldsymbol{\beta}^* = (\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{y}$. Gradient descent not extremely useful in this case.

Data Science Example 2: Logistic Regression (HW5)

- MLE in logistic regression solves $\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \log(1 + e^{-y_i \boldsymbol{\beta}^\top \boldsymbol{x}_i})$
- Note: each $\boldsymbol{x}_i$ has a 1 appended to account for the bias term
- $\boldsymbol{\beta}^*$ does *not* have an easily attainable closed-form solution, must use gradient descent (HW5 Q2)
- Objective $f(\boldsymbol{\beta}) = \sum_{i=1}^{n} \log(1 + e^{-y_i \boldsymbol{\beta}^\top \boldsymbol{x}_i})$ is convex in $\boldsymbol{\beta}$
- $\nabla_{\boldsymbol{\beta}} f(\boldsymbol{\beta}) = \sum_{i=1}^{n} \nabla_{\boldsymbol{\beta}} \log(1 + e^{-y_i \boldsymbol{\beta}^\top \boldsymbol{x}_i}) = \sum_{i=1}^{n} \frac{e^{-y_i \boldsymbol{\beta}^\top \boldsymbol{x}_i}}{1 + e^{-y_i \boldsymbol{\beta}^\top \boldsymbol{x}_i}} y_i \boldsymbol{x}_i$
- Iterate $\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \eta \sum_{i=1}^{n} \frac{e^{-y_i \boldsymbol{\beta}^\top \boldsymbol{x}_i}}{1 + e^{-y_i \boldsymbol{\beta}^\top \boldsymbol{x}_i}} y_i \boldsymbol{x}_i$