COMPSCI 514: ALGORITHMS FOR DATA SCIENCE

Andrew McGregor
Lecture 24
• Final will be Tuesday (5/11) noon to Wednesday (6/11) noon. The exam will be in Gradescope and once you start you have 2 hours 30 minutes to complete and extra 30 minutes spare to upload. Please take care when uploading.

• During the exam, you may consult slides and homework solutions but no other material. If the wording of a question isn’t clear, you may send a private Piazza post to the instructors.

• See Piazza for post about logistics and practice questions.

• Extra office hours at 11am Monday. No office hours tomorrow.
This Class:

• Finish online gradient descent analysis.
• Application to stochastic gradient descent.
• Course wrap up.
• Foundational concepts like convexity (line between any two points on curve is above the curve), convex sets (line between any two points in set in the set), directional derivative (slope of curve if we move in particular direction), and Lipschitzness (slope is bounded).
• Foundational concepts like convexity (line between any two points on curve is above the curve), convex sets (line between any two points in set in the set), directional derivative (slope of curve if we move in particular direction), and Lipschitzness (slope is bounded).

• Gradient descent greedily tries to find the min value of function $f : \mathbb{R}^d \rightarrow d$ by maintaining a vector $\vec{\theta} \in \mathbb{R}^d$ and at each step moving $\vec{\theta}$ “downhill”, i.e., in the direction that minimizes directional derivative
• Foundational concepts like convexity (line between any two points on curve is above the curve), convex sets (line between any two points in set in the set), directional derivative (slope of curve if we move in particular direction), and Lipschitzness (slope is bounded).

• Gradient descent greedily tries to find the min value of function $f : \mathbb{R}^d \to d$ by maintaining a vector $\vec{\theta} \in \mathbb{R}^d$ and at each step moving $\vec{\theta}$ “downhill”, i.e., in the direction that minimizes directional derivative.

• Bounded the number of steps required if $f$ is convex and Lipschitz.
• Foundational concepts like convexity (line between any two points on curve is above the curve), convex sets (line between any two points in set in the set), directional derivative (slope of curve if we move in particular direction), and Lipschitzness (slope is bounded).

• Gradient descent greedily tries to find the min value of function $f : \mathbb{R}^d \rightarrow d$ by maintaining a vector $\vec{\theta} \in \mathbb{R}^d$ and at each step moving $\vec{\theta}$ “downhill”, i.e., in the direction that minimizes directional derivative.

• Bounded the number of steps required if $f$ is convex and Lipschitz.

• Simple extension for optimization over a convex constraint set.
• Foundational concepts like convexity (line between any two points on curve is above the curve), convex sets (line between any two points in set in the set), directional derivative (slope of curve if we move in particular direction), and Lipschitzness (slope is bounded).

• Gradient descent greedily tries to find the min value of function $f : \mathbb{R}^d \to d$ by maintaining a vector $\tilde{\theta} \in \mathbb{R}^d$ and at each step moving $\tilde{\theta}$ “downhill”, i.e., in the direction that minimizes directional derivative.

• Bounded the number of steps required if $f$ is convex and Lipschitz.

• Simple extension for optimization over a convex constraint set.

• Finish up today: Online optimization and stochastic gradient descent.
• Foundational concepts like convexity (line between any two points on curve is above the curve), convex sets (line between any two points in set in the set), directional derivative (slope of curve if we move in particular direction), and Lipschitzness (slope is bounded).

• Gradient descent greedily tries to find the min value of function $f : \mathbb{R}^d \rightarrow d$ by maintaining a vector $\vec{\theta} \in \mathbb{R}^d$ and at each step moving $\vec{\theta}$ “downhill”, i.e., in the direction that minimizes directional derivative.

• Bounded the number of steps required if $f$ is convex and Lipschitz.

• Simple extension for optimization over a convex constraint set.

• Finish up today: Online optimization and stochastic gradient descent.

• Lots that we didn’t cover: accelerated methods, adaptive methods, second order methods (quasi-Newton methods). Gave mathematical tools to understand these methods. See CS 690OP for more!
**Online Optimization**: In place of a single function $f$, we see a different objective function at each step:

$$f_1, f_2, \ldots, f_t : \mathbb{R}^d \rightarrow \mathbb{R}$$

Will make no assumptions on how $f_1, \ldots, f_t$ are related to each other.
Online Optimization: In place of a single function $f$, we see a different objective function at each step:

$$f_1, f_2, \ldots, f_t : \mathbb{R}^d \rightarrow \mathbb{R}$$

- At each step, first pick (play) a parameter vector $\hat{\theta}^{(i)}$.
- Then are told $f_i$ and incur cost $f_i(\hat{\theta}^{(i)})$.
- **Goal:** Minimize total cost $\sum_{i=1}^{t} f_i(\hat{\theta}^{(i)})$.
- **Metric:** Regret $= \sum_{i=1}^{t} f_i(\hat{\theta}^{(i)}) - \sum_{i=1}^{t} f_i(\hat{\theta}^{\text{off}})$ where

$$\hat{\theta}^{\text{off}} = \arg \min_{\theta} \sum_{i=1}^{t} f_i(\theta)$$

Will make no assumptions on how $f_1, \ldots, f_t$ are related to each other.
Assume that:

- \( f_1, \ldots, f_t \) are all convex.
- Each \( f_i \) is \( G \)-Lipschitz, i.e., \( \| \nabla f_i(\vec{\theta}) \|_2 \leq G \) for all \( \vec{\theta} \).
- \( \| \vec{\theta}^{(1)} - \vec{\theta}^{\text{off}} \|_2 \leq R \) where \( \vec{\theta}^{(1)} \) is the first vector chosen.

**Online Gradient Descent**

- Pick some initial \( \vec{\theta}^{(1)} \).
- Set step size \( \eta = \frac{R}{G \sqrt{t}} \).
- For \( i = 1, \ldots, t \)
  - Play \( \vec{\theta}^{(i)} \) and incur cost \( f_i(\vec{\theta}^{(i)}) \).
  - \( \vec{\theta}^{(i+1)} = \vec{\theta}^{(i)} - \eta \cdot \nabla f_i(\vec{\theta}^{(i)}) \)
**Theorem:** For convex $G$-Lipschitz $f_1, \ldots, f_t$, online gradient descent initialized with starting point $\theta^{(1)}$ within radius $R$ of $\theta^{off}$, using step size $\eta = \frac{R}{G\sqrt{t}}$, has regret bounded by:

$$\left[ \sum_{i=1}^{t} f_i(\theta^{(i)}) - \sum_{i=1}^{t} f_i(\theta^{off}) \right] \leq RG\sqrt{t}$$

Upper bound on average regret is at most $RG/\sqrt{t} \to 0$ as $t \to \infty$. 


**Theorem:** For convex $G$-Lipschitz $f_1, \ldots, f_t$, online gradient descent initialized with starting point $\theta^{(1)}$ within radius $R$ of $\theta^{\text{off}}$, using step size $\eta = \frac{R}{G\sqrt{t}}$, has regret bounded by:

$$\left[ \sum_{i=1}^{t} f_i(\theta^{(i)}) - \sum_{i=1}^{t} f_i(\theta^{\text{off}}) \right] \leq RG\sqrt{t}$$

Upper bound on average regret is at most $RG/\sqrt{t} \to 0$ as $t \to \infty$.

**Step 1.1:** For all $i$, $\nabla f_i(\theta^{(i)})^T (\theta^{(i)} - \theta^{\text{off}}) \leq \frac{\|\theta^{(i)} - \theta^{\text{off}} \|_2^2 - \|\theta^{(i+1)} - \theta^{\text{off}} \|_2^2}{2\eta} + \frac{\eta G^2}{2}$. 
Theorem: For convex $G$-Lipschitz $f_1, \ldots, f_t$, online gradient descent initialized with starting point $\theta^{(1)}$ within radius $R$ of $\theta^{\text{off}}$, using step size $\eta = \frac{R}{G \sqrt{t}}$, has regret bounded by:

$$\left[ \sum_{i=1}^{t} f_i(\theta^{(i)}) - \sum_{i=1}^{t} f_i(\theta^{\text{off}}) \right] \leq RG \sqrt{t}$$

Upper bound on average regret is at most $RG/\sqrt{t} \to 0$ as $t \to \infty$.

Step 1.1: For all $i$, $\nabla f_i(\theta^{(i)})^T (\theta^{(i)} - \theta^{\text{off}}) \leq \frac{\|\theta^{(i)} - \theta^{\text{off}}\|_2^2 - \|\theta^{(i+1)} - \theta^{\text{off}}\|_2^2}{2\eta} + \frac{\eta G^2}{2}$.

Convexity $\implies$ Step 1: For all $i$,

$$f_i(\theta^{(i)}) - f_i(\theta^{\text{off}}) \leq \frac{\|\theta^{(i)} - \theta^{\text{off}}\|_2^2 - \|\theta^{(i+1)} - \theta^{\text{off}}\|_2^2}{2\eta} + \frac{\eta G^2}{2}.$$
Theorem: For convex $G$-Lipschitz $f_1, \ldots, f_t$, online gradient descent initialized with starting point $\theta^{(1)}$ within radius $R$ of $\theta^{\text{off}}$, using step size $\eta = \frac{R}{G\sqrt{t}}$, has regret bounded by:

$$\left[ \sum_{i=1}^{t} f_i(\theta^{(i)}) - \sum_{i=1}^{t} f_i(\theta^{\text{off}}) \right] \leq R G \sqrt{t}$$

Step 1: For all $i$, $f_i(\theta^{(i)}) - f_i(\theta^{\text{off}}) \leq \frac{\|\theta^{(i)} - \theta^{\text{off}}\|_2^2 - \|\theta^{(i+1)} - \theta^{\text{off}}\|_2^2}{2\eta} + \frac{\eta G^2}{2}$
**Theorem:** For convex $G$-Lipschitz $f_1, \ldots, f_t$, online gradient descent initialized with starting point $\theta^{(1)}$ within radius $R$ of $\theta^{off}$, using step size $\eta = \frac{R}{G\sqrt{t}}$, has regret bounded by:

$$\left[ \sum_{i=1}^{t} f_i(\theta^{(i)}) - \sum_{i=1}^{t} f_i(\theta^{off}) \right] \leq RG \sqrt{t}$$

**Step 1:** For all $i$, $f_i(\theta^{(i)}) - f_i(\theta^{off}) \leq \frac{||\theta^{(i)} - \theta^{off}||_2^2 - ||\theta^{(i+1)} - \theta^{off}||_2^2}{2\eta} + \frac{\eta G^2}{2} \implies$

$$\left[ \sum_{i=1}^{t} f_i(\theta^{(i)}) - \sum_{i=1}^{t} f_i(\theta^{off}) \right] \leq \sum_{i=1}^{t} \frac{||\theta^{(i)} - \theta^{off}||_2^2 - ||\theta^{(i+1)} - \theta^{off}||_2^2}{2\eta} + \frac{t \cdot \eta G^2}{2}$$

$$\leq \frac{R^2}{2\eta} + \frac{t \cdot \eta G^2}{2} = RG \sqrt{t}$$
Stochastic gradient descent is an efficient **offline optimization method**, seeking $\hat{\theta}$ with

$$f(\hat{\theta}) \leq \min_{\tilde{\theta}} f(\tilde{\theta}) + \epsilon$$
Stochastic gradient descent is an efficient offline optimization method, seeking $\hat{\theta}$ with

$$f(\hat{\theta}) \leq \min_{\tilde{\theta}} f(\tilde{\theta}) + \epsilon$$

- The most popular optimization method in modern machine learning. Easily analyzed as a special case of online gradient descent!
Stochastic gradient descent is an efficient offline optimization method, seeking $\hat{\theta}$ with

$$f(\hat{\theta}) \leq \min_{\tilde{\theta}} f(\tilde{\theta}) + \epsilon$$

- The most popular optimization method in modern machine learning. Easily analyzed as a special case of online gradient descent!
- **Basic Idea**: In gradient descent, we set $\tilde{\theta}_{i+1} = \tilde{\theta}_i - \eta \cdot \nabla f(\tilde{\theta}_i)$. 
Stochastic gradient descent is an efficient offline optimization method, seeking $\hat{\theta}$ with

$$f(\hat{\theta}) \leq \min_{\tilde{\theta}} f(\tilde{\theta}) + \epsilon$$

- The most popular optimization method in modern machine learning. Easily analyzed as a special case of online gradient descent!
- **Basic Idea:** In gradient descent, we set $\tilde{\theta}_{i+1} = \tilde{\theta}_i - \eta \cdot \tilde{\nabla}f(\tilde{\theta}_i)$. In stochastic gradient descent we don’t compute $\tilde{\nabla}f(\tilde{\theta}_i)$ exactly but instead do something random that is correct in expectation. This saves time per step but might increase the number of steps.
Assume that:

• $f$ is convex and decomposable as $f(\theta) = \sum_{j=1}^{n} f_j(\theta)$.
STOCHASTIC GRADIENT DESCENT

Assume that:

- \( f \) is convex and decomposable as \( f(\vec{\theta}) = \sum_{j=1}^{n} f_j(\vec{\theta}) \).
- For example, trying to minimize a loss function over a data set \( \mathbf{X} \), \( L(\vec{\theta}, \mathbf{X}) = \sum_{j=1}^{n} \ell(\vec{\theta}, \vec{x}_j) \) that is a sum of losses of element in data set.
- Each \( f_j \) is \( \frac{G}{n} \)-Lipschitz.
- What does this imply about how Lipschitz \( f \) is?
STOCHASTIC GRADIENT DESCENT

Assume that:

- $f$ is convex and decomposable as $f(\vec{\theta}) = \sum_{j=1}^{n} f_j(\vec{\theta})$.
- For example, trying to minimize a loss function over a data set $X$, $L(\vec{\theta}, X) = \sum_{j=1}^{n} \ell(\vec{\theta}, \vec{x}_j)$ that is a sum of losses of element in data set.
- Each $f_j$ is $\frac{G}{n}$-Lipschitz
- What does this imply about how Lipschitz $f$ is?
- Initialize with $\theta^{(1)}$ satisfying $\|\vec{\theta}^{(1)} - \vec{\theta}^*\|_2 \leq R$. 
Assume that:

• $f$ is convex and decomposable as $f(\vec{\theta}) = \sum_{j=1}^{n} f_j(\vec{\theta})$.
  • For example, trying to minimize a loss function over a data set $X$, $L(\vec{\theta}, X) = \sum_{j=1}^{n} \ell(\vec{\theta}, \vec{x}_j)$ that is a sum of losses of element in data set.
• Each $f_j$ is $G/n$-Lipschitz
  • What does this imply about how Lipschitz $f$ is?
• Initialize with $\theta^{(1)}$ satisfying $\|\vec{\theta}^{(1)} - \vec{\theta}^*\|_2 \leq R$.

Stochastic Gradient Descent

• Pick some initial $\vec{\theta}^{(1)}$.
• Set step size $\eta = \frac{R}{G \sqrt{t}}$.
• For $i = 1, \ldots, t$
  • Pick random $j_i \in 1, \ldots, n$.
  • $\vec{\theta}^{(i+1)} = \vec{\theta}^{(i)} - \eta \cdot \nabla f_{j_i}(\vec{\theta}^{(i)})$
• Return $\hat{\vec{\theta}} = \frac{1}{t} \sum_{i=1}^{t} \vec{\theta}^{(i)}$. 
\[ \bar{\theta}^{(i+1)} = \bar{\theta}^{(i)} - \eta \cdot \nabla f_j(\bar{\theta}^{(i)}) \] vs. \[ \bar{\theta}^{(i+1)} = \bar{\theta}^{(i)} - \eta \cdot \nabla f(\bar{\theta}^{(i)}) \]

**Note that:** \[ \mathbb{E}[\nabla f_j(\bar{\theta}^{(i)})] = \frac{1}{n} \nabla f(\bar{\theta}^{(i)}) \].

Analysis extends to any algorithm that takes the gradient step in expectation (minibatch SGD, randomly quantized, measurement noise, differentially private, etc.)
Theorem – SGD on Convex Lipschitz Functions: SGD run with $t \geq \frac{R^2G^2}{\epsilon^2}$ iterations, $\eta = \frac{R}{G\sqrt{t}}$, and starting point within radius $R$ of $\theta^*$, outputs $\hat{\theta}$ satisfying: $\mathbb{E}[f(\hat{\theta})] \leq f(\theta^*) + \epsilon$. 

Step 1: $f(\hat{\theta}) - f(\theta^*) \leq \frac{1}{t} \sum_{i=1}^{t} \left[ f(\theta_i) - f(\theta^*) \right]$ since $f(\hat{\theta}) = f\left(\frac{1}{t} \sum_{i=1}^{t} \theta_i\right)$ by convexity.

Step 2: $\mathbb{E}[f(\hat{\theta}) - f(\theta^*)] \leq \frac{n}{t} \cdot \mathbb{E}\left[\sum_{i=1}^{t} \left[ f_j(\theta_i) - f_j(\theta^*) \right]\right]$ since $\mathbb{E}[f_j(\theta_i)] = \frac{1}{n} f(\theta_i)$.

Step 3: $\mathbb{E}[f(\hat{\theta}) - f(\theta^*)] \leq \frac{n}{t} \cdot \mathbb{E}\left[\sum_{i=1}^{t} \left[ f_j(\theta_i) - f_j(\theta^*) \right]\right]$.

Step 4: $\mathbb{E}[f(\hat{\theta}) - f(\theta^*)] \leq \frac{n}{t} \cdot R \cdot G \cdot \frac{n}{t} \cdot \sqrt{\frac{1}{t}}$ OGD bound $\leq R\sqrt{\frac{G}{t}}$. 

11
Theorem – SGD on Convex Lipschitz Functions: SGD run with $t \geq \frac{R^2 G^2}{\epsilon^2}$ iterations, $\eta = \frac{R}{G \sqrt{t}}$, and starting point within radius $R$ of $\theta^*$, outputs $\hat{\theta}$ satisfying: $\mathbb{E}[f(\hat{\theta})] \leq f(\theta^*) + \epsilon$.

Step 1: $f(\hat{\theta}) - f(\theta^*) \leq \frac{1}{t} \sum_{i=1}^{t}[f(\theta^{(i)}) - f(\theta^*)]$
Theorem – SGD on Convex Lipschitz Functions: SGD run with $t \geq \frac{R^2G^2}{\epsilon^2}$ iterations, $\eta = \frac{R}{G\sqrt{t}}$, and starting point within radius $R$ of $\theta^*$, outputs $\hat{\theta}$ satisfying: $\mathbb{E}[f(\hat{\theta})] \leq f(\theta^*) + \epsilon$.

Step 1: $f(\hat{\theta}) - f(\theta^*) \leq \frac{1}{t} \sum_{i=1}^{t} [f(\theta^{(i)}) - f(\theta^*)]$ since

$$f(\hat{\theta}) = f\left(\sum_{i=1}^{t} \theta^{(i)}/t\right) \leq \frac{1}{t} \sum_{i=1}^{t} f(\theta^{(i)})$$

by convexity.
Theorem – SGD on Convex Lipschitz Functions: SGD run with \( t \geq \frac{R^2G^2}{\epsilon^2} \) iterations, \( \eta = \frac{R}{G \sqrt{t}} \), and starting point within radius \( R \) of \( \theta^* \), outputs \( \hat{\theta} \) satisfying: \( \mathbb{E}[f(\hat{\theta})] \leq f(\theta^*) + \epsilon. \)

Step 1: \( f(\hat{\theta}) - f(\theta^*) \leq \frac{1}{t} \sum_{i=1}^{t} [f(\theta^{(i)}) - f(\theta^*)] \) since
\[
f(\hat{\theta}) = f\left(\frac{1}{t} \sum_{i=1}^{t} \theta^{(i)}\right) \leq \frac{1}{t} \sum_{i=1}^{t} f(\theta^{(i)}) \text{ by convexity}
\]

Step 2: \( \mathbb{E}[f(\hat{\theta}) - f(\theta^*)] \leq \frac{n}{t} \cdot \mathbb{E} \left[ \sum_{i=1}^{t} [f_{ij}(\theta^{(i)}) - f_{ij}(\theta^*)] \right] \)
**Theorem – SGD on Convex Lipschitz Functions:** SGD run with $t \geq \frac{R^2G^2}{\epsilon^2}$ iterations, $\eta = \frac{R}{G\sqrt{t}}$, and starting point within radius $R$ of $\theta^*$, outputs $\hat{\theta}$ satisfying: $\mathbb{E}[f(\hat{\theta})] \leq f(\theta^*) + \epsilon$.

**Step 1:** $f(\hat{\theta}) - f(\theta^*) \leq \frac{1}{t} \sum_{i=1}^{t} [f(\theta^{(i)}) - f(\theta^*)]$ since

$$f(\hat{\theta}) = f\left(\frac{1}{t} \sum_{i=1}^{t} \theta^{(i)}\right) \leq \frac{1}{t} \sum_{i=1}^{t} f(\theta^{(i)})$$

by convexity

**Step 2:** $\mathbb{E}[f(\hat{\theta}) - f(\theta^*)] \leq \frac{n}{t} \cdot \mathbb{E}\left[\sum_{i=1}^{t} [f_{j_i}(\theta^{(i)}) - f_{j_i}(\theta^*)]\right]$ since

$$\mathbb{E}[f_{j_i}(\theta^{(i)})] = \frac{1}{n} f(\theta^{(i)})$$

since $f(\tilde{\theta}) = \sum_{j=1}^{n} f_j(\tilde{\theta})$.
Theorem – SGD on Convex Lipschitz Functions: SGD run with $t \geq \frac{R^2G^2}{\epsilon^2}$ iterations, $\eta = \frac{R}{G\sqrt{t}}$, and starting point within radius $R$ of $\theta^*$, outputs $\hat{\theta}$ satisfying: $\mathbb{E}[f(\hat{\theta})] \leq f(\theta^*) + \epsilon$.

Step 1: $f(\hat{\theta}) - f(\theta^*) \leq \frac{1}{t} \sum_{i=1}^{t} [f(\theta^{(i)}) - f(\theta^*)]$ since

$$f(\hat{\theta}) = f\left(\frac{\sum_{i=1}^{t} \theta^{(i)}}{t}\right) \leq \frac{1}{t} \sum_{i=1}^{t} f(\theta^{(i)})$$ by convexity

Step 2: $\mathbb{E}[f(\hat{\theta}) - f(\theta^*)] \leq \frac{n}{t} \cdot \mathbb{E}\left[\sum_{i=1}^{t} [f_{j_i}(\theta^{(i)}) - f_{j_i}(\theta^*)]\right]$ since

$$\mathbb{E}[f_{j_i}(\theta^{(i)})] = \frac{1}{n} f(\theta^{(i)})$$ since $f(\tilde{\theta}) = \sum_{j=1}^{n} f_{j}(\tilde{\theta})$

Step 3: $\mathbb{E}[f(\hat{\theta}) - f(\theta^*)] \leq \frac{n}{t} \cdot \mathbb{E}\left[\sum_{i=1}^{t} [f_{j_i}(\theta^{(i)}) - f_{j_i}(\theta^{off})]\right]$. 

11
Theorem – SGD on Convex Lipschitz Functions: SGD run with $t \geq \frac{R^2G^2}{\epsilon^2}$ iterations, $\eta = \frac{R}{G\sqrt{t}}$, and starting point within radius $R$ of $\theta^*$, outputs $\hat{\theta}$ satisfying: $\mathbb{E}[f(\hat{\theta})] \leq f(\theta^*) + \epsilon$.

Step 1: $f(\hat{\theta}) - f(\theta^*) \leq \frac{1}{t} \sum_{i=1}^{t} [f(\theta^{(i)}) - f(\theta^*)]$ since

$$f(\hat{\theta}) = f\left(\frac{\sum_{i=1}^{t} \theta^{(i)}}{t}\right) \leq \frac{1}{t} \sum_{i=1}^{t} f(\theta^{(i)})$$ by convexity

Step 2: $\mathbb{E}[f(\hat{\theta}) - f(\theta^*)] \leq \frac{n}{t} \cdot \mathbb{E}\left[\sum_{i=1}^{t} [f_{ji}(\theta^{(i)}) - f_{ji}(\theta^*)]\right]$ since

$$\mathbb{E}[f_{ji}(\theta^{(i)})] = \frac{1}{n} f(\theta^{(i)}) \text{ since } f(\tilde{\theta}) = \sum_{j=1}^{n} f_j(\tilde{\theta})$$

Step 3: $\mathbb{E}[f(\hat{\theta}) - f(\theta^*)] \leq \frac{n}{t} \cdot \mathbb{E}\left[\sum_{i=1}^{t} [f_{ji}(\theta^{(i)}) - f_{ji}(\theta^{off})]\right]$.

Step 4: $\mathbb{E}[f(\hat{\theta}) - f(\theta^*)] \leq \frac{n}{t} \cdot G \cdot \sqrt{t} = \frac{RG}{\sqrt{t}}$. 

\[\text{OGD bound}\]
SGD vs. GD

Stochastic gradient descent generally makes more iterations than gradient descent.

Each iteration is much cheaper (by a factor of $n$).

$$\vec{\nabla} \sum_{j=1}^{n} f_j(\vec{\theta}) \text{ vs. } \vec{\nabla} f_j(\vec{\theta})$$
When $f(\mathbf{\theta}) = \sum_{j=1}^n f_j(\mathbf{\theta})$ and $\|\nabla f_j(\mathbf{\theta})\|_2 \leq \frac{G}{n}$:

**Theorem – SGD:** After $t \geq \frac{R^2 G^2}{\epsilon^2}$ iterations outputs $\hat{\theta}$ satisfying:

$$\mathbb{E}[f(\hat{\theta})] \leq f(\theta^*) + \epsilon.$$

When $\|\nabla f(\mathbf{\theta})\|_2 \leq \bar{G}$:

**Theorem – GD:** After $t \geq \frac{R^2 \bar{G}^2}{\epsilon^2}$ iterations outputs $\hat{\theta}$ satisfying:

$$f(\hat{\theta}) \leq f(\theta^*) + \epsilon.$$
Methods for working with (compressing) high-dimensional data
Methods for working with (compressing) high-dimensional data

- Started with randomized dimensionality reduction and the JL lemma: compression from \( \text{any} \) \( d \)-dimensions to \( O(\log n/\epsilon^2) \) dimensions while preserving pairwise distances.
Methods for working with (compressing) high-dimensional data

- Started with randomized dimensionality reduction and the JL lemma: compression from any $d$-dimensions to $O(\log n/\epsilon^2)$ dimensions while preserving pairwise distances.
Methods for working with (compressing) high-dimensional data

• Started with randomized dimensionality reduction and the JL lemma: compression from *any* d-dimensions to $O(\log n/\epsilon^2)$ dimensions while preserving pairwise distances.

• Dimensionality reduction via low-rank approximation and optimal solution with PCA/eigendecomposition/SVD.
Methods for working with (compressing) high-dimensional data

- Started with randomized dimensionality reduction and the JL lemma: compression from any d-dimensions to $O(\log n/\epsilon^2)$ dimensions while preserving pairwise distances.
- Dimensionality reduction via low-rank approximation and optimal solution with PCA/eigendecomposition/SVD.
- Low-rank approximation of similarity matrices and entity embeddings (e.g., LSA, word2vec).
Methods for working with (compressing) high-dimensional data

• Started with randomized dimensionality reduction and the JL lemma: compression from \( \text{any} \) \( d \)-dimensions to \( O(\log n/\epsilon^2) \) dimensions while preserving pairwise distances.

• Dimensionality reduction via low-rank approximation and optimal solution with PCA/eigendecomposition/SVD.

• Low-rank approximation of similarity matrices and entity embeddings (e.g., LSA, word2vec).

• Spectral graph theory – nonlinear dimension reduction and spectral clustering for community detection.
Methods for working with (compressing) high-dimensional data

• Started with randomized dimensionality reduction and the JL lemma: compression from any d-dimensions to $O(\log n/\epsilon^2)$ dimensions while preserving pairwise distances.

• Dimensionality reduction via low-rank approximation and optimal solution with PCA/eigendecomposition/SVD.

• Low-rank approximation of similarity matrices and entity embeddings (e.g., LSA, word2vec).

• Spectral graph theory – nonlinear dimension reduction and spectral clustering for community detection.

• In the process covered linear algebraic tools that are very broadly useful in ML and data science: eigendecomposition, singular value decomposition, projection, norm transformations.
• Let $\vec{\pi} \in \mathbb{R}^d$ have random $\mathcal{N}(0, 1)$ entries. Then for any $\vec{x} \in \mathbb{R}^d$,

$$\mathbb{E}[\langle \vec{\pi}, \vec{x} \rangle^2] = ||\vec{x}||_2^2$$
• Let $\mathbf{\pi} \in \mathbb{R}^d$ have random $\mathcal{N}(0, 1)$ entries. Then for any $\mathbf{x} \in \mathbb{R}^d$,

$$E[\langle \mathbf{\pi}, \mathbf{x} \rangle^2] = \|\mathbf{x}\|_2^2$$

• Let $\mathbf{\Pi} \in \mathbb{R}^{k \times d}$ where $k = O(\epsilon^{-2} \log n)$ with $\mathcal{N}(0, 1/k)$ entries, then for any $\mathbf{x} \in \mathbb{R}^d$,

$$(1 - \epsilon)\|\mathbf{x}\|_2^2 \leq \|\mathbf{\Pi}\mathbf{x}\|_2^2 \leq (1 + \epsilon)\|\mathbf{x}\|_2^2$$
• Let $\vec{\pi} \in \mathbb{R}^d$ have random $\mathcal{N}(0, 1)$ entries. Then for any $\vec{x} \in \mathbb{R}^d$,

$$
\mathbb{E}[(\langle \vec{\pi}, \vec{x} \rangle)^2] = \|\vec{x}\|^2
$$

• Let $\Pi \in \mathbb{R}^{k \times d}$ where $k = O(\epsilon^{-2} \log n)$ with $\mathcal{N}(0, 1/k)$ entries, then for any $\vec{x} \in \mathbb{R}^d$,

$$(1 - \epsilon)\|\vec{x}\|_2^2 \leq \|\Pi \vec{x}\|_2^2 \leq (1 + \epsilon)\|\vec{x}\|_2^2$$

• Furthermore, for any $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n \in \mathbb{R}^d$,

$$(1 - \epsilon)\|\vec{x}_i - \vec{x}_j\|_2^2 \leq \|\Pi \vec{x}_i - \Pi \vec{x}_j\|_2^2 \leq (1 + \epsilon)\|\vec{x}_i - \vec{x}_j\|_2^2$$

i.e., random projections preserve distances between vectors.
• The $\mathcal{V}$ be the $k$-dimension subspace of $\mathbb{R}^d$ and let $\mathbf{V} \in \mathbb{R}^{d \times k}$ be the matrix whose columns are an orthonormal basis for $\mathcal{V}$. Then,

$$\mathbf{VV}^T \bar{x} = \arg \min_{\bar{z} \in \mathcal{V}} \| \bar{z} - \bar{x} \|_2$$

If we have $n$ points (rows of $\mathbf{X} \in \mathbb{R}^{n \times d}$), and want to project them all into a $k$-dimensional space $\mathcal{V}$, how to we choose $\mathcal{V}$ to minimizes the total error? Best $\mathcal{V}$ is the one spanned by top $k$ eigenvectors of $\mathbf{X}^T \mathbf{X}$. I.e., if $\mathbf{V}_k$ is the matrix with the first $k$ eigenvectors as columns, $\mathbf{V}_k = \arg \min_{\text{orthonormal}} \| \mathbf{X} - \mathbf{X} \mathbf{V}_k \mathbf{V}_k^T \|_F$ and $\| \mathbf{X} - \mathbf{X} \mathbf{V}_k \mathbf{V}_k^T \|_F = \lambda_{k+1} + \lambda_{k+2} + \ldots$ where $\lambda_1 \geq \lambda_2 \geq \ldots$ are the eigenvalues of $\mathbf{X}^T \mathbf{X}$.
• The $\mathcal{V}$ be the $k$-dimension subspace of $\mathbb{R}^d$ and let $\mathbf{V} \in \mathbb{R}^{d \times k}$ be the matrix whose columns are an orthonormal basis for $\mathcal{V}$. Then,

$$\mathbf{V} \mathbf{V}^T \bar{\mathbf{x}} = \arg \min_{\bar{\mathbf{z}} \in \mathcal{V}} ||\bar{\mathbf{z}} - \bar{\mathbf{x}}||_2$$

• If we have $n$ points (rows of $\mathbf{X} \in \mathbb{R}^{n \times d}$), and want to project them all into a $k$-dimensional space $\mathcal{V}$, how to we chose $\mathcal{V}$ to minimizes the total error?

Best $\mathcal{V}$ is the one spanned by top $k$ eigenvectors of $\mathbf{X}^T \mathbf{X}$
• The $\mathcal{V}$ be the $k$-dimension subspace of $\mathbb{R}^d$ and let $\mathbf{V} \in \mathbb{R}^{d \times k}$ be the matrix whose columns are an orthonormal basis for $\mathcal{V}$. Then,

$$\mathbf{VV}^T \bar{\mathbf{x}} = \arg\min_{\bar{\mathbf{z}} \in \mathcal{V}} \| \bar{\mathbf{z}} - \bar{\mathbf{x}} \|_2$$

• If we have $n$ points (rows of $\mathbf{X} \in \mathbb{R}^{n \times d}$), and want to project them all into a $k$-dimensional space $\mathcal{V}$, how to we chose $\mathcal{V}$ to minimizes the total error?

Best $\mathcal{V}$ is the one spanned by top $k$ eigenvectors of $\mathbf{X}^T \mathbf{X}$

• I.e., if $\mathbf{V}_k$ is the matrix with the first $k$ eigenvectors as columns,

$$\mathbf{V}_k = \arg\min_{\text{orthonormal } \mathbf{V}} \| \mathbf{X} - \mathbf{X} \mathbf{V} \mathbf{V}^T \|_F$$

and $\| \mathbf{X} - \mathbf{X} \mathbf{V}_k \mathbf{V}_k^T \|_F = \lambda_{k+1} + \lambda_{k+2} + \ldots$ where $\lambda_1 \geq \lambda_2 \geq \ldots$ are the eigenvalues of $\mathbf{X}^T \mathbf{X}$. 
• **Power Method:** The most fundamental iterative method for approximate SVD/eigendecomposition.

• **Goal:** Given a matrix $A \in \mathbb{R}^{d \times d}$, find an approximation to the top eigenvector $\vec{v}_1$ of $A$.

• **Algorithm:**
  - Choose $\vec{z}^{(0)}$ randomly: each $\vec{z}^{(0)}(i) \sim \mathcal{N}(0, 1)$.
  - For $i = 1, \ldots, t$
    - $\vec{z}^{(i)} := A \cdot \vec{z}^{(i-1)}$
    - $\vec{z}_i := \vec{z}^{(i)}/\|\vec{z}^{(i)}\|_2$
  
  Return $\vec{z}_t$

• With high probability, after $t = O\left(\gamma^{-1}\ln(d/\epsilon)\right)$ steps $\|\vec{z}^{(t)} - \vec{v}_1\|_2 \leq \epsilon$
  where $\gamma = 1 - |\lambda_2|/|\lambda_1|$.
• Any symmetric matrix $A$ can be written as $V\Lambda V^T$ corresponding to eigenvectors and eigenvectors.

• The Singular Value Decomposition (SVD) extends eigendecomposition to arbitrary matrices. The ‘swiss army knife’ of modern linear algebra.

• Any $X \in \mathbb{R}^{n \times d}$ with rank($X$) = $r$ can be written as $X = U\Sigma V^T$.
  • $U$ has orthonormal columns $\vec{u}_1, \ldots, \vec{u}_r \in \mathbb{R}^n$ (left singular vectors).
  • $V$ has orthonormal columns $\vec{v}_1, \ldots, \vec{v}_r \in \mathbb{R}^d$ (right singular vectors).
  • $\Sigma$ is diagonal with elements $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$ (singular values).

• Eigen-decomposition of $XX^T$ is $V\Sigma^2V^T$
Let $U_k$, $\Sigma_k$, $V_k$ be truncations of $U$, $\Sigma$, $V$ to first $k$ columns. The best rank $k$ approximation of $X$ is $X V_k V_k^T = U_k U_k^T X = U_k \Sigma_k V_k^T$. 

Applications include: Approximating an “incomplete” matrix $X$ by a low rank in the hope that the approximation “fills in” the missing values. LSA uses the rows of $U$ to approximate the documents in the a document/term matrix. Applications to graphs: Given adjacency matrix $A$ projecting nodes on the top $k$ eigenvalues of $A^T A$ allows us to map nodes to $k$-dimensional space such that close nodes are still close. Spectral Clustering The Laplacian $L = D - A$ has the property $\vec{v}^T L \vec{v} = \sum_{ij \in E} (v_i - v_j)^2$. The first second smallest eigenvector of $L$ gives way to decompose the graph into roughly balanced groups such that the number of cross edges in minimized.
Let $U_k, \Sigma_k, V_k$ be truncations of $U, \Sigma, V$ to first $k$ columns. The best rank $k$ approximation of $X$ is $XV_k V_k^T = U_k U_k^T X = U_k \Sigma_k V_k^T$.

Applications include: Approximating an “incomplete” matrix $X$ by a low rank in the hope that the approximation “fills in” the missing values. LSA uses the rows of $U$ to approximate the documents in the a document/term matrix.

Applications to graphs: Given adjacency matrix $A$, projecting nodes on the top $k$ eigenvalues of $A^T A$ allows us to map nodes to $k$-dimensional space such that close nodes are still close.

Spectral Clustering The Laplacian $L = D - A$ has the property $\vec{v}^T L \vec{v} = \sum_{ij \in E} (v_i - v_j)^2$. The first second smallest eigenvector of $L$ gives way to decompose the graph into roughly balanced groups such that the number of cross edges in minimized.
Let $U_k$, $\Sigma_k$, $V_k$ be truncations of $U$, $\Sigma$, $V$ to first $k$ columns. The best rank $k$ approximation of $X$ is $X V_k V_k^T = U_k U_k^T X = U_k \Sigma_k V_k^T$.

**Applications include:** Approximating an “incomplete” matrix $X$ by a low rank in the hope that the approximation “fills in” the missing values. LSA uses the rows of $U$ to approximate the documents in the document/term matrix.

**Applications to graphs:** Given adjacency matrix $A$ projecting nodes on the top $k$ eigenvalues of $A^T A$ allows us to map nodes to $k$-dimensional space such that close nodes are still close.
Let $U_k$, $\Sigma_k$, $V_k$ be truncations of $U$, $\Sigma$, $V$ to first $k$ columns. The best rank $k$ approximation of $X$ is $XV_kV_k^T = U_kU_k^TX = U_k\Sigma_kV_k^T$.

Applications include: Approximating an “incomplete” matrix $X$ by a low rank in the hope that the approximation “fills in” the missing values. LSA uses the rows of $U$ to approximate the documents in the a document/term matrix.

Applications to graphs: Given adjacency matrix $A$ projecting nodes on the top $k$ eigenvalues of $A^TA$ allows us to map nodes to $k$-dimensional space such that close nodes are still close.

Spectral Clustering The Laplacian $L = D - A$ has the property $\bar{v}^T L \bar{v} = \sum_{ij \in E} (v_i - v_j)^2$. The first second smallest eigenvector of $L$ gives way to decompose the graph into roughly balanced groups such that the number of cross edges in minimized.
Randomization as a computational resource for massive datasets.
Randomization as a computational resource for massive datasets.

- Focus on problems that are easy on small datasets but hard at massive scale – set size estimation, load balancing, distinct elements counting (MinHash), checking set membership (Bloom Filters), frequent items counting (Count-min sketch), near neighbor search (locality sensitive hashing).
Randomization as a computational resource for massive datasets.

• Focus on problems that are easy on small datasets but hard at massive scale – set size estimation, load balancing, distinct elements counting (MinHash), checking set membership (Bloom Filters), frequent items counting (Count-min sketch), near neighbor search (locality sensitive hashing).

• Just the tip of the iceberg on randomized streaming/sketching/hashing algorithms.
Randomization as a computational resource for massive datasets.

- Focus on problems that are easy on small datasets but hard at massive scale – set size estimation, load balancing, distinct elements counting (MinHash), checking set membership (Bloom Filters), frequent items counting (Count-min sketch), near neighbor search (locality sensitive hashing).

- Just the tip of the iceberg on randomized streaming/sketching/hashing algorithms.

- In the process covered probability/statistics tools that are very useful beyond algorithm design: concentration inequalities, higher moment bounds, law of large numbers, central limit theorem, linearity of expectation and variance, union bound, median as a robust estimator.
• We want to compute something about the stream $x_1, x_2, \ldots, x_m$ with only one pass over the stream and limited space.
We want to compute something about the stream $x_1, x_2, \ldots, x_m$ with only one pass over the stream and limited space.

Let $f_i$ be the number of values in stream that equal $i$.

Distinct Items: Can estimate $D = |\{i: f_i > 0\}|$ up to a factor $1 + \epsilon$ with probability $1 - \delta$ in $O(\epsilon^{-2} \log 1/\delta)$ space.

Frequently Elements Items: Can return a set $S$ such that:

- $f_i \geq m/k$ implies $i \in S$ and $i \not\in S$ implies $f_i \geq m(1 - \epsilon)/k$ with probability $1 - \delta$ in $O(k/\epsilon \cdot \log 1/\delta)$ space.

Sum of Squares: Can estimate $\sum f_i^2$ up to a factor $1 + \epsilon$ with probability $1 - \delta$ in $O(\epsilon^{-2} \log 1/\delta)$ space. Also investigated estimating $\sum f_i^k$ for general $k$.

In midterm, we say small space algorithms for exactly computing the mean and variance of a stream.
• We want to compute something about the stream $x_1, x_2, \ldots, x_m$ with only one pass over the stream and limited space.

• Let $f_i$ be the number of values in stream that equal $i$.

  • **Distinct Items:** Can estimate $D = |\{i : f_i > 0\}|$ up to a factor $1 + \epsilon$ with probability $1 - \delta$ in $O(\epsilon^{-2} \log 1/\delta)$ space.

  • **Frequently Elements Items:** Can return a set $S$ such that:
    - $f_i \geq m/k$ implies $i \in S$ and $i \notin S$ implies $f_i \geq m(1 - \epsilon)/k$ with probability $1 - \delta$ in $O(k/\epsilon \cdot \log 1/\delta)$ space.

• In midterm, we say small space algorithms for exactly computing the mean and variance of a stream.
• We want to compute something about the stream $x_1, x_2, \ldots, x_m$ with only one pass over the stream and limited space.

• Let $f_i$ be the number of values in stream that equal $i$.
  • **Distinct Items:** Can estimate $D = |\{i : f_i > 0\}|$ up to a factor $1 + \epsilon$ with probability $1 - \delta$ in $O(\epsilon^{-2} \log 1/\delta)$ space.
  • **Frequently Elements Items:** Can return a set $S$ such that:

$$f_i \geq m/k \text{ implies } i \in S \quad \text{and} \quad i \in S \text{ implies } f_i \geq m(1 - \epsilon)/k$$

with probability $1 - \delta$ in $O(k/\epsilon \cdot \log 1/\delta)$ space.
We want to compute something about the stream $x_1, x_2, \ldots, x_m$ with only one pass over the stream and limited space.

Let $f_i$ be the number of values in stream that equal $i$.

- **Distinct Items**: Can estimate $D = |\{i : f_i > 0\}|$ up to a factor $1 + \epsilon$ with probability $1 - \delta$ in $O(\epsilon^{-2} \log 1/\delta)$ space.

- **Frequently Elements Items**: Can return a set $S$ such that:

  $$f_i \geq m/k \text{ implies } i \in S \quad \text{and} \quad i \in S \text{ implies } f_i \geq m(1 - \epsilon)/k$$

  with probability $1 - \delta$ in $O(k/\epsilon \cdot \log 1/\delta)$ space.

- **Sum of Squares**: Can estimate $\sum f_i^2$ up to a factor $1 + \epsilon$ with probability $1 - \delta$ in $O(\epsilon^{-2} \log 1/\delta)$ space. Also investigated estimating $\sum f_i^k$ for general $k$.

- In midterm, we say small space algorithms for exactly computing the mean and variance of a stream.
Thanks for a great semester!