CS-GY 6763: Lecture 6
Online and Stochastic Gradient Decent

NYU Tandon School of Engineering, Prof. Christopher Musco

PROJECT

- Midterm in class next Tuesday. .
  - See Ed post.
  - If you have permission to take remotely, please email asap so I know who you are.
  - List of topics covered and practice problems are on the course webpage.
- Tomorrow in the reading group **Hayden Edelson** will present Estimating Sizes of Social Networks via Biased Sampling. See you there!
- Thanks to **Robert Ronan** for the presentation last week.

First Order Optimization: Given a function f and a constraint set $\mathcal{S}$, assume we have:

- **Function oracle**: Evaluate $f(\mathbf{x})$ for any $\mathbf{x}$.
- **Gradient oracle**: Evaluate $\nabla f(\mathbf{x})$ for any $\mathbf{x}$.
- **Projection oracle**: Evaluate $P_{\mathcal{S}}(\mathbf{x})$ for any $\mathbf{x}$.

Goal: Find $\hat{\mathbf{x}} \in \mathcal{S}$ such that $f(\hat{\mathbf{x}}) \leq \min_{\mathbf{x} \in \mathcal{S}} f(\mathbf{x}) + \epsilon$.

Projected gradient descent:

- Select starting point $\mathbf{x}^{(0)}$, learning rate η.
- For $i = 0, \ldots, T$:
  - $\mathbf{z} = \mathbf{x}^{(i)} - \eta \nabla f(\mathbf{x}^{(i)})$
  - $\mathbf{x}^{(i+1)} = P_{\mathcal{S}}(\mathbf{z})$
- Return $\hat{\mathbf{x}} = \arg\min_i f(\mathbf{x}^{(i)})$.

Conditions for convergence:

- **Convexity:** f is a convex function, $\mathcal{S}$ is a convex set.
- **Bounded initial distance:**

$$\|\mathbf{x}^{(0)} - \mathbf{x}^*\|_2 \leq R$$

- **Bounded gradients (Lipschitz function)**:

$$\|\nabla f(\mathbf{x})\|_2 \leq G \text{ for all } \mathbf{x} \in \mathcal{S}.$$

Theorem: Projected Gradient Descent returns $\hat{\mathbf{x}}$ with $f(\hat{\mathbf{x}}) \leq \min_{\mathbf{x} \in \mathcal{S}} f(\mathbf{x}) + \epsilon$ after

$$T = \frac{R^2 G^2}{\epsilon^2}$$

iterations.

Convexity:

$$0 \leq [f(\mathbf{y}) - f(\mathbf{x})] - \nabla f(\mathbf{x})^T(\mathbf{y} - \mathbf{x})$$

α-strong-convexity and β-smoothness:

$$\frac{\alpha}{2}\|\mathbf{x} - \mathbf{y}\|_2^2 \leq [f(\mathbf{y}) - f(\mathbf{x})] - \nabla f(\mathbf{x})^T(\mathbf{y} - \mathbf{x}) \leq \frac{\beta}{2}\|\mathbf{x} - \mathbf{y}\|_2^2.$$

Number of iterations for ϵ error:

	G-Lipschitz	β-smooth
R bounded start	$O\left(\frac{G^2R^2}{\epsilon^2}\right)$	$O\left(\frac{\beta R^2}{\epsilon}\right)$
α-strong convex	$O\left(\frac{G^2}{\alpha\epsilon}\right)$	$O\left(\frac{\beta}{\alpha}\log(1/\epsilon)\right)$

Theorem (GD for β-smooth, α-strongly convex.)

Let f be a β-smooth and α-strongly convex function. If we run GD for T steps (with step size $\eta = \frac{1}{\beta}$) we have:

$$\|\mathbf{x}^{(T)} - \mathbf{x}^*\|_2^2 \leq e^{-(T-1)\frac{\alpha}{\beta}} \|\mathbf{x}^{(1)} - \mathbf{x}^*\|_2^2$$

Corollary: If $T = O\left(\frac{\beta}{\alpha} \log(R\beta/\epsilon)\right)$ we have:

$$f(\mathbf{x}^{(T)}) - f(\mathbf{x}^*) \leq \epsilon$$

We will prove this in the special case of

$$f(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$$

where $\mathbf{x} \in \mathbb{R}^d, \mathbf{A} \in \mathbb{R}^{n \times d}, \mathbf{b} \in \mathbb{R}^n$.

Let f be a twice differentiable function from $\mathbb{R}^d \to \mathbb{R}$. Let the **Hessian** $\mathbf{H} = \nabla^2 f(\mathbf{x})$ contain all of its second derivatives at a point $\mathbf{x}$. So $\mathbf{H} \in \mathbb{R}^{d \times d}$. We have:

$$\mathbf{H}_{i,j} = \left[\nabla^2 f(\mathbf{x})\right]_{i,j} = \frac{\partial^2 f}{\partial x_i x_j}.$$

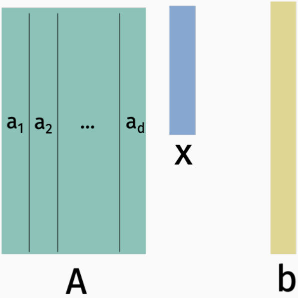
For vector $\mathbf{x}, \mathbf{v}$:

$$\nabla f(\mathbf{x} + t\mathbf{v}) \approx \nabla f(\mathbf{x}) + t\left[\nabla^2 f(\mathbf{x})\right]\mathbf{v}.$$

Let f be a twice differentiable function from $\mathbb{R}^d \to \mathbb{R}$. Let the **Hessian** $\mathbf{H} = \nabla^2 f(\mathbf{x})$ contain all of its second derivatives at a point $\mathbf{x}$. So $\mathbf{H} \in \mathbb{R}^{d\times d}$. We have:

$$\mathbf{H}_{i,j} = \left[\nabla^2 f(\mathbf{x})\right]_{i,j} = \frac{\partial^2 f}{\partial x_i x_j}.$$

Example: Let $f(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$. Recall that $\nabla f(\mathbf{x}) = 2\mathbf{A}^T(\mathbf{A}\mathbf{x} - \mathbf{b})$.



Claim: If f is twice differentiable, then it is convex if and only if the matrix $\mathbf{H} = \nabla^2 f(\mathbf{x})$ is positive semidefinite for all $\mathbf{x}$.

Definition (Positive Semidefinite (PSD))

A square, symmetric matrix $\mathbf{H} \in \mathbb{R}^{d \times d}$ is positive semidefinite (PSD) for any vector $\mathbf{y} \in \mathbb{R}^d$, $\mathbf{y}^T \mathbf{H} \mathbf{y} \geq 0$.

This is a natural notion of "positivity" for symmetric matrices. To denote that $\mathbf{H}$ is PSD we will typically use "Loewner order" notation (`\succeq` in LaTex):

$$\mathbf{H} \succeq 0.$$

We write $\mathbf{B} \succeq \mathbf{A}$ or equivalently $\mathbf{A} \preceq \mathbf{B}$ to denote that $(\mathbf{B} - \mathbf{A})$ is positive semidefinite. This gives a partial ordering on matrices.

Claim: If f is twice differentiable, then it is convex if and only if the matrix $\mathbf{H} = \nabla^2 f(\mathbf{x})$ is positive semidefinite for all $\mathbf{x}$.

Definition (Positive Semidefinite (PSD))

A square, symmetric matrix $\mathbf{H} \in \mathbb{R}^{d \times d}$ is positive semidefinite (PSD) for any vector $\mathbf{y} \in \mathbb{R}^d$, $\mathbf{y}^T\mathbf{H}\mathbf{y} \geq 0$.

For the least squares regression loss function: $f(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$, $\mathbf{H} = \nabla^2 f(\mathbf{x}) = 2\mathbf{A}^T\mathbf{A}$ for all $\mathbf{x}$.

We know that H is PSD because:

$$\mathbf{x}^T\mathbf{H}\mathbf{x} = 2\mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x} = 2\|\mathbf{A}\mathbf{x}\|_2^2 \geq 0.$$

If f is β-smooth and α-strongly convex then at any point $\mathbf{x}$, $\mathbf{H} = \nabla^2 f(\mathbf{x})$ satisfies:

$$\alpha \mathbf{I}_{d\times d} \preceq \mathbf{H} \preceq \beta \mathbf{I}_{d\times d},$$

where $\mathbf{I}_{d\times d}$ is a $d \times d$ identity matrix.

This is the natural matrix generalization of the statement for scalar valued functions:

$$\alpha \leq f''(x) \leq \beta.$$

$$\alpha \mathbf{I}_{d\times d} \preceq \mathbf{H} \preceq \beta \mathbf{I}_{d\times d}.$$

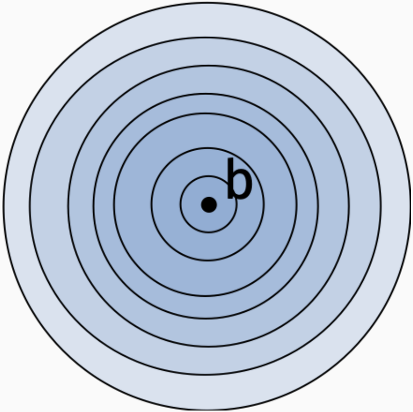
Equivalently for any $\mathbf{z}$,

$$\alpha\|\mathbf{z}\|_2^2 \leq \mathbf{z}^T\mathbf{H}\mathbf{z} \leq \beta\|\mathbf{z}\|_2^2.$$

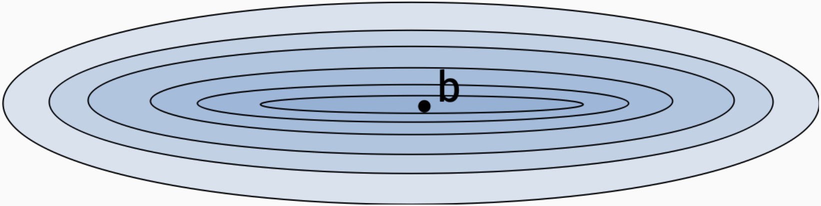
Let $f(\mathbf{x}) = \|\mathbf{D}\mathbf{x} - \mathbf{b}\|_2^2$ where $\mathbf{D}$ is a diagaonl matrix. For now imagine we're in two dimensions: $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, $\mathbf{D} = \begin{bmatrix} d_1 & 0 \\ 0 & d_2 \end{bmatrix}$.

What are α, β for this problem?

$$\alpha\|\mathbf{z}\|_2^2 \leq \mathbf{z}^T\mathbf{H}\mathbf{z} \leq \beta\|\mathbf{z}\|_2^2$$

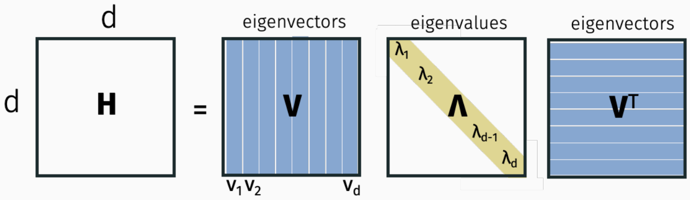


Level sets of $\|\mathbf{D}\mathbf{x} - \mathbf{b}\|_2^2$ when $d_1^2 = 1, d_2^2 = 1$.



Level sets of $\|\mathbf{D}\mathbf{x} - \mathbf{b}\|_2^2$ when $d_1^2 = \frac{1}{3}, d_2^2 = 2$.

Any symmetric matrix $\mathbf{H}$ has an orthogonal, real valued eigendecomposition.

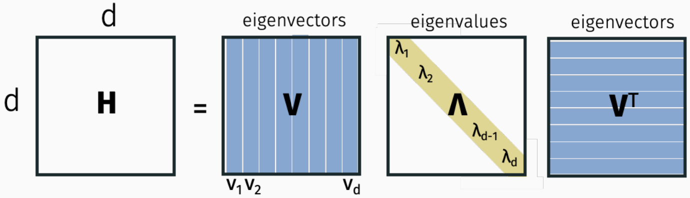


Here $\mathbf{V}$ is square and orthogonal, so $\mathbf{V}^T\mathbf{V} = \mathbf{V}\mathbf{V}^T = \mathbf{I}$. And for each $\mathbf{v}_i$, we have:

$$\mathbf{H}\mathbf{v}_i = \lambda_i \mathbf{v}_i.$$

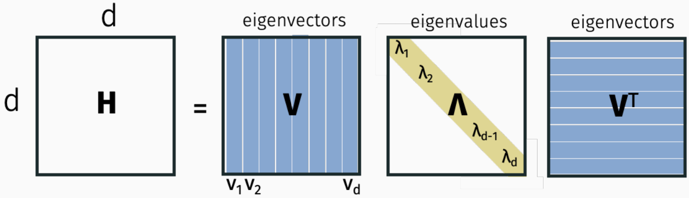
By definition, that's what makes $\mathbf{v}_1, \ldots, \mathbf{v}_d$ eigenvectors.

Recall $\mathbf{V}\mathbf{V}^T = \mathbf{V}^T\mathbf{V} = \mathbf{I}$.



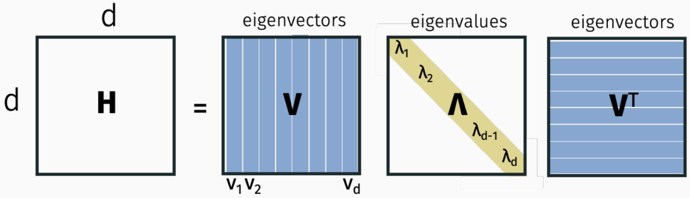
Claim: H is PSD $\Leftrightarrow \lambda_1, ..., \lambda_d \geq 0$.

Recall $\mathbf{V}\mathbf{V}^T = \mathbf{V}^T\mathbf{V} = \mathbf{I}$.



Claim: $\alpha\mathbf{I} \preceq \mathbf{H} \preceq \beta\mathbf{I} \Leftrightarrow \alpha \leq \lambda_d \leq \ldots \leq \lambda_1 \leq \beta$.

Recall $\mathbf{V}\mathbf{V}^T = \mathbf{V}^T\mathbf{V} = \mathbf{I}$.



In other words, if we let $\lambda_{\max}(\mathbf{H})$ and $\lambda_{\min}(\mathbf{H})$ be the smallest and largest eigenvalues of $\mathbf{H}$, then for all $\mathbf{z}$ we have:

$$\mathbf{z}^T\mathbf{H}\mathbf{z} \leq \lambda_{\max}(\mathbf{H}) \cdot \|\mathbf{z}\|^2$$
$$\mathbf{z}^T\mathbf{H}\mathbf{z} \geq \lambda_{\min}(\mathbf{H}) \cdot \|\mathbf{z}\|^2$$

If the maximum eigenvalue of $\mathsf{H} = \nabla^2 f(\mathsf{x}) = \beta$ and the minimum eigenvalue of $\mathsf{H} = \nabla^2 f(\mathsf{x}) = \alpha$ then $f(\mathsf{x})$ is β-smooth and α-strongly convex.

$$\lambda_{\mathsf{max}}(\mathsf{H}) = \beta$$
$$\lambda_{\mathsf{min}}(\mathsf{H}) = \alpha$$

Theorem (GD for β-smooth, α-strongly convex.)

Let f be a β-smooth and α-strongly convex function. If we run GD for T steps (with step size $\eta = \frac{1}{\beta}$) we have:

$$\|\mathbf{x}^{(T+1)} - \mathbf{x}^*\|_2 \leq e^{-T/\kappa}\|\mathbf{x}^{(1)} - \mathbf{x}^*\|_2$$

Goal: Prove for $f(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$.

Let $\lambda_{\max} = \lambda_{\max}(\mathbf{A}^T\mathbf{A})$. Gradient descent update is:

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \frac{1}{2\,\lambda_{\max}} 2\mathbf{A}^T(\mathbf{A}\mathbf{x}^{(t)} - \mathbf{b})$$

Richardson Iteration view:

$$(\mathbf{x}^{(t+1)} - \mathbf{x}^*) = \left(\mathbf{I} - \frac{1}{\lambda_{\max}}\mathbf{A}^T\mathbf{A}\right)(\mathbf{x}^{(t)} - \mathbf{x}^*)$$

What is the maximum eigenvalue of the symmetric matrix $\left(\mathbf{I} - \frac{1}{\lambda_{\max}}\mathbf{A}^T\mathbf{A}\right)$ in terms of the eigenvalues $\lambda_{\max} = \lambda_1 \geq \ldots \geq \lambda_d = \lambda_{\min}$ of $\mathbf{A}^T\mathbf{A}$?

$$(\mathbf{x}^{(T+1)} - \mathbf{x}^*) = \left(\mathbf{I} - \frac{1}{\lambda_{\max}}\mathbf{A}^T\mathbf{A}\right)^T (\mathbf{x}^{(1)} - \mathbf{x}^*)$$

Approach: Show that the maximum eigenvalue of $\left(\mathbf{I} - \frac{1}{\lambda_{\max}}\mathbf{A}^T\mathbf{A}\right)^T$ is small – i.e., bounded by $e^{-T/\kappa} = \epsilon$.

Conclusion:

- $\|\mathbf{x}^{(T+1)} - \mathbf{x}^*\|_2^2 = (\mathbf{x}^{(1)} - \mathbf{x}^*)^T \left(\mathbf{I} - \frac{1}{\lambda_{\max}}\mathbf{A}^T\mathbf{A}\right)^{2T} (\mathbf{x}^{(1)} - \mathbf{x}^*)$
- Since $\lambda_{\max}(\mathbf{M}) = \max_{\mathbf{z}} \frac{\mathbf{z}^T\mathbf{M}\mathbf{z}}{\|\mathbf{z}\|_2^2}$, we have:

$$\|\mathbf{x}^{(T+1)} - \mathbf{x}^*\|_2^2 \leq \lambda_{\max}\left(\left(\mathbf{I} - \frac{1}{\lambda_{\max}}\mathbf{A}^T\mathbf{A}\right)^{2T}\right)$$

So we have $\|\mathbf{x}^{(T+1)} - \mathbf{x}^*\|_2 \leq$

$$(\mathbf{x}^{(T+1)} - \mathbf{x}^*) = \left(\mathbf{I} - \frac{1}{\lambda_{\mathrm{max}}}\mathbf{A}^T\mathbf{A}\right)^T (\mathbf{x}^{(1)} - \mathbf{x}^*)$$

What is the maximum eigenvalue of the symmetric matrix $\left(\mathbf{I} - \frac{1}{\lambda_{\mathrm{max}}}\mathbf{A}^T\mathbf{A}\right)^T$?

ACCELERATION

Nesterov's accelerated gradient descent:

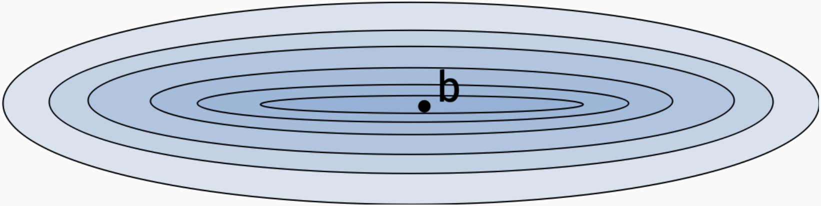
- $\mathbf{x}^{(1)} = \mathbf{y}^{(1)} = \mathbf{z}^{(1)}$
- For $t = 1, \ldots, T$
  - $\mathbf{y}^{(t+1)} = \mathbf{x}^{(t)} - \frac{1}{\beta}\nabla f(\mathbf{x}^{(t)})$
  - $\mathbf{x}^{(t+1)} = \left(1 + \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)\mathbf{y}^{(t+1)} + \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\left(\mathbf{y}^{(t+1)} - \mathbf{y}^{(t)}\right)$

Theorem (AGD for β-smooth, α-strongly convex.)

Let f be a β-smooth and α-strongly convex function. If we run AGD for T steps we have:

$$f(\mathbf{x}^{(t)}) - f(\mathbf{x}^*) \leq \kappa e^{-(t-1)\sqrt{\kappa}}\left[f(\mathbf{x}^{(1)}) - f(\mathbf{x}^*)\right]$$

Corollary: If $T = O\left(\sqrt{\kappa}\log(\kappa/\epsilon)\right)$ achieve error ϵ.



Level sets of $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$.

Other terms for similar ideas:

- Momentum
- Heavy-ball methods

What if we look back beyond <u>two iterates</u>?

BREAK

Second part of class:

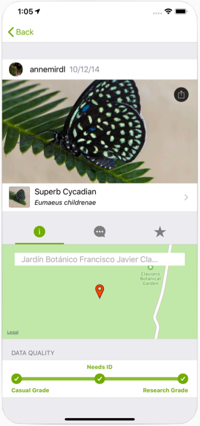
- Basics of Online Learning + Optimization.
- Introduction to Regret Analysis.
- Application to analyzing Stochastic Gradient Descent.

Many machine learning problems are solved in an online setting with constantly changing data.

- Spam filters are incrementally updated and adapt as they see more examples of spam over time.
- Image classification systems learn from mistakes over time (often based on user feedback).
- Content recommendation systems adapt to user behavior and clicks (which may not be a good thing...)

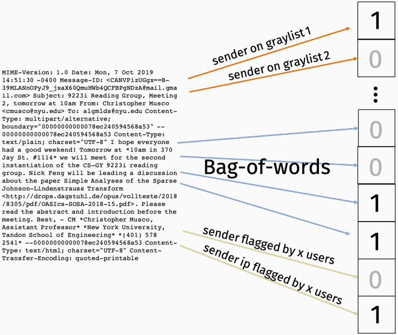
Plant identification via iNaturalist app.

(California Academy of Science + National Geographic)



- When the app fails, image is classified via crowdsourcing (backed by huge network of amateurs and experts).
- Single model that is updated constantly, not retrained in batches.

ML based email spam/scam filtering.



Markers for spam change overtime, so model might change.

ML based email spam/scam filtering.


Re:SAFTY CORONA VIRUS AWARENESS WHO

WO World Health Organization

World Health Organization

Dear Sir,

Go through the attached document on safety measures regarding the spreading of corona virus.

Click on the button below to download

Safety measures

Symptoms common symptoms include fever,coughcshortness of breath and breathing difficulties.

Regards,

Dr. Stella Chungong
Specialist wuhan-virus-advisory

Markers for spam change overtime, so model might change.

Choose some model $M_{\mathbf{x}}$ parameterized by parameters $\mathbf{x}$ and some loss function ℓ. At time steps $1, \ldots, T$, receive data vectors $\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(T)}$.

- At each time step, we pick ("play") a parameter vector $\mathbf{x}^{(i)}$.
- Make prediction $\tilde{y}^{(i)} = M_{\mathbf{x}^{(i)}}(\mathbf{a}_i)$.
- Then told true value or label $y^{(i)}$.
- Goal is to minimize cumulative loss:

$$L = \sum_{i=1}^{n} \ell(\mathbf{x}^{(i)}, \mathbf{a}^{(i)}, y^{(i)})$$

For example, for a regression problem we might use the ℓ_2 loss:

$$\ell(\mathbf{x}^{(i)}, \mathbf{a}^{(i)}, y^{(i)}) = \left| \langle \mathbf{x}^{(i)}, \mathbf{a}^{(i)} \rangle - y^{(i)} \right|^2.$$

For classification, we could use logistic/cross-entropy loss.

Abstraction as optimization problem: Instead of a single objective function f, we have a single (initially unknown) function $f_1, \ldots, f_T : \mathbb{R}^d \to \mathbb{R}$ for each time step.

- For time step $i \in 1, \ldots, T$, select vector $\mathbf{x}^{(i)}$.
- Observe f_i and pay cost $f_i(\mathbf{x}^{(i)})$
- Goal is to minimize $\sum_{i=1}^{T} f_i(\mathbf{x}^{(i)})$.

We make <u>no assumptions</u> that $f_1, \ldots, f_T$ are related to each other at all!

REGRET BOUND

In offline optimization, we wanted to find $\hat{\mathbf{x}}$ satisfying $f(\hat{\mathbf{x}}) \leq \min_{\mathbf{x}} f(\mathbf{x})$. Ask for a similar thing here.

Objective: Choose $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(T)}$ so that:

$$\sum_{i=1}^{T} f_i(\mathbf{x}^{(i)}) \leq \left[\min_{\mathbf{x}} \sum_{i=1}^{T} f_i(\mathbf{x})\right] + \epsilon.$$

Here ϵ is called the regret of our solution sequence $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(T)}$.

Regret compares to the best fixed solution in hindsight.

$$\sum_{i=1}^{T} f_i(\mathbf{x}^{(i)}) \leq \left[\min_{\mathbf{x}} \sum_{i=1}^{T} f_i(\mathbf{x})\right] + \epsilon.$$

It's very possible that $\sum_{i=1}^{T} f_i(\mathbf{x}^{(i)}) < \left[\min_{\mathbf{x}} \sum_{i=1}^{T} f_i(\mathbf{x})\right]$. Could we hope for something stronger?

Exercise: Argue that the following is impossible to achieve:

$$\sum_{i=1}^{T} f_i(\mathbf{x}^{(i)}) \leq \left[\sum_{i=1}^{T} \min_{\mathbf{x}} f_i(\mathbf{x})\right] + \epsilon.$$

Convex functions:

$$\begin{aligned} f_1(x) &= |x - h_1| \\ &\vdots \\ f_n(x) &= |x - h_T| \end{aligned}$$

where $h_1, \ldots, h_T$ are i.i.d. uniform $\{0, 1\}$.

$$\sum_{i=1}^{T} f_i(\mathbf{x}^{(i)}) \leq \left[\min_{\mathbf{x}} \sum_{i=1}^{T} f_i(\mathbf{x})\right] + \epsilon.$$

Beautiful balance:

- Either $f_1, \ldots, f_T$ are similar, so we can learn predict f_i from earlier functions.
- Or $f_1, \ldots, f_T$ are very different, in which case $\min_{\mathbf{x}} \sum_{i=1}^{T} f_i(\mathbf{x})$ is large, so regret bound is easy to achieve.
- Or we live somewhere in the middle.

Online Gradient descent:

- Choose $\mathbf{x}^{(1)}$ and $\eta = \frac{R}{G\sqrt{T}}$.
- For $i = 1, \ldots, T$:
    - Play $\mathbf{x}^{(i)}$.
    - Observe f_i and incur cost $f_i(\mathbf{x}^{(i)})$.
    - $\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \eta \nabla f_i(\mathbf{x}^{(i)})$

If $f_1, \ldots, f_T = f$ are all the same, this looks a lot like regular gradient descent. We update parameters using the gradient ∇f at each step.

$\mathbf{x}^* = \arg\min_{\mathbf{x}} \sum_{i=1}^{T} f_i(\mathbf{x})$ (the offline optimum)

Assume:

- $f_1, \ldots, f_T$ are all convex.
- Each is G-Lipschitz: for all $\mathbf{x}$, i, $\|\nabla f_i(\mathbf{x})\|_2 \leq G$.
- Starting radius: $\|\mathbf{x}^* - \mathbf{x}^{(1)}\|_2 \leq R$.

Online Gradient descent:

- Choose $\mathbf{x}^{(1)}$ and $\eta = \frac{R}{G\sqrt{T}}$.
- For $i = 1, \ldots, T$:
    - Play $\mathbf{x}^{(i)}$.
    - Observe f_i and incur cost $f_i(\mathbf{x}^{(i)})$.
    - $\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \eta \nabla f_i(\mathbf{x}^{(i)})$

Let $\mathbf{x}^* = \arg\min_{\mathbf{x}} \sum_{i=1}^{T} f_i(\mathbf{x})$ (the offline optimum)

Theorem (OGD Regret Bound)

After T steps, $\epsilon = \left[\sum_{i=1}^{T} f_i(\mathbf{x}^{(i)})\right] - \left[\sum_{i=1}^{T} f_i(\mathbf{x}^*)\right] \leq RG\sqrt{T}$.

Average regret overtime is bounded by $\frac{\epsilon}{T} \leq \frac{RG}{\sqrt{T}}$.

Goes $\rightarrow 0$ as $T \rightarrow \infty$.

All this with no assumptions on how $f_1, \ldots, f_T$ relate to each other! They could have even been chosen **adversarially** – e.g. with f_i depending on our choice of $\mathbf{x}_i$ and all previous choices.

Theorem (OGD Regret Bound)

After T steps, $\epsilon = \left[\sum_{i=1}^{T} f_i(\mathbf{x}^{(i)})\right] - \left[\sum_{i=1}^{T} f_i(\mathbf{x}^*)\right] \leq RG\sqrt{T}$.

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

$$f_i(\mathbf{x}^{(i)}) - f_i(\mathbf{x}^*) \leq \frac{\|\mathbf{x}^{(i)} - \mathbf{x}^*\|_2^2 - \|\mathbf{x}^{(i+1)} - \mathbf{x}^*\|_2^2}{2\eta} + \frac{\eta G^2}{2}$$

(Same proof as last class. Only uses convexity of f_i.)

Theorem (OGD Regret Bound)

After T steps, $\epsilon = \left[\sum_{i=1}^{T} f_i(\mathbf{x}^{(i)})\right] - \left[\sum_{i=1}^{T} f_i(\mathbf{x}^*)\right] \leq RG\sqrt{T}$.

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

$$f_i(\mathbf{x}^{(i)}) - f_i(\mathbf{x}^*) \leq \frac{\|\mathbf{x}^{(i)} - \mathbf{x}^*\|_2^2 - \|\mathbf{x}^{(i+1)} - \mathbf{x}^*\|_2^2}{2\eta} + \frac{\eta G^2}{2}$$

Telescoping Sum:

$$\sum_{i=1}^{T} \left[f_i(\mathbf{x}^{(i)}) - f_i(\mathbf{x}^*)\right] \leq \|\mathbf{x}^{(1)} - \mathbf{x}^*\|_2^2 - \|\mathbf{x}^{(T)} - \mathbf{x}^*\|_2^2 + \frac{T\eta G^2}{2}$$

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

STOCHASTIC GRADIENT DESCENT (SGD)

Efficient offline optimization method for functions f with finite sum structure:

$$f(\mathbf{x}) = \sum_{i=1}^{n} f_i(\mathbf{x}).$$

Goal is to find $\hat{\mathbf{x}}$ such that $f(\hat{\mathbf{x}}) \leq f(\mathbf{x}^*) + \epsilon$.

- The most widely use optimization algorithm in modern machine learning.
- Easily analyzed as a special case of online gradient descent!

Recall the machine learning setup. In empirical risk minimization, we can typically write:

$$f(\mathbf{x}) = \sum_{i=1}^{n} f_i(\mathbf{x})$$

where f_i is the loss function for a particular data example $(\mathbf{a}^{(i)}, y^{(i)})$.

Example: least squares linear regression.

$$f(\mathbf{x}) = \sum_{i=1}^{n} (\mathbf{x}^T\mathbf{a}^{(i)} - y^{(i)})^2$$

Note that by linearity, $\nabla f(\mathbf{x}) = \sum_{i=1}^{n} \nabla f_i(\mathbf{x})$.

Main idea: Use random approximate gradient in place of actual gradient.

Pick random $j \in 1, \ldots, n$ and update $\mathbf{x}$ using $\nabla f_j(\mathbf{x})$.

$$\mathbb{E}\left[\nabla f_j(\mathbf{x})\right] = \frac{1}{n}\nabla f(\mathbf{x}).$$

$n\nabla f_j(\mathbf{x})$ is an unbiased estimate for the true gradient $\nabla f(\mathbf{x})$, but can often be computed in a $1/n$ fraction of the time!

Trade slower convergence for cheaper iterations.

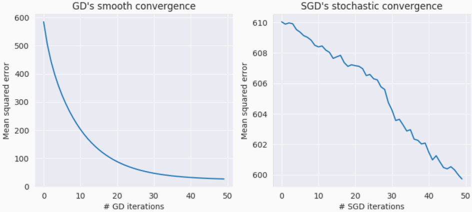
Stochastic first-order oracle for $f(\mathbf{x}) = \sum_{i=1}^{n} f_i(\mathbf{x})$.

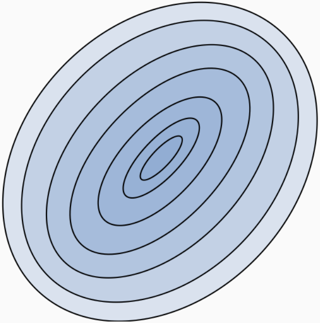
- **Function Query:** For any chosen $j, \mathbf{x}$, return $f_j(\mathbf{x})$
- **Gradient Query:** For any chosen $j, \mathbf{x}$, return $\nabla f_j(\mathbf{x})$

Stochastic Gradient descent:

- Choose starting vector $\mathbf{x}^{(1)}$, learning rate η
- For $i = 1, \ldots, T$:
  - Pick random $j_i \in 1, \ldots, n$.
  - $\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \eta \nabla f_{j_i}(\mathbf{x}^{(i)})$
- Return $\hat{\mathbf{x}} = \frac{1}{T} \sum_{i=1}^{T} \mathbf{x}^{(i)}$

VISUALIZING SGD





Assume:

- Finite sum structure: $f(\mathbf{x}) = \sum_{i=1}^{n} f_i(\mathbf{x})$, with $f_1, \ldots, f_n$ all convex.
- Lipschitz functions: for all $\mathbf{x}$, j, $\|\nabla f_j(\mathbf{x})\|_2 \leq \frac{G'}{n}$.
  - What does this imply about Lipschitz constant of f?
- Starting radius: $\|\mathbf{x}^* - \mathbf{x}^{(1)}\|_2 \leq R$.

Stochastic Gradient descent:

- Choose $\mathbf{x}^{(1)}$, steps T, learning rate $\eta = \frac{D}{G'\sqrt{T}}$.
- For $i = 1, \ldots, T$:
  - Pick random $j_i \in 1, \ldots, n$.
  - $\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \eta \nabla f_{j_i}(\mathbf{x}^{(i)})$
- Return $\hat{\mathbf{x}} = \frac{1}{T} \sum_{i=1}^{T} \mathbf{x}^{(i)}$

Approach: View as online gradient descent run on function sequence $f_{j_1}, \ldots, f_{j_T}$.

Only use the fact that step equals gradient in expectation.

Claim (SGD Convergence)

After $T = \frac{R^2 G'^2}{\epsilon^2}$ iterations:

$$\mathbb{E}\left[f(\hat{\mathbf{x}}) - f(\mathbf{x}^*)\right] \leq \epsilon.$$

Claim 1:

$$f(\hat{\mathbf{x}}) - f(\mathbf{x}^*) \leq \frac{1}{T}\sum_{i=1}^{T}\left[f(\mathbf{x}^{(i)}) - f(\mathbf{x}^*)\right]$$

Prove using Jensen's Inequality:

Claim (SGD Convergence)

After $T = \frac{R^2 G'^2}{\epsilon^2}$ *iterations:*

$$\mathbb{E}\left[f(\hat{\mathbf{x}}) - f(\mathbf{x}^*)\right] \leq \epsilon.$$

$$\begin{aligned}
\mathbb{E}[f(\hat{\mathbf{x}}) - f(\mathbf{x}^*)] &\leq \frac{1}{T}\sum_{i=1}^{T} \mathbb{E}\left[f(\mathbf{x}^{(i)}) - f(\mathbf{x}^*)\right] \\
&= \frac{1}{T}\sum_{i=1}^{T} n\mathbb{E}\left[f_{j_i}(\mathbf{x}^{(i)}) - f_{j_i}(\mathbf{x}^*)\right] \\
&= \frac{1}{T}\sum_{i=1}^{T} n\mathbb{E}\left[f_{j_i}(\mathbf{x}^{(i)}) - f_{j_i}(\mathbf{x}^{offline})\right] \\
&= \frac{n}{T} \cdot \mathbb{E}\left[\sum_{i=1}^{T} f_{j_i}(\mathbf{x}^{(i)}) - f_{j_i}(\mathbf{x}^*)\right] \\
&\leq \frac{n}{T} \cdot \left(R \cdot \frac{G'}{n} \cdot \sqrt{T}\right) \qquad \text{(by OGD guarantee.)}
\end{aligned}$$

Number of iterations for error ϵ:

- **Gradient Descent**: $T = \frac{R^2 G^2}{\epsilon^2}$.
- **Stochastic Gradient Descent**: $T = \frac{R^2 G'^2}{\epsilon^2}$.

Always have $G \leq G'$:

$$\max_{\mathbf{x}} \|\nabla f(\mathbf{x})\|_2 \leq \max_{\mathbf{x}} \left(\|\nabla f_1(\mathbf{x})\|_2 + \ldots + \|\nabla f_n(\mathbf{x})\|_2\right) \leq n \cdot \frac{G'}{n} = G'.$$

So GD converges strictly faster than *SGD*.

But for a fair comparison:

- SGD cost = (# of iterations) $\cdot\, O(1)$
- GD cost = (# of iterations) $\cdot\, O(n)$

We always have $G \leq G'$. When it is much smaller then GD will perform better. When it is closer to this upper bound, SGD will perform better.

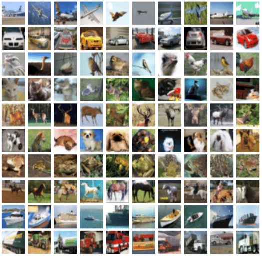
What is an extreme case where $G = G'$?

What if each gradient $\nabla f_i(\mathbf{x})$ looks like random vectors in $\mathbb{R}^d$? E.g. with $\mathcal{N}(0, 1)$ entries?

$$\mathbb{E}\left[\|\nabla f_i(\mathbf{x})\|_2^2\right] =$$

$$\mathbb{E}\left[\|\nabla f(\mathbf{x})\|_2^2\right] = \mathbb{E}\left[\|\sum_{i=1}^{n} \nabla f_i(\mathbf{x})\|_2^2\right] =$$

Takeaway: SGD performs better when there is more structure or repetition in the data set.



PRECONDITIONING

Main idea: Instead of minimizing $f(\mathbf{x})$, find another function $g(\mathbf{x})$ with the same minimum but which is better suited for first order optimization (e.g., has a smaller conditioner number).

Claim: Let $h(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^d$ be an invertible function. Let $g(\mathbf{x}) = f(h(\mathbf{x}))$. Then

$$\min_{\mathbf{x}} f(\mathbf{x}) = \min_{\mathbf{x}} g(\mathbf{x}) \quad \text{and} \quad \arg\min_{\mathbf{x}} f(\mathbf{x}) = h\left(\arg\min_{\mathbf{x}} g(\mathbf{x})\right).$$

First Goal: We need $g(\mathbf{x})$ to still be convex.

Claim: Let $\mathbf{P}$ be an invertible $d \times d$ matrix and let $g(\mathbf{x}) = f(\mathbf{P}\mathbf{x})$.

$g(\mathbf{x})$ is always convex.

Second Goal:

$g(\mathbf{x})$ should have better condition number κ than $f(\mathbf{x})$.

Example:

- $f(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$. $\kappa_f = \frac{\lambda_1(\mathbf{A}^T\mathbf{A})}{\lambda_d(\mathbf{A}^T\mathbf{A})}$.
- $g(\mathbf{x}) = \|\mathbf{A}\mathbf{P}\mathbf{x} - \mathbf{b}\|_2^2$. $\kappa_g = \frac{\lambda_1(\mathbf{P}^T\mathbf{A}^T\mathbf{A}\mathbf{P})}{\lambda_d(\mathbf{P}^T\mathbf{A}^T\mathbf{A}\mathbf{P})}$.

Third Goal: $\mathbf{P}$ should be easy to compute.

Many, many problem specific preconditioners are used in practice. There design is usually a heuristic process.

Example: Diagonal preconditioner.

- Let $\mathbf{D} = \mathrm{diag}(\mathbf{A}^T\mathbf{A})$
- Intuitively, we roughly have that $\mathbf{D} \approx \mathbf{A}^T\mathbf{A}$.
- Let $\mathbf{P} = \sqrt{\mathbf{D}^{-1}}$

$\mathbf{P}$ is often called a **Jacobi preconditioner**. Often works very well in practice!

DIAGONAL PRECONDITIONER

```
A =

    -734      1     33    9111      0
     -31     -2    108    5946    -19
     232     -1    101    3502     10
     426      0    -65   12503      9
    -373      0     26    9298      0
    -236     -2    -94    2398     -1
    2024      0   -132   -6904    -25
   -2258     -1     92   -6516      6
    2229      0      0   11921    -22
     338      1     -5  -16118    -23
```

```
>> cond(A'*A)

ans =

   8.4145e+07
```

```
>> P = sqrt(inv(diag(diag(A'*A))));
>> cond(P*A'*A*P)

ans =

   10.3878
```

Another view: If $g(\mathbf{x}) = f(\mathbf{P}\mathbf{x})$ then $\nabla g(\mathbf{x}) = \mathbf{P}^T \nabla f(\mathbf{P}\mathbf{x})$.

$\nabla g(\mathbf{x}) = \mathbf{P}\nabla f(\mathbf{P}\mathbf{x})$ when $\mathbf{P}$ is symmetric.

Gradient descent on g:

- For $t = 1, \ldots, T$,
  - $\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \eta \mathbf{P}\left[\nabla f(\mathbf{P}\mathbf{x}^{(t)})\right]$

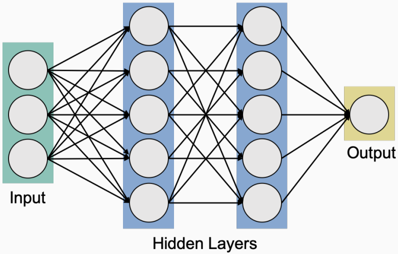
Gradient descent on g:

- For $t = 1, \ldots, T$,
  - $\mathbf{y}^{(t+1)} = \mathbf{y}^{(t)} - \eta \mathbf{P}^2\left[\nabla f(\mathbf{y}^{(t)})\right]$

When **P** is diagonal, this is just gradient descent with a different step size for each parameter!

Algorithms based on this idea:

- AdaGrad
- RMSprop
- Adam optimizer



(Pretty much all of the most widely used optimization methods for training neural networks.)

COORDINATE DESCENT

Main idea: Trade slower convergence (more iterations) for cheaper iterations.

Stochastic Gradient Descent: When $f(\mathbf{x}) = \sum_{i=1}^{n} f_i(\mathbf{x})$, approximate $\nabla f(\mathbf{x})$ with $\nabla f_i(\mathbf{x})$ for randomly chosen i.

Main idea: Trade slower convergence (more iterations) for cheaper iterations.

Stochastic Coordinate Descent: Only compute a single random entry of $\nabla f(\mathbf{x})$ on each iteration:

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(\mathbf{x}) \\ \frac{\partial f}{\partial x_2}(\mathbf{x}) \\ \vdots \\ \frac{\partial f}{\partial x_d}(\mathbf{x}) \end{bmatrix} \qquad \nabla_i f(\mathbf{x}) = \begin{bmatrix} 0 \\ \frac{\partial f}{\partial x_i}(\mathbf{x}) \\ \vdots \\ 0 \end{bmatrix}$$

Update: $\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} + \eta \nabla_i f(\mathbf{x}^{(t)})$.

When $\mathbf{x}$ has d parameters, computing $\nabla_i f(\mathbf{x})$ often costs just a $1/d$ fraction of what it costs to compute $\nabla f(\mathbf{x})$

Example: $f(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$ for $\mathbf{A} \in \mathbb{R}^{n \times d}, \mathbf{x} \in \mathbb{R}^d, \mathbf{b} \in \mathbb{R}^n$.

- $\nabla f(\mathbf{x}) = 2\mathbf{A}^T\mathbf{A}\mathbf{x} - 2\mathbf{A}^T\mathbf{b}$.
- $\nabla_i f(\mathbf{x}) = 2\left[\mathbf{A}^T\mathbf{A}\mathbf{x}\right]_i - 2\left[\mathbf{A}^T\mathbf{b}\right]$.

Stochastic Coordinate Descent:

- Choose number of steps T and step size η.
- For $i = 1, \ldots, T$:
    - Pick random $j_i \in 1, \ldots, d$.
    - $\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \eta \nabla_{j_i} f(\mathbf{x}^{(i)})$
- Return $\hat{\mathbf{x}} = \frac{1}{T} \sum_{i=1}^{T} \mathbf{x}^{(i)}$.

Theorem (Stochastic Coordinate Descent convergence)

Given a G-Lipschitz function f with minimizer $\mathbf{x}^*$ *and initial point* $\mathbf{x}^{(1)}$ *with* $\|\mathbf{x}^{(1)} - \mathbf{x}^*\|_2 \leq R$*, SCD with step size* $\eta = \frac{1}{Rd}$ *satisfies the guarantee:*

$$\mathbb{E}[f(\hat{\mathbf{x}}) - f(\mathbf{x}^*)] \leq \frac{2GR}{\sqrt{T/d}}$$

Often it doesn't make sense to sample i uniformly at random:

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -.5 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & -2 & 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} 10 \\ 42 \\ -11 \\ -51 \\ 34 \\ -22 \end{bmatrix}$$

Select indices i proportional to $\|\mathbf{a}_i\|_2^2$:

$$\Pr[\text{select index } i \text{ to update}] = \frac{\|\mathbf{a}_i\|_2^2}{\sum_{j=1}^d \|\mathbf{a}_j\|_2^2} = \frac{\|\mathbf{a}_i\|_2^2}{\|\mathbf{A}\|_2^2}$$