Gradient Descent and Convergence

Mathematics for Machine Learning

Lecturer: Matthew Wicker

Taught Material Status

- I have written the exam 🎉
- As review, here are the core goals of the class:
  - For you to understand definitions and how they interact in ML contexts (e.g., Hessians + Optimization)
  - For you to use these definitions to analyze a given ML model
  - For you to use course knowledge to come up with mathematical formulations to problems we have not seen

Material covered so far:

Models: Linear models, basis expansion, neural networks

Techniques: Least squares estimation, forward AD, reverse AD, computational graphs

Settings: Regression (discussion of: classification, unsupervised learning)

This lecture: gradient descent, convergence, convexity, Hessians, Lipschitz continuity

Doubling back:

- Matrix shape accounting during AD
- Computational graphs + general reverse-mode
- Complexity of automatic differentiation

Minor Update: Gradients as Row Vectors

Following our taking vectors to be columns by default, we must also take our gradients to be rows. This is also so we are consistent with legacy material

$$f(x) : \mathbb{R}^n \to \mathbb{R}, \quad \frac{\partial f}{\partial x} \in \mathbb{R}^{1\times n}$$

gradient = row vector
by convention

$$f(x) : \mathbb{R}^n \to \mathbb{R}^m, \quad \frac{\partial f}{\partial x} \in \mathbb{R}^{m\times n}$$

If you prefer vectors as rows/gradients as cols, that is fine just be clear when specifically writing out your shapes when asked!

Minor Update: Gradients as Row Vectors

Following our taking vectors to be columns by default, we must also take our gradients to be rows. This is also so we are consistent with legacy material

$$f(x) : \mathbb{R}^n \to \mathbb{R}, \qquad \frac{\partial f}{\partial x} \in \mathbb{R}^{1 \times n}$$

Update to previous slides to correct the identity

$$\nabla_\theta \underbrace{\mathbf{c}^\top \theta}_{\text{vector val fn}} = \mathbf{c}^\top$$

want row vector ∴ use $c^\top$ to get row vector

If you prefer vectors as rows/gradients as cols, that is fine just be clear when specifically writing out your shapes when asked!

Computational Graphs

$$f(x) = \sqrt{x^2 + \exp(x^2)} + cos(x^2 + exp(x^2))$$

Neural networks are actually a particularly easy/nice case for adjoint/reverse mode auto. diff. so let's look at something a little harder!

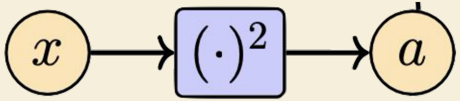
Computational Graphs

$$f(x) = \sqrt{x^2 + \exp(x^2)} + cos\left(x^2 + exp(x^2)\right)$$

1. Identify the elementary operations and look for the ones that occur most and for shared structure. If none, move to step 2
2. We will have variable nodes connected by operation edges. Start with the inputs
3. Start with the most common operations and then build the graph in the order of the computations

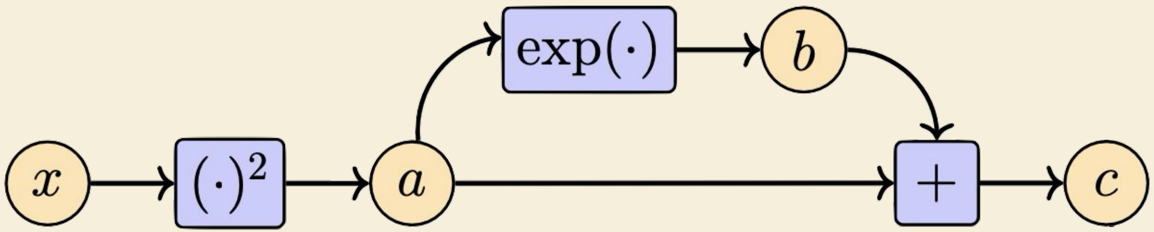
Computational Graphs

$$f(x) = \sqrt{x^2 + \exp(x^2)} + cos\left(x^2 + exp(x^2)\right)$$



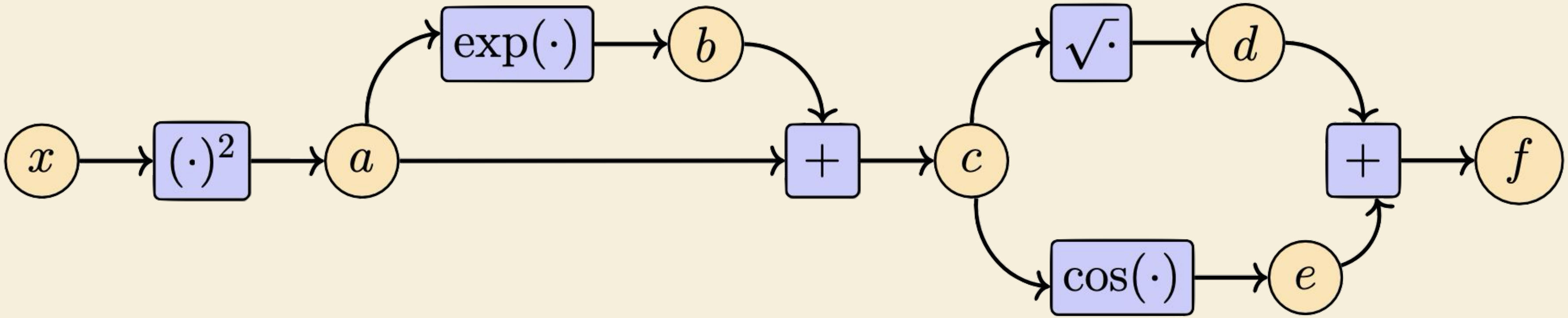
Computational Graphs

$$f(x) = \sqrt{x^2 + \exp(x^2)} + cos\left(x^2 + exp(x^2)\right)$$

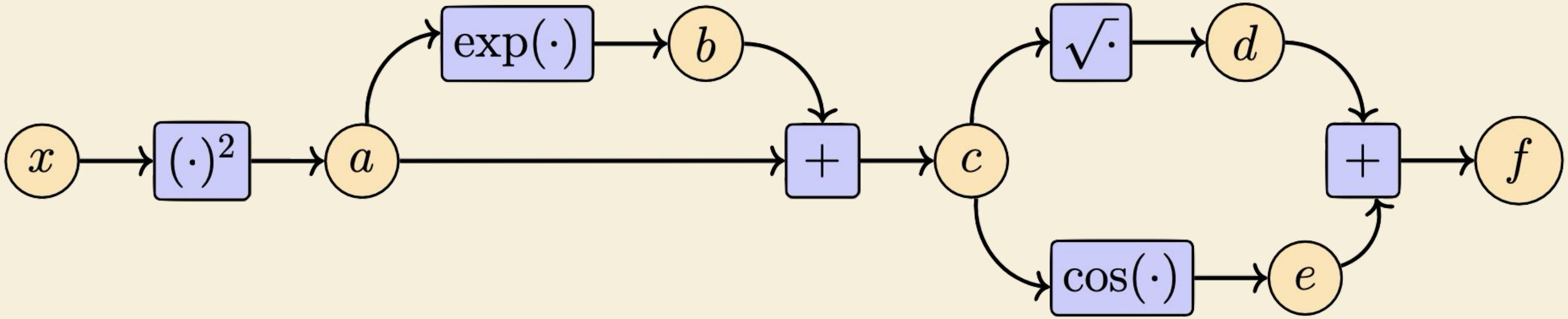


Computational Graphs

$$f(x) = \sqrt{x^2 + \exp(x^2)} + cos\left(x^2 + exp(x^2)\right)$$



Computational Graphs



$$\frac{\partial a}{\partial x} = 2x$$

$b = e^a$
$$\frac{\partial b}{\partial a} = \exp(a)$$

$c = a + b$
$$\frac{\partial c}{\partial a} = 1 = \frac{\partial c}{\partial b}$$

$d = \sqrt{c}$
$$\frac{\partial d}{\partial c} = \frac{1}{2\sqrt{c}}$$

$e = \cos(c)$
$$\frac{\partial e}{\partial c} = -\sin(c)$$

$f = d + e$
$$\frac{\partial f}{\partial d} = 1 = \frac{\partial f}{\partial e}.$$

Make sure that for each of these you can identify the shapes! Here all reals

Computational Graphs

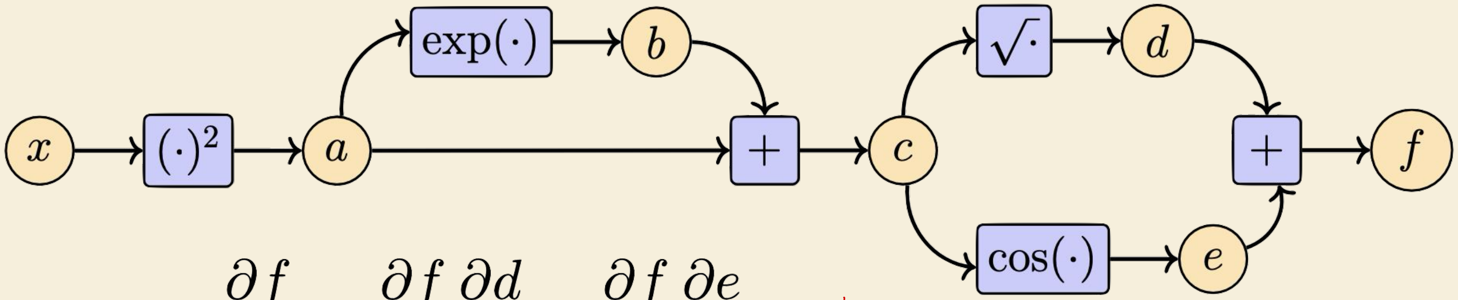
$$f(x) = \sqrt{x^2 + \exp(x^2)} + cos(x^2 + exp(x^2))$$

To compute the derivative we want, we have to back propagate through all of the nodes in the graph using the chain rule for each incoming node and summing them together

$$\frac{\partial f}{\partial x_i} = \sum_{x_j : x_i \in \mathrm{Pa}(x_j)} \frac{\partial f}{\partial x_j} \frac{\partial x_j}{\partial x_i} = \sum_{x_j : x_i \in \mathrm{Pa}(x_j)} \frac{\partial f}{\partial x_j} \frac{\partial g_j}{\partial x_i}$$

Prout of x_j

Computational Graphs



$$\frac{\partial f}{\partial c} = \frac{\partial f}{\partial d}\frac{\partial d}{\partial c} + \frac{\partial f}{\partial e}\frac{\partial e}{\partial c}$$ $= 1 \times \frac{1}{2\sqrt{c}} + 1 \times -\sin(c) = \frac{-\sin(c)}{2\sqrt{c}}$

$$\frac{\partial f}{\partial b} = \frac{\partial f}{\partial c}\frac{\partial c}{\partial b}$$

$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial b}\frac{\partial b}{\partial a} + \frac{\partial f}{\partial c}\frac{\partial c}{\partial a}$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial a}\frac{\partial a}{\partial x}.$$

Here you would use the shapes that you identified before to ensure that you are computing the correct shape of the derivative

$$\frac{\partial a}{\partial x} = 2x$$

$d = \sqrt{c}$ $$\frac{\partial d}{\partial c} = \frac{1}{2\sqrt{c}}$$

$b = e^a$ $$\frac{\partial b}{\partial a} = \exp(a)$$

$e = \cos(c)$ $$\frac{\partial e}{\partial c} = -\sin(c)$$

$c = a+b$ $$\frac{\partial c}{\partial a} = 1 = \frac{\partial c}{\partial b}$$

$f = d+e$ $$\frac{\partial f}{\partial d} = 1 = \frac{\partial f}{\partial e}.$$

Computational Complexity

$$f(x) : \mathbb{R}^n \rightarrow \mathbb{R}^m, \quad \frac{\partial f}{\partial x} \in \mathbb{R}^{m \times n}$$

Finite difference

Algorithm sketch:

- We are given a particular input
- For each input dimension evaluate the model after slightly perturbing the dimension

Complexity: O(2nC(f))

Computational Complexity

$$f(x) : \mathbb{R}^n \to \mathbb{R}^m, \quad \frac{\partial f}{\partial x} \in \mathbb{R}^{m \times n}$$

Finite difference

Algorithm sketch:

- We are given a particular input
- For each input dimension evaluate the model after slightly perturbing the dimension

Complexity: O(2nC(f))

Forward Mode

Algorithm sketch:

- We are given a particular input
- As we compute one forward pass, we also keep track of our derivatives
- Computes a single directional derivative

Complexity: O(2nC(f))

Computational Complexity

$$f(x) : \mathbb{R}^n \to \mathbb{R}^m, \qquad \frac{\partial f}{\partial x} \in \mathbb{R}^{m \times n}$$

Finite difference

Algorithm sketch:
- We are given a particular input
- For each input dimension evaluate the model after slightly perturbing the dimension

Complexity: O(2nC(f))

Forward Mode

Algorithm sketch:
- We are given a particular input
- As we compute one forward pass, we also keep track of our derivatives
- Computes a single directional derivative

Complexity: O(2nC(f))

forward pass w.r.t
Single input

Reverse Mode

Algorithm sketch:
- We are given a particular input
- We compute one forward pass storing all intermediate computations
- We use our output to compute adjoint derivatives backwards through our graph

Complexity: O(2mC(f))

Learning Objectives

- Define convexity + geometric intuition
- Definition of gradient descent
- Proof of convergence of GD in linear models
- Lipschitz continuity + Smoothness
- General convergence theorem

OLS has a direct solution

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

$$\theta^\star = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

$$(X\theta - y)^T (X\theta - y) = \theta^T X^T X \theta - \theta^T X^T y - y^T X \theta + y^T y$$

$$= \theta^T X^T X \theta - 2 y^T X \theta + y^T y$$

$$\nabla_\theta = 2\theta^T X^T X - 2 y^T X = 0$$

$$\theta^T = y^T X (X^T X)^{-1}$$

$$\theta = (X^T X)^{-T} X^T y = (X^T X)^{-1} X^T y$$

$y^T X \theta \to a^T \theta$

diff $\to a^T$

$((X^T X)^{-1})^T = ((X^T X)^T)^{-1}$ if matrix is square and invertible

We saw in previous lectures that for the ordinary least squares formulation of linear regression, we have a direct solution for the best parameter(s)

Ridge regression does not

$$\mathcal{L}_{\text{lasso}}(\theta) = ||\mathbf{X}\theta - \mathbf{y}||_2^2 + \overbrace{\lambda \sum |\theta_i|}^{\text{penalty for large valued params}}$$

$$\text{no closed form } \theta^\star$$

However, desirable modifications of our objective leave us with no closed form for the best parameters. But, we know that a global minimizer can be (easily) found

Ridge regression does not

$$\mathcal{L}_{\text{lasso}}(\theta) = ||\mathbf{X}\theta - \mathbf{y}||_2^2 + \lambda \sum |\theta_i|$$

We know that a global minimizer can be quickly found because the parameter is constrained in a *convex set*, the mean squared error is *convex*, and the added loss term is *convex*.

Convex Functions

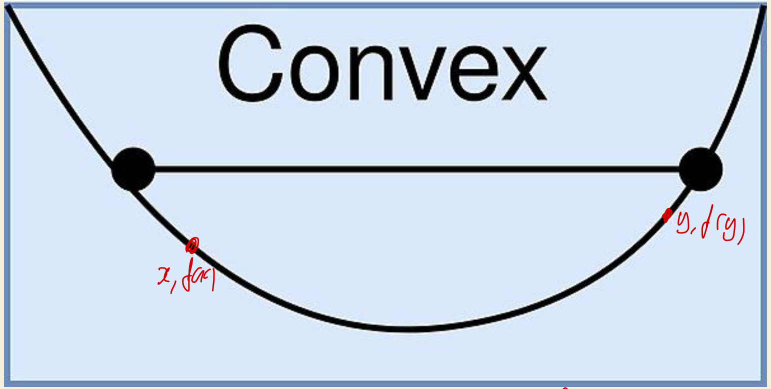
domain

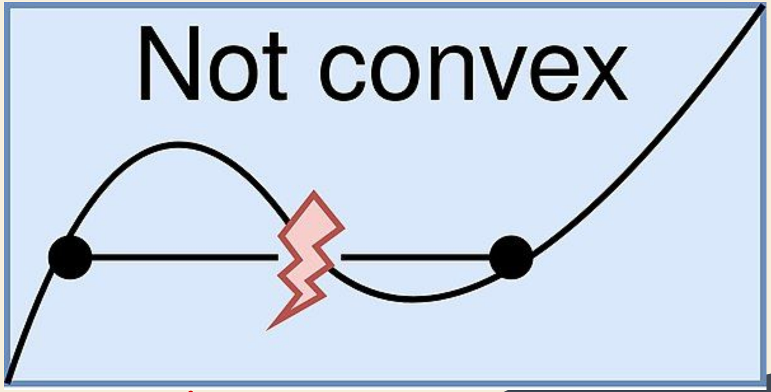
$$\forall x, y \in \mathrm{dom}(f), \forall t \in [0,1]$$

$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y)$

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y)$$

$t=1$: $f(x) \leq f(x)$





take 2 point, x & y, f evaluated at any combo of x & y should be ≤ same combo of f(x) & f(y)

Strictly Convex Functions

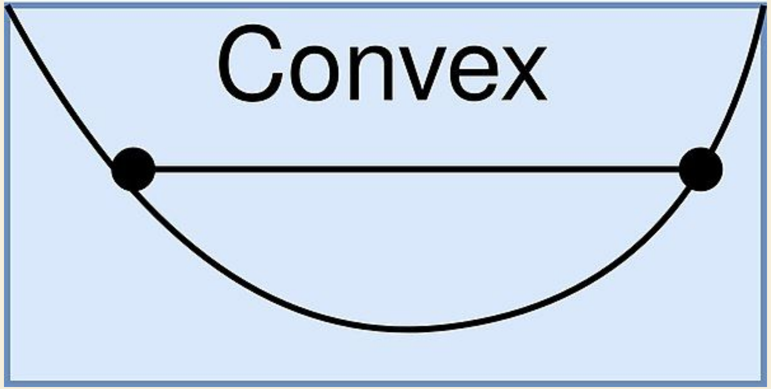
for points x,y in domain of f,

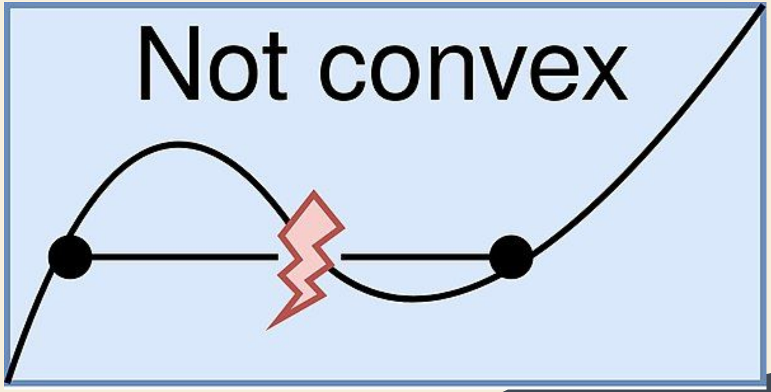
$$\forall x, y \in \mathrm{dom}(f), \forall t \in [0,1]$$

ie $0 \le t \le 1$

$$f(tx + (1-t)y) < tf(x) + (1-t)f(y)$$

$f(0.8x + 0.2y) < 0.8f(x) + 0.2f(y)$



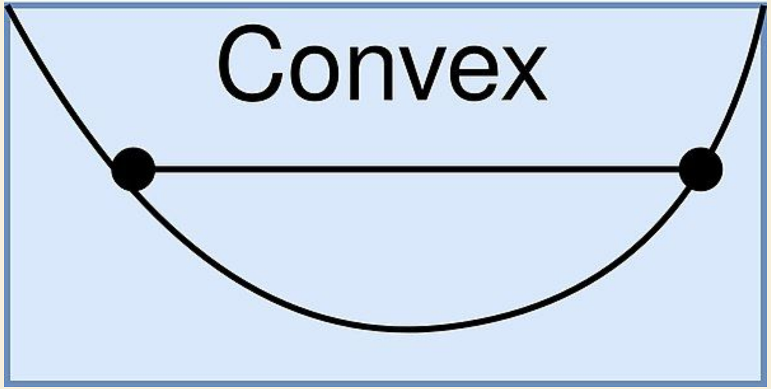


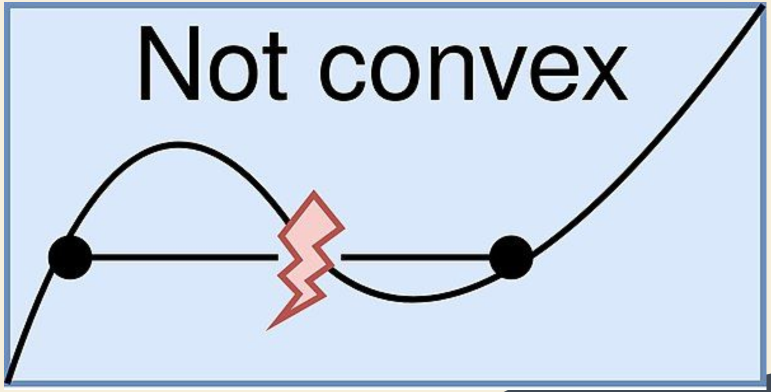
Strictly Convex Functions

$$\forall x, y \in \boxed{\mathrm{dom}(f)}, \forall t \in [0, 1]$$

Needs to be a convex set

$$f(tx + (1-t)y) < tf(x) + (1-t)f(y)$$

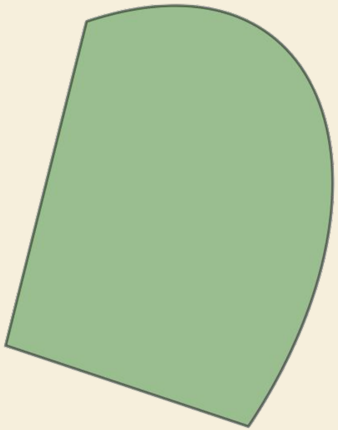


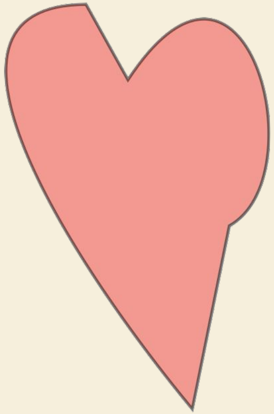


Convex Sets

$C \subseteq \mathbb{R}^n$ is convex if $\forall x, y \in C$ and $\forall t \in 0 \leq t \leq 1$

$$tx + (1-t)y \in C$$



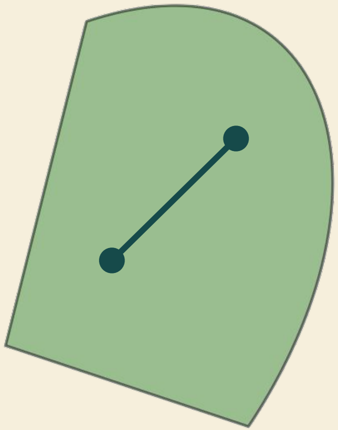


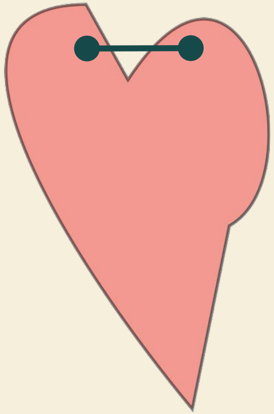
Convex Sets

$$C \subseteq \mathbb{R}^n \text{ is convex if } \forall x, y \in C \text{ and } \forall t \in 0 \leq t \leq 1$$

$$tx + (1-t)y \in C$$

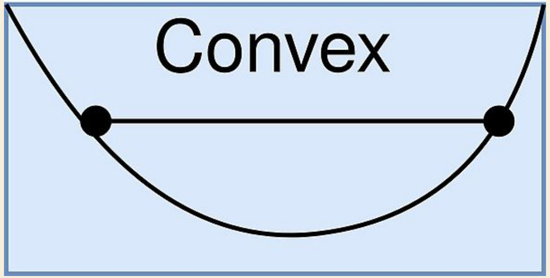
combo of x & y should be in set





Properties of convex functions

- Sum of convex functions is convex
- If a function and its negation are both convex, the function is affine ← linear transformation, doesn't preserve the origin
- The maximum of two convex functions is convex
- Scalar multiples of convex functions are convex



Properties of convex functions

- Sum of convex functions is convex
- If a function and its negation are both convex, the function is affine
- The maximum of two convex functions is convex
- *Scalar multiples of convex functions are convex*

Simply plug the modification $\alpha f(x)$ into the definition of convexity and check it still holds:

Properties of convex functions

- Sum of convex functions is convex
- If a function and its negation are both convex, the function is affine
- The maximum of two convex functions is convex
- *Scalar multiples of convex functions are convex*

Simply plug the modification $\alpha f(x)$ into the definition of convexity and check it still holds:

$$\alpha f(tx + (1-t)y) \leq t\alpha f(x) + (1-t)\alpha f(y)$$

Convex and differentiable functions

We started by observing there is no closed form for the lasso objective because it is not differentiable. But if a function is convex and differentiable, then there is even more we can say!

$f : \mathbb{R}^n \to \mathbb{R}$ is convex iff

$$f(y) \geq f(x) + \nabla f(x)(y - x) \quad \forall x, y \in \mathbb{R}^n$$

Suppose $f : R^n \to R$ is twice differentiable over an open domain. Then, the following are equivalent:

(i) f is convex.

(ii) $f(y) \geq f(x) + \nabla f(x)^T (y - x)$, for all $x, y \in \mathrm{dom}(f)$.

(iii) $\nabla^2 f(x) \succeq 0$, for all $x \in \mathrm{dom}(f)$.

Convex and differentiable functions

$f : \mathbb{R}^n \to \mathbb{R}$ is convex iff

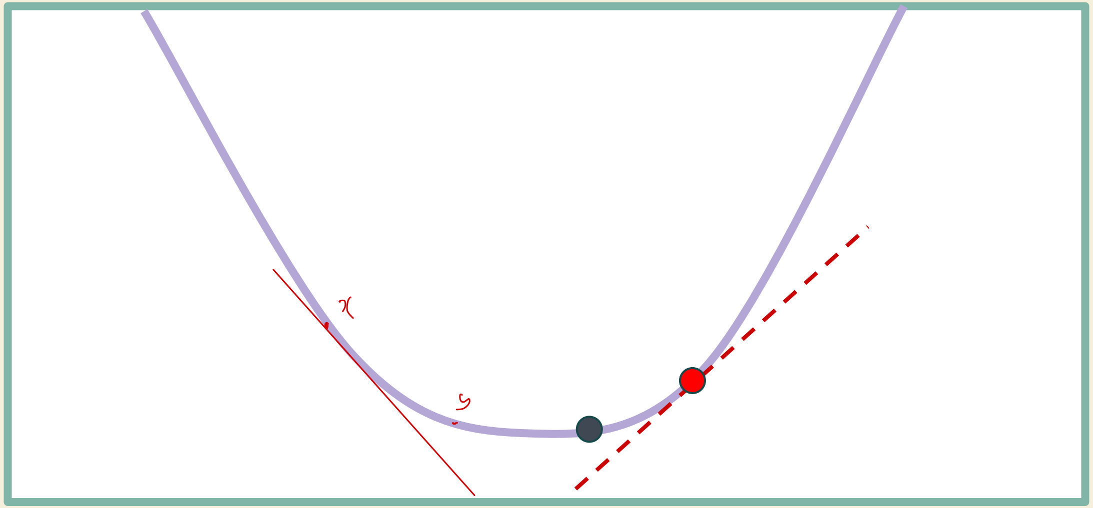
predicted change in f^n when going from x to y

$$f(y) \geq f(x) + \overbrace{\nabla f(x)(y - x)}$$

Before we defined convexity in terms of the functions secant. Here we providing the same definition in terms of the tangent

touches f^n twice

touches f^n once

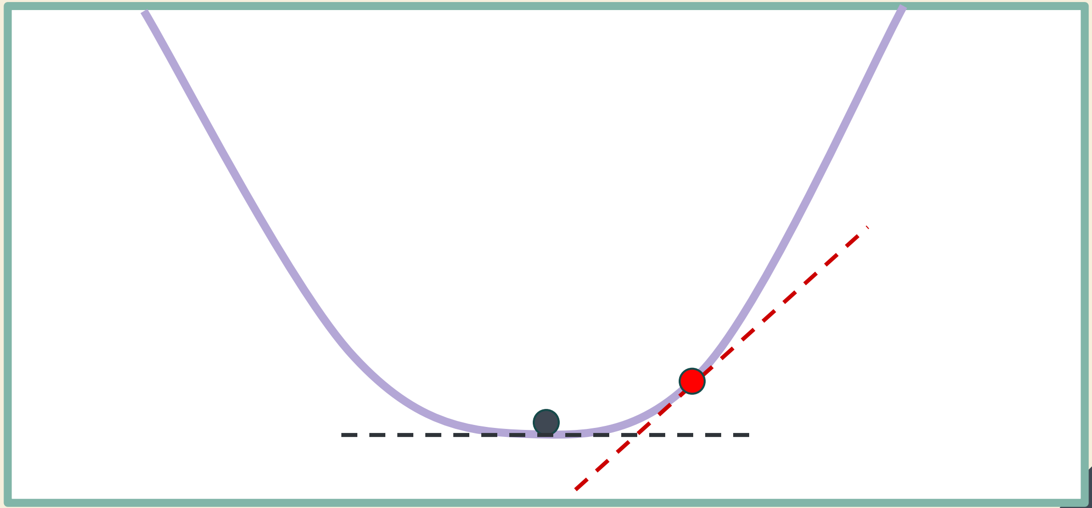


tangent at x is below f^n as you move to y

Global optimality from convexity

$f : \mathbb{R}^n \to \mathbb{R}$ $\qquad f(y) \geq f(x) + \nabla f(x)(y - x)$

$x^\star$ is a global minimizer iff $\nabla f(x^\star) = 0$



Global optimality from convexity

$f : \mathbb{R}^n \to \mathbb{R}$ $\qquad f(y) \geq f(x) + \nabla f(x)(y - x)$

$$x^\star \text{ is a global minimizer iff } \nabla f(x^\star) = 0$$

Proof:

$$f(y) \geq f(x^\star) + \nabla f(x^\star)(y - x^\star)$$

Global optimality from convexity

$$f : \mathbb{R}^n \to \mathbb{R} \qquad f(y) \geq f(x) + \nabla f(x)(y - x)$$

$$x^\star \text{ is a global minimizer iff } \nabla f(x^\star) = 0$$

Proof:

$$f(y) \geq f(x^\star) + \nabla f(x^\star)(y - x^\star)$$

$$\nabla f(x^\star) = 0$$

Global optimality from convexity

$$f : \mathbb{R}^n \to \mathbb{R} \qquad\qquad f(y) \geq f(x) + \nabla f(x)(y - x)$$

$$x^\star \text{ is a global minimizer iff } \nabla f(x^\star) = 0$$

Proof:

$$f(y) \geq f(x^\star) + \overbrace{\nabla f(x^\star)}^{=0}(y - x^\star) = f(x^\star)$$

$$\nabla f(x^\star) = 0$$

$\therefore f(y) \geq f(x^\star) \quad \therefore x^\star$ must be global min

Defining the Hessian

$$f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}, \qquad \nabla^2_{\mathbf{x}} f(\mathbf{x}) \in \mathbb{R}^{n \times n}$$

Example:

Defining the Hessian

$$f(\mathbf{x}) : \mathbb{R}^2 \to \mathbb{R}, \quad \nabla^2_{\mathbf{x}} f(\mathbf{x}) \in \mathbb{R}^{2\times 2}$$

Example:

$$\nabla^2_{[x_1, x_2]} f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2}, & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_1 \partial x_2}, & \frac{\partial^2 f}{\partial x_2^2} \end{bmatrix}$$

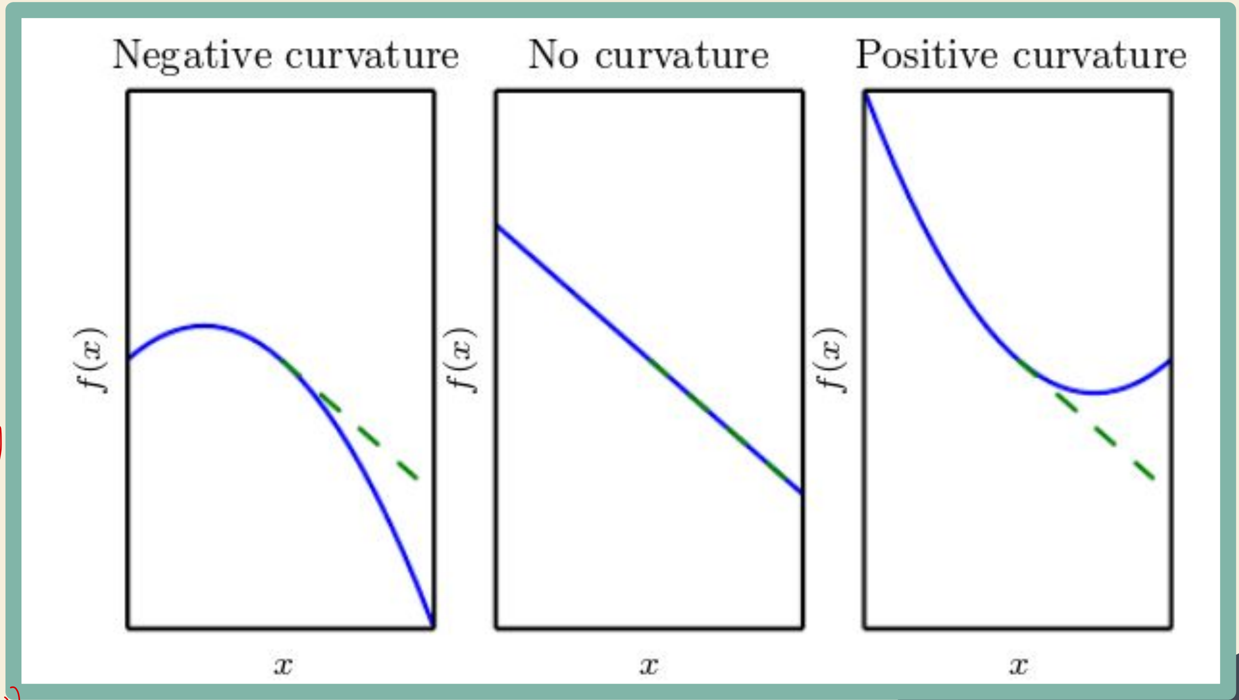
$$\frac{\partial^2 f}{\partial x_i \partial x_j}$$

Visualizing the Hessian

If the derivative does not change at all then the Hessian is zero and we have no curvature

If the hessian is positive then the derivative is "speeding up" and we get positive curvature (min at $\nabla f = 0$)

If the hessian is negative then the derivative is "slowing down" and we get negative curvature (max at $\nabla f = 0$)

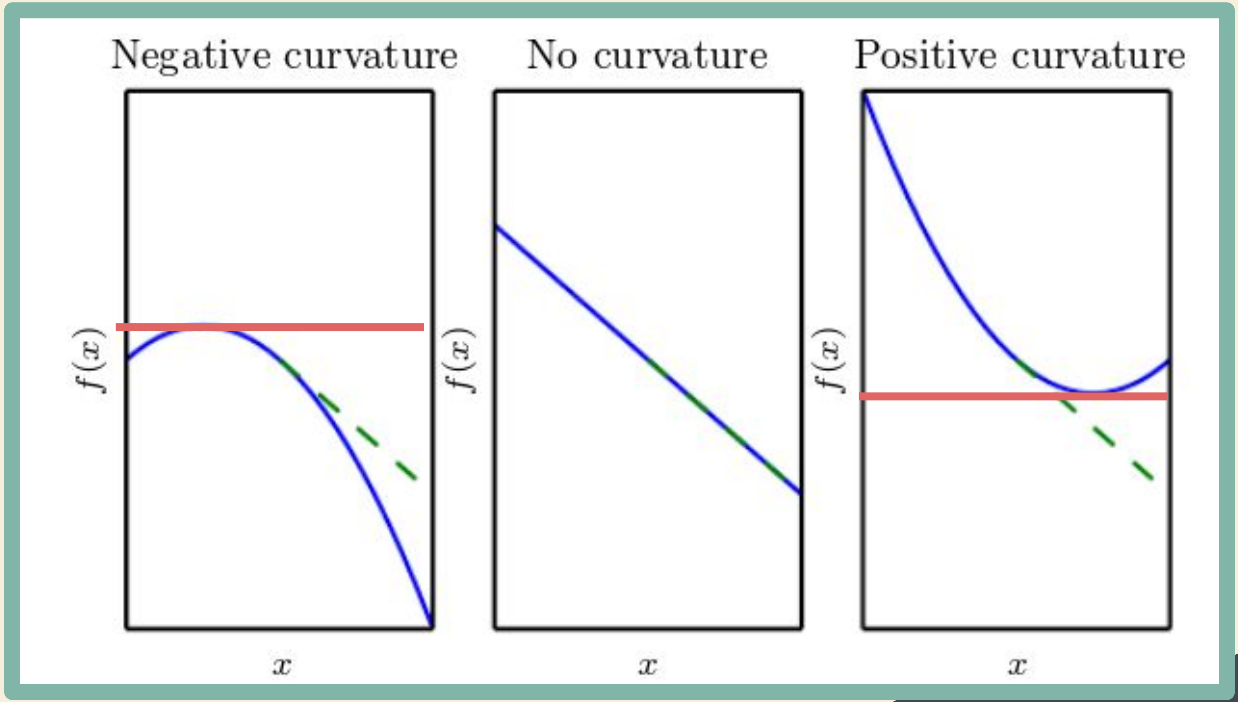


Checking local extrema

If the gradient of our function is zero at a point then the Hessian can tell us if we are at an extremum and which kind:

positive semi-definite : $\lambda \geq 0$

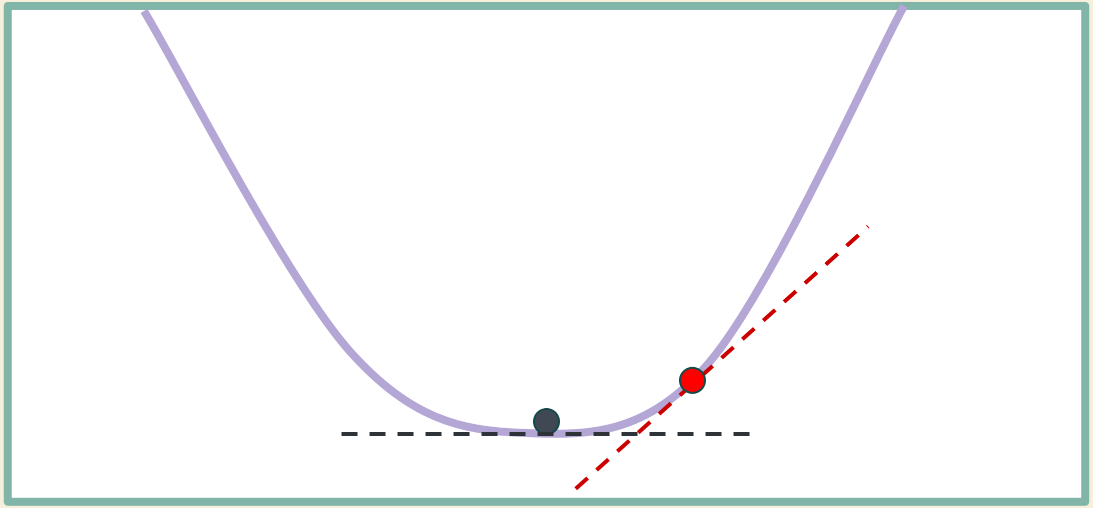
* PSD Hessian - Minimum
$\lambda \leq 0$ * NSD Hessian - Maximum
* Indefinite Hessian - Saddle point



Hessians relationship to convexity

A twice differentiable function is convex if and only if the Hessian is PSD for every input in the domain.

$\nabla^2 f = PSD$



Break time!

Gradient Descent as an Algorithm

Algorithm 1 Gradient Descent

Input: X - Inputs, Y - Labels, γ - Learning rate, K - Number of iterations

$\theta_1 \leftarrow$ Random Initialization
for $i \in [K]$ **do**
  $l = \mathcal{L}(Y, f^{\theta}(X))$ find loss of out & prediction
  $\theta_{i+1} \leftarrow \theta_i - \gamma \nabla_{\theta} l$ new: old - γ x diff. loss w.r.t params
end for
return θ_K

The gradient of a function tells us the direction of steepest ascent. Of course, we want to find the minimum so we move in the opposite (negative) direction of the gradient.

Gradient Descent as an Algorithm

Algorithm 1 Gradient Descent

Input: X - Inputs, Y - Labels, γ - Learning rate, K - Number of iterations

$\theta_1 \leftarrow$ Random Initialization
for $i \in [K]$ **do**
  $l = \mathcal{L}(Y, f^{\theta}(X))$
  $\theta_{i+1} \leftarrow \theta_i - \gamma \nabla_{\theta} l$
end for
return θ_K

For convex functions, gradient descent converges to the global optimum in finite time with bounded error!
So for our lasso objective, we can still find the optimal parameter

What do we mean with convergence?

Definition 2.1. Convergence A series $x_1, x_2, \ldots, x_n$ is said to *converge* to a limit L if for any $\epsilon > 0$ we have an integer K such that $\forall M > K, |x_M - L| < \epsilon$.

all points in a series beyond a certain index ie $x_{K+1}, x_{K+2}, \ldots, x_n$ are less than ϵ from the limit

less than a number

What do we mean with convergence?

Definition 2.1. Convergence A series $x_1, x_2, \ldots, x_n$ is said to *converge* to a limit L if for any $\boxed{\epsilon > 0}$ we have an integer K such that $\forall M > K, |x_M - L| < \epsilon$.

- We get arbitrarily close to the limit

What do we mean with convergence?

Definition 2.1. Convergence A series $x_1, x_2, \ldots, x_n$ is said to *converge* to a limit L if for any $\boxed{\epsilon > 0}$ we have an integer K such that $\boxed{\forall M > K, |x_M - L| < \epsilon.}$

- We get arbitrarily close to the limit
- We stay at least arbitrarily close for the rest of the sequence

What is our sequence?

Algorithm 1 Gradient Descent

Input: X - Inputs, Y - Labels, γ - Learning rate, K - Number of iterations

$\theta_1 \leftarrow$ Random Initialization

for $i \in [K]$ **do**

$l = \mathcal{L}(Y, f^{\theta}(X))$

$\theta_{i+1} \leftarrow \theta_i - \gamma \nabla_{\theta} l$

end for

return θ_K

updating θ

What is our sequence?

Algorithm 1 Gradient Descent

Input: X - Inputs, Y - Labels, γ - Learning rate, K - Number of iterations

---

$\theta_1 \leftarrow$ Random Initialization
for $i \in [K]$ **do**
  $l = \mathcal{L}(Y, f^\theta(X))$
  $\theta_{i+1} \leftarrow \theta_i - \gamma \nabla_\theta l$
end for
return θ_K

$$\theta_0, \theta_1, \ldots, \theta_K$$

But why would this converge?

Proving gradient convergence in linear regression

For this loss: $$\mathcal{L}(\theta) = \frac{1}{2}||\mathbf{y} - \mathbf{X}\theta||_2^2$$

We hope to converge to: $$(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}$$

Proving gradient convergence in linear regression

$$\theta_{i+1} = \theta_i - \gamma_i \nabla_{\theta_i} \mathcal{L}(\underbrace{\mathbf{y}, f^{\theta_i}(\mathbf{X})}_{\text{loss between y and estimate}})$$

Proving gradient convergence in linear regression

$$\begin{bmatrix} \hat{y}^{(1)}, \\ \hat{y}^{(2)}, \\ \dots, \\ \hat{y}^{(N)}, \end{bmatrix} = \begin{bmatrix} x_1^{(1)}, \dots, x_n^{(1)} \\ x_1^{(2)}, \dots, x_n^{(2)} \\ \vdots, \ddots, \vdots \\ x_1^{(N)}, \dots, x_n^{(N)} \end{bmatrix} \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

$f(x) = X\theta$

pred $= X\theta$

loss $= (y - \hat{y})^2 = (y - X\theta)^2$

$$\theta_{i+1} = \theta_i - \gamma_i \nabla_{\theta_i} \mathcal{L}(\mathbf{y}, f^{\theta_i}(\mathbf{X}))$$

$$= \theta_i - \gamma \nabla_{\theta_i} \frac{1}{2} ||\mathbf{y} - \mathbf{X}\theta_i||_2^2$$

$(L2 \text{ norm})^2$

ie $||[a \quad b \quad c]||_2^2$

$= \left(\sqrt{a^2+b^2+c^2}\right)^2$

$= a^2+b^2+c^2$

ie Squaring and adding all values

$(y - X\theta)^T (y - X\theta)$

$= y^T y - 2y^T X\theta + \theta^T X^T X\theta$

$\nabla_\theta: \quad -2y^T X + 2\theta^T X^T X = d'$

$\to -X^T y + X^T X\theta = X^T(X\theta - y)$

Proving gradient convergence in linear regression

$$
\begin{aligned}
\theta_{i+1} &= \theta_i - \gamma_i \nabla_{\theta_i} \mathcal{L}(\mathbf{y}, f^{\theta_i}(\mathbf{X})) \\
&= \theta_i - \gamma \nabla_{\theta_i} \frac{1}{2} ||\mathbf{y} - \mathbf{X}\theta_i||_2^2 \\
&= \theta_i - \gamma \mathbf{X}^\top (\mathbf{X}\theta_i - \mathbf{y})
\end{aligned}
$$

$X: m \times n$
$\theta: n \times 1$
$y: m \times 1$

$X\theta: m \times n \cdot n \times 1 : m \times 1$
$X^T(\cdot): n \times m \cdot m \times 1 = n \times 1$

$\|y - X\theta\|_2^2 = (y - X\theta)^T (y - X\theta) = (y^T - \theta^T X^T)(y - X\theta) = y^T y - y^T X\theta - \theta^T X^T y + \underbrace{\theta^T X^T X \theta}_{\theta^T A \theta \to A\theta + A^T\theta}$

$= X^T X\theta + (X^T X)^T\theta$
$= X^T X\theta + X^T X\theta$

$= y^T y - 2y^T X\theta + \theta^T X^T X\theta$ $\quad \frac{\partial}{\partial\theta} \to -2y^T X + 2\theta^T X^T X = d^T \to -X^T y + X^T X\theta$

θ: [illegible]

Proving gradient convergence in linear regression

$$= \theta_i - \gamma \mathbf{X}^\top (\mathbf{X}\theta_i - \mathbf{y})$$

$$= (\mathbf{I} - \gamma \mathbf{X}^\top \mathbf{X})\theta_i + \gamma \mathbf{X}^\top \mathbf{y}$$

$= \theta_i - \gamma X^T X \theta_i - \gamma X^T y$

$= (I - \gamma X^T X)\theta_i + \gamma X^T y$

Proving gradient convergence in linear regression

$$\theta_{i+1} = \theta_i - \gamma \mathbf{X}^\top(\mathbf{X}\theta_i - \mathbf{y})$$

$$\theta_{i+1} = (\mathbf{I} - \gamma \underbrace{\mathbf{X}^\top}_{} \mathbf{X})\theta_i + \gamma \mathbf{X}^\top \mathbf{y}$$

m×n n×m : m×m

m×n n×1 : m×1

$$\underset{m \times m}{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \vdots & & & \end{bmatrix}} \underset{m \times 1}{\begin{bmatrix} a \\ b \\ \vdots \end{bmatrix}} = \begin{bmatrix} a \\ b \\ c \\ \vdots \end{bmatrix}$$

F θ

Multiplying by this matrix

Adding this vector

Update structure: Arithmetico-Geometric Series!

$$\underbrace{\boxed{(\mathbf{I} - \gamma \mathbf{X}^\top \mathbf{X})}}_{\text{matrix}}\theta_i + \underbrace{\boxed{\gamma \mathbf{X}^\top \mathbf{y}}}_{\text{vector}}$$

$$\boxed{\mathbf{B}}\theta_t + \boxed{\mathbf{c}}$$

n×n n×1 n×1

Update structure

$$\mathbf{B}\theta_t + \mathbf{c}$$

$$\mathbf{B}(\mathbf{B}(\mathbf{B}\theta + \mathbf{c}) + \mathbf{c}) + \mathbf{c}$$

While the structure is simple, when we expand out and look at the kind of update we are doing, it *could* be nicer. In particular, getting rid of that pesky 'c'

Nicer structure

This is brilliant: $$\mathbf{B}\theta_t + \mathbf{c}$$

I want to go from top to bottom

But I prefer this: $$\mathbf{A}(\theta_t + \beta) - \beta$$

Nicer structure

Applying this update multiple times: $\mathbf{A}(\theta_t + \beta) - \beta$

$$\theta_t = \mathbf{A}^t(\theta_1 + \beta) - \beta$$

$\theta_1 = A(\theta_0 + \beta) - \beta$

$\theta_2 = A[(\theta_1) + \beta] - \beta = A[(A(\theta_0 + \beta) - \cancel{\beta}) \cancel{+\beta}] - \beta$

$= A^2(\theta_0 + \beta) - \beta$

$\theta_3 = A[(\theta_2) + \beta] - \beta = A[(A^2(\theta_0 + \beta) - \cancel{\beta}) \cancel{+\beta}] - \beta = A^3(\theta_0 + \beta) - \beta$

$A(\theta_t + \beta) - \beta$

$\to A^t(\theta_0 + \beta) - \beta$

Algebra to convert structures

$$\mathbf{A}(\theta_t + \beta) - \beta = \mathbf{B}\theta_t + \mathbf{c}$$

$$A\theta_t + A\beta - \beta = A\theta_t + (A - I)\beta$$

$$\mathbf{A}\theta_t + (\mathbf{A} - \mathbf{I})\beta = \mathbf{B}\theta_t + \mathbf{c}$$

Algebra to convert structures

$$\mathbf{A}(\theta_t + \beta) - \beta = \mathbf{B}\theta_t + \mathbf{c}$$

$$\boxed{\mathbf{A}}\theta_t + \boxed{(\mathbf{A} - \mathbf{I})\beta} = \mathbf{B}\theta_t + \mathbf{c}$$

$A = B$

$(A - I)\beta = c$

$\beta = (A - I)^{-1} c$

$= (B - I)^{-1} c$

Algebra to convert structures

$$\mathbf{A}(\theta_t + \beta) - \beta = \mathbf{B}\theta_t + \mathbf{c}$$

$$\boxed{\mathbf{A}}\theta_t + \boxed{(\mathbf{A} - \mathbf{I})\beta} = \mathbf{B}\theta_t + \mathbf{c}$$

$$\mathbf{A} = \mathbf{B}, \quad \beta = (\mathbf{B} - \mathbf{I})^{-1}\mathbf{c}$$

$B(\theta_t + (B-I)^{-1}c) - (B-I)^{-1}c$

use substitutions: $A(\theta_t + \beta) - \beta \to B(\theta_t + (B-I)^{-1}c) - (B-I)^{-1}c$

Algebra to convert structures

$$\mathbf{A} = \mathbf{B}, \quad \beta = (\mathbf{B} - \mathbf{I})^{-1}\mathbf{c}$$

$$\theta_{t+1} = \overset{A}{\mathbf{B}}(\overset{\theta_c}{\theta_t} + \overset{\beta}{(\mathbf{B} - \mathbf{I})^{-1}\mathbf{c}}) \overset{-}{-} \overset{\beta}{(\mathbf{B} - \mathbf{I})^{-1}\mathbf{c}}$$

Algebra to convert structures

$$\mathbf{A} = \mathbf{B}, \quad \beta = (\mathbf{B} - \mathbf{I})^{-1}\mathbf{c}$$

$$\theta_{t+1} = \mathbf{B}(\theta_t + (\mathbf{B} - \mathbf{I})^{-1}\mathbf{c}) - (\mathbf{B} - \mathbf{I})^{-1}\mathbf{c}$$

$$\theta_t = \mathbf{B}^t(\theta_0 + (\mathbf{B} - \mathbf{I})^{-1}\mathbf{c}) - (\mathbf{B} - \mathbf{I})^{-1}\mathbf{c}$$

$$\mathbf{A}(\theta_t + \beta) - \beta$$

What were B and c again?

$$\theta_t = \mathbf{A}^t(\theta_1 + \beta) - \beta$$

Algebra to convert structures

$$\boxed{(\mathbf{I} - \gamma \mathbf{X}^\top \mathbf{X})}\theta_i + \boxed{\gamma \mathbf{X}^\top \mathbf{y}}$$

$$\boxed{\mathbf{B}}\theta_t + \boxed{\mathbf{c}}$$

$A(\theta_t + \beta) - \beta$, $A = B = (I - \gamma X^\top X)$

$\beta = (B - I)^{-1} c = ((I - \gamma X^\top X) - I)^{-1} \gamma X^\top y$

Algebra to convert structures

$$\underbrace{\boxed{(\mathbf{I} - \gamma \mathbf{X}^\top \mathbf{X})}}_{=B}\theta_i + \underbrace{\boxed{\gamma \mathbf{X}^\top \mathbf{y}}}_{=C} \quad \to B\theta_i + C,$$

$A = B, \quad \beta = (B - I)^{-1} C$

$$\mathbf{A} = (\mathbf{I} - \gamma \mathbf{X}^\top \mathbf{X}) = B$$

$$\beta = ((\mathbf{I} - \gamma \mathbf{X}^\top \mathbf{X}) - \mathbf{I})^{-1} \gamma \mathbf{X}^\top \mathbf{y} = (B - I)^{-1} C$$

Let's simplify beta to make our lives easier

Algebra to convert structures

$$\beta = ((\mathbf{I} - \gamma \mathbf{X}^\top \mathbf{X}) - \mathbf{I})^{-1} \gamma \mathbf{X}^\top \mathbf{y}$$

$$((\cancel{\mathbf{I}} - \gamma \mathbf{X}^\top \mathbf{X}) \cancel{- \mathbf{I}})^{-1} \gamma \mathbf{X}^\top \mathbf{y} = (-\gamma \mathbf{X}^\top \mathbf{X})^{-1} \gamma \mathbf{X}^\top \mathbf{y}$$

$$= \underbrace{-(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}}_{-\text{ve of LS solution}}$$

Algebra to convert structures

$$\beta = ((\mathbf{I} - \gamma \mathbf{X}^\top \mathbf{X}) - \mathbf{I})^{-1} \gamma \mathbf{X}^\top \mathbf{y}$$

$$((\mathbf{I} - \gamma \mathbf{X}^\top \mathbf{X}) - \mathbf{I})^{-1} \gamma \mathbf{X}^\top \mathbf{y} = (-\gamma \mathbf{X}^\top \mathbf{X})^{-1} \gamma \mathbf{X}^\top \mathbf{y}$$

$$\boxed{= -(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}}$$

We hope to converge to: $(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$

Plugging back into our desired form

$$\mathbf{A} = (\mathbf{I} - \gamma \mathbf{X}^\top \mathbf{X}) \qquad \beta = -(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

$$\theta^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

$A(\theta_0 + \beta) - \beta \to A^t(\theta_0 + \beta) - \beta = \ldots$ $\qquad \theta^* = -\beta$

$$\theta_t = \underbrace{(\mathbf{I} - \gamma \mathbf{X}^\top \mathbf{X})^t}_{A^t} (\theta_0 \underbrace{- \theta^*}_{+\beta}) \underbrace{+ \theta^*}_{-\beta}$$

Condition for convergence

$$\theta_t = (\mathbf{I} - \gamma \mathbf{X}^\top \mathbf{X})^t (\theta_0 - \theta^*) + \theta^*$$

Convergence if:

$$(\mathbf{I} - \gamma \mathbf{X}^\top \mathbf{X})^t (\theta_0 - \theta^*) \to \mathbf{0}$$

as t $\theta^* = (X^T X)^{-1} X^T y$ = LS solution

How do we reason about this?

$$||\theta_t - \theta^*||_2^2 = ||(\mathbf{I} - \gamma \mathbf{X}^\top \mathbf{X})^t (\theta_0 - \theta^*)||_2^2$$
$$= |(\theta_0 - \theta^*)^\top (\mathbf{I} - \gamma \mathbf{X}^\top \mathbf{X})^{2t} (\theta_0 - \theta^*)|$$

This kind of thing we have
proven before (lecture 2)
proving it is a nice exercise

$||A^t B||_2^2 = (A^t B)^T (A^t B) = B^T A A^T B$

$A = (I - \gamma x^T x)^t, \quad A^T = (I - \gamma x^T x)^t$

$\rightarrow (\theta_0 - \theta^*)^T (I - \gamma x^T x)^{2t} (\theta_0 - \theta^*)$

How do we reason about this?

$$
\begin{aligned}
||\theta_t - \theta^*||_2^2 &= ||(\mathbf{I} - \gamma \mathbf{X}^\top \mathbf{X})^t (\theta_0 - \theta^*)||_2^2 \\
&= |(\theta_0 - \theta^*)^\top (\mathbf{I} - \gamma \mathbf{X}^\top \mathbf{X})^{2t} (\theta_0 - \theta^*)|
\end{aligned}
$$

What is the form of this?

How do we reason about this?

$$||\theta_t - \theta^*||_2^2 = ||(\mathbf{I} - \gamma \mathbf{X}^\top \mathbf{X})^t (\theta_0 - \theta^*)||_2^2$$
$$= |(\theta_0 - \theta^*)^\top (\mathbf{I} - \gamma \mathbf{X}^\top \mathbf{X})^{2t} (\theta_0 - \theta^*)|$$

What is the form of this? $\mathbf{x}^\top \mathbf{A} \mathbf{x}$

We know bounds on this from linear alg. review!

$$\lambda_{min}(\mathbf{A})||\mathbf{x}||_2^2 \leq \mathbf{x}^\top \mathbf{A} \mathbf{x} \leq \lambda_{max}(\mathbf{A})||\mathbf{x}||_2^2$$

Eigenvalues!

Applying eigenvalue bound in our case

We know bounds on this from linear alg. review!

$$\lambda_{min}(\mathbf{A})||\mathbf{x}||_2^2 \leq \mathbf{x}^\top \mathbf{A}\mathbf{x} \leq \lambda_{max}(\mathbf{A})||\mathbf{x}||_2^2$$

Eigenvalues!

lower bound

$$||\theta_t - \theta^*||_2^2 \geq \lambda_{min}((\mathbf{I} - \gamma\mathbf{X}^\top\mathbf{X})^{2t})||\theta_0 - \theta^*||_2^2$$

$$||\theta_t - \theta^*||_2^2 \leq \lambda_{max}((\mathbf{I} - \gamma\mathbf{X}^\top\mathbf{X})^{2t})||\theta_0 - \theta^*||_2^2$$

upper bound

Rules for convergence

$$||\theta_t - \theta^*||_2^2 \geq \lambda_{min}((\mathbf{I} - \gamma \mathbf{X}^\top \mathbf{X})^{2t})||\theta_0 - \theta^*||_2^2$$

$$||\theta_t - \theta^*||_2^2 \leq \lambda_{max}((\mathbf{I} - \gamma \mathbf{X}^\top \mathbf{X})^{2t})||\theta_0 - \theta^*||_2^2$$

1. $\lambda_{max} < 1$: always converge
2. $\lambda_{min} \geq 1$: always diverge
3. $\lambda_{min} < 1$ but $\lambda_{max} \geq 1$: convergence depending on θ_0

So just cross our fingers?

1. $\lambda_{max} < 1$: always converge
2. $\lambda_{min} \geq 1$: always diverge
3. $\lambda_{min} < 1$ but $\lambda_{max} \geq 1$: convergence depending on θ_0

$\therefore$ want $\lambda_{max} < 1$

$\lambda_{min} < 1$

We have control over selecting our learning rate!

$$||\theta_t - \theta^*||_2^2 \geq \lambda_{min}((\mathbf{I} - \gamma\mathbf{X}^\top\mathbf{X})^{2t})||\theta_0 - \theta^*||_2^2$$

Picking the right learning rate

We can compute the eigenvalues of: $\mathbf{X}^\top \mathbf{X}$

We want to know eigenvalues of: $(\mathbf{I} - \gamma \mathbf{X}^\top \mathbf{X})^2$

Picking the right learning rate

$X^TX \to \lambda$

$I - \gamma X^TX \to 1 - \gamma\lambda$

$(I - \gamma X^TX)^2 \to (1 - \gamma\lambda)^2$

We can compute the eigenvalues of: $\mathbf{X}^\top \mathbf{X}$

We want to know eigenvalues of: $(\mathbf{I} - \gamma \mathbf{X}^\top \mathbf{X})^2$

If λ is an eigenvalue of $\mathbf{X}^\top \mathbf{X}$

$\gamma X^TX \to \gamma\lambda$, $I - \gamma X^TX \to I - \gamma\lambda$

$(I - \gamma X^TX)^2 \to (I - \gamma\lambda)^2$

Then $(1 - \gamma\lambda)^2$ is an eigenvalue of $(\mathbf{I} - \gamma \mathbf{X}^\top \mathbf{X})^2$

Picking the right learning rate

If λ is an eigenvalue of $\mathbf{X}^\top \mathbf{X}$

Then $(1 - \gamma\lambda)^2$ is an eigenvalue of $(\mathbf{I} - \gamma \mathbf{X}^\top \mathbf{X})^2$

Picking our learning rate to ensure the largest eigenvalue is less than one then allows us to guarantee convergence from any given initialization! We should to pick:

$$\gamma < \frac{2}{\lambda_{max}(\mathbf{X}^\top \mathbf{X})}$$

want $\lambda_{max} < 1$

$(1 - \gamma\lambda)^2 < 1$

$= 1 - 2\gamma\lambda + \gamma^2\lambda^2 < 1$

$\gamma^2\lambda^2 - 2\gamma\lambda < 0$

$\gamma^2\lambda^2 < 2\gamma\lambda$

$\gamma\lambda < 2 \rightarrow \gamma < \frac{2}{\lambda}$

For conv: $\alpha < \frac{2}{\lambda_{max}}$

For div: $\alpha > \frac{2}{\lambda_{min}}$

Further analysis GD in more general cases

Complexity of GD Step

$$\theta_i - \underbrace{\gamma \mathbf{X}^\top (\mathbf{X}\theta_i - \mathbf{y})}$$

Each step of gradient descent has computational complexity that depends on our entire dataset! But in modern ML we often work with internet-scale data which would make this prohibitive!

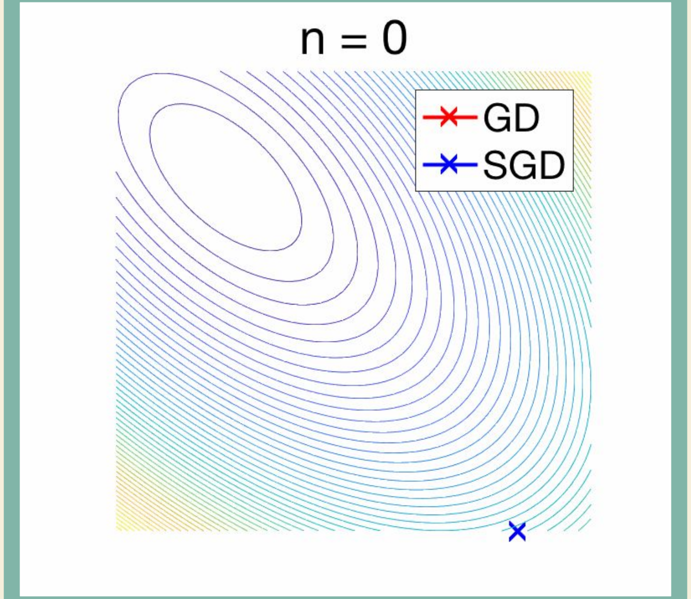
Concept of *stochastic gradient* descent

$$\theta_i - \gamma \mathbf{X}^\top (\mathbf{X}\theta_i - \mathbf{y})$$

$$\theta_i - \gamma \hat{\mathbf{X}}^\top (\hat{\mathbf{X}}\theta_i - \mathbf{y})$$

The critical idea of SGD is to subsample our dataset and use the estimated gradient as our update. This reduces complexity but adds stochasticity

A similar analysis can be carried out!



https://francisbach.com/the-sum-of-a-geometric-series-is-all-you-need/

Convergence under more general assumptions?

We have shown convergence for linear regression by appealing to its analytical tractability. But often we do not have this level of tractability for our models in practice. So what can we do?

Lipschitz Continuity

We say that a function is Lipschtiz continuous if a constant L exists such that:

$$||f(x) - f(y)|| \leq L||x - y||$$

+ve const.

Lipschitz Continuity

We say that a function is Lipschtiz continuous if a constant L exists such that:

$$||f(x) - f(y)|| \leq L||x - y||$$

The smoothness we want to derive a more general convergence result for gradient descent is continuity of the derivative

$$||\nabla f(x) - \nabla f(y)|| \leq L||x - y||$$

Theorem, proof left to notes

Theorem 6.1 *Suppose the function* $f : \mathbb{R}^n \to \mathbb{R}$ *is convex and differentiable, and that its gradient is Lipschitz continuous with constant* $L > 0$, *i.e. we have that* $\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2$ *for any* x, y. *Then if we run gradient descent for* k *iterations with a fixed step size* $t \leq 1/L$, *it will yield a solution* $f^{(k)}$ *which satisfies*

$$f(x^{(k)}) - f(x^*) \leq \frac{\|x^{(0)} - x^*\|_2^2}{2tk}, \tag{6.1}$$

Next week: diving into probabilistic approaches to ML!