## Admin:

-> Quiz 2 grades are out You did well! ☺

---

**Today:** Applications of gradient descent

-> Recap of gradient descent method
-> Linear regression (with mean squared error loss)
-> Solving regression problems via gradient descent
-> Linear system solving
-> Nonlinear regression via monomial dimension expansion

---

**Last time:**

**Problem:** Unconstrained Minimization

Given $f: \mathbb{R}^n \to \mathbb{R}$ f is continuous & smooth

find $x^* = \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} f(x)$

(provided minimum of f exists)

---

**Algorithm:** Gradient descent (GD) method

(locally greedy approach)

-> Start with some $x^{(0)}$
-> For $i = 0, 1, \ldots, T$

$$x^{(i+1)} = x^{(i)} - \eta_i \nabla f(x^{(i)})$$

Step size

Gradient

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

**Intuition:**

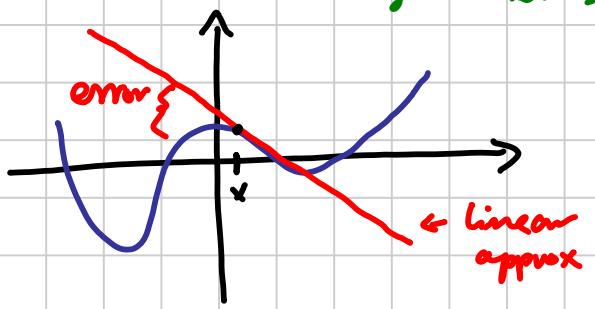
$$f(x+\delta) = f(x) + [\nabla f(x)]^T \delta + \frac{1}{2}\delta^T \nabla^2 f(x)\delta + \dots$$

linear approx. (GD minimizes that)

"error" term (we choose $\eta_i$ to be small enough so our progress is not cancelled by these terms)

Hessian

$$\nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1 \partial x_1} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & & \frac{\partial^2 f(x)}{\partial x_n \partial x_n} \end{bmatrix} = \left[\frac{\partial^2 f(x)}{\partial x_i \partial x_j}\right]_{ij}$$



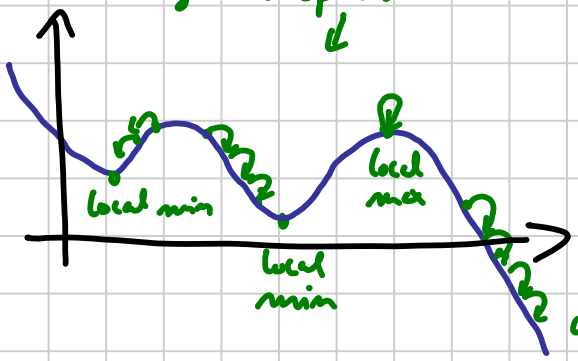

---

**Convergence of GD:**

For $T \to \infty$, $x^{(T)}$ either diverges to $-\infty$, i.e.,

$$\lim_{T\to\infty} f(x^{(T)}) = -\infty$$

Where we end up might depend on starting point!




Or converges to a critical point, i.e.,

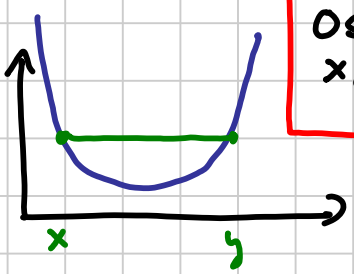
$$\lim_{T\to\infty} x^{(T)} = \hat{x} \text{ s.t. } \nabla f(\hat{x}) = 0$$

$\Rightarrow \hat{x}$ is a local minimum, maximum, or a saddle point

---

**Convexity:** Function $f$ is convex iff

$$\forall_{0\le\lambda\le 1,\ x,y\in\mathbb{R}^n} \quad f(\lambda x + (1-\lambda)y) \le \lambda f(x) + (1-\lambda) f(y)$$




Equivalently, $\forall_{x,\delta\in\mathbb{R}^n} \quad f(x+\delta) \ge f(x) + \nabla f(x)^T \delta$

**Key observation:** If $f$ is convex then **every** critical point is a **global** minimum, ie,

$$\nabla f(\hat{x}) = 0 \iff \hat{x} \text{ is a global minimum.}$$

(this is the key direction)

So, if $f$ is convex & global minimum exists

$\Rightarrow$ GD **provably** finds it

(Alas, many important problems correspond to $f$ that are **NOT** convex.)

---

**Convergence analysis:** (for (strongly) convex functions)

**Def.** $f$ is **$\beta$-smooth** iff

$$\forall_{x,\delta \in \mathbb{R}^n} \quad \delta^T \nabla^2 f(x) \delta \leq \beta \|\delta\|^2$$

for some $\beta \geq 0$

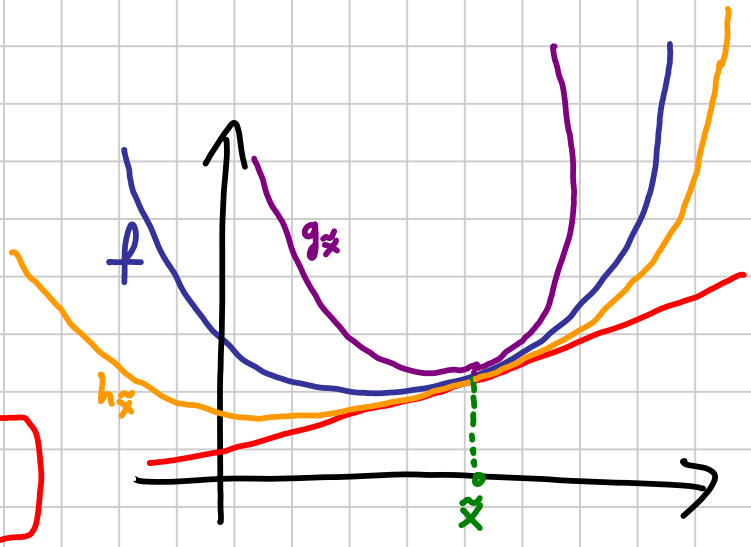
**Def.** $f$ is **$\alpha$-strongly convex** iff

$$\forall_{x,\delta \in \mathbb{R}^n} \quad \delta^T \nabla^2 f(x) \delta \geq \alpha \|\delta\|^2$$

for some $\alpha > 0$

("normal" convexity corresponds to $\alpha = 0$)

$\Rightarrow$ Always $\alpha \leq \beta$




$$g_{\tilde{x}}(x) = f(\tilde{x}) + \nabla f(\tilde{x})^T (x - \tilde{x}) + \frac{\beta}{2} \|x - \tilde{x}\|^2$$

(upper bounding "$\beta$-smoothness" parabola)

$$\Rightarrow \quad f(x) \leq g_{\tilde{x}}(x) \quad \forall_{x,\tilde{x}}$$

$$h_{\tilde{x}}(x) = f(\tilde{x}) + \nabla f(\tilde{x})^T (x - \tilde{x}) + \frac{\alpha}{2} \|x - \tilde{x}\|^2$$

(lower bounding "$\alpha$-strong conv." parabola) $\Rightarrow f(x) \geq h_{\tilde{x}}(x) \quad \forall_{x,\tilde{x}}$

**Theorem** (proved last time)

If $f$ is $\beta$-smooth & $\alpha$-strongly convex then for any $\varepsilon > 0$

$$f(x^{(T)}) - f(x^*) \le \varepsilon \quad \text{whenever} \quad T = \Omega\left(\kappa \log \frac{f(x^{(0)}) - f(x^*)}{\varepsilon}\right)$$

where $\kappa = \frac{\beta}{\alpha}$ is the condition number of $f$

initial sub optimality

**In other words:** For such $f$, GD gives an $\varepsilon$-approx solution after $\le O\left(\kappa \log \frac{f(x^{(0)}) - f(x^*)}{\varepsilon}\right)$ iterations

---

**Intuition on the dependence on $\kappa$:**

→ $\kappa$ measures the quality of the local approximation of $f$ at $\tilde{x}$ via upper bounding "parabola" $g_{\tilde{x}}$ and lower bounding parabola $h_{\tilde{x}}$

**Geometric intuition:**

Consider $f(x) = \frac{1}{2} a \cdot x_1^2 + \frac{1}{2} x_2^2$ for $a \ge 1$

$$\nabla f(x) = \begin{bmatrix} a \\ 1 \end{bmatrix}$$

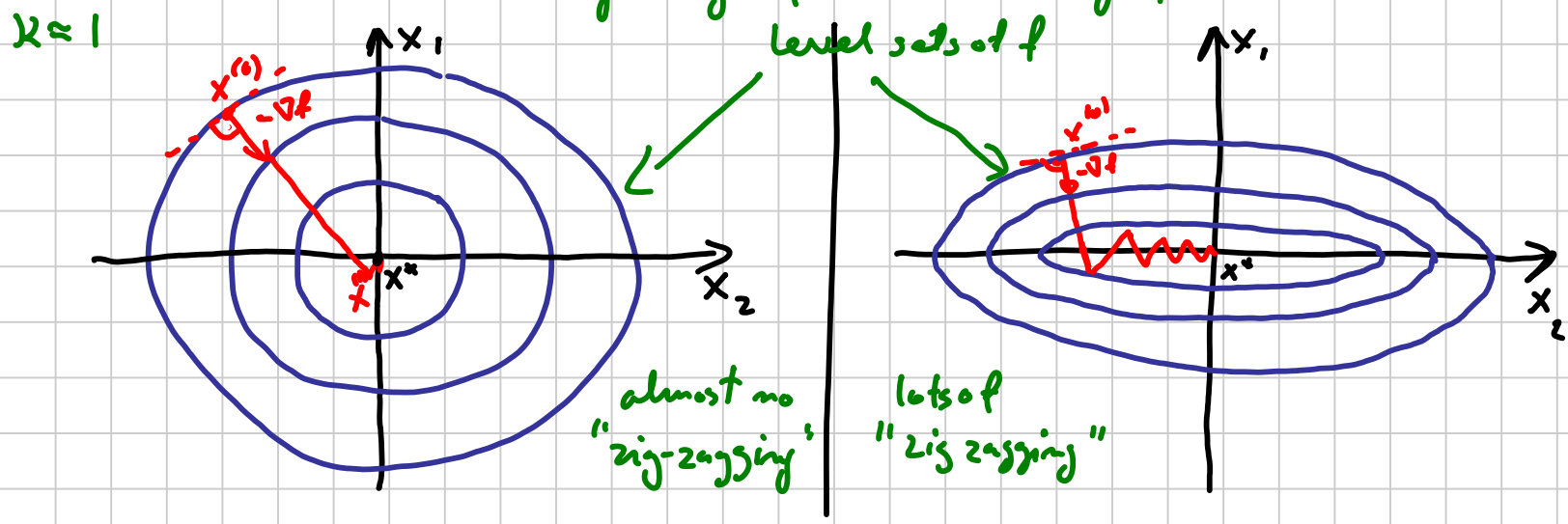
$$\nabla^2 f(x) = \begin{bmatrix} a & 0 \\ 0 & 1 \end{bmatrix}$$

⇒ $f$ is $a$-smooth & 1-strongly convex

⇒ $\kappa = a$ ⇒ $x^* = (0,0)$

Visualization of a trajectory of GD starting from some $x^{(0)}$

$\kappa \approx 1$



Key application domain of gradient descent is training machine learning models

**Illustrative example:** Linear regression

**Input:** $m$ datapoints $x^{(1)}, \dots, x^{(m)} \in \mathbb{R}^n$ & $m$ labels $y^{(1)}, \dots, y^{(m)} \in \mathbb{R}$

**Goal:** Find a linear function $h_w$ that "predicts" each $y^{(j)}$ given $x^{(j)}$, i.e., we want

$$h_w(x^{(j)}) = \sum_{i=1}^{n} w_i \cdot x_i^{(j)} \approx y^{(j)} \quad \text{for each } j = 1, \dots, m$$

**Think:** Predict the score of 6.046 students on the final exam based on their performance during the semester

→ each data point $x^{(j)}$ corresponds to 6.046 student
→ each coordinate of $x^{(j)}$ encodes score on one of the psets or the quizzes
→ each $y^{(j)}$ is the score on the final

---

**Note:** Wlog, we allow our function $h_w$ to have a constant term → just make one "dummy" coordinate of datapoints $x^{(1)}, \dots, x^{(m)}$ be always equal to 1

---

So, our goal is to compute $w \in \mathbb{R}^n$ s.t. the resulting linear function

$$h_w(x) = \sum_{i=1}^{n} w_i x_i = w^T x$$

has "best fit" to data points

⇒ good fit indicates underlying linear relationship

⇒ hope that such $h_w$ "generalizes" to yet unseen data

**How to quantify that?**

**Popular choice:** Mean squared error (MSE)

$$L(v) = \frac{1}{m} \sum_{j=1}^{m} E_j(v)^2$$

where $E_j(v) = h_v(x^{(j)}) - y^{(j)}$ = "residual error" on $x^{(j)}$, i.e. the difference between $h_v$ "prediction" for $x^{(j)}$ and the actual value of $y^{(j)}$

**Our goal:**

Compute $v^* = \underset{v \in \mathbb{R}^n}{\operatorname{argmin}} L(v)$

(Corresponds to a max likelihood estimate if we assume that the data set has linear dependencies + Gaussian noise)

⇒ This is an unconstrained minimization problem ∇

⇒ We can just use GD

---

**Observe:** $\nabla E_j(v) = \begin{bmatrix} x_1^{(j)} \\ \vdots \\ x_n^{(j)} \end{bmatrix}$

**Remember:** Now, $v$ are the variables (not $x$)

$$\nabla L(v) = \frac{2}{m} \sum_{j=1}^{m} E_j(v) \cdot \nabla E_j(v) = \frac{2}{m} \sum_{j=1}^{m} E_j(v) \cdot x^{(j)}$$

⇒ Each datapoint $x^{(j)}$ contributes $\frac{2}{m} E_j(v) x^{(j)}$ to the gradient $= v_j$

⇒ update $v' \leftarrow v - \eta v_j$

reduces the error on data point $x^{(j)}$ since

$$E_j(v') = (v')^T x^{(j)} - y^{(j)} = (v - \eta v_j)^T x^{(j)} - y^{(j)} =$$

$$= E_j(v) - \eta v_j^T x^{(j)} = E_j(v) - \frac{2}{m} \eta E_j(v) \|x^{(j)}\|^2 = E_j(v)\left(1 - \frac{2}{m}\eta\right)$$

But could increase error on other example

$\Rightarrow$ Gradient update: $-\eta \nabla L(w) = -\eta \sum v_j$

takes a "majority" vote of all data point updates

**How fast will it converge (if at all)?**

**Observe:**

$$\nabla^2 L(w) = \frac{1}{m}\sum_{j=1}^{m} \nabla^2 \left(E_j(w)^2\right) = \frac{1}{m}\sum_{j=1}^{m} \left[\frac{\partial^2 (E_j(w)^2)}{\partial k\, \partial l}\right]_{k,l}$$

$$\Rightarrow \nabla^2 L(w) = \frac{2}{m}\sum_{j=1}^{m} x^{(j)} \left(x^{(j)}\right)^T$$

↑ this is (rank-one) $n \times n$ matrix

$$\frac{\partial^2 (E_j(w)^2)}{\partial k\, \partial l} = 2\,\frac{\partial\left(E_j(w) x_l^{(j)}\right)}{\partial k} = 2\, x_k^{(j)} x_l^{(j)}$$

**Recall:** For any vector $v \in \mathbb{R}^n$

the (rank-one) matrix $vv^T$ is positive semi-definite (PSD)

i.e. $\delta^T (vv^T) \delta \geq 0$ for any $\delta \in \mathbb{R}^n$

$\Rightarrow \nabla^2 L(w)$ is PD too

$\Rightarrow L(w)$ is convex

Means that $\nabla^2 L(w)$ is positive definite (PD)

**In fact:** as long as $m \geq n$ & all $x^{(j)}$'s are linearly indep.

$$\delta^T \nabla^2 L(w)\, \delta \geq \alpha \|\delta\|^2 \quad \text{for some } \alpha > 0$$

$\Rightarrow L(w)$ is $\alpha$-strongly convex

**Also:** We always have some $\beta$ s.t.

$$\delta^T \nabla^2 L(w)\, \delta \leq \beta \|\delta\|^2 \quad \text{for some } \beta > 0$$

$\Rightarrow L(w)$ is $\beta$-smooth

$\Rightarrow w^*$ exists & is unique $\Rightarrow$ GD gets $\varepsilon$-close to it after $O\left(\frac{\beta}{\alpha} \log \frac{L(w_0) - L(w^*)}{\varepsilon}\right)$ iter

**In fact:** We can get an explicit formula for $w^*$ (this is quite unusual)

**Recall:** $w^*$ has to be a critical point, i.e., we must have

$$\nabla L(w^*) = 0$$

← Normal equations for $L$

$$\Rightarrow \quad \frac{2}{m} \sum_{j=1}^{m} E_j(w^*) x^{(j)} = 0 \quad \Big| \cdot \frac{m}{2}$$

$$\Rightarrow \quad \sum_{j=1}^{m} \left( (w^*)^T x^{(j)} - y^{(j)} \right) x^{(j)} = 0$$

$$\Rightarrow \quad \sum_{j=1}^{m} (w^*)^T x^{(j)} \cdot x_i^{(j)} = \sum_{j=1}^{m} y^{(j)} x_i^{(j)} \quad \forall_{i=1,\dots,n}$$

$$\Rightarrow \quad (X^T X) w^* = X^T y$$

where

$$X = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_n^{(1)} \\ \vdots & & & \\ x_1^{(m)} & x_2^{(m)} & \dots & x_n^{(m)} \end{bmatrix} \begin{matrix} \} n \\ \} m \end{matrix}$$

Data matrix

$$\Rightarrow \quad \boxed{w^* = (X^T X)^{-1} X^T y}$$

**Note:**

$$\nabla L(w) = \frac{2}{m} \sum_{j=1}^{m} x^{(j)} x^{(j)T} = \frac{2}{m} X^T X$$

$$y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix} \} m$$

Label vector

$\Rightarrow$ [ $X^T X$ invertible (and $w^*$ well-defined) iff $\nabla^2 L(w)$ is PD iff $L(w)$ $\alpha$-strongly convex for $\alpha > 0$

↳ This is a "sanity check", as $w^*$ has to be defined uniquely if $L(w)$ $\alpha$-strongly convex

So, we can use GD to solve (linear) regression but what else?

## Solving linear systems:

Given an invertible $n \times n$ matrix $A$ & vector $b \in \mathbb{R}^n$,

find $x^* \in \mathbb{R}^n$ s.t. $\boxed{Ax = b}$

→ Classic approach:

Compute $x^* = A^{-1}b$ directly via, e.g., Gaussian elimination

Problems: → Always fairly slow $O(n^\omega)$ $\omega \leq 2.373$
(even if matrix is sparse or "nice")

→ Numerically problematic
(dividing by small numbers is bad)

→ Iterative approach:

→ Start with some $x^{(0)}$

→ Iteratively improve your solution $x^{(i)}$ to get solution $x^{(i+1)}$

How to do this? With gradient descent ∇

Consider a function: $\boxed{f_A(x) = \frac{1}{2} x^T (A^T A) x - A^T b}$

Observe: $\boxed{\nabla f_A(x) = (A^T A)x - A^T b}$ ← cheap to compute:

small if A is sparse → $O(n + \#\text{ of non-zero entries of } A)$

**Algorithm:**

→ Start with some $x^{(0)}$, e.g., $x^{(0)} = 0$

→ For $j = 0, 1, \ldots, T$

$$x^{(j+1)} = x^{(j)} - \eta_j \nabla f_A(x^{(j)}) = x^{(j)} - \eta_j A^T(Ax^{(j)} - b)$$

$Ax^{(j)} - b$: residual error of $x^{(j)}$

**Why does it work?**

**Note:** For any critical point $\hat{x}$ we have that

$$\nabla f_A(\hat{x}) = 0 \iff A^T(A\hat{x} - b) = 0$$

$$\iff A\hat{x} = b$$

(we use the fact that $A$ is invertible)

$\Rightarrow$ Any critical point is a solution

**In fact:**

$$\boxed{\nabla^2 f_A(x) = A^T A}$$

$\Rightarrow$ $\nabla^2 f_A(x)$ is PD (for any $x$) if $A$ is invertible

$\Rightarrow$ $f_A$ is $\alpha$-strongly convex for some $\alpha > 0$

(& also $\beta$-smooth)

$\Rightarrow$ Solution $x^*$ is unique and GD will compute an $\varepsilon$-approx. solution after $O\left(\frac{\beta}{\alpha} \log \frac{f_A(0) - f_A(x^*)}{\varepsilon}\right)$ iterations

**Advantage:** Condition number $\kappa = \frac{\beta}{\alpha}$ can often be (made) small

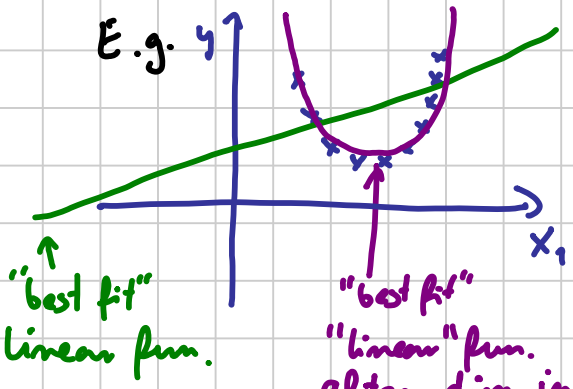
# Beyond Convexity



So far, our classifier $h_w$ was extremely simple, i.e., linear

Turns out that linear classifieres can still be really powerful!

One just needs to increase the number of dimensions




$n=1$

data points $x^{(j)} \in \mathbb{R}$ & $y^{(j)} = (x^{(j)})^2$

$\Rightarrow$ impossible to find a line with a good fit

But: If we expand the dimensions via mapping $\varphi: \mathbb{R} \to \mathbb{R}^2$

$$\varphi(x) = (x, x^2)$$

$\Rightarrow$ "lines" in transformed space corespond to parabolas in the original space

If $z^{(j)} = \varphi(x^{(j)}) \in \mathbb{R}^2$

then a "line" classifier

$$h_w(z) = w_1 \cdot z_1 + w_2 \cdot z_2$$

with $w = (0, 1)$ is such that

$$\forall_j \quad h_w(z^{(j)}) = z_2^{(j)} = (x^{(j)})^2 = y^{(j)}$$

← perfect fit!

So, such expanding of the dimension into the $O(n^d)$-dimensional space of monomials $x_i^k$ for $\substack{1 \le i \le n \\ 0 \le k \le d}$ makes "linear" functions express degree-d polynomials

$\Rightarrow$ Significant boost in expressability

(but at the price of blowing up the dimension!)