Lecture 15.

- Basic Machine Learning
  - Optimization
  - Convexity
  - Gradient Descent
- Private Machine Learning
  - DP Gradient Descent

Announcement: Recitation in person this Fri !

Review basic ML stuff.

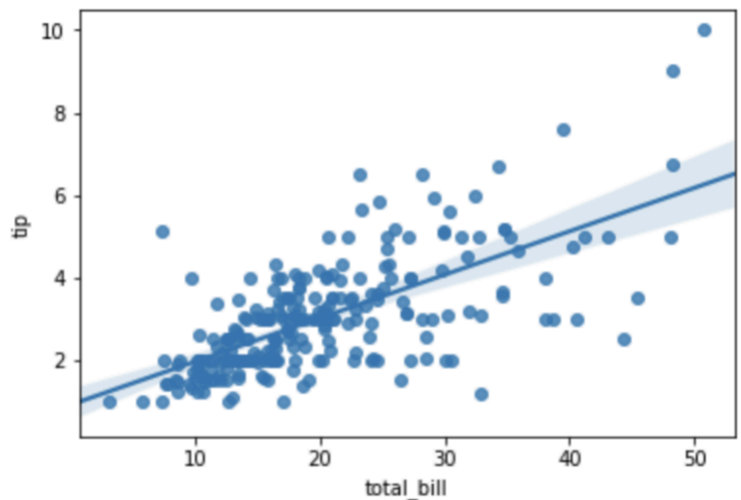
HW2 (?)

TCS 434 7pm Friday

Linear Regression

feature → x y ← label

	total_bill	tip
0	16.99	1.01
1	10.34	1.66
2	21.01	3.50
3	23.68	3.31
4	24.59	3.61
5	25.29	4.71
6	8.77	2.00
7	26.88	3.12
8	15.04	1.96
9	14.78	3.23
10	10.27	1.71
11	35.26	5.00
12	15.42	1.57
13	18.43	3.00
14	14.83	3.02
15	21.58	3.92



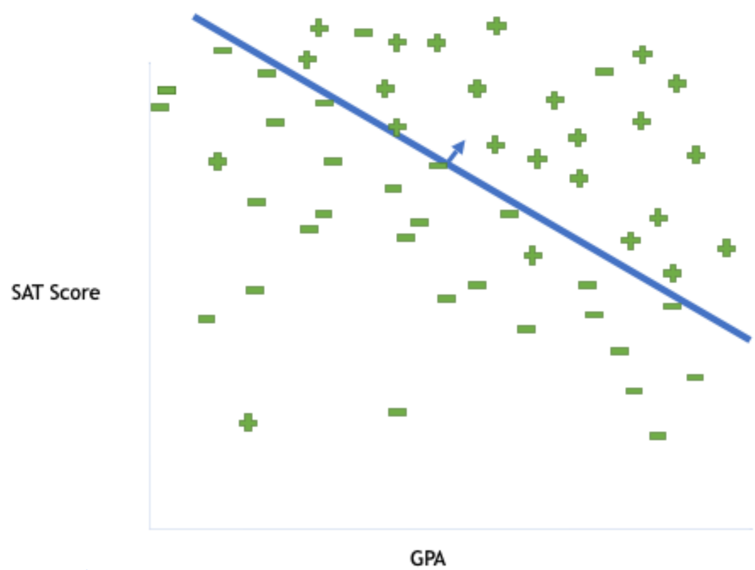
labeled example (x_i, y_i)

x_i = total bill

y_i = tip.

$$\min_{w \in \mathbb{R}} \sum_{i=1}^{n} \underbrace{(w x_i - y_i)^2}_{\text{Squared loss.}}$$

Linear Classification



$x_i = (\text{SAT Score}_i, \ GPA_i)$

$y_i = $ ("college success")
in $\{\pm 1\}$

$z_i = y_i \langle w, x_i \rangle$

y_i: ± 1

$\langle w, x_i \rangle$: want to match sign of y_i

Logistic Loss : $\sum_{i=1}^{n} \ln(1 + \exp(-z_i))$.

Hinge Loss $L(w; x) = \frac{1}{n} \sum_{i=1}^{n} (1 - z_i)_+$

$$(a)_+ = \begin{cases} a & \text{if } a > 0 \\ 0 & \text{o/w.} \end{cases}$$

(Private) Optimization.

Given a data set $X = (x_1, \dots, x_n)$

loss function: ℓ

feasible set of parameters (weights): $C \subseteq \mathbb{R}^d$

Empirical Risk Minimization (ERM):

$$\min_{w \in C} \underbrace{L(w; x)}_{\text{Empirical Risk.}} = \frac{1}{n} \sum_{i=1}^{n} \ell(w; x_i) \quad + \quad \text{Optional Regularization}$$

e.g., $\lambda \|w\|_1$

Often: $C = \mathbb{R}^d$

Empirical Risk: $L(w;x) = \frac{1}{n}\sum_{i=1}^{n} \ell(w;x_i)$

Population Risk: $L(w;P) = \mathbb{E}_{x' \sim P}[\ell(w;x')]$

Usually, ER is a good proxy for PR.

Differential Privacy $\Longrightarrow$ Preventing overfitting.

Examples of loss functions:

$x = ((x_1,y_1), \ldots, (x_n,y_n))$

Squared loss $L(w;x) = \frac{1}{n}\sum_{i=1}^{n} (\langle w, x_i \rangle - y_i)^2$

Hinge Loss (used in support vector machine) $L(w;x) = \frac{1}{n}\sum_{i=1}^{n} (1 - y_i\langle w, x_i \rangle)_+$

$$(a)_+ = \begin{cases} a & \text{if } a > 0 \\ 0 & \text{o/w.} \end{cases}$$

2 Criteria: ① Captures predictive accuracy

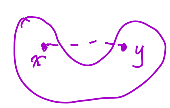
② Easy to optimize.

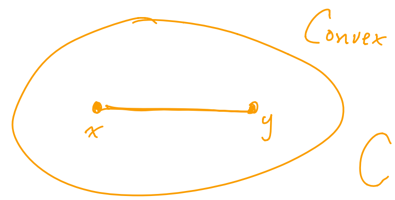
Convexity. (Sets and functions)

- $C \subseteq \mathbb{R}^d$ is convex if $\forall x, y \in C$, $t \in [0,1]$

  $t \cdot x + (1-t) \cdot y \in C$

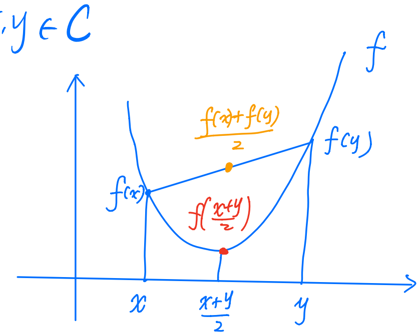
Non-Convex



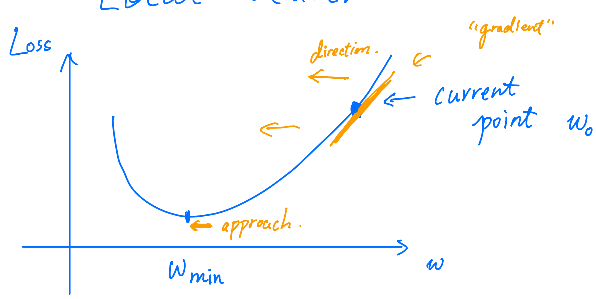


- $f : C \longrightarrow \mathbb{R}$ is convex if $\forall x, y \in C$

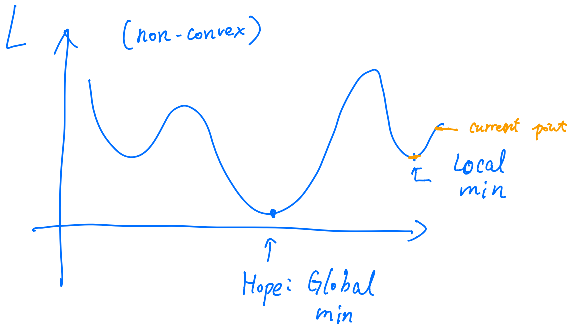
  $f\left(\frac{x+y}{2}\right) \leq \frac{f(x) + f(y)}{2}$

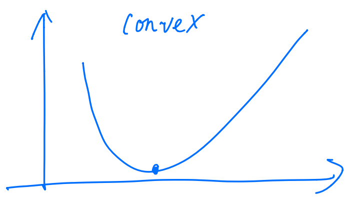


"Local Search"



Loss from Neural Networks.





$\ell : C \times X \longrightarrow \mathbb{R}$

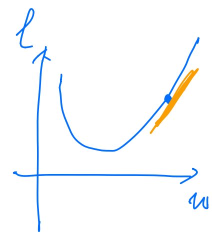
$\ell(w, x)$ measures "loss"

$L : C \longrightarrow \mathbb{R}$

$$L(w) = \frac{1}{n} \sum_{i=1}^{n} \ell(w, x_i)$$

Gradients.

1-dim example $(w \in \mathbb{R})$



$$\ell(w; (x_i, y_i)) = (w x_i - y_i)^2$$

In 1-d, gradient = derivative

$$\nabla_w \ell(w; (x_i, y_i)) = 2(w x_i - y_i)\, x_i$$ ← "Chain Rule"

gradient w.r.t. parameter w

Multi-dim example $(w, x_i \in \mathbb{R}^d)$

$$\ell(w; (x_i, y_i)) = (\langle w, x_i \rangle - y_i)^2$$

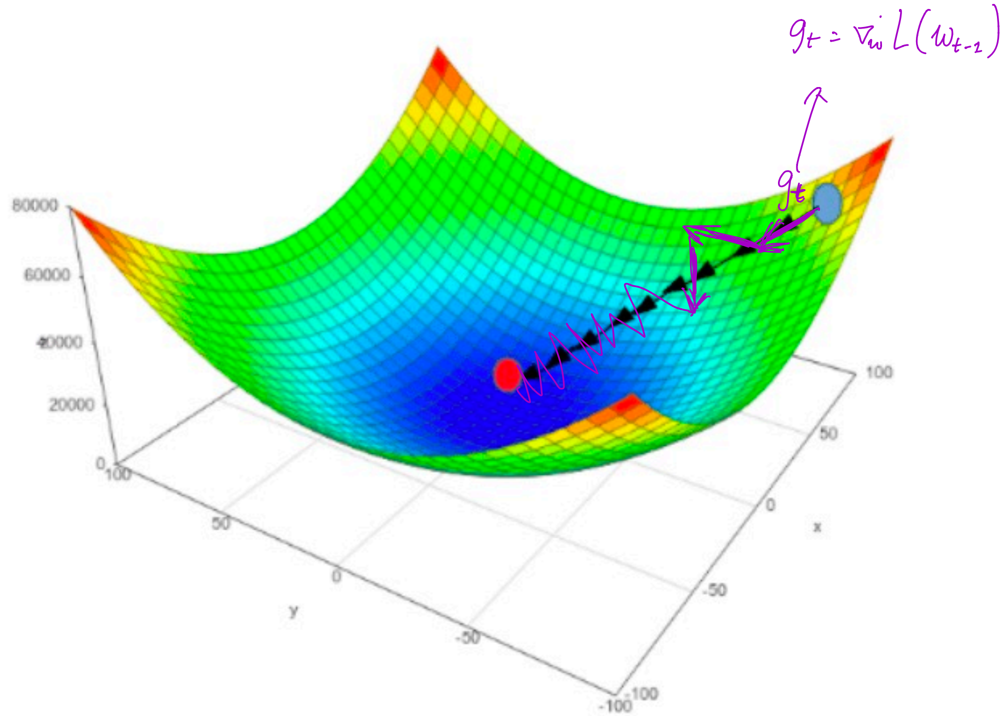
$$\nabla_w \ell(w; (x_i, y_i)) = 2(\langle w, x_i \rangle - y_i)\, x_i$$

in $\mathbb{R}^d$

ERM minimizes

$$L(w; x) = \frac{1}{n} \sum_{i=1}^{n} \ell(w; (x_i, y_i))$$

$$\nabla_w L(w; x) = \underbrace{\frac{1}{n} \sum_{i=1}^{n} \nabla_w \ell(w; (x_i, y_i))}_{\text{avg over all gradients}}.$$


$g_t = \nabla_w L(w_{t-1})$
g_t
80000
60000
40000
20000
0
100
50
0
-50
-100
y
x

Projected Gradient Descent (PGD)

$$\mathrm{PGD}(\underset{\text{Loss}}{L}, \underset{\text{Feasible Set}}{C}, \underset{\text{Learning Rate}}{\eta}, \underset{\#\text{ rounds}}{T}):$$

Init: $w_0 \in C$ (any point)

For $t = 1, \dots, T$:

gradient: (Backpropagation) $g_t = \nabla L(w_{t-1})$

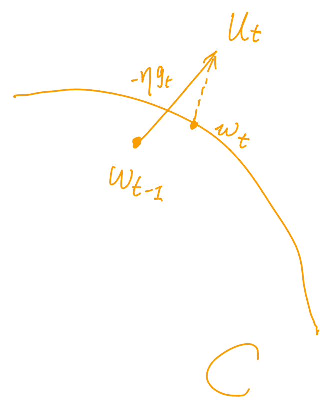
$u_t \leftarrow w_{t-1} - \eta\, g_t$

projection: $w_t \leftarrow \underset{w \in C}{\operatorname{argmin}} \|w - u_t\|_2$

Output: w_T ← Last iterate

or

$\frac{1}{T}\sum_{t=1}^{T} w_t$ ← Average iterate.



Robustness to noise in gradient estimation. (g_t)

Two sources of noise:

→ For efficiency:

Sample a minibatch $\overset{50}{\downarrow}B \subseteq \{1, 2, \dots, \overset{50 \text{ million}}{\downarrow}n\}$

gradient estimate $\tilde{g}_t = \frac{1}{|B|} \sum_{i \in B} \nabla_w L(w_{t-1}, x_i)$.

→ For privacy: Add Gaussian Noise

$$\tilde{g}_t = g_t + N(0, \sigma^2 I_d)$$

from Gaussian mech.

In both cases,

$\tilde{g}_t$ is an unbiased estimate of g_t

$\mathbb{E}[\tilde{g}_t] = g_t$