THEORY OF MACHINE LEARNING

LECTURE 21

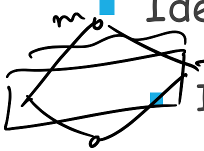
GRADIENT DESCENT FOR NN: NEURAL TANGENT KERNEL

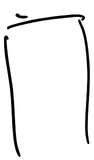
ANNOUNCEMENTS

- **Homework 3 due on Monday April 11**
- Ideas
  - Implicit regularization
  - Stability in optimization
  - Depth vs width (converse)
  - Learning parities – earliest lower bound for deep learning

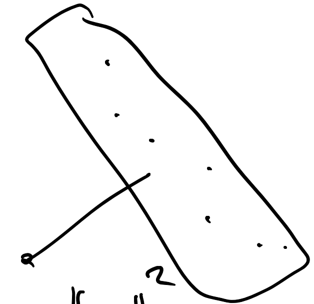
(possible via other algos; not via deep learning).
efficient (n^3.)

- Last lecture's discussion on feature learning





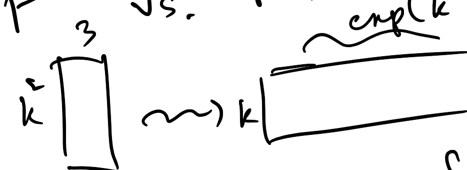
$Ax \approx b$



$\min \|x\|^2$

s.t. $Ax = b$.

F vs. $F + h$.



$\min \|Ax - b\|^2$

→ Soln found by grad descent will be "simple"

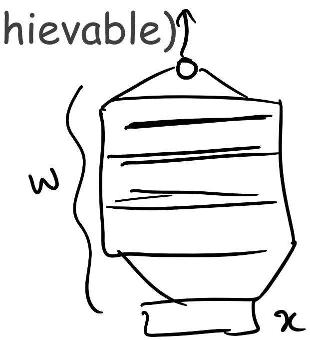
→ This is one reason these models don't overfit.

NEURAL NETWORK TRAINING

$\in \mathbb{R}^d$.

(x_N, y_N)

- **Question (supervised learning):** given data $(x_1, y_1), (x_2, y_2), \ldots$ from some distribution D, find *h* (with given "architecture") that minimizes the risk
  - Really hard theoretically (even if inputs Gaussian and risk zero is achievable)
  - In practice, solved via gradient descent
    [fast implementation (backprop) by Rumelhart, Hinton, Williams]



$f(w; x)$

$$L(w) = \sum_{i=1}^{N} \left(f(w; x_i) - y_i \right)^2$$

- Question for today:
  - How can one analyze dynamics of gradient descent?
  - Are there cases where we can reason about resulting solution?

$w^{(0)} \to w^{(1)} \to \ldots$

OVER-PARAMETRIZATION

parameters >> # inputs (N).
|w|

- **Observation:** modern deep nets pretty overparametrized, but they still don't overfit [Why? ... Belkin et al., "double descent"].
- Question out of desperation: Is GD easier to analyze when network is "heavily" overparametrized? → this is one setting where $L(w^*) = 0$

[AllenZhu, ...].

Theorem. [Jacot, Gabriel, Hongler 18] [Arora, et al. 2019] A width ~ N^3 network (any number of layers) trained via GD from random initialization achieves zero training error. Moreover, the final solution is equivalent to solving a "Kernel regression" problem with a specific kernel.

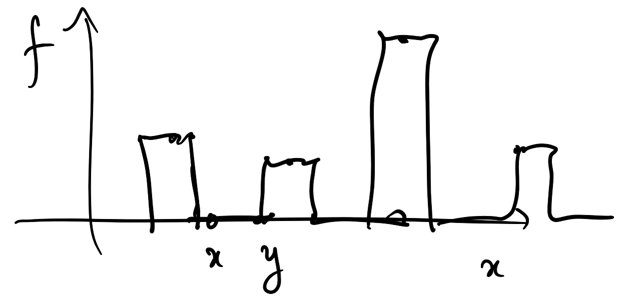
defined "Neural Tangent" kernel.

ASIDE: KERNEL REGRESSION

$(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$

- Ubiquitous motivation: function value known at a bunch of points, "interpolate" to rest of space
  - One way to think of all of ML!
- Suppose K defines "point similarity" K(x,y)
- Consider interpolation via functions of specific form…

K: similarity fn



$$\hat{f}(x) = \sum_{i=1}^{N} \alpha_i K(x, x_i)$$

Goal: Given Kernel (Gaussian), find the best α_i

$\alpha_1 \times$ similarity with given datapoint x_1 $+ \alpha_2 \times$ similarity with x_2 $+ \ldots$

$K(x,y)$: "kernel"

K

$$K(x,y) = \begin{cases} 1 & \text{if } \|x-y\| < r \\ 0 & \text{otherwise.} \end{cases}$$

$e^{-\frac{\|x-y\|^2}{r^2}}$

Simplest / most natural: choose α_i so that $\underbrace{\hat{f}(x_i) = y_i \ \forall i}_{}$

↳ N constraints

want to find coeffs $\alpha_1, \alpha_2, \ldots, \alpha_N$.

$$\hat{f}(x_1) = y_1 \iff \alpha_1 \underbrace{K(x_1, x_1)}_{} + \alpha_2 K(x_1, x_2) + \ldots + \alpha_N K(x_1, x_N) = y_1$$

↳ linear eqn in the α's (since we know the x_i ...)

$$\begin{array}{cc} & j \quad n \\ \begin{array}{c} n \\ i \end{array} & \begin{bmatrix} & & \\ & K & \\ & & \end{bmatrix} \end{array} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \to \text{opt soln: } \alpha = \underline{\underline{K^{-1} y}}.$$

$K(x_i, x_j)$

→ Try to minimize $g(\alpha) = \| K\alpha - y \|^2$ using gradient descent (with param $= \alpha$)

initialize: $\alpha^{(0)}$

$$\underline{\underline{\nabla g^{(\alpha)} = 2K(K\alpha - y)}} .=$$

$$\underline{\underline{\alpha^{(t+1)} = \alpha^{(t)} - \eta \cdot \nabla g(\alpha^{(t)})}}$$

Qn: What happens to the $\hat{f}$ as you do steps of GD?

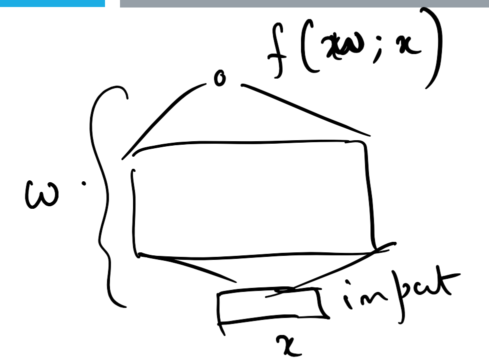
$$\hat{f}(\alpha^{(t)}; x) = \sum_{i=1}^{N} \alpha_i^{(t)} K(x; x_i)$$

$$\hat{f}(\alpha^{(t+1)}; x) - \hat{f}(\alpha^{(t)}; x) = -\eta \left\langle \nabla \hat{f}(\alpha^{(t)}; x), \nabla g(\alpha^{(t)}) \right\rangle$$

$$\underset{\text{(exercise)}}{=} -\eta \left\langle \nabla \hat{f}(\alpha^{(t)}; x), \sum_{i=1}^{N} (\hat{f} - y)_i \cdot \nabla \hat{f}(\alpha^{(t)}; x_i) \right\rangle$$

$$= -\eta \sum_{i=1}^{N} (\hat{f} - y)_i K(x, x_i)$$

NEURAL NET TRAINING REVISITED



$$L(w) = \frac{1}{2}\sum_{i=1}^{N}\left(f(w; x_i) - y_i\right)^2$$

$$\underline{\nabla_w} L(w) = \sum_{i=1}^{N} \underbrace{\left(f(w; x_i) - y_i\right) \cdot \nabla_w f(w; x_i)}$$

$\cancel{f(w^{(t)})}$

$$w^{(t+1)} = w^{(t)} - \eta \cdot \nabla L(w^{(t)}) \quad \text{(Gradient descent)}$$

<u>Qn:</u> how does $f(w^{(t)}, x)$ "evolve" as time progresses?

$$\underline{f(w + \varepsilon; x) - f(w; x)} \approx$$

$$\underline{\langle \nabla f(w; x), \varepsilon \rangle}.$$

$$f(w^{(t+1)}; x) - f(w^{(t)}, x)$$

$$\parallel$$

$$f(w^{(t)} \underbrace{- \eta \nabla L(w^{(t)})}_{-\varepsilon}; x) - f(w^{(t)}; x) \approx -\eta \underbrace{\langle \nabla f(w^{(t)}, x), \nabla L(w^{(t)}) \rangle}$$

Defining $f(w^{(t)}, x_i) = u_i^{(t)}$,

$$= -\eta \left\langle \nabla f(w^{(t)}, x), \sum_{i=1}^{N} \left(u_i^{(t)} - y_i\right) \nabla f(w^{(t)}, x_i) \right\rangle$$

$$= -\eta \sum_{i=1}^{N} \left(u_i^{(t)} - y_i\right) \underbrace{\left\langle \nabla f(w^{(t)}, x), \nabla f(w^{(t)}, x_i) \right\rangle}_{H^{(t)}(x, x_i)}$$

for any x,

$$\boxed{f(w^{(t+1)}; x) - f(w^{(t)}; x)} = \underbrace{-\eta \cdot \sum_{i=1}^{N} \left(u_i^{(t)} - y_i\right) H^{(t)}(x; x_i)}.$$

$\Rightarrow$ Moral: gradient descent $\equiv$ Kernel regression with a time-varying "Tangent" Kernel.

→ For very wide NNs, Kernel remains "nearly fixed" with time.

initialized randomly (with Gaussian)

For "large" amount of time, $w^{(t)}$ doesn't change much!

$w^{(t)}$ can change very slightly & still achieve zero error..

→ w's don't correspond to "useful" features.

(random features [Rahimi; Recht])

KERNEL IN THE INFINITE WIDTH LIMIT