PERSONAL NOTES

Principles of Large Scale ML

Hanfeng Zhai

Disclaimer: These notes are intended solely for personal reference and study purposes. They represent my own understanding of the course material and may contain errors or inaccuracies. The content presented here should not be considered as an authoritative source, and reliance solely on these materials is not recommended. If you notice any materials that potentially infringe upon the copyright of others, please contact me at hz253@cornell.edu so that appropriate action can be taken. Your feedback is greatly appreciated.

2023

Scaling → ML

↑

Principles.

How scale impact performance of ML system.

Data Collection → Data cleaning → Feature Engineering (Embedding) → Training (Robust).

Train / Valide / Test.

Splitting.

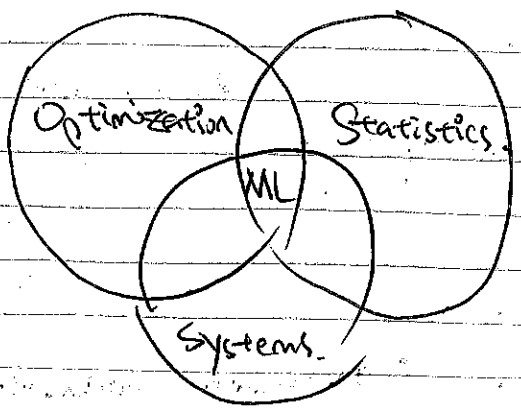
Model selection (Hyperpara. Opt.)

(Online learning)

Deployment

Inference

Monitoring



Principle #1: Optimization. Tasks

Solve it using gradient-base algorithms that are fast & "canned".

backpropagation + numerical linear algebra

- gradient descent.
- empirical risk min.

Principle #2: ✗ whole dataset, → subsample.

- Stochastic gradient d.
  - ↳ subsample the component losses.
  - ↑ compute ↗ faster.
- cross-validation / train / val / test split.
- bagging.
- kernel sampling. → subsample with ~~kernel~~ in CNN
- dropout. - data augmentation.
- VAE. (dimension reduction).

Principle #3 Use algorithms that are compatible w/ hardware. or vice versa.

- batch size. selection.
- numerical precision.
- GPU ↔ NN.
- Caching
  - data loader / streaming
  - ↕
  - GPU.
- using fast numerical linear algebra kernels.
- parallel / distributed computing.
- neuromorphic computing. (chip → NN).

CS.cornell.edu/courses/cs4787.

+ Canvas. + Gradescope → problem.

+ ed discussion. + CMS - programming a. (Python, numpy, torch)

Grading.

Psets

PAs.

Prelims

Final.

Paper reading.

Review sets — linear algebra ..

▷ Vector calculus

▷ Logistics

▷ Computing w/ Python

— office hours

Wednesday · 2-3 pm

Gates 426.

Week 1 . - 2 .

Aug. 24.

Principle 1 : ML as optimization.

↓

Continuous optimization.

↓

optimizing real numbers.

Data.

↳ Embedding steps → $\mathbb{R}^d$

vectors.

eg. pred. ppl.

· 33

· g. avg.

· 2013 . current job / current dun.

· 18,000 / yr.

↳ $\begin{bmatrix} - \\ - \\ - \\ - \end{bmatrix} \Rightarrow \mathbb{R}^4$.

Vector - element in vector space.

$x, y \in V \quad x+y \in V$

+ associated & commutative.

multiplication - -

~~Standard~~ property of a vector space.

Typical example. $\mathbb{R}^n$. $\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$

e.g. matrix $\mathbb{R}^{n \times n}$.

$\mathbb{R}^{n \times n \times p}$. } can all be interpreted as vectors.

Core idea under: basis in linear algebra.

↓

how many numbers does it take / pinpoint that "vector space".

A basis for V is the subset for

$B = \{x_1, x_2, x_3, \ldots\}$. s.t. for every v in V - v can be written as a linear combination of vectors in B.

exists some $x_1, x_2, \ldots x_n \in B$,

and $a_1, a_2, \ldots a_n$.

s.t. $v = \sum_{i=1}^{n} a_i x_i$ $\quad V = \text{span}(B)$

and $\forall x \in B$,

$|B|$ = "Dimension" $\quad x \notin \text{span}(B / \{x\})$

↓

terminology confusion

↙ numerical linear algebra ↘ numpy, SimPy, ...

$V = \mathbb{R}^2$.

$B = \left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\}$

$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = x_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + x_2 \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

numpy faster python.

- multiprocessing
- c++
- cache

* & @ easily mapped

↓ ↓

multiply tensor / matrix multiply

per-element.

broadcast → faster than manually comp.

(1.3.7)
(3.1.7)

(3.1.5)
(3.1.6)

X

(1.3.7) ✓ i.e. loop replicate.
(3.1.7)

(3.3.7)

(3.3.7)

numpy broadcasting.

Week 2 (1) Aug. 29

Automatic diff.

Definition - $\nabla f(w)^T$ is a linear map such that: $\Delta \in \mathbb{R}^d$

$$\nabla f(w)^T \Delta = \lim_{a \to 0} \frac{f(w+a\Delta) - f(w)}{a}$$

$f(x) = x^T A x.$

$$\lim_{a \to 0} \frac{(x+a\Delta)^T A (x+a\Delta) - x^T A x}{a}$$

$$\lim_{a \to 0} \frac{x^T A x + a\Delta^T A x + a x^T \Delta x + a^2 \Delta^T A \Delta - x^T A x}{a}$$

$$\lim_{a \to 0} \Delta^T A x + x^T A \Delta + a \Delta^T A \Delta.$$

$$\Delta^T A x + x^T A \Delta$$

$$\Delta^T (A x + A^T x).$$

$$\nabla f(x) = A x + A^T x$$

all elements.

$$f(x) = \|x\|_2^2 = \sum_{i=1}^{d} x_i^2$$

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

$$\nabla f(x) = 2x. \qquad f(x+\alpha\Delta) = (x+\alpha\Delta)^T (x+\alpha\Delta)$$

Euclidean norm

L_1 Norm.

$$f(x) = \|x\|_1 = \sum_{i=1}^{d} |x_i|$$

$$\frac{\partial f}{\partial x_i} = \frac{\partial}{\partial x_i} |x_i| = \text{sign}(x_i) = \frac{x_i}{|x_i|}$$

$$\nabla f(x) = \text{sign}(x).$$

- Symbolic differentiation.

write function in mathematical expression.

↓

apply chain rule, ...

↓

code.

→ Problems.

had to do it manually.

- converting code into mathe not trivial
- no guarantee → comput. structure.

why important?

differentiating f_x function.

→ Numerical differentiation.

- high-order different.
-
- error accumulating,

Problems.

- numerical imprecision.
  - ↳ number of operations increases.
- f not smooth → wrong results.
- unclear how to setup ε
- for a vector → scalar function, you have to compute each partial individually, meaning $O(d)$ blowup in cost.
- if ε too small, may get zeros or NaN

Automatic differentiation.

- Forward mode
- Reverse mode.

replace y with a tuple & a derivative

* operator overloading

Problems w/ Forward Mode AD.

- Differentiate with respect to one input
- Blow-up proportional to dimension d.

if we are computing a gradient of f:

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

Week 2 - 2

Aug 31

Backpropagation.

Review for Forward Mode AD.

$x \in \mathbb{R}$.

↓

$\frac{\partial y}{\partial x}$ → Store the tuple: $(y, \frac{\partial y}{\partial x})$.

* Python supports operator overloading

Reverse - Mode AD

→ fix one output l over $\mathbb{R}$.

↳ compute partial derivative $\frac{\partial l}{\partial y}$.

$(y, \nabla_y l)$.

Same shape

- Derive backprop. thru chain rule.

$l = f(u), u = g(y)$

$h = f(u_1, u_2, \dots, u_k)$, $u_i = g_i(y)$

$$\nabla_y h = \sum_{i=1}^{k} Dg_i(y)^T \nabla_{u_i} h$$

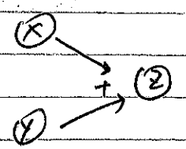
$Dg_i(y)$: derivative

is a Jacobian

compute gradient w.r.t. y.

$$\nabla_y h = \sum_{i=1}^{k} Dg_i(y)^T \nabla_{u_i} h$$

process y — process u_i

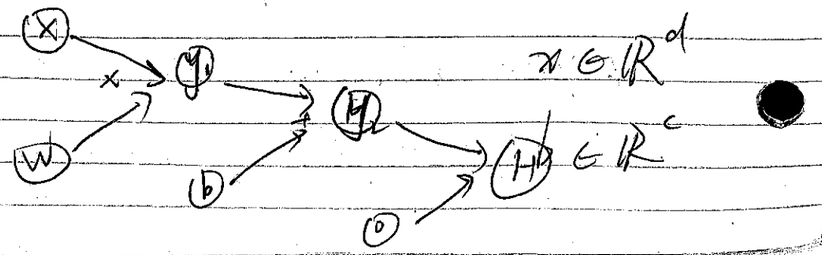


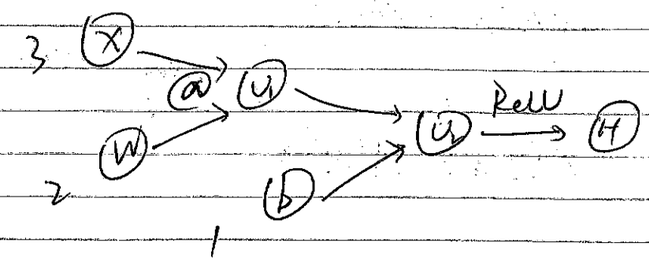
$z = x + y$

$H = \max(0, Wx + b)$. $W \in \mathbb{R}^{c \times d}$

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

$H \in \mathbb{R}^c$





$$\nabla_x H = \frac{\partial H}{\partial U_2} \frac{\partial U_2}{\partial g} \frac{\partial g}{\partial U_1} \frac{\partial U_1}{\partial x}$$

$H = f(U_2)$. $U_2 = g(U_1)$, $U_1 = h(x)$.

$\nabla_x H = ReLU'$

initialize

$\nabla_{U_2} H = \nabla_H H = \nabla_W H = \nabla_x H = \nabla_b H = 0$

$\nabla_{U_2} H \mathrel{+}= ReLU'(U_2) \cdot \nabla_H H = 1 \cdot 1 = 1$

Problems

1.2. Derivative of x^TAx.

b. $\Delta^TAx + x^TA\Delta$

↓ why $x^TA\Delta = (x^TA\Delta)^T$?

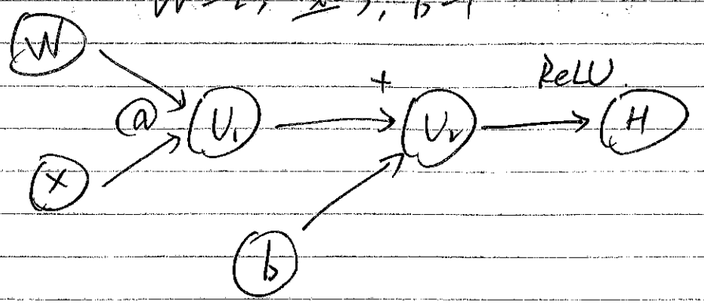
$\Delta^TAx + \Delta^TA^Tx$

d. → a Jacobian.

2.5. why sum(log) in np.

Review: Backprop.

$W=2,\ x=3,\ b=1$



Forward

- $U_1 = W * x = 2*3 = 6$
- $U_2 = U_1 + b = 7$
- $H = ReLU(U_2) = ReLU(Wx+b) = ReLU(7)$

- $\nabla_H H = 1$
- initialize: $\nabla_{U_2} H = \nabla_{U_1} H = \nabla_W H$
  $= \nabla_x H = \nabla_b H = 0$
- $\nabla_{U_2} H \mathrel{+}= ReLU'(U_2)\nabla_H H = 1*1 = 1$
- $\nabla_{U_1} H \mathrel{+}= \nabla_{U_2} H = 1$
- $\nabla_b H \mathrel{+}= \nabla_{U_2} H = 1$
- $\nabla_W H \mathrel{+}= x \nabla_{U_1} H = 3*1 = 3$

- $\nabla_x H += W \nabla_{v_1} H = 2 * 1 = 2$

$F(x) \rightarrow DF(x)$

$\Delta \mapsto$

Λ^{-1}

Labor day.

Week 3 - $\frac{1}{2}$.

Machine learning systems

$f : U \mapsto V$

$Df(x) \quad \mathcal{L}(U, V)$

$f : \mathbb{R}^m \to \mathbb{R}^n$.

$Df(x) \in \mathcal{L}(\mathbb{R}^m, \mathbb{R}^n)$.

$\mathbb{R}^{m \times n}$

If $F(x) = x^2$.

$\lim_{\alpha \to 0} \frac{F(x+\alpha\Delta) - F(x)}{\alpha}$.

$A \in \mathbb{R}^{n \times n}$.

$\lim_{\alpha \to 0} \frac{x^2 + \alpha\Delta x + \alpha x \Delta + \alpha^2\Delta^2 - x^2}{\alpha}$.

$= A x + x \Delta$

$DF(x) = (\Delta \mapsto \Delta x + x \Delta)$

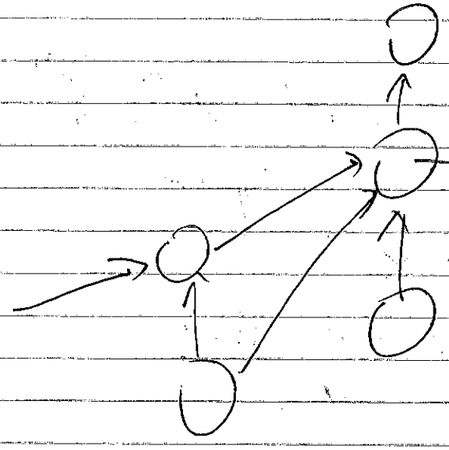
or $(DF(x))(\Delta) = \Delta x + x \Delta$

Why Backprop.

gradient has ↓ output l.

backprop manifest gradient

$\nabla_y l$ for intermediates y.



class / object

↓

knows how

to compute itself

↓

knows its dependencies

+

```
def __add__(self, another):
```

two ways:

(lazy)

EAGER v.s. GRAPH only later

compute immediately — do/can compute, but know the dependencies

Eager

↓ cutoff

↓ error

- take storage

- debugging & development (X).

- if / branch / control flow (✓).

immutabelized

↑

Graph.

- first computation.
- simplification (transform. ✓)
- reuse the graph. (✓)
- good for caching (?).
- memory locality (✓).

- less python overhead (✓)

* Deployment.

↓

convert it into some form that "isn't python"

- optimization

TF —Default→ Graph

PT ⟶ Eager

ML framework.

- numerical linear algebra library.
- hardware accelerator support.
- backprop engine. (gradient auto).
- library for writing Deep Nets.

PyTorch & TensorFlow

MxNet Jax. Flux (Julia)
(C++)

Caffe (ancient) → 1st GPU acc.

```
model = torch.nn.Sequential(
    nn.Linear( ... )
    nn.Linear( ... )
)
```

hardware - accelerator.

```
X. to ("cuda")
X. to ("cpu")
Y. to ("cpu")
Z = X + Y.
```

throw an exception when two diff devs or diff hardwares.

Programming Assignment #1.

def get_next_order
    return to a "+1" order

def to_backprop
    convert array to backprop

* class Backprop Array.

def __init__ (np array).

def __repr__ ( )
    ↦ String.

(1.1)

def all_dependencies( )
    find all the dependencies

use graph search alg.

(1.2)

def backward( )
    backward ⟹ find gradients

Start from the dependencies reverse mode

def grad_fn( )

"Define math operators in backprop"

def add
    → BA_Add( )

def sub
    → BA_Sub( )

def mul
    → BA_Mul( )

def truediv
    → BA_Div( )

def sum ( )
    → BA_Sum( ).

def reshape ( )
    → BA_Reshape( )

def transpose ( )
    → BA_Transpose( )

Different classes

Doesn't matter what tensor operation for (2.2)

Tensor operation

(2.1)

(2.3)

Compare "grad"

implement for scalar-arrays.

(1.3)

compare ad, nd, sd

(1.5)

global funs.

- def numerical_diff( ).
- def numerical_grad( ).
- def backprop_diff( ).

(2.3) (2.4) (2.5) (2.6)

Store test functions in a class:

class TestFxs( ):

def {f1, df1dx, f2, ...}

test implementation: TestFxs.df3dx (1.4)

wish'd input

if __name__ = "__main__":

scalar output

write the script to execute main functions.

for #3:

$$\lim_{\alpha \to 0} \frac{\log(x+\alpha\Delta) - \log(x)}{\alpha} = D(f(x))\Delta$$

$$\frac{\exp(x+\alpha\Delta) - \exp(x)}{\alpha}$$

$$= \frac{\exp(x)\cdot\exp(\alpha\Delta) - \exp(x)}{\alpha}$$

$$= \frac{\exp(x)(\exp(\alpha\Delta) - \exp(0))}{\alpha \cdot \exp(0)}$$

$$=$$

Week 4 - Lecture 7.

GD

we compute gradient in large-scale opt. problem to solve learning task

Examples will most be on supervised learning. but not limited to ...

in SL : $f(w) = \frac{1}{n} \sum_{x,y \in D} \ell(h_w(x), y)$.

$= \frac{1}{n} \sum_{x \in D} f(w; x)$

n: size of dataset.

in framework, we strive to:

Minimize $f(w)$ over $w \in \mathbb{R}^d$.

computing f takes $O(n)$ time.

Init $w_0 = 0$ (not nec. $= 0$.)

$w_{t+1} = w_t - \alpha \nabla f(w_t)$.

↳ gradient descent.

gradient.

★ How much time does it cost?

→ $O(ndk)$.

★ How much memory required?

→ $O(nd)$ or $O(d)$

↓

• parallel way

running backprop thru whole thg.

in a NAIVE way → $O(nd)$.

Newton's Method

$w_{t+1} = w_t - [\nabla^2 f(w_t)]^{-1} \nabla f(w_t)$

↳ converges faster.

"BUT" more expensive that GD

So, ★ How much time? & ★ How m. mem?

store Hessian matrix.

→ $O(nd^2k)$. → $O(d^2)$

(Q) scared away from Newton's method cuz the large memory required, i.e. quadratic GD.

✡ why are we confident that GD will converge?.

the value of gradient as compared based on particular weight, if move a lil' bit, it will drastically change grad.

we want to make it a bust.

How close GD is?

→ bound. Derivative.

↳ second-order D. is also Bd.

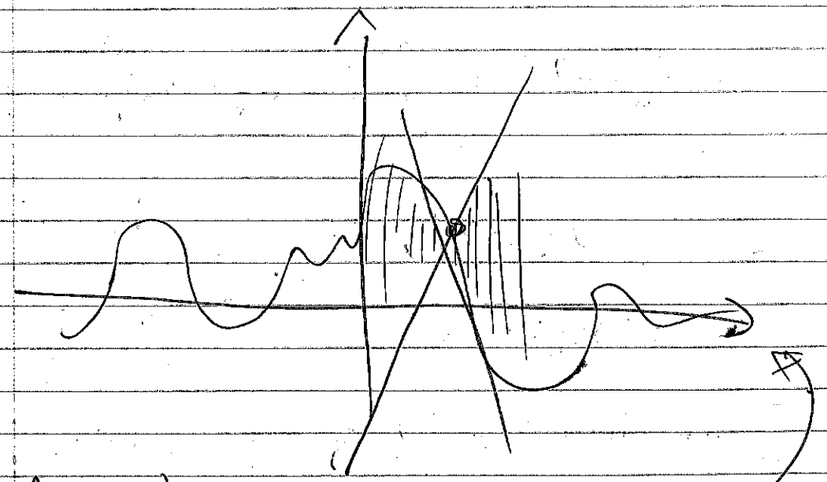
Gradient ∇f is L- Lipshitz continuous

$$\|\nabla f(u) - \nabla f(v)\|_2 \le L\|u-v\|_2$$

Assume $\exists f$ s.t. $f(w) \ge f^*$.

$\forall w, \forall u, \|u\|=1$.

$$\left\| \frac{\partial^2}{\partial \alpha^2} f(w+\alpha u) \right\| \le L.$$



~~AAA~~ Lipshitz continuous

↳ L- smooth.

How to prove GD converges ???

$$f(w_{t+1}) = f(w_t - \alpha \nabla f(w_t))$$

fundamental thm. of calc.

$$= f(w_t) + \int_0^\alpha \frac{\partial}{\partial \eta} f(w_t - \eta \nabla f(w_t))\, d\eta$$

$$f(b) - f(a) = \int_a^b \frac{\partial}{\partial x} f(x)\, dx.$$

$$= f(w_t) + \int_0^\alpha \left(-\nabla f(w_t)\right)^T \nabla f(w_t - \eta \nabla f(w_t))\, d\eta.$$

how? → close to grad.

↳ imagine = 0.

Let $h(\eta) = f(w_t - \eta \nabla f(w_t))$

$$h(\alpha) = h(0) + \int_0^\alpha h'(\eta)\, d\eta$$

Expect η to be small ← expect α small

$$\frac{\partial}{\partial \eta} f(w_t - \eta \nabla f(w_t))$$

$$= \lim_{\delta \to 0} \frac{f(w_t - (\eta+\delta)\nabla f(w_t)) - f(w_t - \eta \nabla f(w_t))}{\delta}$$

$$= \lim_{\delta \to 0} \frac{f(w_t - \eta \nabla f(w_t) - \delta \nabla f(w_t)) - f(w_t - \eta \nabla f(w_t))}{\delta}.$$

$$= \lim_{\delta \to 0} \frac{f(\hat{w} + \delta(-\nabla f(w_t))) - f(\hat{w})}{\delta}$$

$$= -(\nabla f(w_t))^T \nabla f(\hat{w}).$$

⇒ close to norm. → always positive ⇒ gd works to min.

→ cont.

$$= f(w_t) + \int_0^{\alpha} (-\nabla f(w_t)^T \nabla f(w_t))\, d\eta$$

$$+ \int_0^{\alpha} (-\nabla f(w_t))^T (\nabla f(w_t - \eta \nabla f(w_t)) - \nabla f(w_t))\, d\eta.$$

Apply Cau. theorem:

$$A \cdot B \leq \|A\| \|B\|$$

$$\{ \|AB\|, \}$$

$$\leq f(w_t) + \int_0^{\alpha} (-\nabla f(w_t)^T \nabla f(w_t))\, d\eta$$

$$+ \int_0^{\alpha} \|-\nabla f(w_t)\| \, \|\nabla f(w_t - \eta \nabla f(w_t)) - \nabla f(w_t)\|\, d\eta.$$

$$\leq f(w_t) + \int_0^{\alpha} -\|\nabla f(w_t)\|^2 \, d\eta$$

$$\frac{\alpha^2}{2} = \int_0^{\alpha} \eta \, d\eta$$

$$+ \int_0^{\alpha} \|-\nabla f(w_t)\| \cdot L \cdot \|\eta \nabla f(w_t)\|\, d\eta$$

$$\Rightarrow f(w_t) \leq f(w_t) - \alpha \|\nabla f(w_t)\|^2 + \frac{\alpha^2 L}{2} \|\nabla f(w_t)\|^2$$

$$\leq f(w_t) - \alpha\left(1 - \frac{\alpha L}{2}\right) \|\nabla f(w_t)\|^2$$

$\Rightarrow$ How does constant L effect how we learn tasks in scales

$\downarrow$

that's why

Assume: $\alpha L < 1$.

$$\leq f(w_t) - \frac{\alpha}{2} \|\nabla f(w_t)\|^2$$

$$\frac{\alpha}{2}\|\nabla f(w_t)\|^2 \le f(w_t) - f(w_{t+1})$$

↳ Single iteration of GD.

$$\frac{\alpha}{2}\sum_{t=0}^{k}\|\nabla f(w_t)\|^2 \le \sum_{t=0}^{k-1}\left(f(w_t) - f(w_{t+1})\right)$$

↳ k iterations $< f(w_0) - f(w_k)$.

$$\frac{\alpha}{2}\sum_{t=0}^{k-1}\|\nabla f(w_t)\|^2 \le f(w_0) - f(w_t)$$

assume f is bounded from below. ★★★

$$= \le f(w_0) - f^*.$$

$$\frac{\alpha}{2}k \min_{t\in\{0,\dots,k-1\}}\|\nabla f(w_t)\|^2 \le f(w_0) - f(w_t)$$

$$\min_{t\in\{0,\dots,k-1\}}\|\nabla f(w_t)\|^2 \le \frac{2f(w_0) - f^*)}{\alpha k}.$$

↳ require $\alpha k \le 1$.

LHS: run to iteration of GD

return the optimal weights.

RHS upper bound of func.

Step size α → to be large as possible to $\frac{1}{\alpha}$ → small

the largest step

↳ $\le \frac{2(f(w_0) - f^*)}{k}$.

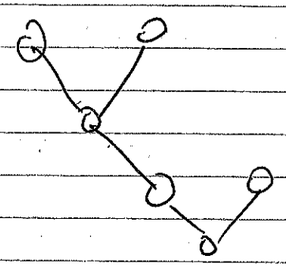
GD only find local opt.

OH (Sep. 13)

(1.1)

- Breadth-first search ?

what are the dependencies



(1.2)

what is the utility of the grad-fn

why zero out ".grad" ?

(1.3)

1.3.4. — why ?

(2)

generally, how write the help function to manipulate the dimensions ?

(2.5) numerical-grad. — hints ?

(2.7).

$\mathbb{R}^d \longrightarrow \mathbb{R}$

confused on the implementation.

② writing a for loop.

while the len

① add empty row / cols. for extra dim. → new axis.

check the dims are targeting the same

⇒ broadcasting.

A: 1 x 6 x 5

↕

B: (1) x 1 x 2

→ iterating over the axis.

1: adding 1.

2: repeating axis (iterating)

helper_fns

BA - And

just recall += helper(a, b)

Lecture 8 . . (Wk. 4).

HW Review:

$\nabla_{\underline{x}} \ell$ has same shape as $\underline{X}$

$\underline{X}$.grad.shape = $\underline{X}$.data.shape.

* "+=" in numpy mutates the vector

self.x.grad += stuff

self.x.grad = self.x.grad + stuff.

* init both as arrays as all zeros

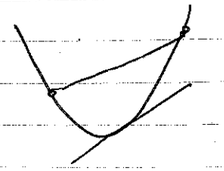
GD continued.

... last time.

$w_{t+1} = w_t - \alpha \nabla f(w_t)$

→ most simple condition - convexity

0th-order convexity



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

for $\alpha \in [0, 1]$

▽ 1st order def. — considering whether func. conv.

$$f(x) + (y-x)^T \nabla f(x) \leq f(y)$$

▽ 2nd order def.

$$\frac{\partial^2}{\partial \alpha^2} f(x + \alpha u) \geq 0 \quad \text{for any } \alpha, u \in \mathbb{R}^d$$

"parabola has a const. sec. deriv."

- Convex has a unique global optimum.

"that's why we care 'bout conv"

Strongly convex function

"a function that can be bounded from below"

μ- strongly convex function.

(scalar positive num.)

$$\frac{\partial^2}{\partial \alpha^2} f(x + \alpha u) \geq \mu \quad (\text{for } \|u\| = 1)$$

e.g. "Strongly convex".

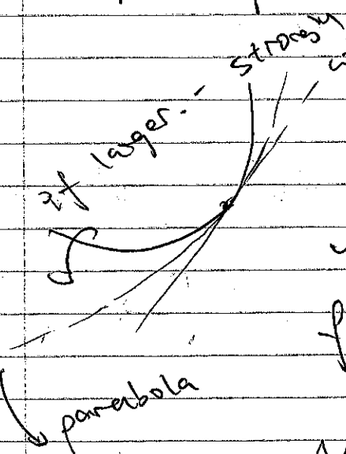
$f(x) = \max(x, 0)$

Convexity: vector space ↦ real num.
i.e. scalar.

Strictly convex vs. Strongly convex

e.g. $f(x) = e^x$ ↳ but not larger than a positive val.

$$f(y) \geq f(x) + (y-x)^T \nabla f(x) + \frac{\mu}{2}\|y-x\|^2$$



f is μ-strongly convex, iff $\exists g$ s.t.

$$f(w) = g(w) + \frac{\mu}{2}\|w\|^2$$

& g is convex.

☆☆☆

Use in a lot of analysis.

Polyak-Łojasiewicz (PL)

... TBC

$$\|\nabla f(w)\|^2 \geq 2\mu \underbrace{(f(w) - f^*)}_{\text{min val.}}$$

Strongly convex.

but does not imply strongly convex

e.g. $f(w) = \frac{\mu}{2}\|w\|^2$.

$\nabla f(w) = \mu w$

$\|\nabla f(w)\|^2 = \|\mu w\|^2 = \mu^2 \|w\|^2$

$= 2\mu \left(\frac{\mu}{2}\|w\|^2\right) = 2\mu (f(w) - f^*)$

☆ the gradient is only close to zero when its only around global optimum.

satisfies PL yet not strongly cv.

$f(w_{t+1}) = f(w_t - \alpha \nabla f(w_t))$

$\leq f(w_t) - \alpha\left(1 - \frac{\alpha L}{2}\right)\|\nabla f(w_t)\|^2$

non-convex of gd.

$\leq f(w_t) - 2\mu\alpha\left(1 - \frac{\alpha L}{2}\right)(f(w_t) - f^*)$

$f(w_{t+1}) - f^* \leq (f(w_t) - f^*) - 2\mu\alpha\left(1 - \frac{\alpha L}{2}\right)(f(w_t) - f^*)$

$= \left(1 - \underbrace{2\mu\alpha\left(1 - \frac{\alpha L}{2}\right)}_{\text{call it } \gamma}\right)(f(w_t) - f^*)$

$\leq \gamma (f(w_t) - f^*)$ ← min, ↳ min is γ

$f(w_k) - f^* \leq \gamma^k (f(w_0) - f^*)$ if $\gamma > 0$.

☆ How should we chose α? ↳ to show $1 - \frac{\mu}{L}$ is min

$$2\mu\alpha - \frac{2\mu\alpha^2 L}{2} \to 0$$

$$1 - \alpha L = 0$$

$$\alpha L = 1$$

$$\alpha = \frac{1}{L}.$$

we got $\gamma = 1 - L\mu\frac{1}{L}\left(1 - \frac{\frac{1}{L}L}{2}\right) = 1 - \frac{\mu}{2}$

$$f(w_k) - f^* \leq \left(1 - \frac{\mu}{L}\right)^k \left(f(w_0) - f^*\right)$$

$$\leq \exp\left(-\frac{\mu k}{L}\right) \cdot \left(f(w_0) - f^*\right)$$

convergence @ a linear rate

even tho is decreasing exponentially

converges with errors a num. of digits.

going to be linear wrt. ...

"roughly linear with digits not. sense".

Goal: output $\hat{w}$ with

$f(\hat{w}) - f^* \in \Sigma \mapsto$ Error tol.

$(\varepsilon \in \mathbb{R},\ \varepsilon > 0)$.

It suffices to run T iterations of (num timestep)

GD, s.t.

$$\exp\left(-\frac{\mu}{L}T\right)\left(f(w_0) - f^*\right) \leq \varepsilon.$$

we can solve this for T

$$\exp\left(\frac{\mu}{L}T\right) \geq \frac{f(w_0) - f^*}{\varepsilon}$$

$$\frac{\mu}{L}T \geq \log\left(\frac{f(w_0) - f^*}{\varepsilon}\right)$$

$$T \geq \frac{L}{\mu}\log\left(\frac{f(w_0) - f^*}{\varepsilon}\right)$$

hessian for sc. pos. def symm.

ratio of largest eigen val of hessian / the smallest val.

$\frac{L}{\mu} = K$ "condition number".

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

$\left|\frac{\partial^2}{\partial \alpha^2} f(x+\alpha u)\right| \le L$

(for $\|u\| = 1$)

eqly. $-L I \le H f(x) \le L I$

$\nabla^2 f$

L large → lot of steps

L small → a few steps.

$L \gg 1 \hookrightarrow k = 1$ isotropic quadratic

Condition number ∝ num. of steps

— to solve the opt. problem.

the other thing impact time of GD.

→ size of dataset.

↓

Stochastic GD ← GD goes bad when DS are large.

↓

Subsampling + approx.

↓

Principle #2 of LS ML.

Spoiler: SGD faster than GD.

GD # steps. $\sim \vartheta(K)$.

SGD # time steps. $\sim \vartheta(nK)$

GD

Lecture 7

Week 3.

GD time

Subscription

$O(nk)\ \log\left(\frac{1}{\varepsilon}\right)$

Num. of iterations of GD.

size

training set

$f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$.

$$\text{GD}\begin{cases} w_{t+1} = w_t - \alpha \nabla f(w_t) \\ = w_t - \frac{\alpha}{n}\sum_{i=1}^{n} \nabla f_i(w_t) \end{cases}$$

SGD: sample a random $i \in \{1, \dots, n\}$.

$w_{t+1} = w_t - \alpha \nabla f_i(w_t)$.

mini-batch SGD: sampling B

$w_{t+1} = w_t - \frac{\alpha}{|B|}\sum_{b=1}^{B} \nabla f_{i_b}(w_t)$

batch size

sample i_b uniformly from $\{1, \dots, n\}$ for each $b \in \{1, \dots, B\}$

Stochastic GD (mini-batch)

$$\mathbb{E}\left[w_{t+1} \mid w_t\right] = w_t - \alpha \nabla f(w_t)$$

$$\mathbb{E}\left[\nabla f_i(w_t) \mid w_t\right] = \sum_{j=1}^{n} P(i=j) \nabla f_j(w_t)$$

$$= \sum_{j=1}^{n} \frac{1}{n} \nabla f_j(w_t).$$

$$= \nabla f(w_t).$$

Assume $\exists L > 0$ s.t.

$$\|\nabla f(w) - \nabla f(v)\| \le L\|w - v\|$$

Assume. $f(w) \ge f^*$

let g_t denote $\frac{1}{B}\sum_{b=1}^{B} \nabla f_{i_b}(w_t) - g_t$

$f(w_{t+1}) = f(w_t - \alpha g_t) = f(w_t) + \int_0^{\alpha} \frac{\partial}{\partial \eta} f(w_t - \eta g_t)\, d\eta = f(w_t) + \int_0^{\alpha} \nabla f(w_t - \eta g_t)(-g_t)\, d\eta$

fwd loss function

$$= f(w_t) - \alpha \nabla f(w_t)^T q_t + \int_0^\alpha (\nabla f(w_t - \eta q_t) - \nabla f(w_t))^T (-q_t)\, d\eta$$

$$\leq f(w_t) - \alpha \nabla f(w_t)^T q_t + \int_0^\alpha \|\nabla f(w_t - \eta q_t) - \nabla f(w_t)\| \, \|q_t\| \, d\eta$$

$$\leq f(w_t) - \alpha \nabla f(w_t)^T q_t + \int_0^\alpha L \eta \|q_t\| \|q_t\| \, d\eta$$

$$\leq f(w_t) - \alpha \nabla f(w_t)^T q_t + \frac{\alpha^2 L}{2} \|q_t\|^2$$

full-batch grad

$$f(w_{t+1}) \leq f(w_t) - \alpha \nabla f(w_t)^T q_t + \frac{\alpha^2 L}{2} \|q_t\|^2$$

$\nabla f(w_t)$ ← Stochastic grad.

$$\mathbb{E}[f(w_{t+1}) | w_t] \leq f(w_t) - \alpha \nabla f(w_t)^T \mathbb{E}[q_t | w_t] + \frac{\alpha^2 L}{2} \mathbb{E}[\|q_t\|^2 | w_t]$$

$$\leq f(w_t) - \alpha_t \nabla f(w_t)^T \nabla f(w_t) + \frac{\alpha^2 L}{2} \mathbb{E}[\|q_t\|^2 | w_t]$$

$$\leq f(w_t) - \alpha \|\nabla f(w_t)\|^2 + \frac{\alpha^2 L}{2} \mathbb{E}[\|q_t\|^2 | w_t]$$

New Assumption (uniform over w_t)

$$\frac{1}{n} \sum_{i=1}^n \|\nabla f_i(w_t) - \nabla f(w_t)\|^2 \leq \sigma^2$$

for any $w \in \mathbb{R}^d$.

$$\mathbb{E}[\|q_t\|^2 | w_t] = \mathbb{E}\left[\left\| \frac{1}{B} \sum_{b=1}^B \nabla f_{i_b}(w_t) \right\|^2 | w_t\right]$$

$$= \mathbb{E}\left[\left(\frac{1}{B} \sum_{b=1}^B \nabla f_{i_b}(w_t)\right)^T \left(\frac{1}{B} \sum_{c=1}^B \nabla f_{i_c}(w_t)\right) \middle| w_t\right]$$

$$= \frac{1}{B^2} \sum_{b=1}^B \sum_{c=1}^B \mathbb{E}[\nabla f_{i_b}(w_t)^T \nabla f_{i_c}(w_t) | w_t]$$

$\mathbb{E}[\ ]$ → samples that $b \neq c$; independent of each other → $b \neq c$

$$= \frac{1}{B^2}(B^2 - B) \|\nabla f(w_t)\|^2 + \frac{1}{B^2} \sum_{b=1}^B \mathbb{E}[\|\nabla f_{i_b}(w_t)\|^2 | w_t]$$

same

$$= \frac{B-1}{B} \|\nabla f(w_t)\|^2 + \frac{1}{B n} \sum_{i=1}^{n} \|\nabla f_i(w_t)\|^2$$

Recall the assumption:

$$\frac{1}{n}\sum_{i=1}^{n} \|\nabla f_i(w_t)\|^2 - \frac{2}{n}\sum_{i=1}^{n} \nabla f_i(w)^T \nabla f(w) + \frac{1}{n}\sum_{i=1}^{n} \|\nabla f(w)\|^2 \le \sigma^2$$

$$-2\|\nabla f(w)\|^2$$

Easy to bound $\|\nabla f(w)\|^2$.

$$\frac{1}{n}\sum_{i=1}^{n} \|\nabla f_i(w)\|^2 - \|\nabla f(w)\|^2 \le \sigma^2$$

$$\le \frac{B-1}{B}\|f(w_t)\|^2 + \frac{1}{B}\left(\sigma^2 + \|\nabla f(w)\|^2\right)$$

$$= \|\nabla f(w_t)\|^2 + \frac{\sigma^2}{B}$$

Coming back to 3rd assumption

$$\mathbb{E}\left[f(w_{t+1}) \mid w_t\right] \le f(w_t) - \alpha \|\nabla f(w_t)\|^2 + \frac{\alpha^2 L}{2}\|\nabla f(w_t)\|^2 + \frac{\alpha^2 \sigma^2 L}{2B}$$

Variance of the square error of SGD

$\mathbb{E}[Y] = \mathbb{E}[\mathbb{E}[Y|X]]$

Also assume: $\|\nabla f(w)\|^2 \ge 2\mu(f(w) - f^*)$

PL condition

$$\le f(w_t) - \alpha\left(1 - \frac{\alpha L}{2}\right) 2\mu (f(w_t) - f^*) + \frac{\alpha^2 \sigma^2 L}{2B}$$

$$\mathbb{E}\left[f(w_{t+1}) - f^* \mid w_t\right] \le \left(1 - \alpha\left(1 - \frac{\alpha L}{2}\right)2\mu\right)(f(w_t) - f^*) + \frac{\alpha^2 \sigma^2 L}{2B}$$

SGD additional "noise" term.

$$\Leftarrow \rho_t = \mathbb{E}\left[f(w_t) - f^*\right]$$

$$\rho_{t+1} \le \left(1 - \alpha\left(1 - \frac{\alpha L}{2}\right)2\mu\right)\rho_t + \frac{\alpha^2 \sigma^2 L}{2B}$$

$\Rightarrow$ Assume $\alpha L \le 1$.

$$\le (1 - \alpha\mu)\rho_t + \frac{\alpha^2 \sigma^2 L}{2B}$$

$$\rho_\alpha = (1-\alpha/\mu)\rho_\alpha + \frac{\alpha^2\sigma^2 L}{2B}$$

$$\alpha/\mu \rho_\alpha = \frac{\alpha\sigma^2 L}{2B}$$

$$\rho_\alpha = \frac{\alpha\sigma^2 L}{2\mu B} = \frac{\alpha\sigma^2 K}{2B}$$

$$\rho_{t+1} - \frac{\alpha\sigma^2 L}{2\mu B} \le (1-\alpha/\mu)\rho_t - \frac{\alpha\sigma^2 L}{2\mu B}$$

$$+ \frac{\alpha^2\sigma^2 L}{2B}$$

$$\le (1-\alpha/\mu)\left(\rho_t - \frac{\alpha\sigma^2 L}{2\mu B}\right)$$

$$\rho_T - \frac{\alpha\sigma^2 L}{2\mu B} \le (1-\alpha/\mu)^T \left(\rho_0 - \frac{\alpha\sigma^2 L}{2\mu B}\right)$$

Step:

$$\le (1-\alpha/\mu)^T \rho_0$$

$$\mathbb{E}[f(w_T) - f^*] \le (1-\alpha/\mu)^T (f(w_0) - f^*)$$

$$+ \frac{\alpha^2\sigma^2 L}{2\mu B} \rightarrow \text{noise wall / noise floor}$$

$$\Rightarrow 1 - x \le \exp(-x) \quad (\text{plug in})$$

$$\text{LHS} \le \exp(-\alpha\mu T)(f(w_0) - f^*) + \frac{\alpha\sigma^2 L}{2\mu B}$$

Lecture 9. Week 5 - 2.

Rev - Last time.

$$\mathbb{E}[f(w_t) - f^*] \le \exp(-\alpha\mu T)(f(w_0) - f^*) + \frac{\alpha L \sigma^2}{2B\mu}.$$

Goal: $\mathbb{E}[f(w_{out}) - f^*] \le \varepsilon$.

suffice for $\varepsilon \ge \exp(-\alpha\mu T)(f(w_0)-f^*) + \frac{\alpha L\sigma^2}{2B\mu}$.

also suffice $\frac{\varepsilon}{2} \ge \exp(-\alpha\mu T)(f(w_0)-f^*)$

and $\frac{\varepsilon}{2} \ge \frac{\alpha L\sigma^2}{2B\mu}$.

$$\Rightarrow \alpha \le \frac{B\mu\varepsilon}{L\sigma^2}.$$

$$\log\left(\frac{\varepsilon}{2(f(w)-f^*)}\right) \ge -\alpha\mu T.$$

$$\Rightarrow T \ge \frac{1}{\alpha\mu}\cdot\log\left(\frac{2(f(w_0)-f^*)}{\varepsilon}\right)$$

also $\alpha L \le 1$.

$$\alpha \le \min\left(\frac{B\mu\varepsilon}{\sigma^2}, 1\right)\cdot\frac{1}{L}$$

$$T \ge \max\left(\frac{\sigma^2}{B\mu\varepsilon}, 1\right)\frac{L}{\mu}\log\left(\cdots \frac{2(f(w_0)-f^*)}{\varepsilon}\right)$$

$$\ge \max\left(\frac{\sigma^2}{B\mu\varepsilon}, 1\right)\cdot K\cdot\log(\cdots)$$

(runtime of SGD): $\mathcal{O}\left(\max\left(\frac{\sigma^2}{B\mu\varepsilon}, 1\right)\right)\cdot K\cdot\log\left(\frac{1}{\varepsilon}\right)$

usually, ε are small

$\tilde{\Theta}\left(\frac{\sigma^2}{B\mu\varepsilon}K\right)$ → batch B cancels

↳ ignoring "log" term.

runtime of GD. $\mathcal{O}(nK\log(\frac{1}{\varepsilon}))$

for GD $n \longrightarrow \frac{\nabla^2}{B\varepsilon}$.

GD	SGD
• convex (strongly).	• non-convex
• Global opt. guaranteed (unique).	• multi-objective
• large memory. (GPU). parallel	• huge dataset • huge n $\varepsilon \rightarrow$ large "noisy" or not too small
• really small ε	
• large ∇^2	

How to set batch size B

⇓

★ Hardware consideration.

eg. $f_i(w) = \frac{1}{2}(x_i^T w - y_i)^2$.

$\nabla f_i(w) = x_i(x_i^T w - y_i)$ → dot, axpy

minibatch $X \in \mathbb{R}^{B\times d}$. $y \in \mathbb{R}^B$

$\nabla f_{batch}(w) = \frac{1}{B} X^T (Xw - Y)$

mmpy $(B,d)\,@\,(d,1)$,

faster

$(d,B)\,@\,(B,1)$

$(1024,1024)\,@\,(1024,1024)$ a lot less than $1024\times$

$(1024,1024)\,@\,(1024,1)$

minibatch size ∝ hardware itself

e.g. $B = 256, 64, \ldots$

SGD → GD: sampling w/o replacement.

$\max(n, 1)$ → is GD scenario

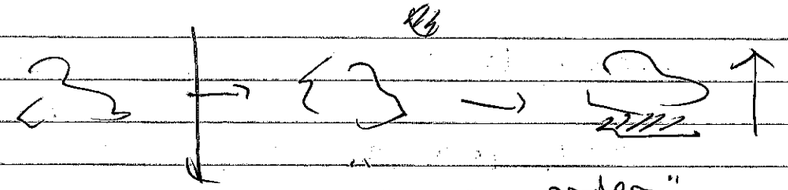
larger, slower

Random reshuffling

sample w/o repl., only reuse example after every example has been used.

epoch ⇒ 1 past thru the training set (e.g. 100 ~ 200).

Shuffle-once



Statistically not -

use a fixed step size.

Diminishing step size (assuming $\alpha_t L \le 1$)

$$\mathbb{E}[f(w_{t+1}) - f^*] \le (1 - \alpha_t \mu)\,\mathbb{E}[f(w_t) - f^*] + \frac{\alpha_t^2 \sigma^2 L}{2B}$$

$$\rho_{t+1} \le (1 - \alpha_t \mu)\rho_t + \frac{\alpha_t^2 \sigma^2 L}{2B}$$

$$\Rightarrow 0 = -\mu \rho_t + \frac{\alpha_t \sigma^2 L}{B}$$

$$\Rightarrow \alpha_t \triangleq \frac{\rho_t \mu B}{\sigma^2 L}$$

$$\rho_{t+1} \le \rho_t - \frac{\mu^2 B}{2\sigma^2 L}\rho_t^2$$

$$\frac{1}{\rho_{t+1}} \ge \frac{1}{\rho_t - \frac{\mu^2 B}{2\sigma^2 L}\rho_t^2} \quad (\text{let } \rho_t)$$

$$= \frac{1}{\rho_t}\left(1 - \frac{\mu^2 B}{2\sigma^2 L}\rho_t\right)^{-1}$$

$$\ge \frac{1}{\rho_t}\left(1 + \frac{\mu^2 B}{2\sigma^2 L}\rho_t\right) = \frac{1}{\rho_t} + \frac{\mu^2 B}{2\sigma^2 L}$$

$$\Rightarrow \frac{1}{\rho_T} \ge \frac{1}{\rho_0} + \frac{\mu^2 B T}{2\sigma^2 L} \Rightarrow \rho_T = \mathbb{E}[f(w_T) - f^*]$$

$$\leq \left(\frac{1}{f(w_0) - f^*} + \frac{\mu^2 BT}{2\sigma^2 L} \right)^{-1}$$

$$= \vartheta^2 \left(\frac{1}{T} \right)$$

$$\simeq \vartheta \left(\frac{\sigma^2 K}{\mu BT} \right)$$

Lecture 11 Week 6.

Momentum.

Rev: $K \sim \frac{L}{\mu}$; grad. noise $\propto \left(\frac{\sigma^2}{\mu} \right)$

Announcement: • Pset 3
• PA 2.

Rev: GD.

$$\ell \to f(w_T) - f^* \leq \exp\left(-\frac{T}{K} \right) (f(w_0) - f^*)$$

gap after T epochs

Q. Simplest model with large K?

a 2D problem → Quadratic

e.g. $f(w) = \frac{1}{2}(w_1^2 L + w_2^2 \mu)$

$$= \frac{1}{2} w^T \begin{bmatrix} L & 0 \\ 0 & \mu \end{bmatrix} w$$

$\nabla^2 f(w)$

$\Rightarrow H(w) = \begin{bmatrix} L & 0 \\ 0 & \mu \end{bmatrix}$ Suppose $L > \mu > 0$

$$\nabla f(w) = \begin{bmatrix} L & 0 \\ 0 & \mu \end{bmatrix} w$$

$$w_{t+1} = w_t - \alpha \nabla f(w_t)$$

$$= w_t - \alpha \begin{bmatrix} L & 0 \\ 0 & \mu \end{bmatrix} w_t$$

$$= \begin{pmatrix} 1-\alpha L & 0 \\ 0 & 1-\alpha\mu \end{pmatrix} w_t$$

power "T"

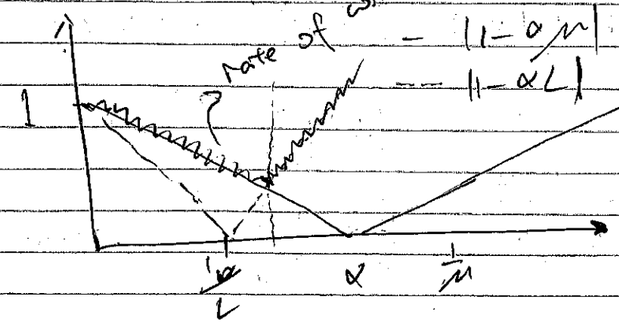
$$\Rightarrow w_T = \begin{bmatrix} 1-\alpha L & 0 \\ 0 & 1-\alpha\mu \end{bmatrix}^T w_0$$

$$= w_T - \begin{bmatrix} (1-\alpha L)^T & 0 \\ 0 & (1-\alpha\mu)^T \end{bmatrix} w_0$$

$$f(w_T) = \frac{1}{2}\left(L(1-\alpha L)^{2T} (w_0)_1^2 + \mu(1-\alpha\mu)^{2T} (w_0)_2^2 \right)$$

rate of convergence $= |1-\alpha\mu|$

$= |1-\alpha L|$



$$\alpha = \frac{2}{L+\mu}$$

$$= \frac{1-\frac{\mu}{L}}{1+\frac{\mu}{L}} = \frac{1-\frac{\mu}{L}}{k+1}$$

$$= 1 - \frac{2}{k+1}$$

$$\Rightarrow f(w_T) = \sigma\left(\left(1-\frac{2}{k+1}\right)^{2T}\right) = \mathcal{O}\left(\exp\left(-\frac{4T}{k+1}\right)\right)$$

$$w_{t+1} = w_t - \alpha \nabla f(w_t) + \beta(w_t - w_{t-1})$$

previous step — the more step, step size too large, overshooting the object.

"Self-tune" the step size

$$w_{t+1} - w_t = \cancel{-\alpha \nabla f(w_t)}$$

$$\beta(w_t - w_{t+1}) - \alpha \nabla f(w_t)$$

we need — particle undergoes "some" force term

$0 < \beta < 1$

"friction" term

- or momentum.

How Polyak momentum applies to 2D system?

$$w_{t+1} = w_t - \alpha \underbrace{\begin{bmatrix} L & 0 \\ 0 & \mu \end{bmatrix}}_{A} w_t + \beta(w_t - w_{t-1})$$

$$\begin{bmatrix} w_{t+1} \\ w_t \end{bmatrix} = \begin{bmatrix} w_t - \alpha A w_t + \beta(w_t - w_{t-1}) \\ w_t \end{bmatrix}$$

$$= \begin{bmatrix} (1+\beta)I - \alpha A & -\beta \\ I & 0 \end{bmatrix} \begin{bmatrix} w_t \\ w_{t-1} \end{bmatrix}$$

$$\begin{bmatrix} w_{T+1} \\ w_T \end{bmatrix} = \begin{bmatrix} (1+\beta)I - \alpha A & -\beta I \\ I & 0 \end{bmatrix}^T \begin{bmatrix} w_1 \\ w_0 \end{bmatrix}$$

4×4 block matrix

$$\begin{bmatrix} (1+\beta) - \alpha L & -\alpha & -\beta & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & (1+\beta) - \alpha\lambda & -\beta \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Eigenvalue

use χ to find eigenvalues.

$$\begin{vmatrix} (1+\beta) - \alpha\lambda - \chi & -\beta \\ 1 & 0 - \chi \end{vmatrix} \overset{(2a)}{=} \chi^2 - (1+\beta-\alpha\lambda)\chi + \beta$$

$$\chi = \frac{1+\beta-\alpha\lambda \pm \sqrt{(1+\beta-\alpha\lambda)^2 - 4\beta}}{2}$$

complex vals.

Polyak Sives:

$$\alpha = \frac{1+2\beta}{L+\mu}, \quad \sqrt{\beta} = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$$

$$|\chi|^2 = \frac{1}{4}\left[(1+\beta-\alpha\lambda)^2 + 4\beta - (1+\beta-\alpha\lambda)^2\right]$$

$$T \approx \sqrt{\kappa} \cdot \sqrt{\beta}^T \approx \exp\left(-\frac{L}{\sqrt{\kappa}+1}T\right)$$

$$T \approx \sqrt{\kappa}$$

Works when $L(\beta \geq (1+\beta-\alpha\lambda)^2$

$$|2\sqrt{\beta}| \geq |1+\beta+\alpha\lambda|$$

$$\hookrightarrow 2\sqrt{\beta} \geq 1+\beta+\alpha\lambda \geq -2\sqrt{\beta}$$

$\mu \leq \lambda \leq L$

Lecture 11. - Week 6.

Conditioning

Rev. - momentum.

$T = \mathcal{O}\left(nK \log\left(\frac{1}{\varepsilon}\right)\right)$

$\Uparrow$

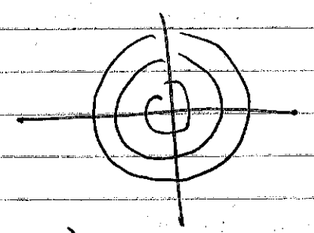
condition num.

Could be too large / too small.

but lec.

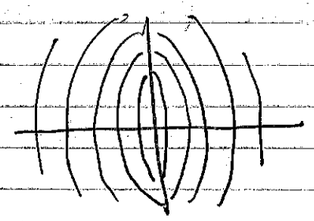
$\hookrightarrow T_{mom} = \mathcal{O}\left(\alpha \sqrt{K} \log\left(\frac{1}{\varepsilon}\right)\right)$.

$f(\omega_1, \omega_2) = \frac{1}{2}\omega_1^2 + \frac{1}{2}\omega_2^2 \quad (K=1)$



$C = \omega_1^2 + \omega_2^2$

$f(\omega_1, \omega_2) = \frac{10}{2}\omega_1^2 + \frac{1}{2}\omega_2^2$



$C = a\omega_1^2 + b\omega_2^2$

"poorly-conditioned" prob.

$w = Pu$

$\min_w f(w) = \min_u f(Pu)$

ALA. $|P| \neq 0$

↙ invertible

$\operatorname{argmin}_w f(w) = P\left(\operatorname{argmin}_u f(Pu)\right)$ ← any matrix, invertible

let $g(u) = f(Pu)$.

graphically, this is essentially "stretching" the poorly-conditioned problem, i.e. apply linear operation

⇓

map back ← reduced K

⇓

solve for optimal

preconditioning.

instead of saying minimize $f(w)$ over w — solve minimize $f(Pu)$ over u (for clever P).

e.g. $f(w) = \frac{1}{2} w^T A w$. $A^T = A$.

- how to set P?.. what cond. can we get?.. (PD, $A > 0$, all eigenval. A are positive)

$P^T A P = I$ ← symmetric.

$2f(Pu) = (Pu)^T A(Pu) = u^T P^T A P u$

$= u^T u = \|u\|^2$

suppose P symmetric, invertible

- $PAP = I$

- $P^{-1}PAPP^{-1} = P^{-1} I P^{-1}$ ← replace eigenval.

$A = P^{-2} = P = A^{-\frac{1}{2}}$

in transformed space.

$$g(w) = f(Pw) \rightarrow \nabla g(w) = P^T \nabla f(Pw)$$

$$u_{t+1} = u_t - \alpha \nabla g(u_t) = u_t - \alpha P^T \nabla f(Pu_t)$$

assume $w = Pu$; $w_t = Pu_t$

$$Pu_{t+1} = Pu_t - \alpha P P^T \nabla f(Pu_t)$$

$$w_{t+1} = w_t - \alpha P P^T \nabla f(w_t)$$

R ← Preconditioner

$$w_{t+1} = w_t - \alpha R \nabla f(w_t)$$

positive, invertible, definite, symmetric.

Preconditioned GD

$R = A^{-1}$

$$x^T R x = x^T P P^T x$$
$$= (P^T x)^T (P^T x)$$
$$= \|P^T x\|^2 > 0$$

chose R to be diagonal preconditioner

R diagonal ⟶ stretch along the coordinates' axis

memory $\mathcal{O}(d)$.

time multiply $\mathcal{O}(d)$.

$$(w_{t+1})_i = (w_t)_i - \alpha R_{ii} (\nabla f(w_t))_i$$

Let $\alpha_i = \alpha R_{ii}$.

$$(w_{t+1})_i = (w_t)_i - \alpha_i (\nabla f(w_t))_i$$

$$w_{t+1} = w_t - \alpha \odot \nabla f(w_t)$$

element-wise

- run GD in implicit scaled space.

AdaGrad

loop

$g \leftarrow \nabla f_i(w)$, random i

$r_i \leftarrow r_i + g_i^2$

$\left((r_i)_t = \sum_{k=0}^{t} (g_i^2)_t\right)$

$w_i \leftarrow w_i \oplus - \frac{\alpha}{\sqrt{r_i}} g_i \quad (\alpha_i = \frac{\alpha}{\sqrt{r_i}})$

- AdaGrad.

↳ "Adaptive Optimization" Alg.

Lecture 12. Week 7.

AdaGrad.

init $w \in \mathbb{R}^d$, $r = 0 \in \mathbb{R}^d$.

loop:

get example grad $g \in \mathbb{R}^d$

$g = \nabla f(w; x)$ minibatch etc

↑ large example

update $r \leftarrow r + g^2$

update $w \leftarrow w - \frac{\alpha}{\sqrt{r} + \varepsilon} \cdot g$

↳ small number.

- done in element-wise

implicitly.

depends too much on the history

infinite memory. ⇐

$r_i \approx \mathbb{E}[g_i^2]$

mul. num. of steps.

↳ $\approx \tau(\mathbb{E}(g_i)^2 + Var(g_i))$

could be dominant

fix this: RMSProp.

RMSProp

hyperparameters $\alpha > 0$, $\rho \in (0, 1)$, ε

init $w \in \mathbb{R}^d$, $r = 0 \in \mathbb{R}^d$

loop:

get gradient sample g at w.

update $r \leftarrow \rho r + (1-\rho) g^2$

update $w \leftarrow w - \dfrac{\alpha}{\sqrt{r} + \varepsilon} g$

$r_i \approx \mathbb{E}[g_i^2]$

typical $\rho = 0.999$

Combining momentum + RMSProp + correction for 0-init

init $w \in \mathbb{R}^d$, $r = 0 \in \mathbb{R}^d$

$s = 0 \in \mathbb{R}^d$

momentum:

$$w_{t+1} = w_t - \alpha g_t + \beta (w_t - w_{t-1})$$

$$(w_{t+1} - w_t) = \beta (w_t - w_{t-1}) - \alpha g_t$$

$$\begin{cases} v_{t+1} = \beta v_t - \alpha g_t \to \beta v_t - (1-\beta)\left[\frac{\alpha}{(1-\beta)} g_t\right] \\ w_{t+1} = w_t + v_t \end{cases}$$

α, β_1, β_2 hyperparam.

loop:

get gradient sample g @ w.

$$s \leftarrow \beta_1 s + (1-\beta_1) g$$

$$r \leftarrow \beta_2 r + (1-\beta_2) g^2$$

update:

$$w \leftarrow w - \frac{\alpha}{\sqrt{\hat{r}} + \varepsilon} \hat{s}$$

$$\hat{s} \leftarrow \frac{s}{1-\beta_1^t}$$

t = # of steps

$$\hat{r} \leftarrow \frac{r}{1-\beta_2^t}$$

if g_t denotes the g drawn at step t, $s_t = \sum_{k=0}^{t-1} \beta_1^k (1-\beta_1) g_{t-k}$

s after t steps

$$\sum_{k=0}^{t-1} \beta_1^k (1-\beta_1)$$

$$= (1-\beta_1)\left(\frac{1-\beta_1^t}{1-\beta_1}\right)$$

SI: exponential MA:

sequence $x_0, x_1, x_2, \ldots$ (recurrence)

$s_0 = 0$; $s_{t+1} = \beta s_t + (1-\beta) x_t$.

Evolving - parameters:

Subsampling - training number n.

momentum - k.

variance of the gradient

↳ avoid having

Variance Reduction

→ SVRG

↳ Strive to behave GD

- yet SGD each steps.

↳ Large-Scale convex optimization

⇒ great !!

for DL - Not so good.

GD SVRG.

$O(nk\log(\frac{1}{\varepsilon}))$ $O((n+k)\log(\frac{1}{\varepsilon}))$

Polyak Averaging

- Output an average:

if W_t is parameters vector after t steps

↳ $\frac{1}{T}\sum_{t=0}^{T-1} W_t$

Stochastic Weight averaging (SWA).

(wed.)

Lecture 15/14. Week 7.

Rev...

$\mathcal{O}(n/\kappa \log(\frac{1}{\varepsilon}))$ - GD.

$\mathcal{O}(\kappa/\varepsilon^2)$ - SGD

Momentum: $\mathcal{O}(n\sqrt{\kappa}\log(\frac{1}{\varepsilon}))$ - MSGD.

$\mathcal{O}((n+\kappa)\log(\frac{1}{\varepsilon}))$ SVRG.

- Polyak averaging / Stochastic weight averaging

$\mathcal{O}(n\kappa \cdot d \cdot \log(\frac{1}{\varepsilon})) \Leftrightarrow \mathcal{O}(\kappa\sigma^2 d/\varepsilon^2)$

$\mathcal{O}(n\sqrt{\kappa} d \log(\frac{1}{\varepsilon}))$

linear model: $w \in \mathbb{R}^d$.

$f(w) = \frac{1}{n}\sum_{x,y\in D} \mathcal{L}(w^T x, y)$.

time to compute an example grad

$\mathcal{O}(d)$

$\nabla^2 f(x)$ → covariance matrix. $(d\times d)$.

computationally expensive

Random Projection. (Johnson-Lindenstrauss

$x \in \mathbb{R}^D \Rightarrow Ax \in \mathbb{R}^d$ (transform)

feature length

$A \in \mathbb{R}^{d\times D}$ random.

$A_{ij} \sim \mathcal{N}(0, \sigma^2)$.

$\|Ax - Ay\|^2 \leq$

$(1-\varepsilon)\|x_i - x_j\|^2 \leq \|Ax_i - Ax_j\|^2 \leq (1+\varepsilon)\|x_i - x_j\|^2$

$\forall x_i, x_j \in D$

$d \geq \frac{8\log(n)}{\varepsilon^2}$.

Why? choose σ^2 s.t.

$\mathbb{E}\left[\|Ax_i - Ax_j\|^2\right] = \|x_i - x_j\|^2$.

$\mathbb{E}[\|A(x_i - x_j)\|^2] = \mathbb{E}[(x_i - x_j)^T A^T A (x_i - x_j)]$

$= (x_i - x_j)^T \mathbb{E}[A^T A] (x_i - x_j)$

↳ Gaussian matrix.

$\mathbb{E}[A^T A]_{ij} = \sum_{k=1}^{d} \mathbb{E}(A_{ki} A_{kj})$

$= \begin{cases} 0 & \text{if } i \neq j \\ \sum_{k=1}^{d} \mathbb{E}[A_{ki}^2] & \text{if } i = j \end{cases}$

σ^2

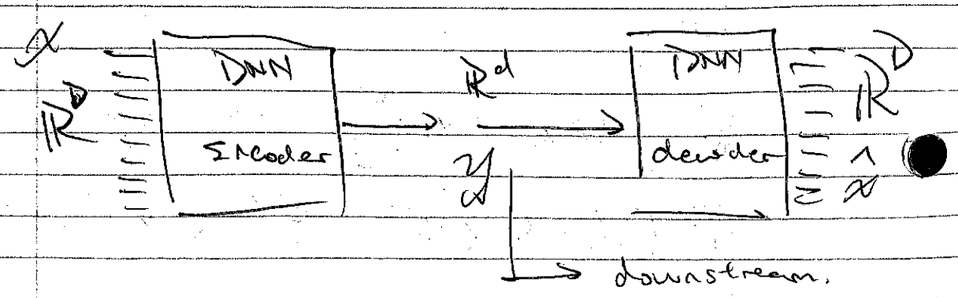
$\mathbb{E}[A^T A] = \sigma^2 d I$

$= (x_i - x_j)^T (\sigma^2 d I)(x_i - x_j)$

$= \sigma^2 d \|x_i - x_j\|^2 \overset{\text{if}}{=} \|x_i - x_j\|^2$

↳ $\sigma^2 = \frac{1}{d}$.

AutoEncoders



$\ell(x, \hat{x})$

e.g. $\frac{1}{|D|} \sum_{x \in D} \|x - \hat{x}\|_2^2$

↑ unsupervised learning.

Sparsity

density of x = $\frac{\#\text{ of nonzero entries } x}{\text{size of } x \cdot d}$

$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 2 \\ 0 \\ 0 \\ 3 \end{bmatrix} = \{(4, 2.0), (7, 3.0)\}$ = indices = [4, 7]

values = [2.0, 3.0]

16 ish

Sparse matrix.

COO

↳ expensive to comp.
Easy to write

$\begin{bmatrix} 0 & .5 & 0 & 0.3 \\ 0 & 1.0 & 0 & 0.0 \end{bmatrix} \mapsto$

row idx. [1, 1, 2]

col. idx. [3, 6, 2]

values [5, 3, 1]

CSR — vs. — CSC

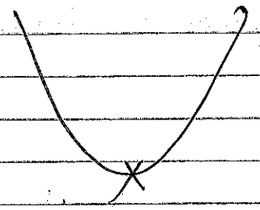
row offsets. [0, 2] [3, 6] [2] ⇒ [3, 6, 2]

col idxs. [5.0, 3.0] [1.0] [5.0, 3.0, 1.2]

values.

Week 8. Lecture 15/16.

Deep learning

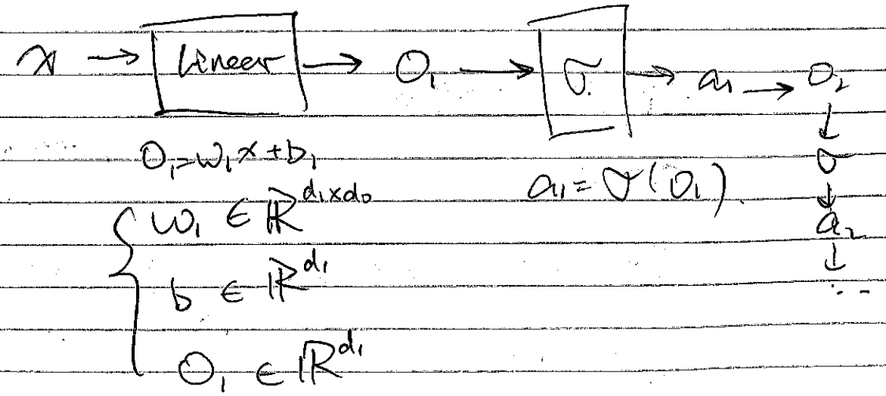


convex opt.

Neural Networks

- MLP

$x \in \mathbb{R}^{d_0}$



Nonlinear: function acts element-wise as the activation function.

$\sigma(x) = ReLU(x) = \max(x, 0)$.

universal: for continuous input → output. there exists a function that approx. with bounded parameters (good recovery).

capacity $\xrightarrow{\text{imply}}$ overfit.

NN tend not to overfit!

→ larger models generalize better.

▷ double-descent.

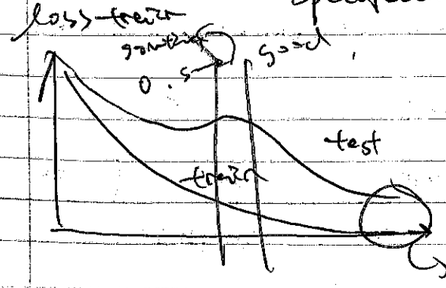
$(x, y) \in \mathbb{R}^d \times \mathbb{R}$.

$|\mathcal{D}| = n$. $n = d$.

$\mathcal{L}(w) = \frac{1}{n} \sum_{i=1}^{n} (x_i^T w - y_i)^2$ + regularization

larger model generalize better

Specifically for SGD, Adam. or sth similar.



"double-descent" phenomenon.

$a_\ell, o_\ell \in \mathbb{R}^{d_\ell}$

$a_\ell = W_\ell \cdot o_{\ell-1} + b_\ell \rightarrow \mathcal{O}(\;)$

$\mathbb{R}^{d_\ell} \quad [\mathbb{R}^{d_{\ell-1}}] \in \mathbb{R}^{d_\ell}$

$o_\ell = \sigma_\ell(a_\ell) \rightarrow \mathcal{O}(\;)$

$d_1(d_0 + d_2 - 1) + d_1$

$3 d_1 \ell.$

$W_1 \in \mathbb{R}^{d_1 \times d_0}$

$b_1 \in \mathbb{R}^{d_1}$

$x \in \mathbb{R}^{d_0}$

Image processing (- CV.

$x \in \mathbb{R}^{C \times W \times H}$ → height of image

→ width of image

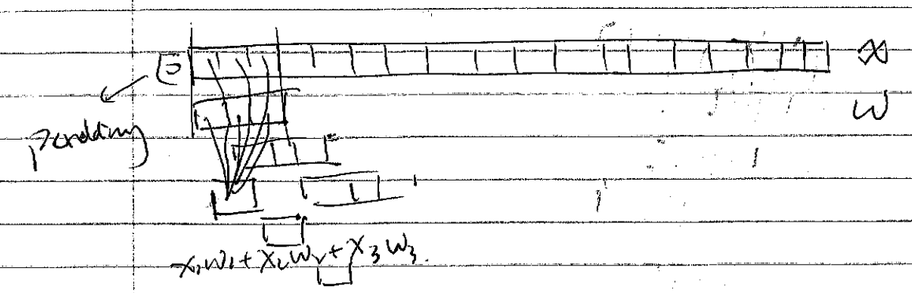
↑ channels.

1) convolutional neural network

- restricted subset

convolutional layers. in place of the linear layers.

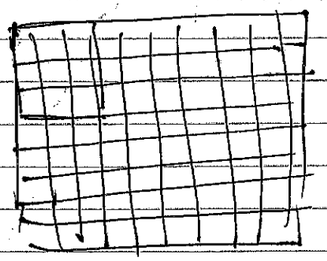
convolution in 1D:



filter: arbitrarily set of parameters.

$x_{k+1} w_1 + x_{k+2} w_2 + x_{k+3} w_3$

in 2D



Conv. layer

diff. channels

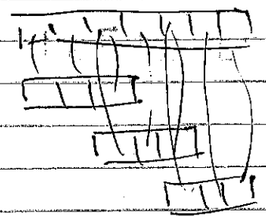
input $x \in \mathbb{R}^{C_0 \times W_0 \times H_0}$ output. $a \in \mathbb{R}^{C_1 \times W_1 \times H_1}$

parameters $W \in \mathbb{R}^{C_1 \times C_0 \times k_W \times k_H}$

$a_i = \sum_{j=1}^{C_0} \text{conv.}(W_{ij}, x_j)$ (W_{ij}: matrix) $W \in \mathbb{R}^{C_1 \times C_0 \times k_W \times k_H}$ ↑ ↑ ↑ ↑ i-th image

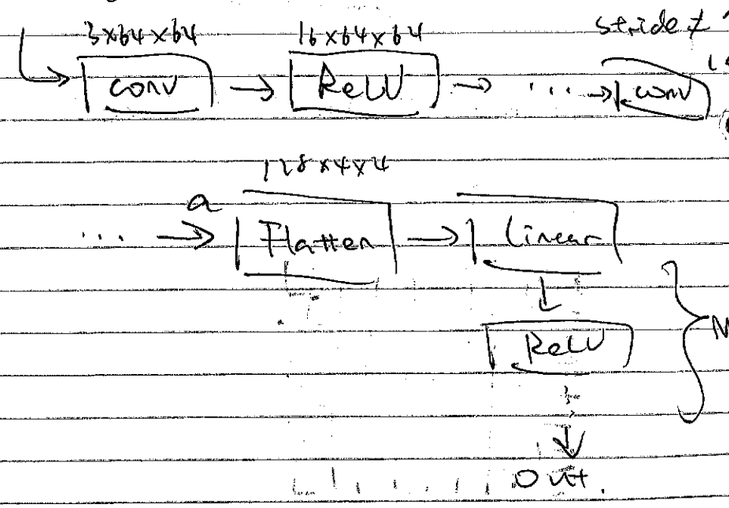
$b \in \mathbb{R}^{C_1}$

• stride.



output: 3

input image:



Week 9. Lecture 17.

Overfitting / Underfitting ← low capacity

↖ model has high capacity.

(overparametrized).

↓

DNNs high ← representational

many weights.

weights coming ← effective.

from the learning algorithm.

Dropout.

↳ increase the representation capacity.

During training → randomly drop

(independently each step) neurons.

↓

forces the learn to employ

more neurons.

Week 9 Lecture 18

Batch Norm.

Let $u \in \mathbb{R}^n$.

$$u \to \frac{u - \mu}{\sqrt{\sigma^2}} = \frac{u - \mu}{\sigma}$$

↳ Instead, look at $u \in \mathbb{R}^B$

— conv. ⇒ image ⇒ per channel

— saves a running [exp.] average,

of the mean & variance

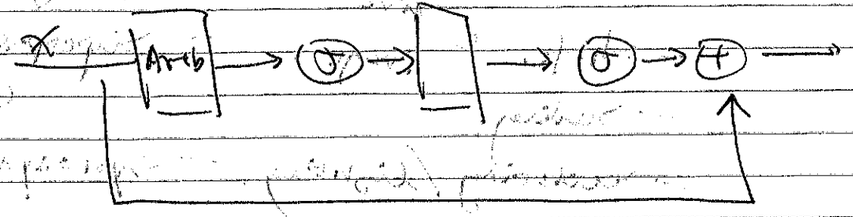
μ , σ.

Model.train() ⇒ turning on/off batch norm.
Model.eval()

$$u \to \frac{u - \mu}{\sigma} \cdot \gamma + \beta$$

broadcast. scalar learned (SGD)

Residual / connections / skip connections



⇒ ResNet.

⇒ DenseNet.

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

$(x_i, y_i) \in \mathcal{D} \subseteq \mathcal{X} \times \mathcal{Y}$

▷ sparse data (prob. ρ)

n too small.

☆ Data Augmentation.

randomness.

Aug: function $\mathcal{A} \cdot \mathcal{X} \xrightarrow{r \in [0,1]} \mathcal{X}$. ← random noise

$$w_{t+1} = w_t - \alpha \nabla \ell(w_t; \mathcal{A}(x_t; \eta_t), y_t)$$

example chosen @ t.

IMAGES:

- Translation
- Rotation
- Shift
- Scaling
- recoloring / lighting

make image

- Noise / Anti fact Addition
- Cropping
- Superposition (Mixup)
- synonym swapping
- deletion
- back translations

Semi-supervised learning:

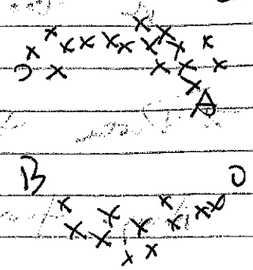
"data-cheap" "label-expens."

lots of data, few labels

assump.

Similar inputs have similar outputs.

↳ (kNN)



clustering → data is clustered

labels by cluster.

Manifold → data lie on low-dimensional manifold

PCA Auto encoders

Week 10. Lecture 19.

Sequence models.

$x \in \mathbb{R}^{d \times n}$ ← different per example.

discussion:

- takes sequence as input.

↳ generate sequence output.

○ handle sequence input/output?

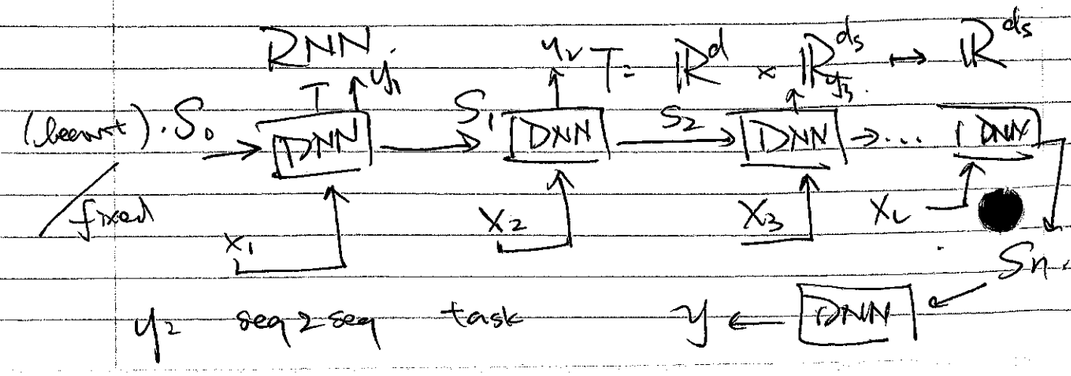
- padding → max length.

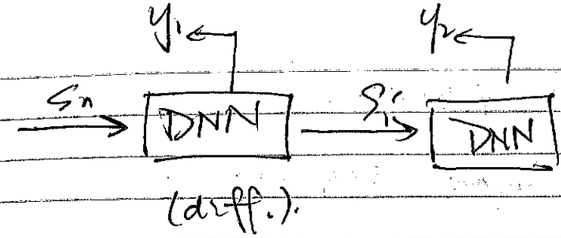
- counts / sum

- RNN ≈ sequential

- Transformers. ≈ parallel

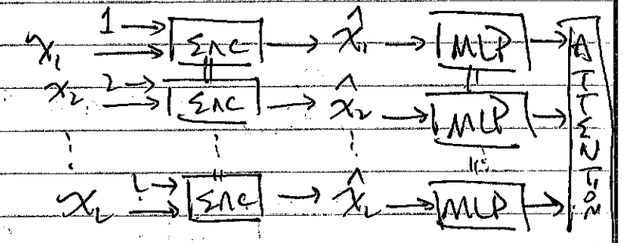
DfA: $S_{next} = T(S_{current}, X_i)$. $\times \mathbb{R}^c \to \mathbb{R}^{d_s}$





- LSTM.

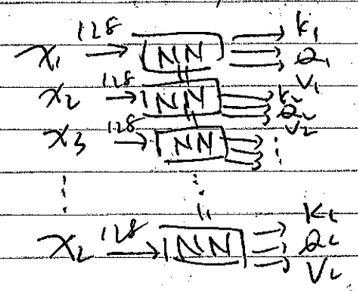
- Transformers.



(input sequence

$x_i = Enc(x_i, \psi)$.

Mathematically attention layer:



$k_i \in \mathbb{R}^{d_k}$

$V_i \in \mathbb{R}^{d_v}$.

$Q_i \in \mathbb{R}^{d_k}$.

$Q, k \in \mathbb{R}^{n \times d_k}$.

$V \in \mathbb{R}^{n \times d_v}$.

$\text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \cdot V$

approximate table lookup:

↓

Attention block.

Multihead Attention

Same input sequence.

↓

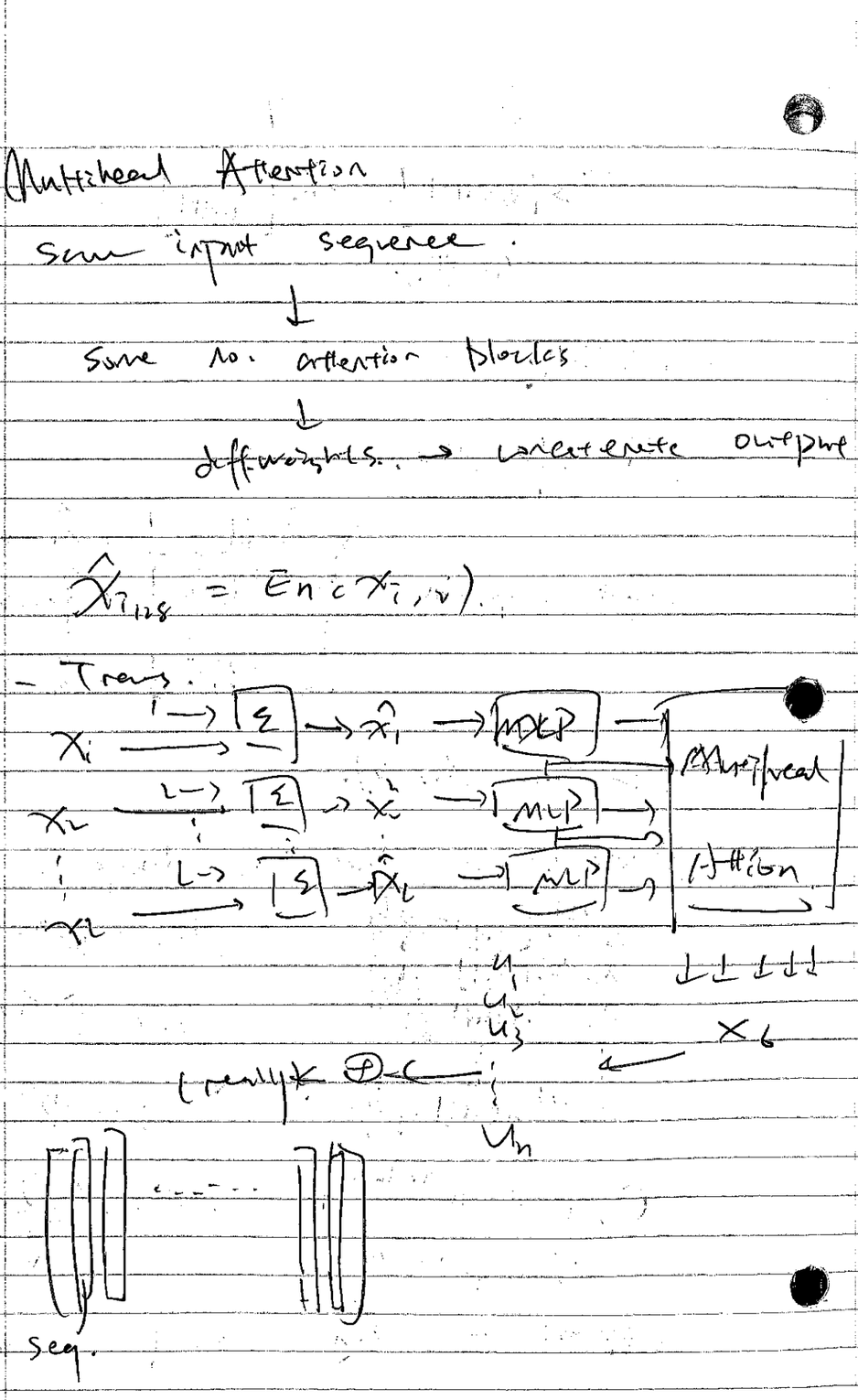
Same no. attention blocks

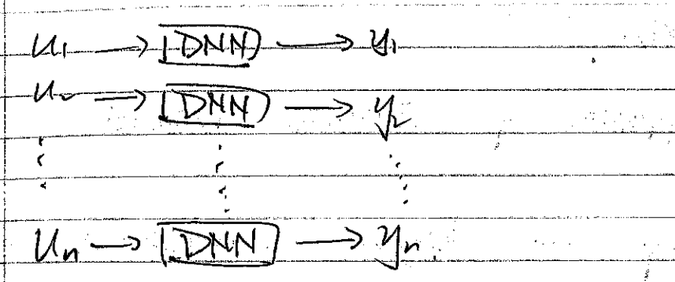
↓

diff weights → concatenate output

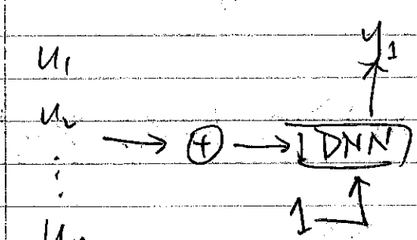
$\hat{X}_{i_{128}} = Enc(X_{i,v})$

- Trans.





instead, we do



Lecture 20.

Kernels.

opposite of dimension reduction.

map into higher dimensional space.

$$\phi(x) = \mathbb{R}^d \mapsto \mathbb{V}$$

$$\langle \phi(x_1), \phi(x_2) \rangle = k(x_1, x_2).$$

$$k: \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$$

Examples

- Gaussian Kernels

$$k(x_1, x_2) = \exp(-\gamma \|x_1 - x_2\|^2).$$

a.k.a. RBF kernel

- linear kernel

$$k(x_1, x_2) = x_1^T x_2.$$

- exponential kernel (laplacian)

$$k(x_1, x_2) = \exp(-\gamma \|x_1 - x_2\|)$$

- Polynomial kernels

$$k(x_1, x_2) = (1 + x_1^T x_2)^p.$$

Kernel definition.

$$k: \mathcal{X} \times \mathcal{X} \to \mathbb{R}.$$

$$k(u, v) = k(v, u)$$

→ kernel must be symmetric.

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j k(x_i, x_j) \geq 0.$$

$$\| \sum_{i=1}^{n} c_i \phi(x_i) \|^2 \geq 0$$

if $K_{ij} = k(x_i, x_j) \Rightarrow K$ is positive semidefinite

k is a kernel implies:

$\exists \mathbb{V}$ vector space & ϕ s.t.

$\langle \phi(x_1), \phi(x_2) \rangle = k(x_1, x_2)$

if k_1 & k_2 are kernels. w/ feature maps ϕ_1 & ϕ_2, then $k = k_1 + k_2$

$k(u, v) = k_1(u, v) + k_2(u, v)$.

$\phi(x) = \begin{bmatrix} \phi_1(x) \\ \phi_2(x) \end{bmatrix}$ ← is a kernel.

$k(u,v) = c\,k_1(u,v)$ if $c \geq 0$.

$\phi(x) = \sqrt{c}\,\phi_1(x)$.

$k(u,v) = k_1(u,v)\,k_2(u,v)$.

$\phi(x) = \phi_1(x) \otimes \phi_2(x)$.

$k(u,v) = f_1(u)\,k_1(u,v)\,f_2(v)$.

$\phi(x) = f(x)\,\phi(x)$.

$f : x \to \mathbb{R}$.

component loss ℓ: $f(w, x, y) = \ell(w^T \phi(x), y)$.

$= \ell(\langle w, \phi(x) \rangle, y)$.

$f(w) = \frac{1}{n} \sum_{i=1}^{n} f(w, x_i, y_i)$.

$= \frac{1}{n} \sum_{i=1}^{n} \ell(\langle w, \phi(x_i) \rangle, y_i)$.

$\nabla_w f(w, x, y) = \ell'(w^T \phi(x), y)\,\phi(x)$.

$\Rightarrow$ for GD / SGD.

$w_k \in \text{span}\{\phi(x_1), \phi(x_2), \ldots \phi(x_n)\}$

$w_k = \sum_{i=1}^{n} \alpha_i \phi(x_i)$

$w_k = \sum_{i=1}^{n} a_i \phi(x_i)$.

$F(a, x, y) = \ell(\langle \sum_{i=1}^{n} a_i \phi(x_i), \phi(x) \rangle, y)$.

$= \ell(\sum_{i=1}^{n} a_i \langle \phi(x_i), \phi(x) \rangle, y)$

$= \ell(\sum_{i=1}^{n} a_i k(x_i, x), y)$

$f(a) = \frac{1}{n} \sum_{j=1}^{n} \ell(\sum_{i=1}^{n} a_i k(x_i, x_j), y)$.

let $K_{ij} = k(x_i, x_j)$

$= \frac{1}{n} \cdot \sum_{j=1}^{n} \ell((Ka)_j, y)$.

① compute $\phi(x_i)$ on-the-fly as needed.

② precompute & store $\phi(x_i)$ before training. ✓ train in the transformed space

③ compute k on-the-fly as needed

④ precompute K & store.

Lecture 21.

kernels function $k: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$.

ways to do learning:

(1) Learn in the transformed space, compute ϕ on-the-fly.

(2) learn " " precompute ϕ.

(3) learn in the "kernel" space while computing k on-the-fly.

(4) learn in "kernel trick" space, precompute $K_{ij} = k(x_i, x_j)$

$f_i(w) = \ell(w^T \phi(x_i), y_i)$

$\nabla f_i(w) = \ell'(w^T \phi(x_i), y_i)\, \phi(x_i)$

Suppose n examples, $x_i \in \mathbb{R}^d$, and k takes $\Theta(d)$ to compute.

$\phi(x_i) \in \mathbb{R}^m$.

run for T steps.

cost to compute $\phi \Rightarrow \Theta(md)$.

— Vanilla non-kernel linear model: $\mathbb{R}^d$.

	Compute	Memory
(1)	(H) (Td)	(H) (d) (not inc. train. set)
(2)		
(3)		
(4)		

— learn in the transformed space

	Compute	Memory
(1)	(H) (Tmd). precompute (H) (nm).	"kernel"
(2)	(H) $(Tm + nmd)$.	(H) (nm). (training d.)
(3)	(H) (Tnd).	(H) (n).
(4)	(H) $(Tn + n^2 d)$	(H) (n^2).

$\nabla f_i(w) = \ell'(\sum_{j=1}^{n} u_j k(x_i, x_j), y_i)\, \phi(x_i)$

$\nabla f_i(w) = \ell'(\phi(x_i)^T \sum_{j=1}^{n} u_j \phi(x_j), y_i)\, \phi(x_i)$

an SGD step ⇒ compute $\hat{}$

$$u_i \leftarrow u_i - \alpha \ell'\left(\sum_{j=1}^{n} u_j k(x_i, x_j); y\right)$$

For kernels - subsampling.

$$k(x_i, x_j) \approx \langle \psi(x_i), \psi(x_j) \rangle$$

$$\psi(x) \in \mathbb{R}^D.$$

Approximate feature map.

↳ Random Fourier features.

if kernel : $k(x_i, x_j) = k(x_i - x_j)$.

then $k(x_i, x_j) = \mathbb{E}\left[2\cos(w^T x_i + b), \cos(w^T x_j + b)\right]$.

$b \sim \text{Unif}[0, 2\pi]$

$w \sim \mathcal{F}(k)$ Fourier transform of k.

("Rep" k)

RBF kernel:

$$w \sim \mathcal{N}(0, 2\gamma I),$$

$$b \sim \text{Uniform}([0, 2\pi]).$$

$$\mathbb{E}\left[2\cos(w^T x_1 + b), \cos(w^T x_2 + b)\right]$$

$$= \exp\left(-\gamma \|x_1 - x_2\|^2\right).$$

draw $w_1, w_2, \ldots, w_D \sim \mathcal{N}(0, 2\gamma I)$

$b_1, b_2, \ldots, b_D \sim \text{Unif}([0, 2\pi])$

Set $\psi_i(x) = \frac{1}{\sqrt{D}} \sqrt{2} \cos(w_i^T x + b_i)$

$$\Rightarrow \mathbb{E}\left[\langle \psi(x_i), \psi(x_j) \rangle\right] = \exp\left(-\gamma \|x_i - x_j\|^2\right)$$

; (TBC)

⑤ Learn w/ approx. feature map comp. ψ on-the-fly.

⑥. Learn w/ approx. f. m. compute & cache

Vanilla linear model.

compute. Memory.

(E) (H)(TDd). (H)(D)

(b) (H)(TD) + nDd) (H)(nD).

large D - high "accuracy" in repr. R.

- high compute & memory cost.

Lecture 22.

▷ Hyperparameter optimization

- Step size / lr / α → diverge.

$\alpha \uparrow$ — too large noise ball.

$\alpha \downarrow$ — get "stuck" in non-convex.

- λ for regularization

↓

over/under regularize.

converge too slowly — wrong level of expressivity.

poor generalization.

- β momentum. → if $\beta \notin (0, 1)$.

↓

bad.

- B batch size. → too small can affect noise ball.

too large → much like GD.

performance matching parallel capabilities of HW.

- choice of architecture : DNN

↓

Generalize poorly.

Systems implications.

- Non-DNN -

- tree depth → weak learner

overfit & lots of mem.

- feature vector length for

Random features.

→ poor fidelity,

high compute/memory

cost.

- # of epochs → high loss/error

compute cost.

- dimension → poor accuracy.

high compute cost

- k in kNN & k-means → under/overfit

more compute for larger k.

- the tasks for assigning hp:

→ HPO - Any process assign hps.

Standard: β 0.9 0.99

(β_1, β_2) (0.9, 0.999)

B power of 2, e.g. 256.

- Grid Search

↳ CoD

- Random Search.

less reproducible than GS.

Parallelism.

Early stopping

- droping bad hyperparameters

that perform poor each epoch.

goal: minimize $F(\alpha, \beta, B, k, \lambda)$.

"derivative-free" optimization. (DFO)

Encode our knowledge abt the space & the problem $\rightarrow$ HPO

Bayesian Opt — continued.

Mean / Median $f_0\ f_1\ f_2\ f_3\ f_4\ \ldots$

HPO

$P(f \mid f(x_1)=y_1,\ f(x_2)=y_2)$.

look at: $P(f(x_*) \mid f(x_1)=y_1,\ f(x_2)=y_2, \ldots)$

$f \in \mathbb{R}^d$, $f \sim \mathcal{N}(0, \Sigma_1)$

↓ ↓

mean covariance

↓

{ Set by ~~the~~ according to intuition.

♣ Fact: if (X, Y) are jointly (multivariate) gaussian.

the $X \mid Y = y$ is also Gaussian.

♣ Fact: "$\sim$", then X is Gaussian.

$f(x_*) \mid f(x_1)=y_1,\ f(x_2)=y_2, \ldots$

$\sim \mathcal{N}(\mu, \sigma^2)$

How to pick the next point?

$$x_{next} = \arg\min_{x^*} a(\mu(x^*), \sigma^2(x^*))$$

$a(\mu, \sigma) = \mu - k\sigma$.

(lower confidence bound)

a = acquisition function.

$a(\mu, \sigma) = P(f(x^*) \le f_{best})$.

$\hookrightarrow$ P ~ ...

(Probability of Improvement).

$f(x^*) \sim N(\mu, \sigma^2)$

$= \Phi\left(\frac{f_{best} - \mu}{\sigma}\right)$

Φ

accumulated distribution function.

$\hookrightarrow \Phi(u) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{u} e^{-\frac{z^2}{2}} dz$

$P(f(x^*) \le f_{best} | obs) = P(Z \le f_{best})$

for $Z \sim N(\mu, \sigma^2)$

$= P(\sigma u + \mu \le f_{best})$

for $u \sim N(0, 1)$

$= P(u \le \frac{f_{best} - \mu}{\sigma}) = \int_{-\infty}^{(f_{best}-\mu)/\sigma} P(u)\,du$

$a(\mu, \sigma) = \mathbb{E}[\underbrace{f(x_*) - f_{best}}_{\min(f(x_*), f_{best}) | obs}]$.

Expected Improvement.

by convention:

$= \mathbb{E}[\min(f(x_*) - f_{best}, 0) | obs]$

Thm 3: (Strong max/min principle for heat eqn.)

Assume u is C_C^2 in space & C^1 in time, all within U_T & it is continuous on $\bar{U}_T$, & it satisfies the heat eqn. in U_T.

Then (1) $\max_{\bar{U}_T} u = \max_{\partial U_T} u$.

(2) if Ω is connected, & there exists $\exists (x_0, t_0) \in U_T$, s.t. $u(x_0 t_0) = \max_{\bar{U}_T} u$

then u is constant in $\bar{U}_{t_0}$.

If u attains a max/min @ an interior pt. within U_T, then u must have been constant for all earlier times.

$\Rightarrow$ If u satisfies $u_t - \Delta u = 0$. in U_T.

B.C. $u = 0$, on $\partial\Omega \times [0, T)$.

I.C. $u = g$. on $\Omega \times \{0\}$.

where $g \geq 0$, then $u(x,t) > 0$. if $g(x) > 0$. for some $x \in \Omega$:

another example of infinite propagation speed of info. within ht. eqn.

Thm 5. (uniqueness in Poisson's Eqn.)

If $g \in C^1(\partial\Omega)$ & $f \in C^1(\Omega)$, then $\exists!$ soln $u \in C^2(\Omega) \cap C^1(\bar{\Omega})$.

to

$$-\Delta u = f \quad \text{in } \Omega$$

$$u = g \quad \text{on } \partial\Omega$$

Thm 6. (Regularity in Laplaces Eqn)

If $u \in C^2(\Omega)$ & also satisfies $\fint_{\partial B_r(x)} u\,ds = \int_{B_r(x)} u\,dy$ for any ball $B_r(x) \subset \Omega$ then $u \in C^\infty(\Omega)$

↑ guarantees nth about what happens on $\partial\Omega$

Thm 7 (Harnacks inequality)

Let $V \subset\subset \Omega$ be compact, connected set, &

$\forall v \subset V$, $\exists$ a positive constant, c, for each such v. where $\sup_v u \leq c \cdot \inf_v u$

holds for all non-negative harmonic functions v, defined on Ω.

$\Rightarrow$ The values of a non-negative harmonic function, within an arbitrary V, can't be that different from one another: Laplace's equation smooths things out away from the boundary $\partial\Omega$.

The point of all these thms. is to illustrate that, once we have a mathematical model, we can analyze it to understand its properties, & the properties of its solution ... and potentially this transfers back to the real world & its properties.

Also, we need to understand the properties of our continuous solutions before we can properly discretize things for the computer.

$\not\Rightarrow$ good for code debugging.

Let's go back to Poisson eqn.

Lecture 21

Let's recall Poisson eqn. $\Delta u(x) = f(x)$ & use it to come up with a general viewpoint for thinking about classical solutions to Linear PDE

- The Green's function (GF)

$$G(x, \tilde{x})$$

The whole idea comes down to finding a solution for a special instance of, eg. the Poisson eqn.:

$$\Delta G(x, \tilde{x}) = \delta(x, \tilde{x})$$

Dirac Delta

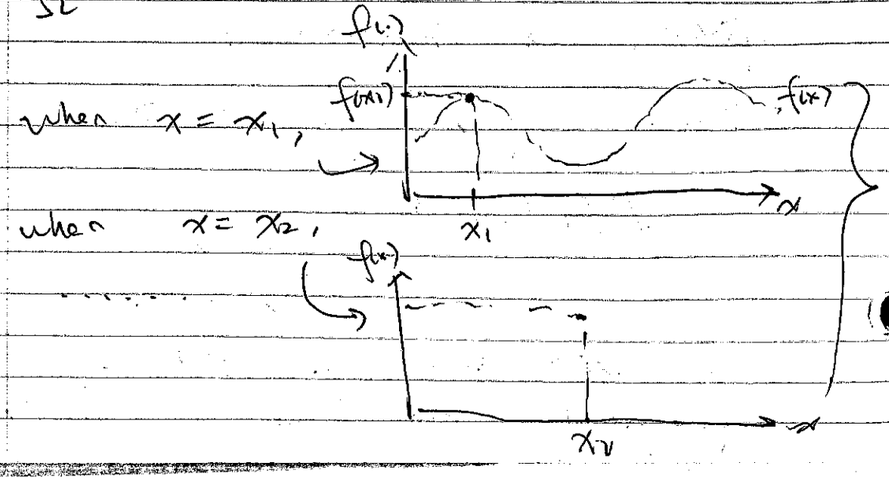
The Dirac Delta is kind of linear functional, known as a distribution.

This means δ likes to operate on some smooth function to produce a real number. The "operation" involves an integral.

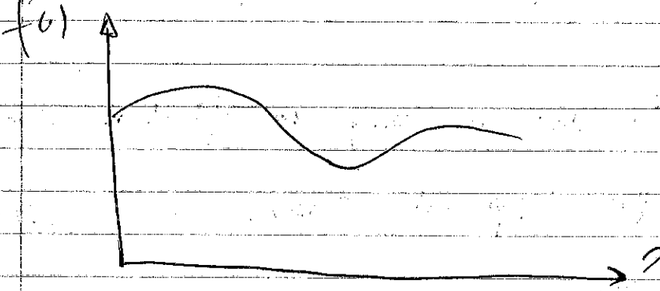
$$\delta(x) = \begin{cases} \infty & x = 0 \\ 0 & x \neq 0 \end{cases} \qquad \text{s.t.} \int_{-\infty}^{\infty} \delta(x)\,dx = 1$$

We can use these interesting properties of δ to "break up" our source term, $f(x)$, into "impulses" meaning

$$\int_{\Omega} \delta(x, \tilde{x}) f(\tilde{x})\, d\tilde{x} = f(x) \qquad (+)$$



"taking" an infinity of such pts, we get a smooth curve"



but we also know that if we substitute a Green's function in $\Delta(\cdot)$. we get the Dirac Delta: $\Delta(G(x,\tilde{x})) = \delta(x,\tilde{x})$ so we rewrite (+): $\int_{\Omega} \Delta(G(x,\tilde{x})) f(\tilde{x})\, d\tilde{x} = f(x)$

Integration is a linear differential operator Ω is fixed.

$$\Delta\left(\int_{\Omega} G(x,\tilde{x}) f(\tilde{x})\, d\tilde{x}\right) = f(x).$$

is a soln to PS, since it satisfies the PDE $\Delta(u(x)) = f(x)$

thus $u(x) = \int_\Omega G(x,\tilde{x}) f(\tilde{x})\, d\tilde{x}$

"Ω is fixed & f is nice".

G.f. not so easy to find ... available only for special cases. w/ simple geometries.

Eg. The Green's function for Poisson's eqn. on the unit disk w/ homogeneous Dirichlet B.C.s. i.e. $(u(x)=0, \forall x \in \partial\Omega)$. is

$$G(x,\tilde{x}) = \frac{1}{2\pi}\left(\ln|x-\tilde{x}| - \ln|x^*-\tilde{x}^*| - \ln|\tilde{x}|\right)$$

Heat Eqn. (Diffusion).

$$u_t - \Delta u = 0$$

if there is a source term within the domain, eq. becomes $u_t - \Delta u = f$.

where $t \geq 0$ & $x \in \Omega \subset \mathbb{R}^n$. The solution $u(x,t)$ is the function satisfying the PDE & I.C. & B.C. such that $u: \bar{\Omega} \times [0,\infty) \to \mathbb{R}$, i.e. our soln is a scalar field over $\bar{\Omega}$.

But where did this PDE come from?

Let's begin from the case with heat flowing: First we need some terminology.

Thermal Energy ≡ instantaneous molecular kinetic Energy.

Heat ≡ Thermal Energy in transit.

Temperature ≡ Average of thermal Energy

Physics Theory = Conservation of Energy

Rate of change in Thermal Energy = Heat Generated by sources within Ω + flow of heat into Ω from outside

taking arbitrary subd. $\Omega' \subset \Omega$

Note: Thermal Energy contained in a body is tied to "heat capacity", C.

$C \equiv$ amount of heat generated per unit mass, per rise in temp, μ.

$\rho \equiv$ mass density.

The total thermal energy with Ω" @ a given temp. t is:

$$\int_{\Omega} C(x)\,\rho(x)\,\underbrace{\mu(x,t)}_{\text{Temp. field}}\,dx$$

if $f(x,t)$ is our heat source, specified per unit volume, per unit time, then the total heat generated by the source within Ω' is: $\int_{\Omega'} f(x,t)\,dx$

Lastly, heat flux thru $\partial\Omega'$ is $q_\theta(x,t)$; thus the total heat flux into the body @ time "t", is given by:

$$\int_{\partial\Omega'} q_\theta \cdot \underbrace{\nu}_{\text{unit outward normal to } \partial\Omega'}\,dS.$$

Rate of change in thermal Energy within Ω = Heat generated in Ω + flow of heat into domain

considering arbitrary sub-domain Ω'

$$-\int_{\partial\Omega'} q_\theta \cdot \nu \cdot ds$$

$$\frac{d}{dt}\int_{\Omega'} C(x)\,\rho(x)\,\mu(x,t)\,dx \qquad \downarrow \text{divergence thm}$$

$$= \int_{\Omega'} f(x,t)\,dx - \int_{\Omega'} \nabla \cdot q_\theta\,dx.$$

$$\int_{\Omega'} \underbrace{\left[C(x)\,\rho(x)\,\frac{\partial \mu(x,t)}{\partial t} - f(x,t) + \nabla \cdot q_\theta\right]}_{=0} dx = 0$$

not the same.

$$\Rightarrow c\rho \frac{\partial \mu}{\partial t} + \nabla \cdot q = f \quad (\#)$$ we now need

a constitutive model (Fourier's law) relating

heat flux w/ gradients in temp.

$$q_q = -k \nabla \mu \quad (*)$$

(k: thermal conductivity)

$(*) \rightarrow (\#)$

$$\frac{\partial \mu}{\partial t} - \left(\frac{k}{c\rho}\right) \nabla \cdot \nabla \mu = f$$

$$\frac{\partial \mu}{\partial t} - \chi \nabla^2 \mu = f$$

(χ: thermal diffusivity)

(RHS=0)

The fundamental sol'n for homogeneous heat eqn.

$$\phi(x,t) := \begin{cases} \frac{1}{(4\pi t)^{n/2}} e^{-\frac{|x|^2}{4t}} & (x \in \mathbb{R}^n, t>0) \\ 0 & (x \in \mathbb{R}^n, t<0) \end{cases}$$

Note singularity @ pt. $(0,0)$.

$|x| = (x_1^2 + x_2^2 + \dots + x_n^2)^{1/2}$

(another radial sol'n)

sol'n is normalized s.t.

$$\int_{\mathbb{R}^n} \phi(x,t)\, dx = 1.$$

Let's consider IVP heat equation.

$$\mu_t - \Delta \mu = 0 \quad \text{in } \mathbb{R}^n \times (0,\infty)$$

$$\mu = g \quad \text{on } \mathbb{R}^n \times \{t=0\}.$$

We note that the fundamental sol'n, ϕ, solves the homogeneous heat eqn. away from the singularity @ $(0,0)$, as it does for any shift in spatial coordinate, as in $\phi(x-y,t)$ for each fixed $y \in \mathbb{R}^n$. As a result, the following convolution is a solution to our IVP.

Thm 1: (Initial value heat eqn.)

Assume $g \in C(\mathbb{R}^n) \cap L^\infty(\mathbb{R}^n)$, and define "$\mu$" as, $\mu(x,t) = \int_{\mathbb{R}^n} \phi(x-y,t)\, g(y)\, dy$

$$= \frac{1}{(4\pi t)^{n/2}} \int_{\mathbb{R}^n} e^{-\frac{|x-y|^2}{4t}} g(y)\, dy.$$

$(x \in \mathbb{R}^n,\ t > 0)$

then,

1) $u \in C^\infty(\mathbb{R}^n \times (0,\infty))$ ← soln ↑ smooth

2) $u_t(x,t) - \nabla^2 u(x,t) = 0, \quad x \in \mathbb{R}^n,\ t > 0.$

&

3) $\lim u(x,t) = g(x^0)$ for each $x^0 \in \mathbb{R}^n$.
$(x,t) \to (x^0, 0)$

Since g is bounded & continuous, if we also make it possitive semi definite, then $u(x,t)$ is postive for all Pts $x \in \mathbb{R}^n$ & times $t > 0$.

implies prop. speed for info. regarding I.C.

$u_t - \Delta u = f$ in $\mathbb{R}^n \times (0,\infty)$

$u = 0$ on $\mathbb{R}^n \times \{0\}$.

Thm 2: (forced heat eqn.)

Let $u(x,t)$ be of the form

$$u(x,t) = \int_0^t \int_{\mathbb{R}^n} \phi(x-y, t-s) f(y,s)\, dy\, ds.$$

$$= \int_0^t \underbrace{\frac{1}{(4\pi(t-s))^{n/2}} \int_{\mathbb{R}^n} e^{-\frac{|x-y|^2}{4(t-s)}} f(y,s)\, dy\, ds}_{\text{treating our forcing as I.C. @ "s"}}$$

then ① $u \in C_c^2(\mathbb{R}^n),\ C_c'([0,\infty))$.

② $u_t - \Delta u = f \quad (x \in \mathbb{R}^n,\ t > 0)$.

and

③ $\lim_{(x,t) \to (x^0, 0)}$

↳ $u(x,t) = 0$ for each $x^0 \in \mathbb{R}^n$

Combining Thms 1 & 2, we can solve

$$u_t - \Delta u = f \quad \text{in} \quad \mathbb{R}^n \times (0, \infty)$$

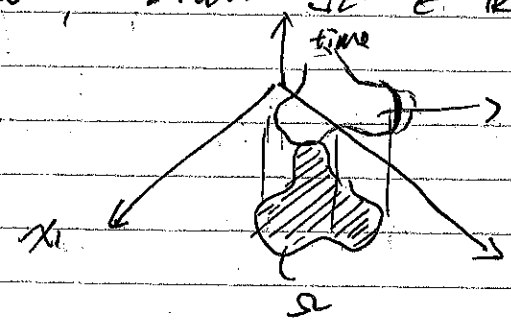
$$u = g \quad \text{on} \quad \mathbb{R}^n \times \{0\}.$$

with $$u(x,t) = \int_{\mathbb{R}^n} \phi(x-y, t)\, g(y)\, dy$$

$$+ \int_0^t \int_{\mathbb{R}^n} \phi(x-y, t-s)\, f(y,s)\, dy\, ds$$

moving away from all $\mathbb{R}^n$, now, let's define an open region of space serving as our spatial domain: $\Omega \in \subset \mathbb{R}^n$.

For now, draw $\Omega \in \mathbb{R}^2$ & time for 3').



$U_T := \Omega \times (0, T]$

↑ open set. | open | final time included

$\partial U_T := \bar{U}_T - U_T$.