Learning

PAC Learning

Classification

Vc Dim Vc Thm

Time complexity

Regression

Linear OLS

Regularization

Probabilistic Classification

Logistic regression

NN

architec

SGD

Generation

Vibe definition, indistinguishability

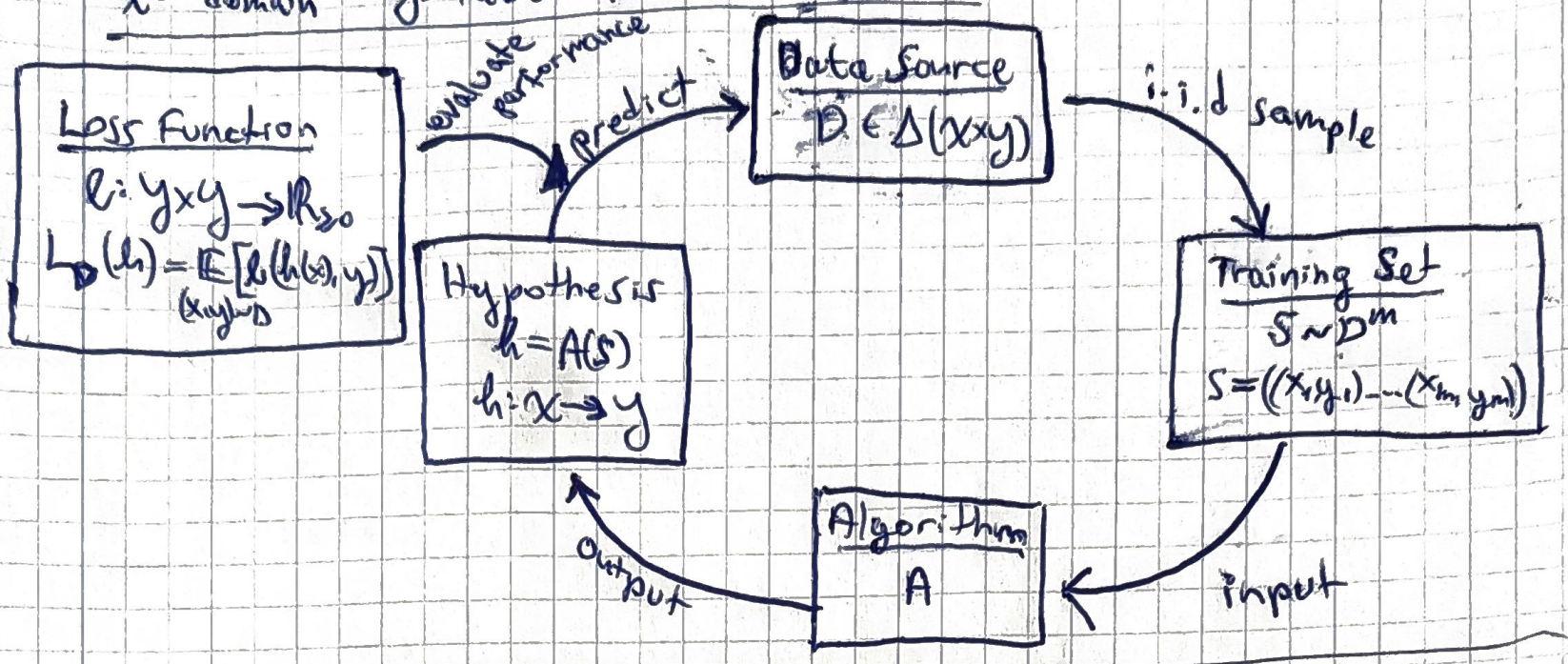
Next token prediction

RLHF

Diffusion

1. Learning

$\mathcal{X}$ – domain $\mathcal{Y}$ – label space



Instantiations

- (Binary) classification with '0-1' loss
- Regression with ℓ_2 loss
- Probabilistic Classification with CE loss

Example

Will treatment cure patient?

How big will baby be?

Want: low loss

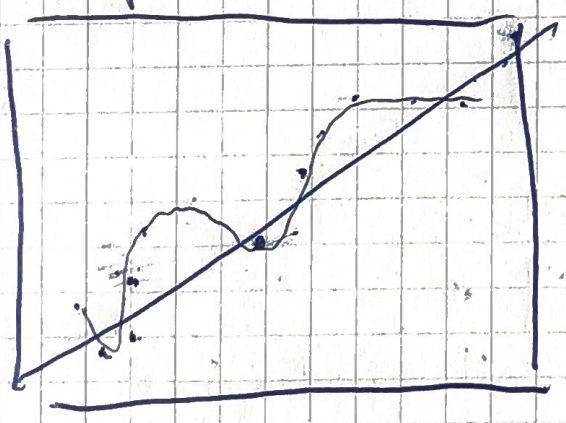
$$L_S(h) = \frac{1}{m}\sum_{i=1}^{m} \ell(y_i, h(x_i))$$

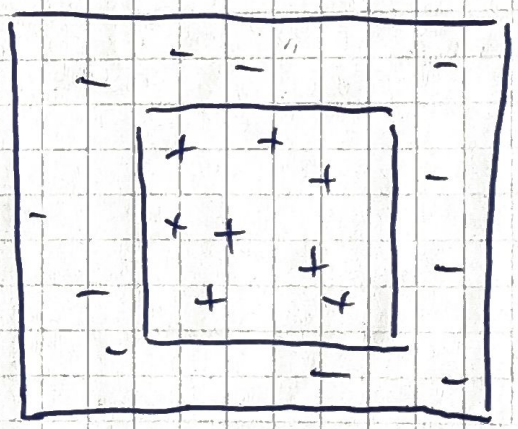
Fitting to data – L_S small

Overfitting – fitting too well to S can be bad

Examples:

1)





vs memorization

Resources: data ("sample complexity")

compute { train time, inference time

memory

randomness

1.1 PAC Learning

What is good enough?

In practice — based on use case (gambling, stock market vs. medical decision)

Ideally: Learning = "require $L_D(h) = 0$" (such h doesn't always $\exists$)

Simple case where 0-loss hypothesis exists:

Def (Realizable dist). D is $\mathcal{H}$-realizable if $\exists h \in \mathcal{H} : L_D(h) = 0$

Q: Can we get 0 loss in realizable case?

A:

Limitation	Solution in Definition	Name
Limited training set	$L_D(h) \leq \varepsilon$ "success"	"Approx Correct"
"Bad Luck" Bad samples	$P[\text{success}] \geq 1-\delta$	"Probably"
Other resources time, memory...	ignore	

Def (PAC Learning - Realizable Case). Let $X, Y, \ell : Y^2 \to \mathbb{R}_{\geq 0}$, $\mathcal{H} \subseteq Y^X$.

We say that $A : (X \times Y)^* \to Y^X$ **PAC Learns** $\mathcal{H}$ if:

$$\forall \varepsilon > 0 \; \forall \delta > 0 \; \exists m = m(\varepsilon, \delta) \in \mathbb{N} \; \forall \mathcal{H}\text{-realizable } D \in \Delta(X \times Y):$$

$$\underset{S \sim D^m}{P}\left[L_D(A(S)) \leq \varepsilon \right] \geq 1 - \delta .$$

Q: If 0-loss h doesn't exist? Don't want to assume anything about D?

A: compare to benchmark \ competitive with $\mathcal{H}$

Def (Agnostic PAC Learning). Let $X, Y, \ell, \mathcal{H}$.

We say that $A : (X \times Y)^* \to Y^X$ **agnostically PAC Learns** $\mathcal{H}$ if

$$\forall \varepsilon > 0 \; \forall \delta > 0 \; \exists m = m(\varepsilon, \delta) \in \mathbb{N} \; \forall D \in \Delta(X \times Y):$$

$$\underset{S \sim D^m}{P}\left[L_D(A(S)) \leq \underbrace{\inf_{h \in \mathcal{H}} L_D(h)}_{\text{OPT}} + \varepsilon \right] \geq 1 - \delta .$$

Lemma. Let $\mathcal{H}$ finite, $\ell: \mathcal{Y}^2 \to [0,1]$ (bounded). Then $\mathcal{H}$ is:

(1) PAC Learnable with samp comp $m(\epsilon,\delta) \le \frac{\ln(|\mathcal{H}|/\delta)}{\epsilon}$.

(2) Agnostic PAC Learnable — " — $m(\epsilon,\delta) \le O\left(\frac{\ln(|\mathcal{H}|/\delta)}{\epsilon^2}\right)$

$ERM_{\mathcal{H}}(S)$ ("Empirical Risk Minimization")

choose arbitrary $h \in \operatorname{argmin}_{h' \in \mathcal{H}} L_S(h')$

return h

Pf of Lemma.

Let $S \in (\mathcal{X}\times\mathcal{Y})^*$

Def. Fix $h \in \mathcal{H}$, $D \in \Delta(\mathcal{X}\times\mathcal{Y})$. We say that S is bad for h if $L_S(h) = 0$ and $L_D(h) > \epsilon$ ("misleading" S)

(1) $$\mathbb{P}_{S\sim D^m}\left[L_D(ERM(S)) > \epsilon\right] \le \mathbb{P}_{S\sim D^m}\left[\exists h \in \mathcal{H} : S \text{ is bad for } h\right]$$

$$\le \mathbb{P}_S\left[\bigcup_{h\in\mathcal{H}} \{S \text{ is bad for } h\}\right] \le \sum_{h\in\mathcal{H}} \mathbb{P}_S\left[S \text{ is bad for } h\right]$$

$$\le \sum_{h\in\mathcal{H}} (1-\epsilon)^m \le |\mathcal{H}| e^{-\epsilon m} \le \delta.$$

($1+x \le e^x$; $m \ge m(\epsilon,\delta)$)

$\left(\text{recall } \mathbb{P}_{(x,y)\sim D}[\ell(h(x),y) > 0] \ge \epsilon\right)$

((2) is similar)

2. Classification

Q: what is the best we can do? (samp comp)

A: VCDim (next page)

Thm. (VC68) For PAC Learning, optimal samp comp is

$$m(\epsilon,\delta) = \Theta\left(\frac{VC(\mathcal{H}) + \log(1/\delta)}{\epsilon}\right)$$

(ERM is optimal up to $\log(1/\epsilon)$ factor). optimal alg given by Hanneke (2016).

For agnostic PAC Learning: ERM is optimal with samp comp

$$m(\epsilon,\delta) = \Theta\left(\frac{VC(\mathcal{H}) + \log(1/\delta)}{\epsilon^2}\right)$$

Def (Restriction). Let $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$, $A \subseteq \mathcal{X}$, $h \in \mathcal{H}$.

- $h|_A = g$ s.t. $g: A \to \{0,1\}$ and $\forall a \in A: g(a) = h(a)$
- $\mathcal{H}|_A = \{h|_A : h \in \mathcal{H}\}$

Def (Shattering). Let $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$, $A \subseteq \mathcal{X}$, $|A| < \infty$.
We say that $\mathcal{H}$ shatters A if $|\mathcal{H}|_A| = 2^{|A|}$

Def (VC Dimension). Let $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$.

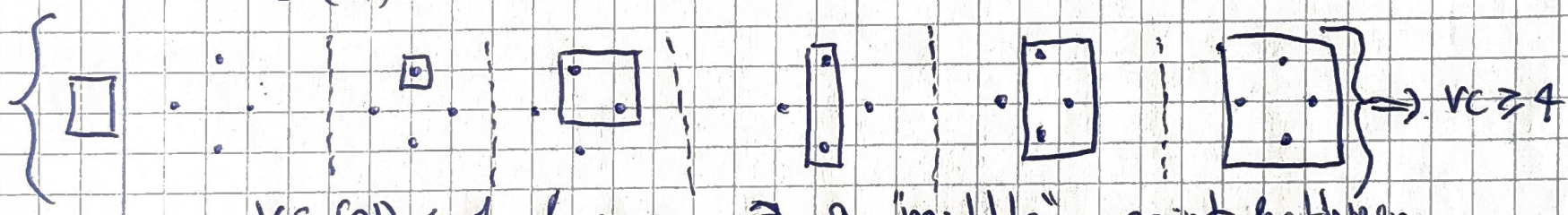
$$VC(\mathcal{H}) = \sup\{k \in \mathbb{N} : \exists A \subseteq \mathcal{X} \ |A| = k \wedge \mathcal{H} \text{ shatters } A\}.$$

Examples

- $|\mathcal{H}| \leq 1$: $VC(\mathcal{H}) = 0$
- $|\mathcal{H}| = 2$: $VC(\mathcal{H}) = 1$
- Thresholds: $\mathcal{H} = \{\mathbb{1}(x \geq t) : t \in \mathbb{R}\}$. $VC(\mathcal{H}) = 1$, But $|\mathcal{H}| = \infty$.
- Rectangles: $h_{x_{max}, x_{min}, y_{max}, y_{min}}(x,y) = (x_{min} \leq x \leq x_{max}) \wedge (y_{min} \leq y \leq y_{max})$

$\mathcal{H} = \{h_{x_1, x_2, y_1, y_2} : (x_1, x_2, y_1, y_2) \in \mathbb{R}^4\}$

$VC(\mathcal{H}) = 4$



$VC(\mathcal{H}) \leq 4$ because $\exists$ a "middle" point between extreme pts.
$\Rightarrow$ if extreme pts labeled 1 then middle also.

3. Time Complexity

- ERM can be computationally hard, E.g.:

Thm (Ben-David, Eiron, Long 2003). Let $\mathcal{X} = \mathbb{R}^n$ and $\mathcal{H}$ be the class of axis-aligned rectangles in $\mathbb{R}^n$. If $P \neq NP$ then $\forall \varepsilon > 0$ $\nexists$ poly(n)-time algorithm that given input $S \in (\mathcal{X} \times \{0,1\})^*$, outputs $h \in \mathcal{H}$ s.t. $L_S(h) \leq \left(\frac{770}{767} - \varepsilon\right) \min_{h' \in \mathcal{H}} L_S(h')$.

- ERM can be undecidable. $h_{Halt}(x) = \mathbb{1}(x \text{ is a TM that halts})$, $h_{\overline{Halt}}(x) = 1 - h_{Halt}(x)$. $\mathcal{H} = \{h_{Halt}, h_{\overline{Halt}}\}$. $S = \{(x, 1)\}$.

Def (LPN Search Problem). $\mathcal{X} = \mathbb{F}_2^n$, $f_s(x) = \langle s, x \rangle \in \mathbb{F}_2$
$f_s : \mathcal{X} \to \mathbb{F}_2$

$\mathcal{H} = \{ f_s : s \in \mathbb{F}_2^n \}$.

$D_s \in \Delta(\mathcal{X} \times \mathbb{F}_2)$ s.t.:

(1) $x \sim U(\mathcal{X})$; (2) $e \sim \text{Ber}(\frac{1}{8})$; (3) $y = f_s(x) \oplus e$

output: (x, y)

[redacted]

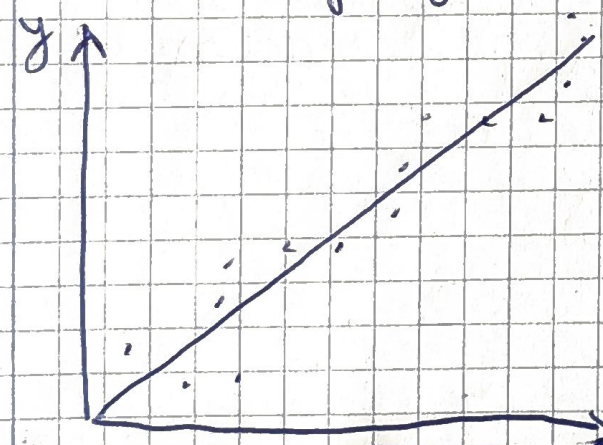
Problem: given $\left((x^{(1)}, y^{(1)}) \dots (x^{(m)}, y^{(m)}) \right) \sim (D_s)^m$, find s.

Hardness assumption: LPN is hard. I.e. $\forall$ poly(n)-time algorithm A:

$$\mathbb{P}_{\substack{s \sim U(\mathcal{X}) \\ X \sim U(\mathbb{F}_2^{m \times n}) \\ e \sim (\text{Ber}(\frac{1}{8}))^m}} \left[A(X, X \cdot s \oplus e) = s \right] \leq \text{negl}(n).$$

4. Regression

- Label space $\mathcal{Y}$ is continuous, no longer hope/expect to predict perfectly, $\hat{y} = y$, enough to be close, $\hat{y} \approx y$.
- Concretely: $\mathcal{Y} = \mathbb{R}$, $\ell(y, \hat{y}) = (y - \hat{y})^2$ (ℓ_2 loss)



$\mathcal{X} = \mathbb{R}^n$, $\mathcal{H} = \{ h_w : w \in \mathbb{R}^n \}$

$h_w(x) = \langle w, x \rangle$, $h_w : \mathbb{R}^n \to \mathbb{R}$

$=$ ERM

OLS(S): ("Ordinary Least Squares"
$S \in (\mathbb{R}^n \times \mathbb{R})^*$, $S = ((x^1, y^1) \dots (x^m, y^m))$

return $h_w \in \text{argmin} \frac{1}{m} \sum_{i=1}^{m} (h_w(x^i) - y^i)^2$

$f(w) = L_S(h_w)$ is convex (sum of squares of linear functions) in w, so solving

$$\nabla f(w) = 0$$

gives a closed-form solution for OLS.

4.1 Regularization

Often we get better generalization if we have inductive bias towards "simpler" functions using regularization:

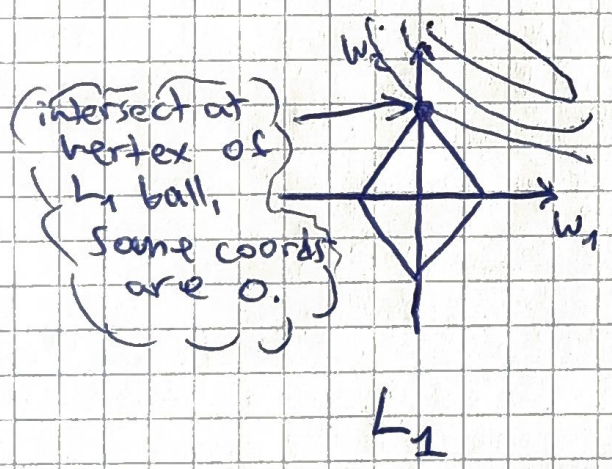
Select h_w that minimizes:

- L_2 / Ridge: $L_S(h_w) + \lambda\|w\|_2^2 = \frac{1}{n}\sum_i (\langle w, x^i\rangle - y^i)^2 + \lambda\|w\|_2^2$
- L_1 / Lasso: $L_S(h_w) + \lambda\|w\|_1 = \text{—"—} + \lambda\|w\|_1$

Regularization limits fit to noise (L_1 & L_2)

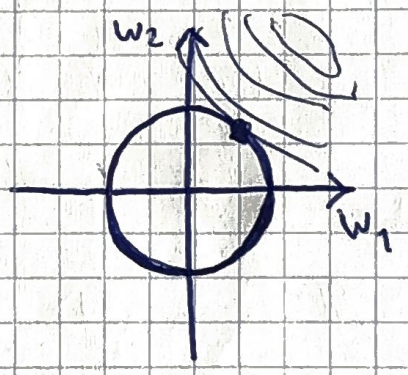
Intuition:

- To fit outliers with large $|y_i|$, we need $\|w\|$ large. But if outlier is noise / not representative, then this large $\|w\|$ will give large population loss.
- For $x \in \mathbb{R}^n$, if there are some directions (covariates) x_i that are not informative (indep. of y_i), OLS fits w_i to noise in direction i
- L_1 induces sparse solutions. $L_S(h_w)$ is quadratic function of w. Contours are ellipsoids.



L_1

Sparse solutions



L_2

Non-sparse solution

5. Probabilistic Classification

$y = \{0,1\}$ but $\mathcal{H} \subseteq [0,1]^X$.

$\hat{p} = h(x)$ = "probability that label is 1"

Ex: Predict whether medication will cure patient

(We could replace $h(x)$ with e.g. $\mathbb{1}(h(x) \geq \frac{1}{2})$ to make binary predictions, but $\hat{p} \in [0,1]$ useful in many cases.

- conceptually, represents uncertainty
- good for optimization

★ another intuition:

Suppose x_1, x_2 have correlation with each other, $(x_1 \approx x_2)$, and with y (x_1 a bit better)

e.g. [illegible]

e.g. L_2 is minimized by any w_m with

$$w_1 + w_2 = 1$$

then L_2 regularization would give $w_1 \approx w_2 \approx \frac{1}{2}$

L_1 $w_1 = 1, w_2 = 0$

Because L_2 penalizes larg abs value, for each w_i

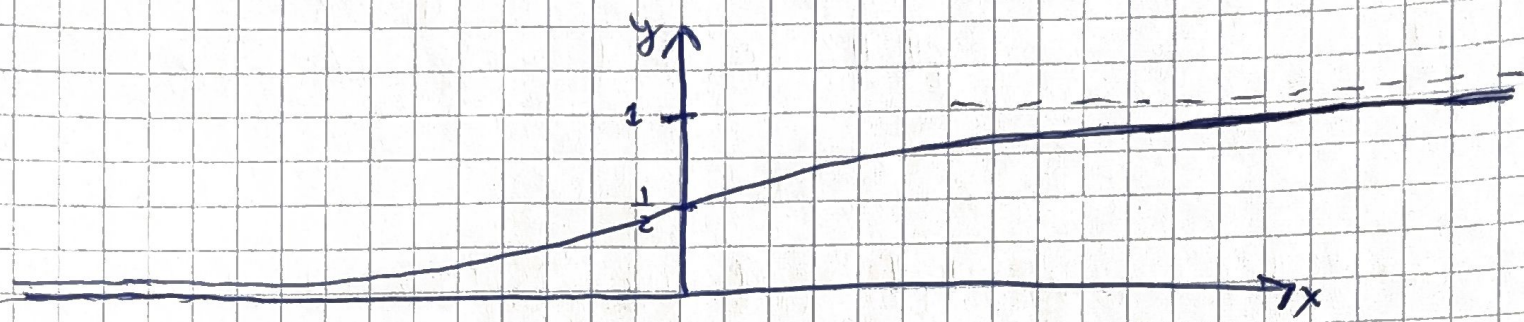
L_1 doesn't, only sum of w_i

Loss function: Cross Entropy (CE) / Log Loss / Suprisal:

$$\ell(y,\hat{p}) = \begin{cases} \ln\frac{1}{\hat{p}} & y=1 \\ \ln\left(\frac{1}{1-\hat{p}}\right) & y=0 \end{cases} = -y\ln\hat{p} + (1-y)\ln\frac{1}{1-\hat{p}} = \mathop{\mathbb{E}}_{y}\left[\ln\frac{1}{\hat{p}_y}\right]$$

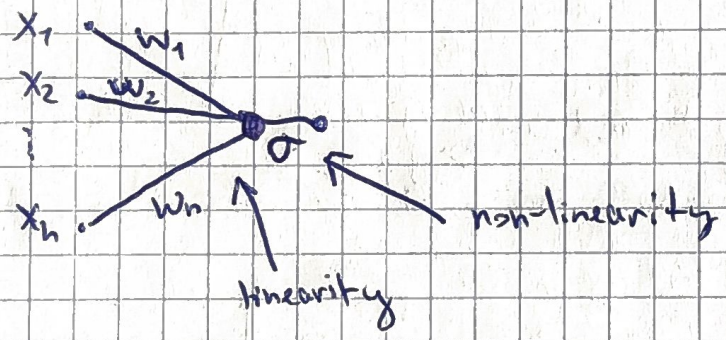
5.1 Logistic Regression

Logistic function: $\sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}$



Logistic Regression: a "generalized linear model":

- $\mathcal{X} = \mathbb{R}^n$, $h_w = \sigma(\langle w, x\rangle + b)$, $\mathcal{H} = \{h_w : w \in \mathbb{R}^n\}$
- One of the most common models in science
- Is a single-neuron NN:



- no closed form solution

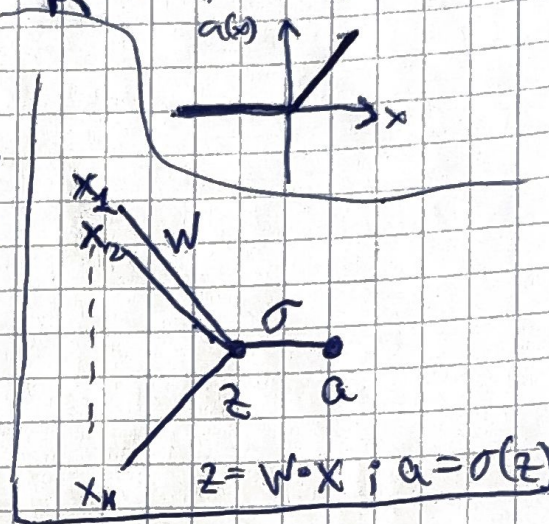
6. Neural Networks

(Very loosely inspired by biology)

Neuron: composition of linear function with non-linear "activation" function, eg.:

- Rectified Linear Unit (ReLU): $R(x) = \max\{0, x\}$
- Logistic: $\sigma(x)$
- Softmax (generalized logistic, maps $x_1 \dots x_k \longmapsto \Delta([k])$):

$$\sigma(x_1, \dots, x_k) = \left(\frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}} \right)_{i=1}^{k}$$



Neural Network: composition of many neurons

Architectures:

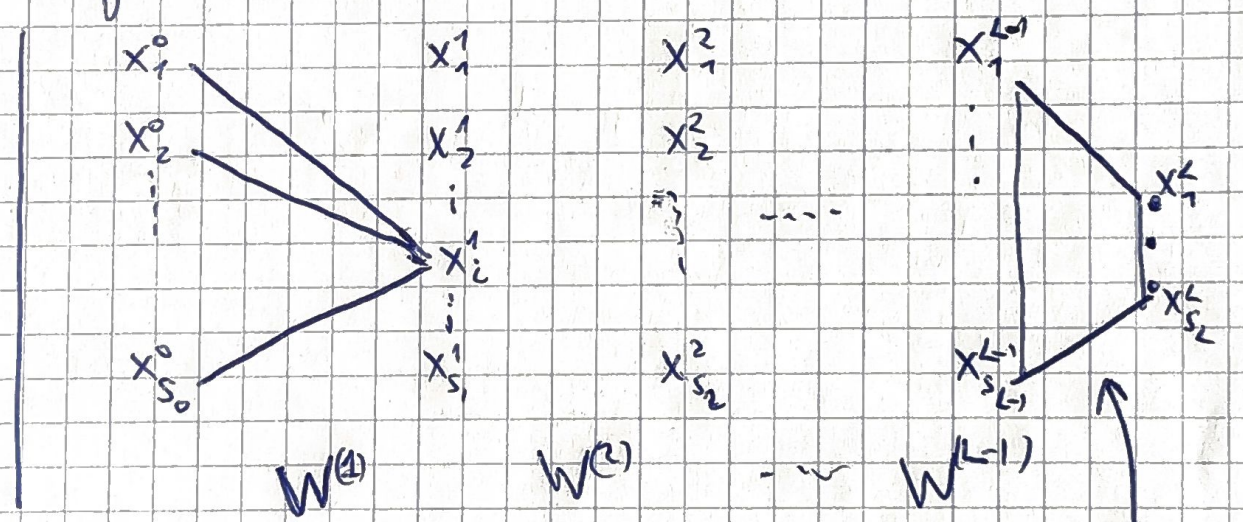
Forward Pass

For $i = 1, 2, \dots, L$:

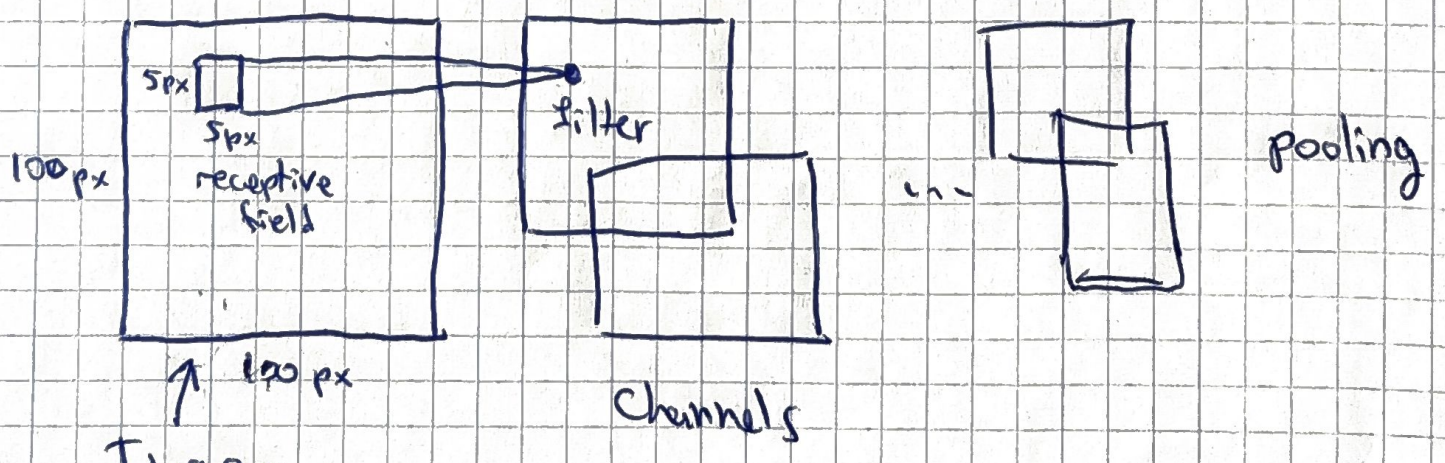
$z^{(i)} \leftarrow W^{(i)} \cdot x^{(i-1)}$

$x^{(i)} \leftarrow \sigma_i(z^{(i)})$

- Fully-connected:



- Skip connections
- Recurrent
- Convolutional (CNN). Common in vision:



- Efficiency:

Fully connected	CNN
10,000 weight per neuron	25 weights per channel

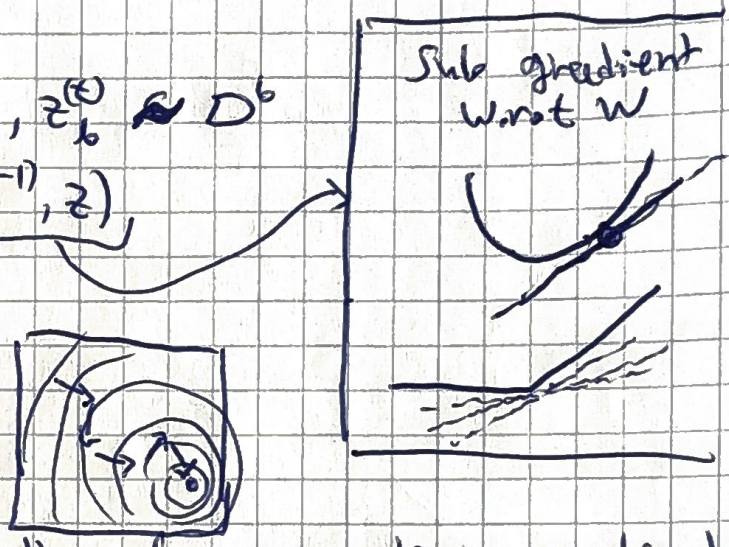
- locality. Filters for local features like Gabor lines
- Translation ~~invariance~~ equivariance

6.1 Training NN

(Minibatch) Stochastic Gradient Descent (SGD)

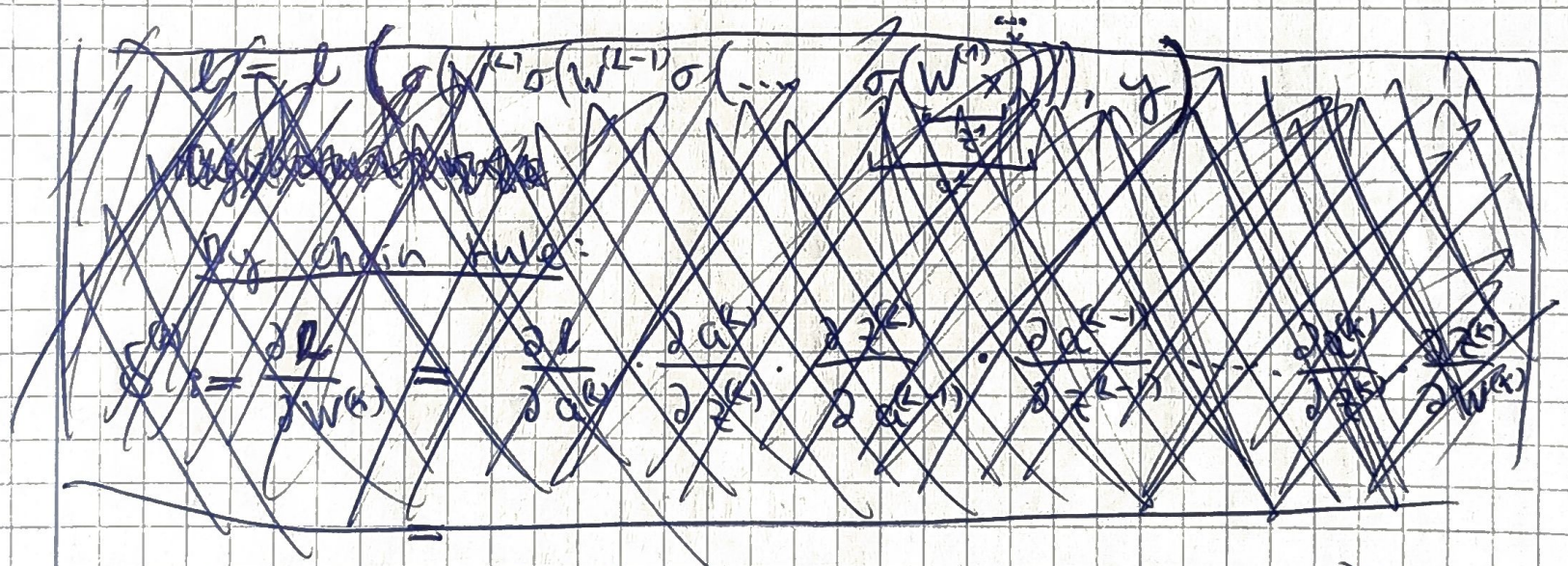
WEIR_params: $\eta > 0$ – step size ; $T \in \mathbb{N}$ – number of rounds ; $b \in \mathbb{N}$ – mini-batch size

Initialize $W^{(0)}$ randomly
for $t = 1, 2, \ldots T$:
  Sample $z_1^{(t)}, z_2^{(t)}, \ldots, z_b^{(t)} \sim D^b$
  Select $v^{(t)} \in \partial \ell(W^{(t-1)}, z)$
  $W^{(t)} \leftarrow W^{(t-1)} - \eta \cdot v^{(t)}$



Back-Propagation

A dynamic-programming algorithm for computing gradients.



Thm (Backpropagation / auto-differentiation – informal)
Suppose $f: \mathbb{R}^d \to \mathbb{R}$ is computed by a differential circuit of size N. Then ∇f can be computed in time $O(N)$.

Idea:

- Compute forward pass $x^0, z^1, x^2, z^2, x^{L-1}, z^L, x^L$, loss $\ell = \ell(x^L, y)$
- Let $\delta^L = \frac{\partial \ell}{\partial z^L}$
- for $k = L-1, L-2, \dots 1$:

$$\delta^k := \frac{\partial \ell}{\partial z^k} = \frac{\partial \ell}{\partial z^{k+1}} \cdot \frac{\partial z^{k+1}}{\partial z^k} = \delta^{k+1} \cdot \frac{\partial z^{k+1}}{\partial x^k} \cdot \frac{\partial x^k}{\partial z^k}$$

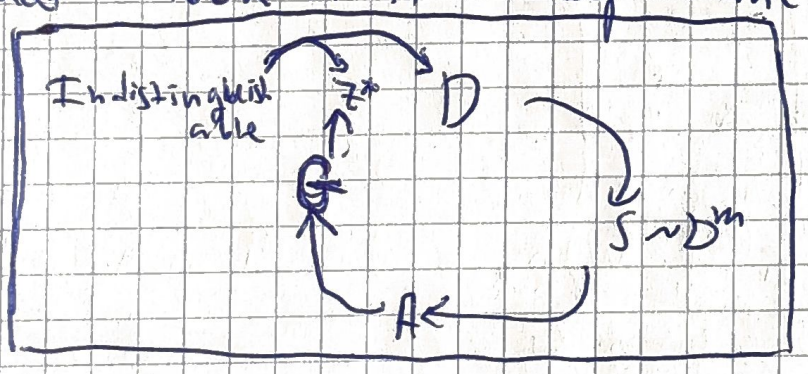
$$= \delta^{k+1} \cdot W^{k+1} \cdot \sigma'(z^k)$$

$$\frac{\partial \ell}{\partial W^k} = \frac{\partial \ell}{\partial z^k} \cdot \frac{\partial z^k}{\partial W^k} = \delta^{(k)} \cdot x^{(k-1)}$$

7. Generation

Example: thispersondoesnotexist.com

General Idea: I show my learning alg iid samps $z_1 \dots z_m \sim D^m$, and now it can generate/produce new z^* that "look as if" they came from D



Ex: After seeing 100 shakespeare sonnets, my system can generat new "Shakespeare sonnets"

What is "indistinguisable"? Unclear, based on use-case
NOT TV etc.

7.1 LLMs

can be used/"prompted" w/ prefixes, not directly conversational

Corpus → LM GPT → "pretraing" Next-token-prediction → RLHF → Chatbot

"Peanut butter and ____"

RAG
Tool Use
CoT

Transformer