BLG 561 E FALL 2021

Deep Learning

19. 10. 2021

Gözde ÜNAL

All ML algorithms have 3 components:

(model)

1. Hypothesis Class: set of functions we consider for mapping input to output
2. Loss function: how "good" our selected hypothesis is
3. Optimization: How do we find a hypothesis ($\equiv$ model parameters) w/ the given loss (objective) fn.

$\Rightarrow$ Probabilistic view of ML $\rightarrow$ relates mainly to both hypothesis fn. & the loss fn.

ML Problem. Given $x^{(1)}, \dots, x^{(m)}$: How do I find a probability distribution that captures this data "well"?

$P(\theta; X)$: How do I find θ (parameters) of this distrib. that fits the data well?

$\rightarrow$ Maximum Likelihood Estimation (MLE) : the prob. of observing this data:

Given observed independent data points

$$\rightarrow P(\underbrace{x^{(1)}, x^{(2)}, \dots, x^{(m)}}; \theta) = \prod_{i=1}^{m} p(x^{(i)}, \theta) \longrightarrow$$

Basic idea in MLE is $\rightarrow$

Find the parameters that maximize the probability of the observed data:

$$\max_\theta \underbrace{\prod_{i=1}^{m} p(x^{(i)}, \theta)}_{\triangleq\, \ell(\theta)} = \max_\theta \underbrace{\ell(\theta)}_{\text{likelihood.}} \Rightarrow \text{maximize log likelihood of our data.}$$

$$\rightarrow \max_\theta \sum_{i=1}^{m} \log p(x^{(i)}; \theta)$$

(many ML algo. can be interpreted as an MLE)

MLE for Linear Regression: $y = \theta^T x + \varepsilon$

$\varepsilon \sim \mathcal{N}(0, \sigma^2)$

Given x, prob. distrib. of y?

$$P(y \mid x ; \theta) = \mathcal{N}(\theta^T x, \sigma^2) \quad \propto \exp\left(-\frac{(y-\theta^T x)^2}{2\sigma^2}\right)$$

MLE: $\max \log p(y \mid x ; \theta)$

$$\max_\theta \sum_{i=1}^{m} \log p(y^{(i)} \mid x^{(i)}; \theta) \rightarrow \min_\theta - \sum_{i=1}^{m} \log p(y^{(i)} \mid x^{(i)}; \theta)$$

(negative log likelihood) go to NLL..

$$\Rightarrow \min_\theta \sum_{i=1}^{m} (y^{(i)} - \theta^T x^{(i)})^2$$

Least squares loss (MSE) can be viewed MLE under ← Gaussian assumption

Exercise: Now, say errors are Laplace-distributed; $p(x) = \frac{1}{2\sigma} e^{-|x-\mu|.\sigma}$

Derive MLE in this case $\rightarrow$ (obtain the MAE)

MLE for Logistic Regression: (Binary Classification) $\{+1, -1\}$

$p(y=+1 \mid x ; \theta) \propto \exp(\theta^T x)$, : $\underline{\theta^T x} \uparrow$.

$p(y=-1 \mid x ; \theta) \propto 1$. : $\theta^T \simeq 0$

$p(y=+1 \mid x^i ; \theta) = \frac{\exp(\theta^T x^i)}{1+\exp(\theta^T x^i)}$; $p(y=-1 \mid x^i ; \theta) = \frac{1}{1+\exp(\theta^T x^i)}$

$\rightarrow$ $\boxed{\text{Hypothesis fn } h_\theta(x) = \frac{1}{1+e^{-y^i.\theta^T x^i}}}$: sigmoid fn. (aka: logistic fn.)

$\rightarrow$ MLE estimate for θ : $\max_\theta \sum_{i=1}^{m} \log p(y^{(i)} \mid x^{(i)} ; \theta) = \max_\theta \sum_i \log \frac{1}{1+e^{-y^i \theta^T x^i}}$

$\xrightarrow[\text{convert to}]{\text{min}}$ $\min_\theta \sum_i \log(1 + e^{-y^{(i)}\theta^T x^{(i)}})$: this is how we obtain the logistic regression loss we've seen last time.

Cross-Entropy Loss: (for Binary Classification)

Recall. Bernoulli r.v.

$$\left.\begin{aligned} p(x=1) &= \phi \\ p(x=-1) &= 1-\phi \end{aligned}\right\} p(x) = \phi^{x}(1-\phi)^{1-x}$$

$$\left.\begin{aligned} p(y=+1 \mid x;\theta) &= h_\theta(x) \\ p(y=-1 \mid x;\theta) &= 1-h_\theta(x) \end{aligned}\right\}$$

Combine $p(y \mid x;\theta) = (h_\theta(x))^{y}(1-h_\theta(x))^{1-y}$.

$$\ell(\theta) = p(y \mid x;\theta) = \prod_{i=1}^{m} p(y^{(i)} \mid x^{(i)};\theta)$$

we assume indep. data samples

$$= \prod_{i=1}^{m} (h_\theta(x^{(i)}))^{y^{(i)}} (1-h_\theta(x^{(i)}))^{(1-y^{(i)})}$$

Maximize the log-likelihood:

$$\max \; \ell(\theta) = \sum_{i=1}^{m} y^{(i)} \log(h_\theta(x^{(i)})) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)}))$$

$H(p) = \sum_p \log p.$

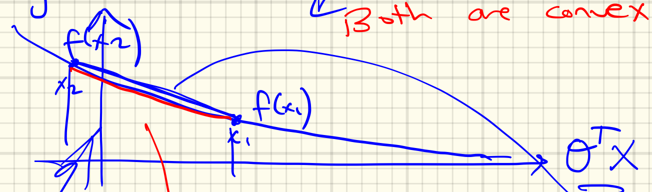
aka Binary Cross Entropy (BCE)

$$\min \; \ell_{BCE} = \sum_{i=1}^{m} \underbrace{-y^{(i)} \log\left(\frac{1}{1+e^{-\theta^T x^{(i)}}}\right)}_{y=+1} \underbrace{- (1-y^{(i)}) \log\left(\frac{1}{1+e^{-\theta^T x^{(i)}}}\right)}_{y=-1}$$

Q: Is BCE a convex fn?

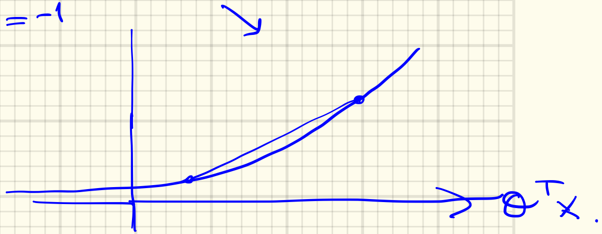
for $y=+1$

Both are convex.



is this convex?

$y=-1$



is this convex?

Convex fn: $f(\alpha x_1 + (1-\alpha) x_2) \le \alpha f(x_1) + (1-\alpha) f(x_2)$

$\alpha \in (0,1)$

BCE: is sum of 2 convex fn.s convex? ✓ Yes.

SOFTMAX Loss Fn. Now y takes values $\{1, \dots, k\}$ w/

$$p(y=j \mid x;\theta) \propto \exp(\theta_j^T x)$$

(unnormalized prob) likelihood fn.

$$\begin{pmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_K \end{pmatrix}$$

normalize

$$p(y=j \mid x;\theta) = \frac{\exp(\theta_j^T x)}{\sum_{l=1}^{k} \exp(\theta_l^T x)}$$

→ maximize log-likelihood.

$$\max_{\theta} \log p(y=j \mid x;\theta) = \theta_j^T x - \log \sum_{l=1}^{k} \exp(\theta_l^T x)$$

softmax loss ⇒

$$\min_{\theta} \sum_{i=1}^{m} \log \sum_{l=1}^{k} \exp(\theta_l^T x) - \theta_{y^{(i)}}^T x^{(i)} \longrightarrow$$

true class label: $y^{(i)}$.

multi-class classification soft max loss

$$\min_{\theta} \left[\sum_{i=1}^{m} \log \sum_{l=1}^{k} \exp(h_{\theta_l}(x^{(i)})) - h_{\theta_y}(x^{(i)}) \right]$$

Multi-Class SVM Loss:

$$l_{hinge}^{(i)} = \sum_{j \neq y_i} \max(0, 1 + S_j - S_{y_i})$$

correct class score.

S_j: prediction hypothesis.

if $\underbrace{S_j + 1}_{\text{+ margin of 1}} \leq \underbrace{S_{y^{(i)}}}$ $\longrightarrow$ No loss is added.

When $S_j + 1 \geq S_{y_i}$ +ve loss (penalty) is added to the loss.

$S = \theta^T x$: linear hypothesis

→ minimize overall SVM loss

$$\theta^* = \arg\min_{\theta} \mathcal{L}_{svm}(\theta) = \frac{1}{m}\sum_{i=1}^{m} \ell_{hinge}^{(i)}$$

Q. $\mathcal{L}_{svm}(\theta^*)$ → is θ^* unique?

Consider $\sum \max(0, \theta_j^T x^i - \theta_{y^i}^T x^i + 1)$ → θ was a minimizer

Double θ 2θ $\sum \max(0, \underbrace{2\theta_j^T x^i - 2\theta_{y^i}^T x^i + 1}_{})$ → 2θ is also a minimizer.

$2\theta_j^T x^i + 1 \geq 2\theta_{y^i}^T x^i$

$\gg$

How to prevent that! $\theta \to k\theta$

Generally:

$$\arg\min_{\theta} \mathcal{L}(\theta) = \underbrace{\arg\min_{\theta} \frac{1}{m}\sum_{i=1}^{m} \ell^{(i)}(h_\theta(x^i), y^{(i)})}_{\text{Data Term.}} + \lambda \cdot R(\theta)$$

Add a Regularizer

Ex: Regularizer $R(\theta) = \|\theta\|_2^2 = \sum_i \theta_i^2$: penalize growth of θ →

L2-constraint

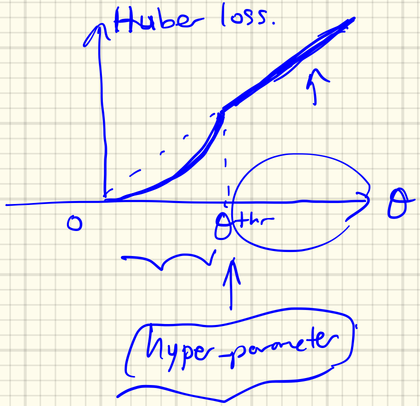
L1 constraint: $R(\theta) = \|\theta\|_1 = \sum_i |\theta_i|$

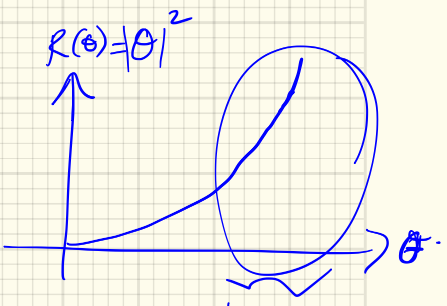
Lasso
Elastic

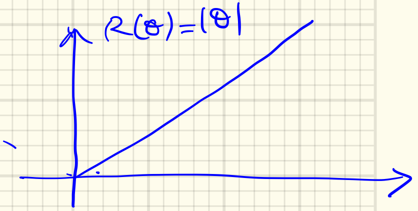
X.

$L2 + L1$ (regularizer)

Huber Loss







→ Last time: predicting electricity demand; we used linear hypothesis

Linear Hypothesis Class w/ Non linear Features.

Peak electricity demand$^{(i)}$ vs. Temp$^{(i)}$ $\{x^{(i)}, y^{(i)}\}$

mean of a given day.

$$x^{(i)} = \begin{bmatrix} (\text{Mean Temp})^{2\,(i)} \\ (\text{Mean Temp}) \\ 1 \end{bmatrix} = \begin{bmatrix} x^{2\,(i)} \\ x^{(i)} \\ 1 \end{bmatrix} = \underline{x} = \begin{bmatrix} x_2 \\ x_1 \\ 1 \end{bmatrix}$$

Same hypothesis class as before:

$$h_\theta(x) = \underline{\theta}^T \underline{x} = \theta_1 x^{2\,(i)} + \theta_2 x^{(i)} + \theta_3$$

: e.g. we used a quadratic polynomial

Generalize: n^{th} order polynomial on the data

$$h_\theta(x^i) = \sum_{j=0}^{10} \theta_j x_j^{(i)}$$

: e.g. a 10^{th} degree polynomial.

→ Next: check the Slides ⟶

Until now, we've seen Linear Hypothesis fns: $y = Wx + b$

$y = \theta^T_{n\times 1} x_{n\times 1}$

→ Nonlinear Hypothesis Examples in ML:

– Nearest-Neighbor Classifier: predict output based on "nearest" ex. in the training set. $\{x_i\}_{i=1}^m$

$$h_\theta(x) = y\{\underset{i\in\{1,\dots,m\}}{\arg\min} \text{"dist"}(x, x^i)\}$$

Non-parametric hypothesis class. pros: no training required.

Con: Keep go over all the training data even at TESTING time!

exercise: Write the hypothesis class for KNN (K-nearest neighbor classifier).

refer to a specific class of

→ Neural Networks: (Nonlinear) Hypothesis Functions.

↳ refer to the 1st element. 1

NN hypothesis ✓

2 Any loss fn? ✓

3 Any optimization method ✓

nonlinear

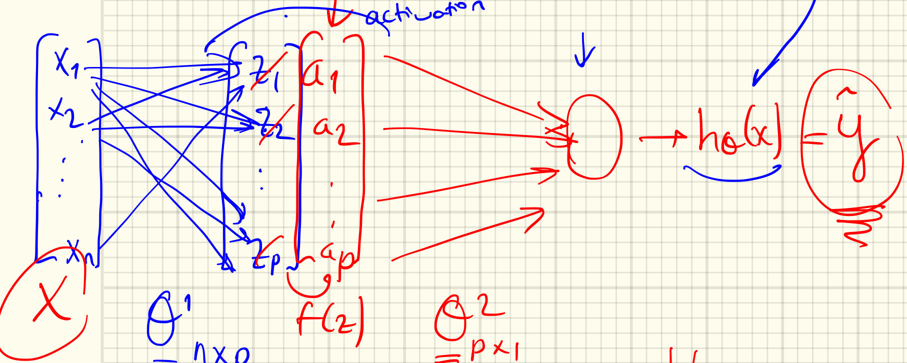
Let's consider a NN: 1 $h_\theta(x) = \cancel{f}\left(\underline{\underline{\theta_2}}^T f(\underline{\underline{\theta_1}}^T \underline{x})\right)$

the hypothesis class for a 2-layer NN

$\equiv$ a single hidden layer NN

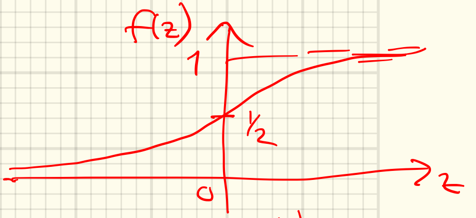
elementwise nonlinearity $f: \mathbb{R} \to \mathbb{R}$:

$$f(z) = \frac{1}{1+e^{-z}}$$



$\underline{\underline{\Theta}}^1_{n\times p}$ $f(z)$ $\underline{\underline{\Theta}}^2_{p\times 1}$

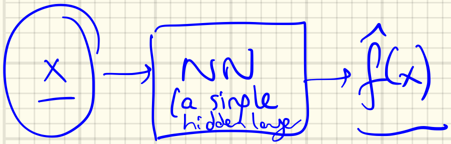
$z_1 = (\Theta^1 x) \to f(z_1)$



Universal Function Approximation Theorem:

Given any smooth fn. $f : \mathbb{R} \to \mathbb{R}$ (closed region $X \subset \mathbb{R}^n$)
$\mathbb{R}^n \to \mathbb{R}$

and $\varepsilon > 0$, we can construct a one hidden-layer neural network, $\hat{f}$ s.t.



$$\max_{x \in X} \left| f(x) - \hat{f}(x) \right| \leq \varepsilon$$

A NN w/ a single hidden layer ($\forall$ enough hidden units) w/ a squashing (activation) fn. is a universal fn. approximator.

Food for thought: $\longrightarrow$ Although a 2 layer NN can represent any function, it may fail due to several reasons: What are they?

→ Importance of the nonlinear activation fn.

Consider a 2 stage hypothesis class.

$$h_\theta(x) = \underline{\underline{W}}_2 \, (\underbrace{\underline{\underline{W}}_1 x + b)}_{a_1 \in \mathbb{R}^{k\times 1}} + b_2 \,, \quad x \in \mathbb{R}^n$$

$$\underline{\theta} = \{ \underline{\underline{W}}_1 \in \mathbb{R}^{k\times n} \,, \; b_1 \in \mathbb{R}^k \,, \; \underline{\underline{W}}_2 \in \mathbb{R}^{1\times k} \,, \; b_2 \in \mathbb{R} \}$$

Final hypothesis :

$$h_\theta(x) = \underbrace{\underline{\underline{W}}_2 \, \underline{\underline{W}}_1}\, \underline{x} + \underbrace{W_2 b + b_2}$$

$$h_\theta(x) = \underline{\underline{\tilde{W}}} \; \underline{x} + \underline{\tilde{b}}$$

The composed model is still affine (linear)

∴ We have not gained any representation power.

$h_\theta(x)$: # parameters in $\underline{\theta}$ $\equiv$ capacity of the model.

NNs introduce a nonlinear fn. after each affine operation.

$$h_\theta(x) = f_2\left(\underline{\underline{W_2}}\, f_1(W_1 x + b_1) + b_2\right)$$

$f_1, f_2 : \mathbb{R} \to \mathbb{R}$ ← nonlinear (elementwise) functions

$$f(\underline{x}) = \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_n) \end{bmatrix}$$

Ex: Learning the XOR Function:

Training data $x^{(i)} = \begin{Bmatrix} 0,0 \\ 0,1 \\ 1,0 \\ 1,1 \end{Bmatrix}$, $y^{(i)} = \begin{Bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{Bmatrix}$

$m = 4$

1. Try a linear hypothesis: $h_\theta(x^i) = \underline{\underline{\theta}}^T x + b$

$$\hat{y}^i = h_\theta(x^i) = \underbrace{[\theta_1 \;\; \theta_2 \;\; b]}_{\underline{\underline{\theta}}} \underbrace{\begin{bmatrix} x_1^i \\ x_2^i \\ 1 \end{bmatrix}}_{\underline{x}}$$

Q: Could a linear ^ work?

2. Let's use MSE: $\mathcal{L}(\theta) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i)^2 = (\underline{\underline{\theta}}^T \underline{x} - \underline{y})^2$

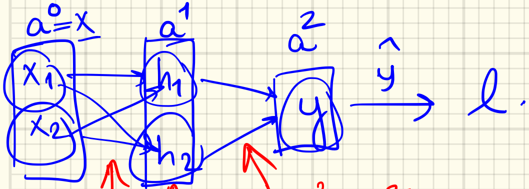
→ Normal eqns. $\underline{\underline{\theta}} = (\underline{\underline{X}}^T \underline{\underline{X}})^{-1} (\underline{\underline{X}}^T y)$ → $\boxed{\underline{\theta}^T = [0 \;\; 0 \;\; 0.5]}$

→ Linear classifier →

$$h_\theta(x) = [\cancel{\theta_1}^{\,0} \quad \cancel{\theta_2}^{\,0} \quad 0.5] \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} = 0.5$$

Linear classifier not able to learn represent XOR fn.

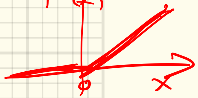
→ Use a NN w/ a single hidden layer



$$h = a^1_{2\times 1} = \underbrace{f}_{\text{nonlinearity}}\big(\underbrace{W^1 a^0 + b^1}_{\text{affine layer oper.}}\big)$$

$$\hat{y} = a^2 = \underbrace{W^2 a^1 + b^2}_{\text{affine layer}}$$

eg. $f(x)$: ReLU



∴ The Nonlinear hypothesis fn for the overall network:

$$\hat{y} = W^2 f(W^1 a^0 + b^1) + b^2 = h_\theta(x), \quad \theta = (W^1, b^1, W^2, b^2)$$

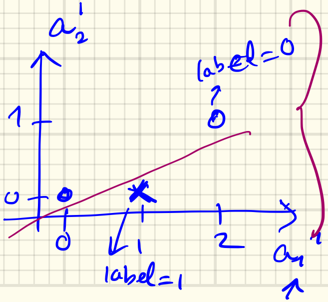
Suppose we ~~know~~ specify the soln to be:

$$W^1 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \quad b^1 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

$$W^2 = \begin{bmatrix} 1 \\ -2 \end{bmatrix}, \quad b^2 = 0$$

Let's check the hidden layer: for 4 training samples

$$a^1 = \begin{Bmatrix} a^{1(1)} \\ a^{1(2)} \\ a^{1(3)} \\ a^{1,(4)} \end{Bmatrix} = \begin{pmatrix} 0,0 \\ 1,0 \\ 1,0 \\ 2,1 \end{pmatrix}; \quad \hat{y} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$



this space is separable btw. classes.

→ ✓ Now, while we increased the capacity of the model ≡ (hypothesis)
we increased our representation power.

& learned (the feature space) to represent Xor function.

Also, we fit to the training data well.

Note: Generalization is another story.