Please fill out course survey

https://go.blueja.io/dEfDbakFQkmiLp__xxDacQ

Final Exam Review

- Exam info in eClass announcement
  - Dec 18, 8:30am-11:30am in Pavilion
  - 40 multiple choice questions
  - 40% Chap 9-11
  - 40% Chap 6-8
  - 20% Chap 1-5

Study tips:

- Review assignments
- Review examples and exercises in course notes
- Review midterms
- Understand everything on the formula sheet

Other notes

- softmax cannot be represented as a single layer NN
- $d^{(b)}$ is the number of non-bias neurons

Math Review

Sets and notation: $\{0,1,2\}$, $\mathbb{N}$, $\mathbb{R}$

$\in$, $\subset$, $\notin$, $\not\subset$, $\cup$, $\cap$, $\setminus$, set^c

set builder notation:

$\{x \in \mathbb{N} \mid x < 3\} = \{0,1,2\}$

$\{f \mid f : \mathbb{R} \to \mathbb{R}\}$

Cartesian products (set of tuples):

$\mathcal{X} \times \mathcal{Y} = \{(x,y) \mid x \in \mathcal{X}, y \in \mathcal{Y}\}$

$\mathbb{R}^3 = \mathbb{R} \times \mathbb{R} \times \mathbb{R} = \{(a,b,c) \mid a,b,c \in \mathbb{R}\}$

Dot product: $\vec{x} \in \mathbb{R}^d$, $\vec{w} \in \mathbb{R}^d$

$$\vec{x}^T \vec{w} = (x_1, \dots, x_d)(w_1, \dots, w_d)^T$$
$$= x_1 w_1 + \dots + x_d w_d$$

Functions:

$f : \mathbb{R} \to \mathbb{R}$, $f(x) = x^2$

$f : \mathbb{R}^2 \to \mathbb{R}$, $f(\vec{x}) = f(x_1, x_2) = 2x_1 + 3x_2^2$

$\mathcal{A} : (\mathbb{R} \times \mathbb{R})^n \to \{ f \mid f : \mathbb{R} \to \mathbb{R} \}, \quad n = 2$

$$D = ((1,2),(3,4))$$

$$\mathcal{A}(D) = f \quad \text{where } f(x) = (4-2)x + 2 + 1$$

Summation and Integration:

$\mathcal{X} = (x_1, x_2, x_3)$, $f(x) = x^2$

$$\sum_{x \in \mathcal{X}} x = \sum_{i=1}^{3} x_i = x_1 + x_2 + x_3$$

$$\sum_{x \in \mathcal{X}} f(x) = \sum_{i=1}^{3} f(x_i) = f(x_1) + f(x_2) + f(x_3)$$

$\mathcal{Y} = [a, b]$, $f(y) = y^c$

$$\int_{\mathcal{Y}} f(y)\,dy = \int_a^b f(y)\,dy = \int_a^b y^c = \left. \frac{y^{c+1}}{c+1} \right|_a^b$$

$f(x,y) = xy$, $\mathcal{X} = (x_1, x_2, x_3)$, $\mathcal{Y} = [a, b]$

$$\int_{\mathcal{Y}} \sum_{x \in \mathcal{X}} f(x,y)\,dy = \int_a^b \sum_{i=1}^{3} x_i y \, dy$$

$\int^b$

$$= \int_a x_1 y + x_2 y + x_3 y \, dy$$

$$= \frac{x_1 y^2}{2}\Big|_a^b + \frac{x_2 y^2}{2}\Big|_a^b + \frac{x_3 y^2}{2}\Big|_a^b$$

(Partial) Derivatives:

$f: \mathbb{R} \to \mathbb{R}, \quad f(x) = x^2, \quad f'(x) = \frac{df}{dx}(x) = 2x, \quad f''(x) = 2$

$f: \mathbb{R}^2 \to \mathbb{R}, \quad f(\vec{x}) = f(x_1, x_2) = x_1^2 + x_2^2$

$\frac{\partial f}{\partial x_1}(x_1) = 2x_1, \quad \frac{\partial f}{\partial x_2}(x_2) = 2x_2$

Common derivatives and properties on formula sheet

Probability

Outcome space and Events:

$\mathcal{X} = \{1,2,3,4,5,6\}$ outcome space

$E \subset \mathcal{X}$ event $\quad E = \{1,2,3\}$

$= \mathcal{X}$

$\neq \{\mathcal{X}\}$

Probability Distribution:

$\mathbb{P}$ takes as input events and outputs values in $[0,1]$

$\mathbb{P}(E)$ where $E \subset \mathcal{X}$ probability of event E

Random Variable:

outcome space

$\mathcal{X}, X, x$ ← variable or instance of a r.v. X

r.v. $X \in \mathcal{X}$ and $\mathbb{P}$

$x \in \mathcal{X}$

$X \in \mathcal{X}$ and has distribution $\mathbb{P}$

$\mathbb{P}(X \in E) \stackrel{\text{def}}{=} \mathbb{P}(E)$ where $E \subset \mathcal{X}$

$\cancel{\mathbb{P}(x \in E)}$

Common notation: If E contains a single outcome

$E = \{x\}$ where $x \in \mathcal{X}$. Then $\mathbb{P}(X = x) \stackrel{\text{def}}{=} \mathbb{P}(X \in \{x\})$

$= \mathbb{P}(\{x\})$

If E is an interval:

$E = [a,b]$ then $\mathbb{P}(a \le X \le b) \overset{\text{def}}{=} \mathbb{P}(X \in [a,b])$

$E = [a,\infty)$ then $\mathbb{P}(X \ge a) \overset{\text{def}}{=} \mathbb{P}(X \in [a,\infty))$

$E = (-\infty, b]$ then $\mathbb{P}(X \le b) \overset{\text{def}}{=} \mathbb{P}(X \in (-\infty, b])$

$E \subset \mathcal{X} = \mathbb{R}$

Discrete r.v.:

Countable outcome space $\quad \mathcal{X} = \{2,3,4,5\}$

Continuous r.v.:

$X \in \mathcal{X}$

Uncountable outcome space $\quad \mathcal{X} = \mathbb{R}$

Calculating Probabilities:

If X is discrete: use pmf $p : \mathcal{X} \to [0,1]$

$$\mathbb{P}(X \in E) \overset{\text{def}}{=} \sum_{x \in E} p(x) \quad \text{where } E \subset \mathcal{X}$$

If Y is continuous: use pdf $p : \mathcal{X} \to [0,\infty)$

$$\mathbb{P}(X \in E) \overset{\text{def}}{=} \int_E p(x)\,dx \quad \text{where } E \subset \mathcal{X}$$

Commonly used discrete and continuous distributions on formula sheet.

Ex: Bernoulli, Normal, Laplace

Multivariate Probability:

$Z = (X, Y) \in \mathcal{X} \times \mathcal{Y}$ r.v.

On formula sheet $E_X \subset \mathcal{X}, E_Y \subset Y$

Joint: $\mathbb{P}(X \in E_X, Y \in E_Y)$

Marginal: $\mathbb{P}_X(X \in E_X)$, $\mathbb{P}_Y(Y \in E_Y)$

Conditional: $\mathbb{P}_{X|Y}(X \in E_X | Y = y)$, $\mathbb{P}_{Y|X}(Y \in E_Y | X = x)$

Product Rule: $p(x, y) = \underbrace{p(y|x)}_{P_{Y|X}(y|x)} p(x) = p(x|y) p(y)$

Independence:

$X = (X_1, \dots, X_n)$ $X_1, \dots, X_n$ are independent if:

$$p(x_1, \dots, x_n) = p_{X_1}(x_1) \, p_{X_2}(x_2) \cdots p_{X_n}(x_n)$$

Functions of r.v.:

A function of a r.v. is a r.v.

If $X \in \mathcal{X}$ is a r.v. then:

$f: \mathcal{X} \to \mathcal{Y}$, $Y = f(X) = X^2$ is a r.v. with

$\bar{X} = g(X_1, \ldots, X_n) = \frac{X_1 + \ldots + X_n}{n}$ outcome space $\mathcal{Y}$

Expectation and Variance:

$Z = (X, Y) \in \mathcal{X} \times \mathcal{Y}$ r.v.

On formula sheet with useful properties

Univariate: $\mathbb{E}[X]$

Function: $\mathbb{E}[f(X)]$

Multivariate: $\mathbb{E}[f(Z)] = \mathbb{E}[f(X, Y)]$

Conditional: $\mathbb{E}[f(Y) \mid X = x]$

Variance: $\mathrm{Var}[X] = \mathbb{E}[(X - \mathbb{E}[X])^2]$

Supervised Learning

Dataset:

$D = ((\vec{X}_1, Y_1), \dots, (\vec{X}_n, Y_n)) \in (\mathcal{X} \times \mathcal{Y})^n$ where

$(\vec{X}_i, Y_i) \sim \mathbb{P}_{\vec{X},Y}$ and independent for all $i \in \{1, \dots, n\}$

$\mathcal{X} = \mathbb{R}^d$ feature vector (always $\mathbb{R}^d$)

$\mathcal{Y}$ Label or target

Learner:

$\mathcal{A} : (\mathcal{X} \times \mathcal{Y})^n \to \{ f \mid f : \mathcal{X} \to \mathcal{Y} \}$

Predictor:

$f : \mathcal{X} \to \mathcal{Y}$

loss function

$\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ $\qquad \ell(f(\vec{X}), Y)$

Expected loss

$L(f) = \mathbb{E}[\ell(f(\vec{X}), Y)]$ $\qquad (\vec{X}, Y) \sim \mathbb{P}_{\vec{X},Y}$

Objective:

Define $\mathcal{A}$ such that $\mathbb{E}[L(\mathcal{A}(D))]$ is small

Regression:

If $\mathcal{Y}$ has a notion of order

Usually $\mathbb{R}$ or an interval $[a, b]$

Use squared or absolute loss

Classification:

$\mathcal{Y}$ does not have a notion of order

Usually finite set like $\{cat, dog, bird\}$

Use 0-1 loss

Learner: ERM input dataset D

Estimation: Use D to estimate $L(f)$ for all $f \in \tilde{\mathcal{F}}$

call estimate $\hat{L}(f)$

Optimization: pick $\hat{f}$ as the $f \in \mathcal{F}$ that minimizes $\hat{L}(f)$

Estimation:

$X \in \mathcal{X}$ is a r.v. with distribution $\mathbb{P}$

Want to estimate $\mathbb{E}[X] = \mu$

Use n i.i.d. samples from $\mathbb{P}$

$(X_1, \dots, X_n)$

Sample mean estimate:

$$\hat{\mu} = \bar{X} = g(X_1, \dots, X_n) = \frac{X_1 + \cdots X_n}{n} = \frac{1}{n} \sum_{i=1}^{n} X_i$$

Expectation and Variance:

$$\mathbb{E}[\bar{X}] = \mathbb{E}[X] = \mathbb{E}[X_1] = \cdots = \mathbb{E}[X_n]$$

$$\mathrm{Var}[\bar{X}] = \frac{\mathrm{Var}[X]}{n} = \frac{\mathrm{Var}[X_1]}{n} = \cdots = \frac{\mathrm{Var}[X_n]}{n}$$

Optimization

$$\min_{w \in W} g(w) = g(w^*) \quad \text{where} \quad w^* = \underset{w \in W}{\operatorname{argmin}}\, g(w)$$

$$w^* = \underset{w \in W}{\operatorname{argmin}}\, g(w) = \underset{w \in W}{\operatorname{argmax}} -g(w)$$

$$\min_{w \in W} g(w) = -\left(\max_{w \in W} -g(w)\right)$$

Solving Optimization problems:

- If W discrete just compare $g(w)$ for all $w \in W$
- If W continuous can use derivatives sometimes

Continuous Optimization:

If $g(w)$ is convex and twice differentiable then:

Cases: 1. If $W = \mathbb{R}$ then w^* is the solution to $g'(w) = 0$

2. If $\mathcal{W} = [a,b]$ then w^* is the solution to $g'(w) = 0$ if this solution is in $[a,b]$. Otherwise, w^* is a or b

Twice doffentiable: The second derivative of $g(w)$ written $g''(w)$ exists for all $w \in \mathcal{W}$

Convex: $g(w)$ is convex if $g''(w) \geq 0$ for all $w \in \mathcal{W}$

"Usually $g(w)$ is bowl shaped"

Multidimensional Minimization

If $\mathcal{W} = \mathbb{R}^d$ for $d > 1$, and $g(\vec{w})$ is convex

Note: it is more complicated to check if $g(\vec{w})$ is convex if $d > 1$. So, I will just tell you

Then we calculate

$$\vec{w}^* = (w_1^*, \ldots, w_d^*)^T = \underset{\vec{w} \in \mathcal{W}}{\operatorname{argmin}} \, g(\vec{w})$$

by setting w_j^* as the solution to

$$\frac{\partial g}{\partial w_j}(w_j) = 0 \quad \text{for all } j \in \{1, \ldots, d\}$$

$$g(\vec{w}^*) = \min_{\vec{w} \in W} g(\vec{w})$$

Linear Regression (Closed form)

$\mathcal{X} = \mathbb{R}^{d+1}$, $\mathcal{Y} = \mathbb{R}$ regression

$\mathcal{F} \subset \{f \mid f : \mathbb{R}^{d+1} \to \mathbb{R}\}$

$D = ((\vec{x}_1, y_1), \dots, (\vec{x}_n, y_n))$

$\hat{L}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(\vec{x}_i), y_i)$ an estimate of $L(f)$ for all $f \in \mathcal{F}$

we want $\boxed{\hat{f} = \underset{f \in \mathcal{F}}{\operatorname{argmin}}\, \hat{L}(f)}$

pick $\mathcal{F} = \{f \mid f : \underset{\mathbb{R}^{d+1}}{\overset{\shortparallel}{\mathcal{X}}} \to \underset{\mathbb{R}}{\overset{\shortparallel}{\mathcal{Y}}}$ and $f(\vec{x}) = \vec{x}^T \vec{w}$ where $\vec{w} \in \mathbb{R}^{d+1}\}$

Learner

$\mathcal{A}(D) = \hat{f} \in \mathcal{F}$

where $\hat{f}(\vec{x}) = \vec{x}^T \hat{\vec{w}}$ and $\hat{\vec{w}} = \underbrace{A^{-1} \vec{b}}_{\text{Depends on } D}$

Algorithm: Closed form linear regression Learner

input: $D = ((\vec{x}_1, y_1), \dots, (\vec{x}_n, y_n))$

$A \leftarrow \sum_{i=1}^{n} \vec{x}_i \vec{x}_i^T$

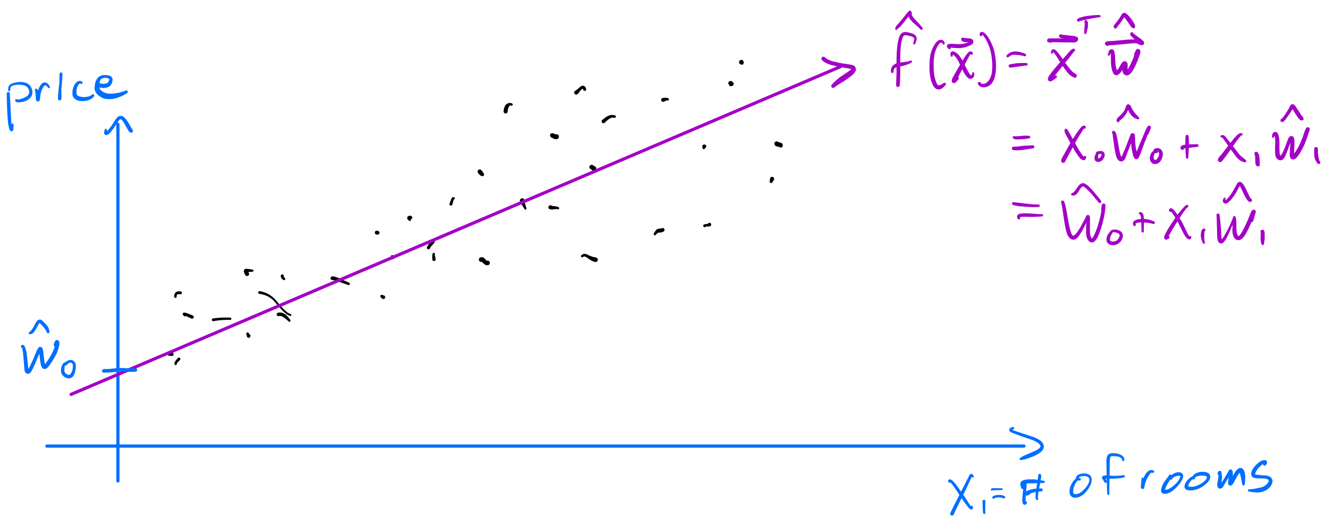
$b \leftarrow \sum_{i=1}^{n} \vec{x}_i y_i$

$\hat{\vec{w}} \leftarrow A^{-1} b$

return $\hat{f}(\vec{x}) = \vec{x}^T \hat{\vec{w}}$

$\underline{\underline{Ex}}$: $d=1$, $\mathcal{X} = \mathbb{R}^{1+1} = \mathbb{R}^2$, $\mathcal{Y} = \mathbb{R}$, $\hat{\vec{w}} = (\hat{w}_0, \hat{w}_1)^T$

$D = ((\vec{x}_1, y_1), \dots, (\vec{x}_n, y_1))$

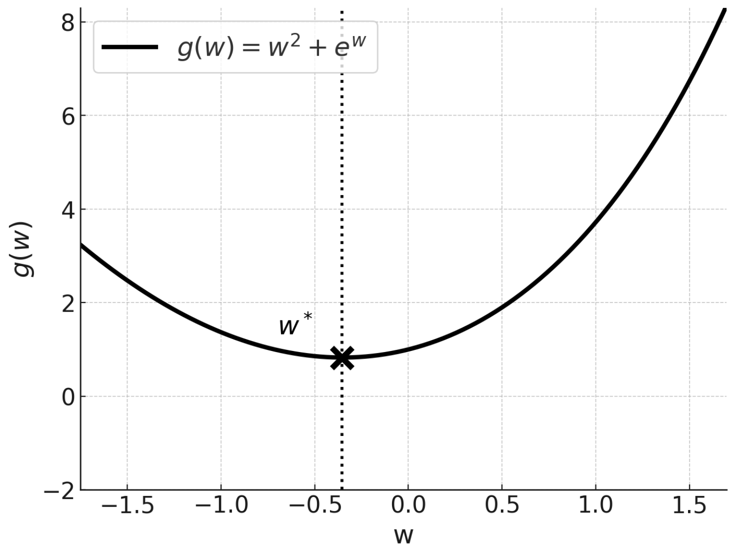


Gradient Descent

Ex: $g(w) = w^2 + e^w$, $g'(w) = 2w + e^w$, $g''(w) = 2 + e^w \geq 0$

Convex

$w \in \mathbb{R} = \mathcal{W}$



$$g'(w) = 2w + e^w = 0$$

$$\underbrace{2w = -e^w}$$

No closed form solution

Second-order Gradient Descent:

$$g(w) \approx g_{w^{(0)}}(w) = g(w^{(0)}) + g'(w^{(0)})(w - w^{(0)}) + \frac{g''(w^{(0)})}{2}(w - w^{(0)})^2$$

$$g'_{w^{(0)}}(w) = g'(w^{(0)}) + g''(w^{(0)})(w - w^{(0)}) = 0$$

$$\Rightarrow \quad w = w^{(0)} - \frac{g'(w^{(0)})}{g''(w^{(0)})}$$

General: $w^{(t+1)} = w^{(t)} - \dfrac{g'(w^{(t)})}{g''(w^{(t)})}$

$w^{(t+1)}$ approaches $w^* = \underset{w \in \mathcal{W}}{\operatorname{argmin}}\ g(w)$

as $t \to \infty$

First-order Gradient Descent

$$w^{(t+1)} = w^{(t)} - \eta^{(t)} g'(w^{(t)})$$

Multivariate Gradient Descent

Objective:

$$\vec{w}^* = (w_1^*, \ldots, w_d^*)^T = \underset{\vec{w} \in \mathcal{W}}{\operatorname{argmin}}\ g(\vec{w})$$

gradient descent update rule:

$$\vec{w}^{(t+1)} = \vec{w}^{(t)} - \eta^{(t)} \nabla g(\vec{w}^{(t)})$$

where $\nabla g(\vec{w}^{(t)}) = \left(\frac{\partial g}{\partial w_1}(\vec{w}^{(t)}), \ldots, \frac{\partial g}{\partial w_d}(\vec{w}^{(t)}) \right)^T \in \mathbb{R}^d$

Selecting the step size

1. constant: $\eta^{(t)} = \eta \in (0, \infty)$

2. Inverse decaying: $\eta^{(t)} = \dfrac{\eta}{1+\lambda t}$ where $\eta, \lambda \in (0,\infty)$

3. Exponential decaying: $\eta^{(t)} = \eta \dfrac{1}{e^{\lambda t}}$ where $\eta, \lambda \in (0,\infty)$

4. Normalized gradient: $\eta^{(t)} = \dfrac{\eta}{\varepsilon + \|\nabla g(\vec{w}^{(t)})\|}$ where $\eta, \varepsilon \in (0,\infty)$

ε is small (ex: $\varepsilon = 10^{-8}$)

$$\|\nabla g(\vec{w}^{(t)})\| = \sqrt{\sum_{j=1}^{d} \left(\frac{\partial g}{\partial w_j}(\vec{w}^{(t)})\right)^2}$$

(Batch) Gradient Descent (BGD)

$$\vec{w}^{(t+1)} = \vec{w}^{(t)} - \eta^{(t)} \nabla \hat{L}(\vec{w}^{(t)})$$

$$\nabla \hat{L}(\vec{w}^{(t)}) = \frac{2}{n} \sum_{i=1}^{n} (\vec{x}_i^T \vec{w} - y_i) \vec{x}_i$$

Algorithm: BGD Linear Regression Learner
(with a constant size)

---

input: $D = ((\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n))$, η, T number of "epochs"

$\vec{w} \leftarrow$ random vector in $\mathbb{R}^{d+1}$

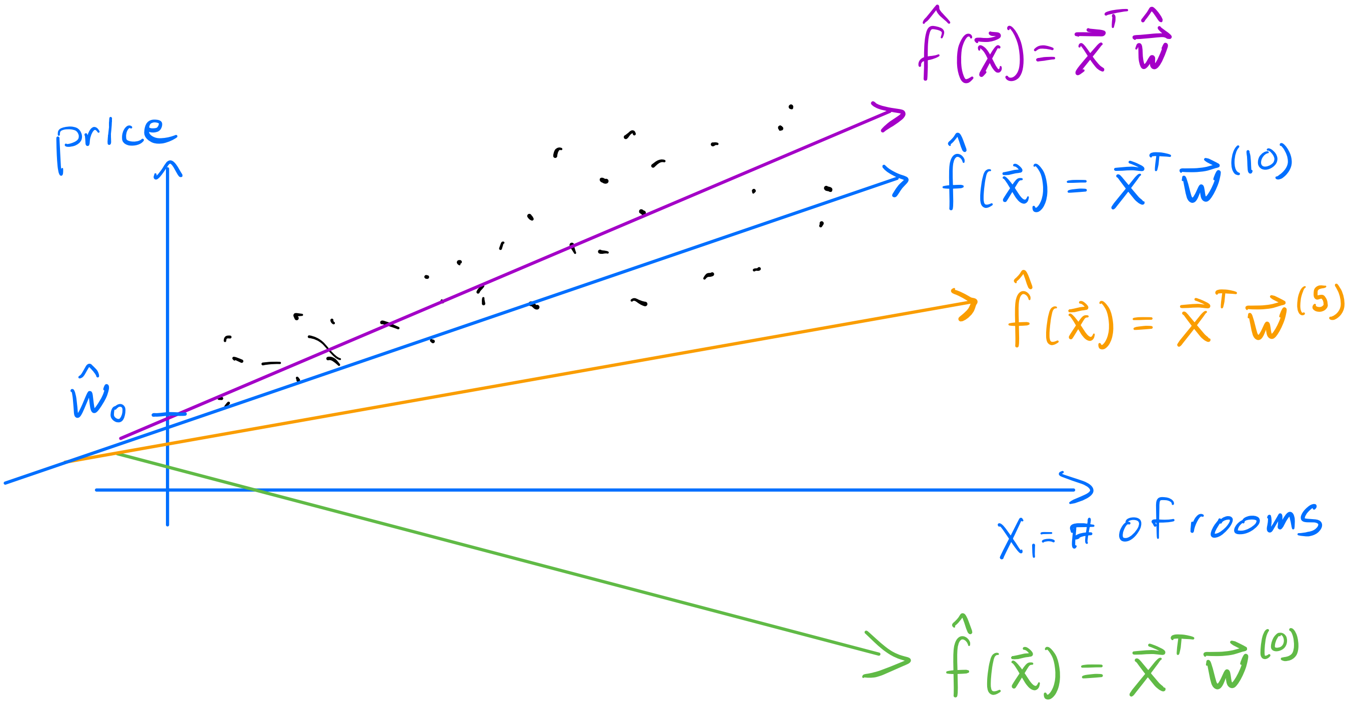
for $t = 1, \ldots, T$

$\quad \nabla \hat{L}(\vec{w}) \leftarrow \frac{2}{n} \sum_{i=1}^{n} (\vec{x}_i^T \vec{w} - y_i) \vec{x}_i$

$\quad \vec{w} = \vec{w} - \eta \nabla \hat{L}(\vec{w})$

return $\hat{f}(\vec{x}) = \vec{x}^T \vec{w}$

Ex:



$\hat{f}(\vec{x}) = \vec{x}^T \hat{\vec{w}}$
price
$\hat{f}(\vec{x}) = \vec{x}^T \vec{w}^{(10)}$
$\hat{f}(\vec{x}) = \vec{x}^T \vec{w}^{(5)}$
$\hat{w}_0$
$x_1 = \#$ of rooms
$\hat{f}(\vec{x}) = \vec{x}^T \vec{w}^{(0)}$

Computation

Closed form: $O(nd^2 + d^3)$

BGD: $O(ndT)$

if $T < d$: $O(ndT) < O(nd^2) \leq O(nd^2 + d^3)$

BGD is more computationally efficient if $T < d$

Mini-Batch Gradient Descent (MBGD)

$$D = \Big((\vec{x}_1, y_1), \dots, (\vec{x}_b, y_b),$$
$$(\vec{x}_{b+1}, y_{b+1}), \dots, (\vec{x}_{2b}, \dots, y_{2b}),$$
$$\vdots$$
$$(\vec{x}_{(M-1)b+1}, y_{(M-1)b+1}), \dots, (\vec{x}_{Mb}, y_{Mb}) \Big)$$

b: mini-batch size

$M = \text{floor}\left(\frac{n}{b}\right)$: number of mini batches we don't use $n - Mb \leq b$ datapoints

$$\hat{L}_m(\vec{w}) = \frac{1}{b} \sum_{i=(m-1)b+1}^{mb} (\vec{x}_i^T \vec{w} - y_i)^2 \qquad m \in \{1, \dots, M\}$$

Algorithm: MBGD Linear Regression Learner (with a constant size)

---

input: $D = ((\vec{x}_1, y_1), \dots, (\vec{x}_n, y_n))$, η, T, b

$\vec{w} \leftarrow$ random vector in $\mathbb{R}^{d+1}$

$M \leftarrow \text{floor}\left(\frac{n}{b}\right)$

for $t = 1, \dots, T$

  Randomly shuffle D

  for $m = 1, \dots, M$

    $\nabla \hat{L}(\vec{w}) \leftarrow \frac{2}{b} \sum_{i=(m-1)b+1}^{mb} (\vec{x}_i^T \vec{w} - y_i) \vec{x}_i$

    $\vec{w} = \vec{w} - \eta \nabla \hat{L}(\vec{w})$

return $\hat{f}(\vec{x}) = \vec{x}^T \vec{w}$

Advantage is MT is now the number of gradient steps

Setting $b = n \Rightarrow M = 1$ gives back BGD

$b = 1 \Rightarrow M = n$ gives "Stochastic GD (SGD)"

Polynomial Regression

$\phi_p : \mathcal{X} \to \mathcal{Z}$ is a degree p polynomial "feature map"

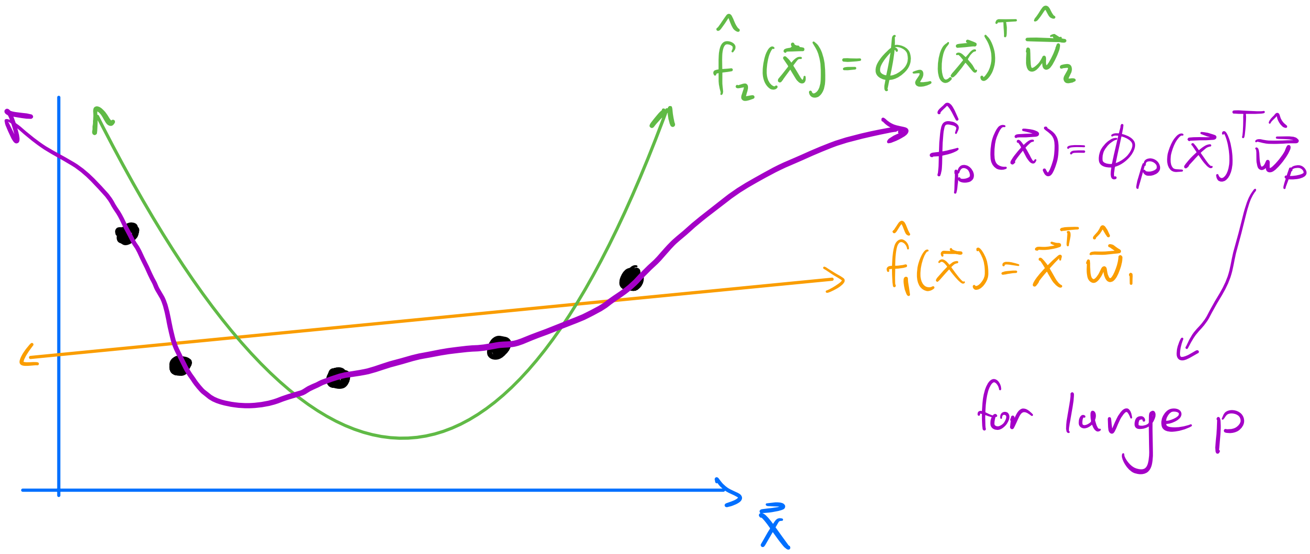
For $\mathcal{X} = \mathbb{R}^{d+1}$, $\mathcal{Z} = \mathbb{R}^{\bar{p}}$ where $\bar{p} = \binom{(d+1)+p-1}{p} = \binom{d+p}{p}$

$$\mathcal{F}_p = \{ f \mid f : \mathbb{R}^{d+1} \to \mathbb{R} \text{ and } f(\vec{x}) = \phi_p(\vec{x})^T \vec{w}, \vec{w} \in \mathbb{R}^{\bar{p}} \}$$

$$\mathcal{F}_1 \subset \mathcal{F}_2 \subset \dots \subset \mathcal{F}_p$$

$$\hat{L}(\hat{f}_1) \geq \hat{L}(\hat{f}_2) \geq \dots \geq \hat{L}(\hat{f}_p) \approx 0$$

Ex:



Evaluating Predictors/Models

Objective (formal):

Define a learner $\mathcal{A}: (\mathcal{X} \times \mathcal{Y})^n \to \{f \mid f: \mathcal{X} \to \mathcal{Y}\}$

such that $\mathbb{E}[L(\mathcal{A}(D))]$ is small

Defining $\mathcal{A}(D)$: Empirical Risk Minimization (ERM)

Estimation:

Use D to estimate $L(f)$ for all $f \in \mathcal{F} \subset \{f \mid f: \mathcal{X} \to \mathcal{Y}\}$

call the estimate $\hat{L}(f)$

Optimization:

pick $\hat{f}$ to be the $f \in \underbrace{\mathcal{F}}_{\text{Function class}}$ that minimizes $\hat{L}(f)$

When should we expect ERM to work well?

- When $\mathcal{F}$ contains an f that can make $L(f)$ small
- When $\hat{L}(\hat{f})$ is a good estimate of $L(\hat{f})$

If $\mathcal{A}(D) = \hat{f}_D$ depends on D then

$$\mathbb{E}[\hat{L}(\hat{f}_D)] \neq \mathbb{E}[L(\hat{f}_D)]$$

$\ell(\hat{f}_D(\vec{X}_i), Y_i)$ are not i.i.d.

$\hat{f}_D$ depends on $(\vec{X}_1, Y_1), \ldots, (\vec{X}_n, Y_n)$!

It can be shown that:

$$\mathbb{E}[L(\hat{f}_D)] - \mathbb{E}[\hat{L}(\hat{f}_D)]$$

- increases as $\mathcal{F}$ gets more complex
- decreases as n increases

Decomposing $\mathbb{E}[L(\mathcal{A}(D))]$

Let $\mathcal{A}(D) = \hat{f}_D$

$\mathcal{F}$ any function class

$$f_{Bayes} = \underset{f \in \{f \mid f: \mathcal{X} \to \mathcal{Y}\}}{\operatorname{argmin}} L(f)$$

$$f^* = \underset{f \in \mathcal{F}}{\operatorname{argmin}} L(f)$$

$$\hat{f}_D = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \hat{L}(f)$$

$$\mathbb{E}[L(\hat{f}_D)] = \underbrace{\mathbb{E}[L(\hat{f}_D)] - L(f^*)}_{\text{Estimation Error (EE)}} + \underbrace{L(f^*) - L(f_{Bayes})}_{\text{Approximation Error (AE)}} + \underbrace{L(f_{Bayes})}_{\text{Irreducible Error (IE)}}$$

Irreducible Error: Due to inherent noise in labels

- Decreases if you gather more/better feature info
- Usually not possible to do "irreducible"

Approximation Error: Due to a small $\mathcal{F}$

- Decreases if you make $\mathcal{F}$ larger

Estimation Error: Due to random dataset D

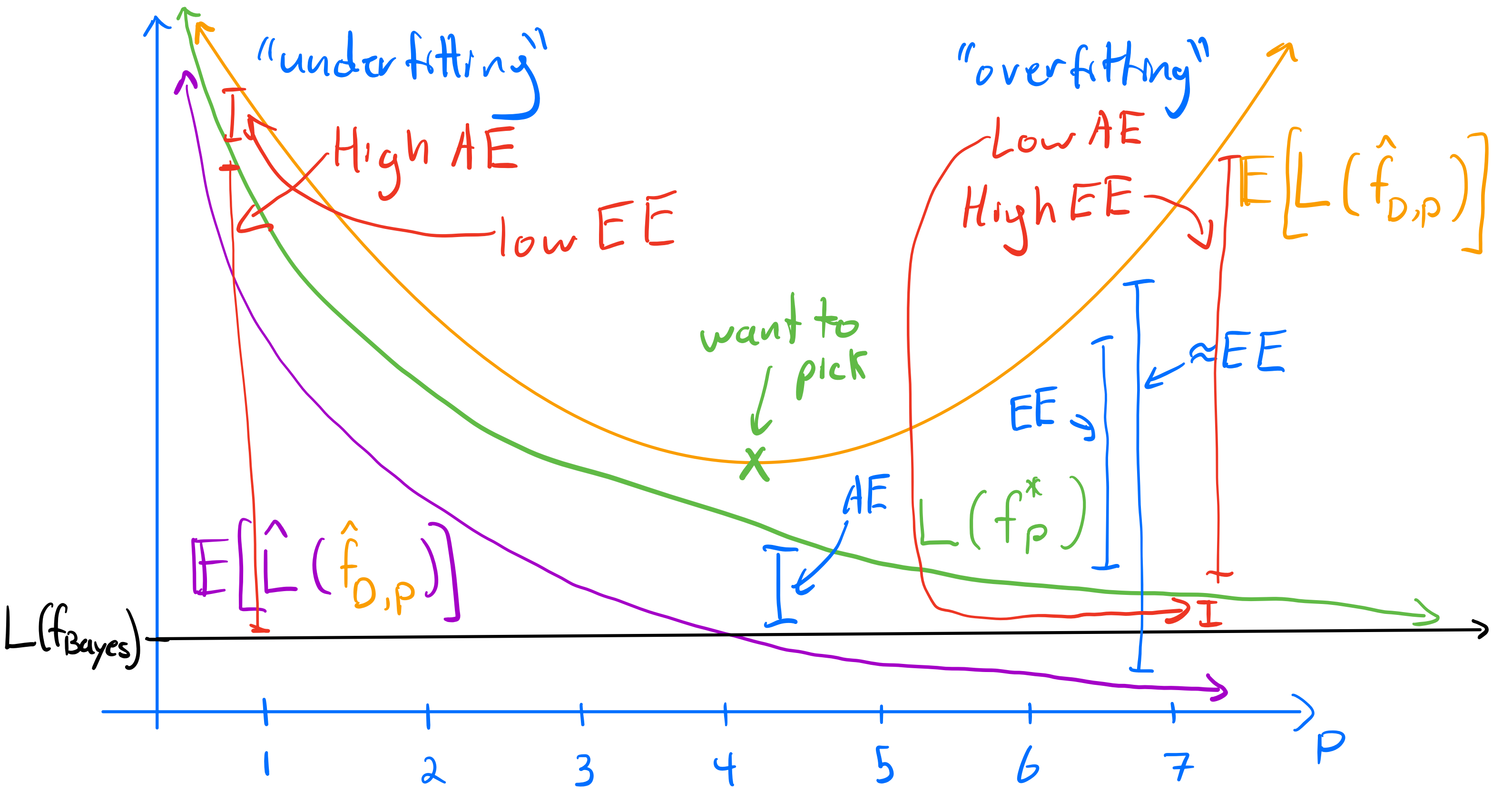
- Decreases if you increase n
- Increases if you increase $\mathcal{F}$

$$\mathcal{A}(D) = \hat{f}_{D,p} = \underset{f \in \tilde{\mathcal{F}}_p}{\arg\min}\ \hat{L}(f) \qquad \tilde{\mathcal{F}}_1 \subset \cdots \subset \tilde{\mathcal{F}}_p$$

$$\mathbb{E}[L(\hat{f}_D)] - \mathbb{E}[\hat{L}(\hat{f}_D)]$$

$$\approx$$

$$\mathbb{E}[L(\hat{f}_D)] = \underbrace{\mathbb{E}[L(\hat{f}_D)] - L(f^*)}_{\substack{\text{Estimation Error} \\ \text{(EE)}}} + \underbrace{L(f^*) - L(f_{Bayes})}_{\substack{\text{Approximation Error} \\ \text{(AE)}}} + \underbrace{L(f_{Bayes})}_{\substack{\text{Irreducible} \\ \text{Error (IE)}}}$$



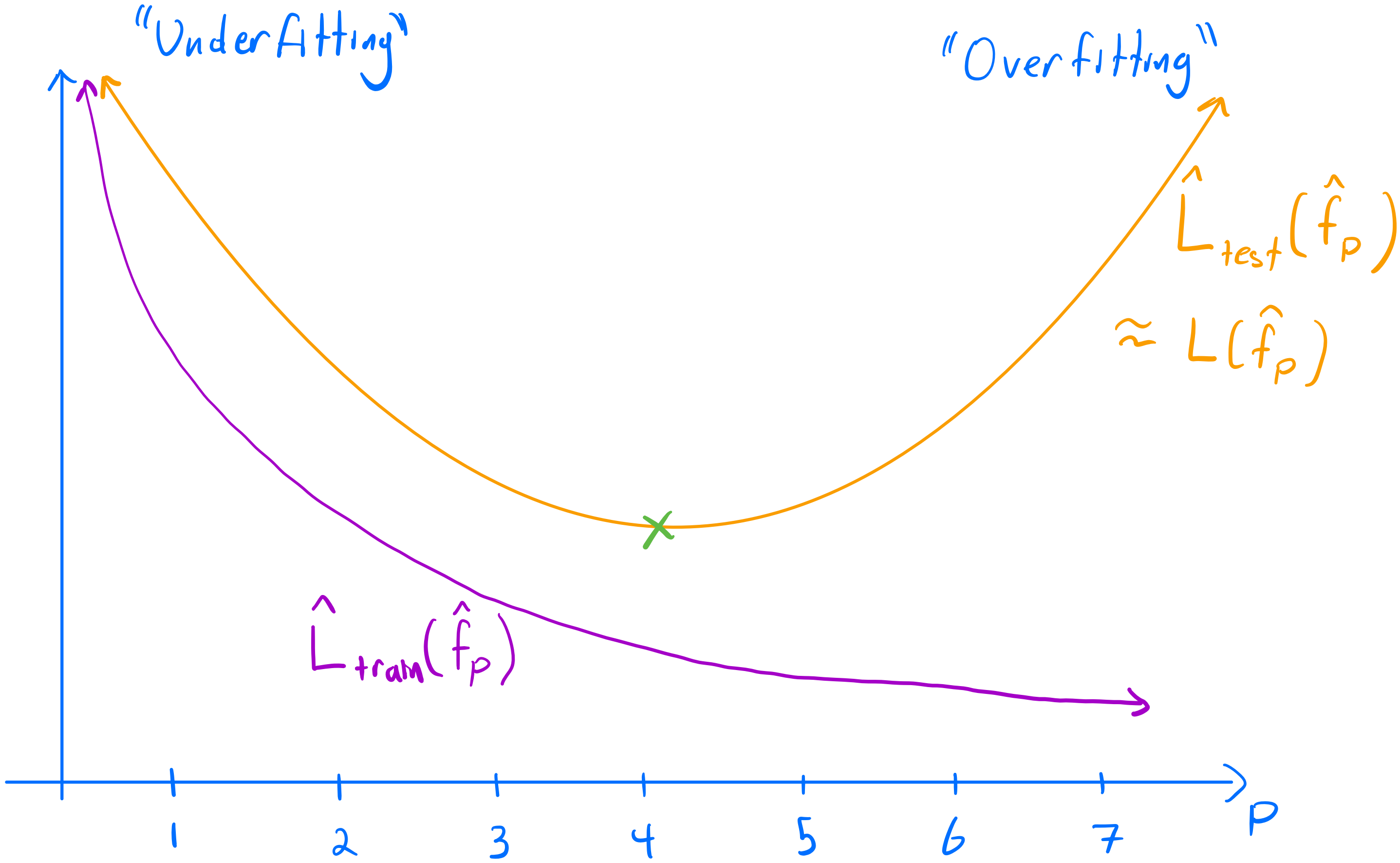
Can't calculate L, so use train, test split

$$D_{train} = \left((\vec{x}_1, y_1), \ldots, (\vec{x}_{n-m}, y_{n-m}) \right)$$

$$D_{test} = \left((\vec{x}_{n-m+1}, y_{n-m+1}), \ldots, (\vec{x}_n, y_n) \right)$$

$$|D_{train}| = n-m \quad , \quad |D_{test}| = m$$

$$\mathcal{A}(D_{train}) = \hat{f}_p = \underset{f \in \tilde{\mathcal{F}}_p}{\operatorname{argmin}} \hat{L}_{train}(f) \qquad \mathcal{F}_1 \subset \cdots \subset \mathcal{F}_p$$



Bias-Variance Tradeoff

$\mathbb{E}[L(\hat{f}_D)]$ Let ℓ be squared loss

if $\bar{f} = f^*$

$$= \mathbb{E}\Big[\overbrace{\underbrace{\mathbb{E}\big[(\hat{f}_D(\vec{x}) - \bar{f}(\vec{x}))^2 \mid \vec{x}\big]}_{}}^{=EE}\Big] + \mathbb{E}\Big[\overbrace{\underbrace{(\bar{f}(\vec{x}) - f_{Bayes}(\vec{x}))^2}_{}}^{=AE}\Big] + \underbrace{L(f_{Bayes})}_{}$$

Variance $= \mathrm{Var}[\hat{f}_D(\vec{x}) \mid \vec{x}]$ Bias IE

where $\bar{f}(\vec{x}) = \mathbb{E}[\hat{f}_D(\vec{x}) \mid \vec{x}]$ "expected predictor"

Effects of changing $\mathcal{F}$, n on Bias, Variance

Bias $\downarrow$ if $\mathcal{F} \uparrow$

Variance $\downarrow$ if $n \uparrow$

$\uparrow$ if $\mathcal{F} \uparrow$

Regularization

let $\vec{w} = (w_0, w_1, \dots, w_{\bar{p}-1})^T \in \mathbb{R}^{\bar{p}}$

Observation: large values of $|w_0|, |w_1|, \dots, |w_{\bar{p}-1}|$ leads to more complex $f_p(\vec{x}) = \phi_p(\vec{x})^T \vec{w}$

Regularization: penalize large weights

If $f_p \in \mathcal{F}_p$:

$$\underbrace{\hat{L}_\lambda(f_p)}_{\text{Regularized estimated loss}} = \overbrace{\frac{1}{n}\sum_{i=1}^{n} \ell(f_p(\vec{x}_i), y_i)}^{=\hat{L}(f_p)} + \underbrace{\frac{\lambda}{n}\sum_{j=1}^{\bar{p}-1} w_j^2}_{\text{"Regularizer"}}$$

Regularization parameter $\lambda \in [0, \infty)$

$j=0$ not included

Bias vs. Variance

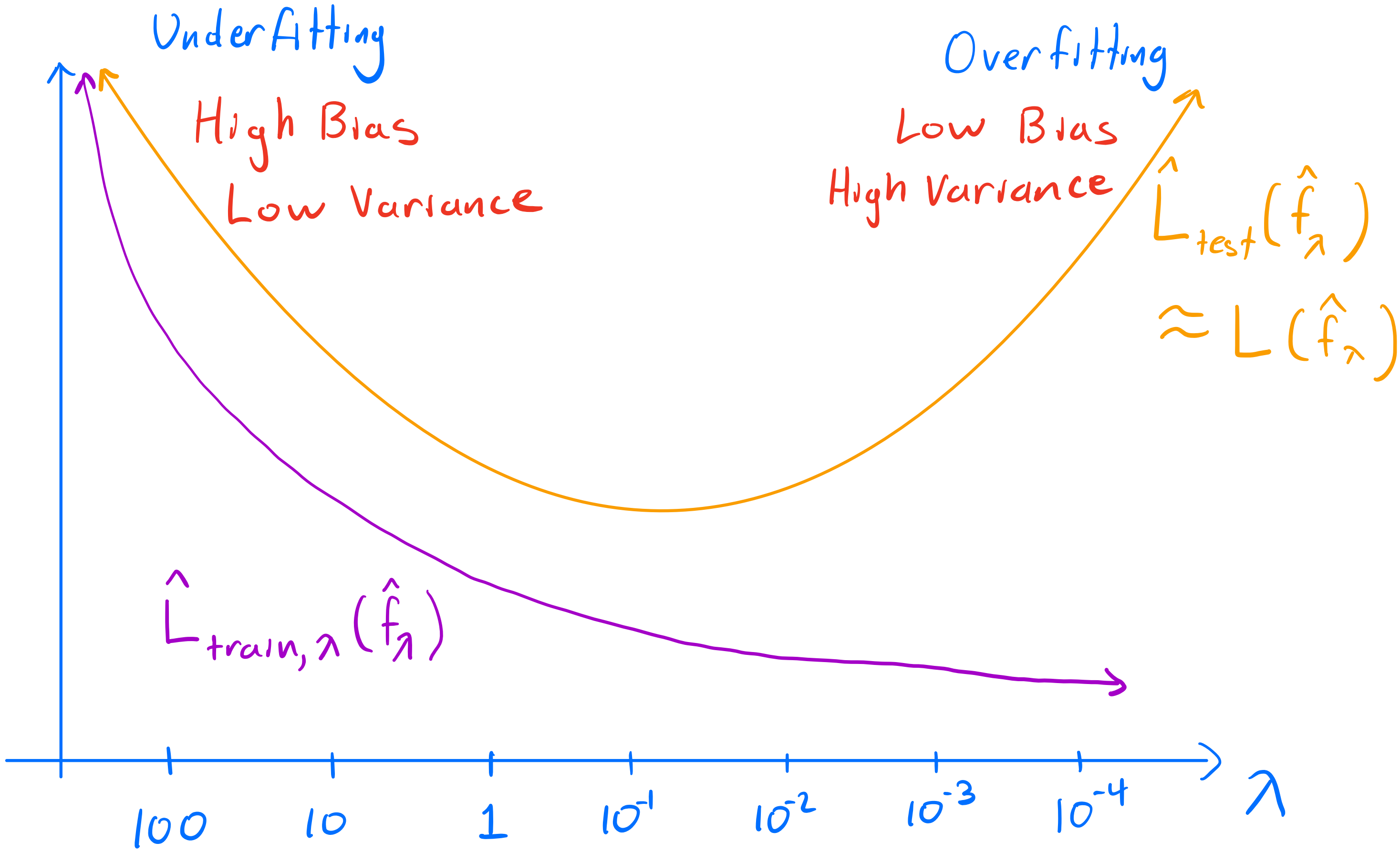
Bias: $(\bar{f}_\lambda(\vec{x}) - f_{Bayes}(\vec{x}))^2$

- Decreases if λ decreases

Variance: $E[(\hat{f}_{D,\lambda}(\vec{x}) - \bar{f}_\lambda(\vec{x}))^2 | \vec{x}]$

- Increases if λ decreases
- Decreases if n increases

$$\hat{f}_\lambda = \underset{f \in \mathcal{F}_{10}}{\operatorname{argmin}} \hat{L}_\lambda(f)$$



Minimizing $\hat{L}_\lambda(f)$

Objective:

$$\hat{\vec{w}}_\lambda = \underset{\vec{w} \in \mathbb{R}^{d+1}}{\operatorname{argmin}} \hat{L}_\lambda(\vec{w})$$ using squared loss, $\mathcal{F}_1$

where $$\hat{L}_\lambda(\vec{w}) = \frac{1}{n}\sum_{i=1}^{n}(\vec{x}_i^T\vec{w} - y_i)^2 + \frac{\lambda}{n}\sum_{j=1}^{d} w_j^2$$

BGD:

$$\vec{w}^{(t+1)} = \vec{w}^{(t)} - \eta^{(t)} \nabla \hat{L}_\lambda(\vec{w}^{(t)})$$

Summary of Errors (Sec 7.6)

gather more informative features

$$\mathbb{E}[L(\hat{f}_D)] = \underbrace{\mathbb{E}[L(\hat{f}_D)] - L(f^*)}_{\text{Estimation Error (EE)}} + \underbrace{L(f^*) - L(f_{\text{Bayes}})}_{\text{Approximation Error (AE)}} + \underbrace{L(f_{\text{Bayes}})}_{\text{Irreducible Error (IE)}}$$

$$\mathbb{E}[L(\hat{f}_D)] = \mathbb{E}\left[\underbrace{\mathbb{E}[(\hat{f}_D(\boldsymbol{X}) - \bar{f}(\boldsymbol{X}))^2|\boldsymbol{X}]}_{\text{Variance}}\right] + \mathbb{E}\left[\underbrace{(\bar{f}(\boldsymbol{X}) - f_{\text{Bayes}}(\boldsymbol{X}))^2}_{\text{Bias}}\right] + \underbrace{L(f_{\text{Bayes}})}_{\text{Irreducible Error}}$$

$$\bar{f}(\boldsymbol{X}) = \mathbb{E}[\hat{f}_D(\boldsymbol{X})|\boldsymbol{X}].$$

$$\hat{f}_D = \underset{f\in\mathcal{F}}{\operatorname{argmin}}\, \hat{L}(f).$$

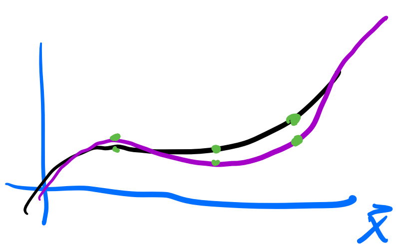
$$f_{\text{Bayes}} = \underset{f\in\{f|f:\mathcal{X}\to\mathcal{Y}\}}{\operatorname{argmin}}\, L(f).$$

$$f^* = \underset{f\in\mathcal{F}}{\operatorname{argmin}}\, L(f).$$

	Always good **Increasing n**	$\mathcal{F}_p$ **Increasing p with $\lambda = 0$**	**Decreasing λ with $p = 10$**
EE	Decreases	Increases	Unclear
AE	Unchanged	Decreases until $f_{\text{Bayes}} \in \mathcal{F}_p$	Unchanged
Variance	Decreases	Increases	~~Decreases~~ Increases
Bias	Unchanged	Decreases until $f_{\text{Bayes}} \in \mathcal{F}_p$	~~Increases~~ Decreases

Table 7.1: How changing n, p, and λ affects the EE, AE, variance, and bias. The terms "Increasing" and "Decreasing" also encompass cases where values remain constant.

$$\underbrace{L(f^*) - L(f_{\text{Bayes}})}_{\text{Approximation Error (AE)}}$$ Small $\xRightarrow{\text{Most of the time}}$ $f^*(\vec{x}) \approx f_{\text{Bayes}}(\vec{x})$ for all $\vec{x} \in \mathcal{X}$



MLE

$$f_{Bayes} = \underset{f \in \{f \mid f: \mathcal{X} \to \mathcal{Y}\}}{\arg\min} L(f)$$

assume squared loss and Regression

$$f_{Bayes}(\vec{x}) = \mathbb{E}[Y \mid \vec{X} = \vec{x}]$$

$$= \int_{\mathcal{Y}} y \, p_{Y|\vec{X}}(y \mid \vec{x}) \, dy$$

Use $\mathcal{D}$ to estimate $p_{Y|\vec{X}}$

MLE Basics

$D = (Z_1, \dots, Z_n)$ and Z_i are i.i.d. with p_Z

Assume p_Z is based on some parameter w^*

You are give a fixed data $D = (z_1, \dots, z_n)$

$$w_{MLE} = \underset{w \in \mathcal{W}}{\arg\max} \overbrace{p(D \mid w)}^{\text{likelihood}}$$

$$= \underset{w \in \mathcal{W}}{\arg\min} \overbrace{-\log(p(D \mid w))}^{\text{negative log-likelihood}}$$

$$= \underset{w \in \mathcal{W}}{\arg\min} -\log\left(\prod_{i=1}^{n} p(z_i | w)\right)$$

$$= \underset{w \in \mathcal{W}}{\arg\min} -\sum_{i=1}^{n} \log\left(p(z_i | w)\right)$$

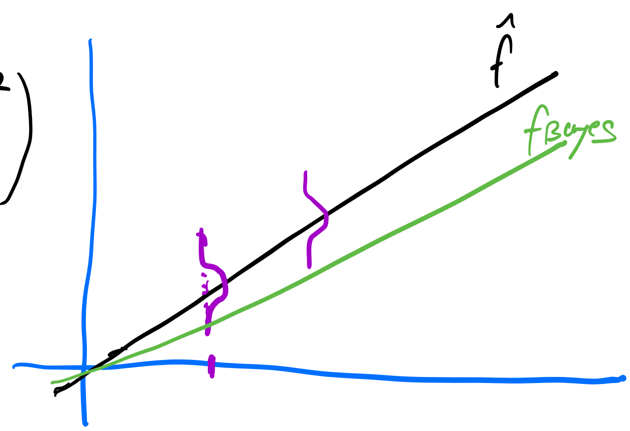
Estimating $P_{Y|\vec{X}}$

$D = ((\vec{X}_1, Y_1), \ldots, (\vec{X}_n, Y_n)) \in (\mathcal{X} \times \mathcal{Y})^n$, $\mathbb{P}_D$, p_D

$(\vec{X}_i, Y_i)$ are i.i.d with $\mathbb{P}_{\vec{X}, Y}$ and $p_{\vec{X}, Y}$

Assume $Y_i | \vec{X}_i = \vec{x}_i \sim \mathcal{N}(\vec{x}_i^T \vec{w}^*, 1)$

$$P_{Y|\vec{X}=\vec{x}}(y|\vec{x}) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(y - \vec{x}^T \vec{w}^*)^2}{2}\right)$$



$$\vec{w}_{MLE} = \underset{\vec{w} \in \mathbb{R}^{d+1}}{\arg\min} \sum_{i=1}^{n} \frac{(Y_i - \vec{X}_i^T \vec{w})^2}{2}$$

$$= \underset{\vec{w} \in \mathbb{R}^{d+1}}{\arg\min} \hat{L}(\vec{w})$$

$$f_{Bayes}(\vec{x}) \approx \vec{x}^T \vec{w}_{MLE}$$

MAP

If $P_{Y|\vec{X}}$ is a function of some parameter $w \in \mathcal{W}$ then we just need to estimate that parameter

$$\text{MLE: } \underset{w \in \mathcal{W}}{\arg\max}\ \underbrace{p(D|w)} = \prod_{i=1}^{n} p(z_i|w)$$

"find w that maximizes the likelihood of the data"

$$\text{MAP: } \underset{w \in \mathcal{W}}{\arg\max}\ \underbrace{p(w|D)}_{} = \text{"posterior"}$$

"find w that is the most likely given the data"

$$w_{MAP} = \underset{w \in \mathcal{W}}{\arg\max}\ p(w|D)$$

$$= \underset{w \in \mathcal{W}}{\arg\max}\ \frac{p(w,D)}{P(D)}$$

prod rule

$$= \underset{w \in \mathcal{W}}{\arg\max}\ \frac{p(D|w)\,p(w)}{P(D)}$$

$$= \underset{w \in \mathcal{W}}{\arg\max}\ \underbrace{p(D|w)}_{\text{likelihood}}\ \underbrace{p(w)}_{\text{prior}}$$

$$= \underset{w \in \mathcal{W}}{\arg\min} -\log\big(p(D|w)\,p(w)\big)$$

$$= \underset{w \in \mathcal{W}}{\arg\min} \Big[-\log\big(p(D|w)\big) - \log\big(p(w)\big)\Big]$$

if n large then: $w_{MAP} \approx w_{MLE}$

if $p(w) = c$ a constant then: $w_{MAP} \approx w_{MLE}$

if $Var[W]$ is large then: $w_{MAP} \approx w_{MLE}$

small then: $w_{MAP} \approx \mathbb{E}[W]$

Estimating $\vec{w}$ for $P_{Y|\vec{X}}$

$$D = \big((\vec{X}_1, Y_1), \ldots, (\vec{X}_n, Y_n)\big) \in (\mathcal{X} \times \mathcal{Y})^n, \ \mathbb{P}_D, \ p_D$$

$(\vec{X}_i, Y_i)$ are i.i.d with $\mathbb{P}_{\vec{X},Y}$ and $p_{\vec{X},Y}$

Assume $Y_i | \vec{X}_i = \vec{x}_i \sim \mathcal{N}(\vec{x}_i^T \vec{w}^*, 1)$

$$p_{Y|\vec{X}=\vec{x}}(y|\vec{x}) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(y - \vec{x}^T \vec{w}^*)^2}{2}\right)$$

Assume $w_j \sim \mathcal{N}(0, \frac{1}{\lambda})$ are i.i.d. for $j \in \{1, \ldots, d\}$

and $\quad w_0 \sim \mathcal{N}(0, \sigma^2)$ for very large σ

$\approx \text{Uniform}(-a, a)$ for large a

w_0 is independent of w_j for all $j \in \{1, \dots, d\}$

$$\vec{w}_{MAP} = \underset{\vec{w} \in \mathbb{R}^{d+1}}{\operatorname{argmax}} \; p(\vec{w} \mid D)$$

$$= \underset{\vec{w} \in \mathbb{R}^{d+1}}{\operatorname{argmin}} \left[\underbrace{\sum_{i=1}^{n} \frac{(y_i - \vec{x}_i^T \vec{w})^2}{2}}_{= \frac{n}{2} \hat{L}} + \underbrace{\frac{\lambda}{2} \sum_{j=1}^{d} w_j^2}_{\text{almost regularizer}} \right]$$

$$= \underset{\vec{w} \in \mathbb{R}^{d+1}}{\operatorname{argmin}} \left[\hat{L}_\lambda(\vec{w}) \right]$$

$$f_{Bayes}(\vec{x}) \approx \vec{x}^T \vec{w}_{MAP}$$

Lasso regression

Assume $W_j \sim \text{Laplace}(0, 1/\lambda)$ are i.i.d for $j \in \{1, \dots, d\}$

and $W_0 \sim \text{Laplace}(0, b)$ for very large b

$\simeq \text{Uniform}(-a, a)$ for large a

W_0 is independent of W_j for all $j \in \{1, \dots, d\}$

$$\vec{w}_{MAP} = \underset{\vec{w} \in \mathbb{R}^{d+1}}{\arg\max} \; p(\vec{w} \mid D)$$

$$= \underset{\vec{w} \in \mathbb{R}^{d+1}}{\arg\min} \left[\sum_{i=1}^{n} \frac{(y_i - \vec{x}_i^T \vec{w})^2}{2} + \lambda \sum_{j=1}^{d} |w_j| \right]$$

Classification

Labels are unordered (and usually finite)

Loss ℓ is 0-1 loss

$$\hat{L}(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(\vec{x}_i), y_i)$$

Not Continuous

$\Rightarrow$ Hard to optimize

$$f_{Bayes} = \underset{f \in \{f \mid f: \mathcal{X} \to \mathcal{Y}\}}{\operatorname{argmin}} L(f)$$

$$L(f) = \mathbb{E}[\ell(f(\vec{X}), Y)]$$

$$f_{Bayes}(\vec{x}) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}}\ p(y \mid \vec{x})$$

Binary Classification $\mathcal{Y} = \{0, 1\}$

MLE to estimate pmf $p(y \mid \vec{x})$

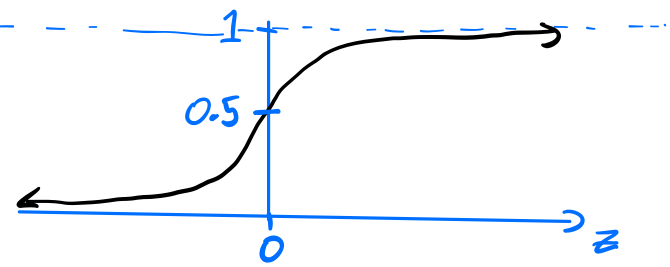
$$D = ((\vec{X}_1, Y_1), \ldots, (\vec{X}_n, Y_n))$$

$(\vec{X}_i, Y_i)$ are i.i.d. with $\mathbb{P}_{\vec{X},Y}$, $p_{\vec{X},Y}$

Assume $Y_i \mid \vec{X}_i = \vec{x}_i \sim \text{Bernoulli}(\alpha^*(\vec{x}_i)) = \text{Bernoulli}(\sigma(\vec{x}_i^T \vec{w}^*))$

$$p(y \mid \vec{x}) = (\sigma(\vec{x}^T \vec{w}^*))^y (1 - \sigma(\vec{x}^T \vec{w}^*))^{(1-y)}$$

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



"logistic" or "sigmoid" function

$$\vec{W}_{MLE} = \underset{\vec{w}\in\mathbb{R}^{d+1}}{\operatorname{argmax}} \; p(D \mid \vec{w})$$

$$= \underset{\vec{w}\in\mathbb{R}^{d+1}}{\operatorname{argmin}} \; \overbrace{-\sum_{i=1}^{n}\left[y_i \log\left(\sigma(\vec{x}_i^T\vec{w}\right) + (1-y_i)\log\left(1-\sigma(\vec{x}_i^T\vec{w})\right)\right]}^{=g(\vec{w}) \quad \text{convex}}$$

$$\frac{\partial g}{\partial w_j}(\vec{w}) = \sum_{i=1}^{n}\left(\sigma(\vec{x}_i^T\vec{w}) - y_i\right) x_{i_j}$$

$$\nabla g(\vec{w}) = \sum_{i=1}^{n}\left(\sigma(\vec{x}_i^T\vec{w}) - y_i\right)\vec{x}_i$$

$$\vec{w}^{(t+1)} = \vec{w}^{(t)} - \eta^{(t)} \nabla g(\vec{w}^{(t)})$$

$$\vec{W}_{MLE} \approx \vec{w}^{(T)}$$

$$\sigma\left(\vec{x}^T\vec{w}^{(T)}\right) \approx \underbrace{\sigma\left(\vec{x}^T\vec{w}_{MLE}\right)}_{= f_{MLE}(\vec{x})} \approx \alpha^*(\vec{x})$$

Binary Classification Learner:

$$\mathcal{A}(D) = \hat{f}_{Bin}$$

$$f_{Bayes}(\vec{x}) = \underset{y \in \mathcal{Y}}{\arg\max}\ p(y|\vec{x})$$

$$\approx \underset{y \in \{0,1\}}{\arg\max}\ p(y|\vec{x}, \vec{w}_{MLE})$$

$$= f_{Bin}(\vec{x})$$

$$p(y=1|\vec{x}, w_{MLE}) = f_{MLE}(\vec{x})$$

$$p(y=0|\vec{x}, w_{MLE}) = 1 - f_{MLE}(\vec{x})$$

$$= \begin{cases} 1 & \text{if} \quad f_{MLE}(\vec{x}) \geq 0.5 \\ 0 & \text{if} \quad f_{MLE}(\vec{x}) < 0.5 \end{cases}$$

$$= \begin{cases} 1 & \text{if} \quad \vec{x}^T \vec{w}_{MLE} \geq 0 \\ 0 & \text{if} \quad \underbrace{\vec{x}^T \vec{w}_{MLE}}_{\text{decision boundary}} < 0 \end{cases}$$

Logistic Regression

$\mathcal{Y} = [0,1]$ representing values of $\alpha^*(\vec{x})$

$\ell(f(\vec{x}), y) = -\left[y \log(f(\vec{x})) + (1-y)\log(1-f(\vec{x}))\right]$ "cross-entropy loss"

$\mathcal{F} = \{f \mid f : \mathbb{R}^{d+1} \to [0,1] \text{ and } f(\vec{x}) = \sigma(\vec{x}^T \vec{w}) \text{ where } \vec{w} \in \mathbb{R}^{d+1}\}$

Learner: $\mathcal{A}(D) = \hat{f}$

$$\hat{f} = \underset{f \in \mathcal{F}}{\arg\min}\ \hat{L}(f)$$

$$= \underset{f \in \mathcal{F}}{\arg\min}\ \frac{1}{n} \sum_{i=1}^{n} -\left[y_i \log(f(\vec{x}_i)) + (1-y_i)\log(1-f(\vec{x}_i))\right]$$

$$= \underset{f \in \mathcal{F}}{\arg\min}\ -\sum_{i=1}^{n} \left[y_i \log(f(\vec{x}_i)) + (1-y_i)\log(1-f(\vec{x}_i))\right]$$

$$= f_{MLE}$$

Multiclass Classification $\mathcal{Y} = \{0, \dots, K-1\}$

MLE to estimate pmf $p(y|\vec{x})$

$D = ((\vec{X}_1, Y_1), \dots, (\vec{X}_n, Y_n))$

$(\vec{X}_i, Y_i)$ are i.i.d. with $\mathbb{P}_{\vec{X},Y}$, $p_{\vec{X},Y}$

Assume $Y_i | \vec{X}_i = \vec{x}_i \sim \text{Categorical}\Big(\overbrace{\alpha_0^*(\vec{x}_i), \dots, \alpha_{K-1}^*(\vec{x}_i)}^{\vec{\alpha}^*(\vec{x}_i)}\Big)$

$$p(y|\vec{x}) = \sigma_y(\vec{x}^T \vec{w}_0^*, \dots, \vec{x}^T \vec{w}_{K-1}^*)$$

$\sigma(\vec{z}) = (\sigma_0(\vec{z}), \dots, \sigma_{K-1}(\vec{z})) \in [0,1]^K$ "softmax"

$$\sigma_y(z_0, \dots, z_{K-1}) = \frac{\exp(z_y)}{\sum_{q=0}^{K-1} \exp(z_q)}$$

$\vec{w}_{MLE,0}, \dots \vec{w}_{MLE,K-1}$

$$= \operatorname*{argmin}_{\vec{w}_0, \dots, \vec{w}_{K-1} \in \mathbb{R}^{d+1}} -\sum_{i=1}^n \log\Big(p(y_i | x_i, \vec{w}_0, \dots, \vec{w}_{K-1})\Big)$$

$$= \operatorname*{argmin}_{\vec{w}_0, \dots, \vec{w}_{K-1} \in \mathbb{R}^{d+1}} \overbrace{-\sum_{i=1}^n \left[\vec{x}_i^T \vec{w}_{y_i} - \log\left(\sum_{q=0}^{K-1} \exp(\vec{x}_i^T \vec{w}_q)\right)\right]}^{= g(\vec{w}_0, \dots, \vec{w}_{K-1})}$$

Convex

$$\frac{\partial g}{\partial w_{yj}}(\vec{w}_0, \ldots, \vec{w}_{K-1}) = \sum_{i=1}^{n} \left(\sigma_y(\vec{x}_i^T \vec{w}_0, \ldots, \vec{x}_i^T \vec{w}_{K-1}) - \mathbb{I}_{\{y_i\}}(y) \right) x_{ij}$$

$$\nabla_{\vec{w}_y} g(\vec{w}_0, \ldots, \vec{w}_{K-1}) = \left(\frac{\partial g}{\partial \vec{w}_{y0}}, \ldots, \frac{\partial g}{\partial \vec{w}_{yd}} \right)^T \in \mathbb{R}^{d+1} \quad \text{for } y \in \mathcal{Y}$$

$$\vec{w}_y^{(t+1)} = \vec{w}_y^{(t)} - \eta^{(t)} \nabla_{\vec{w}_y} g(\vec{w}_0^{(t)}, \ldots, \vec{w}_{K-1}^{(t)})$$

Multiclass Classification Learner:

$$\mathcal{A}(D) = \hat{f}_{Mul}$$

$$\begin{aligned}
f_{Bayes}(\vec{x}) &= \underset{y \in \mathcal{Y}}{\operatorname{argmax}}\ p(y \mid \vec{x}) \\
&\approx \underset{y \in \mathcal{Y}}{\operatorname{argmax}}\ p(y \mid \vec{x}, \vec{w}_{MLE,0}, \ldots, \vec{w}_{MLE,K-1}) \\
&= \underset{y \in \mathcal{Y}}{\operatorname{argmax}}\ \sigma_y(\vec{x}_i^T \vec{w}_{MLE,0}, \ldots, \vec{x}_i^T \vec{w}_{MLE,K-1}) \\
&= \hat{f}_{Mul}
\end{aligned}$$

Softmax Regression

$\mathcal{Y}_{soft} = [0,1]^K$ representing values of $\left(\alpha_0^*(\vec{x}), \ldots, \alpha_{K-1}^*(\vec{x})\right)^T$

Labels: $K=3$, $\vec{y}_i = \text{onehot}(y_i)$

$$\ell(\hat{\vec{y}}, \vec{y}) = -\sum_{q=0}^{K-1}\left[y_q \log(\hat{y}_q)\right]$$ "multiclass cross-entropy loss"

$$\mathcal{F} = \left\{ f \mid f: \mathbb{R}^{d+1} \to [0,1]^K \text{ and } f(\vec{x}) = \sigma\left(\vec{x}^T\vec{w}_0, \ldots, \vec{x}^T\vec{w}_{K-1}\right) \text{ where } \vec{w}_0, \ldots, \vec{w}_{K-1} \in \mathbb{R}^{d+1} \right\}$$

Learner: $\mathcal{A}(D) = \hat{f}_{ERM}$

$$\hat{f}_{ERM} = \underset{f\in\mathcal{F}}{\arg\min}\ \hat{L}(f)$$

$$= \underset{f\in\mathcal{F}}{\arg\min}\ \frac{1}{n}\sum_{i=1}^{n} -\sum_{q=0}^{K-1}\left[y_{iq}\log(f_q(\vec{x}_i))\right]$$

$$= \underset{f\in\mathcal{F}}{\arg\min}\ -\sum_{i=1}^{n}\sum_{q=0}^{K-1}\left[y_{iq}\log(f_q(\vec{x}_i))\right]$$

$$= f_{MLE}$$

$f_{MLE}(\vec{x}) = \sigma\left(\vec{x}^T\vec{w}_{MLE,0}, \ldots, \vec{x}\,\vec{w}_{MLE,K-1}\right)$

Comparison to Logistic Regression

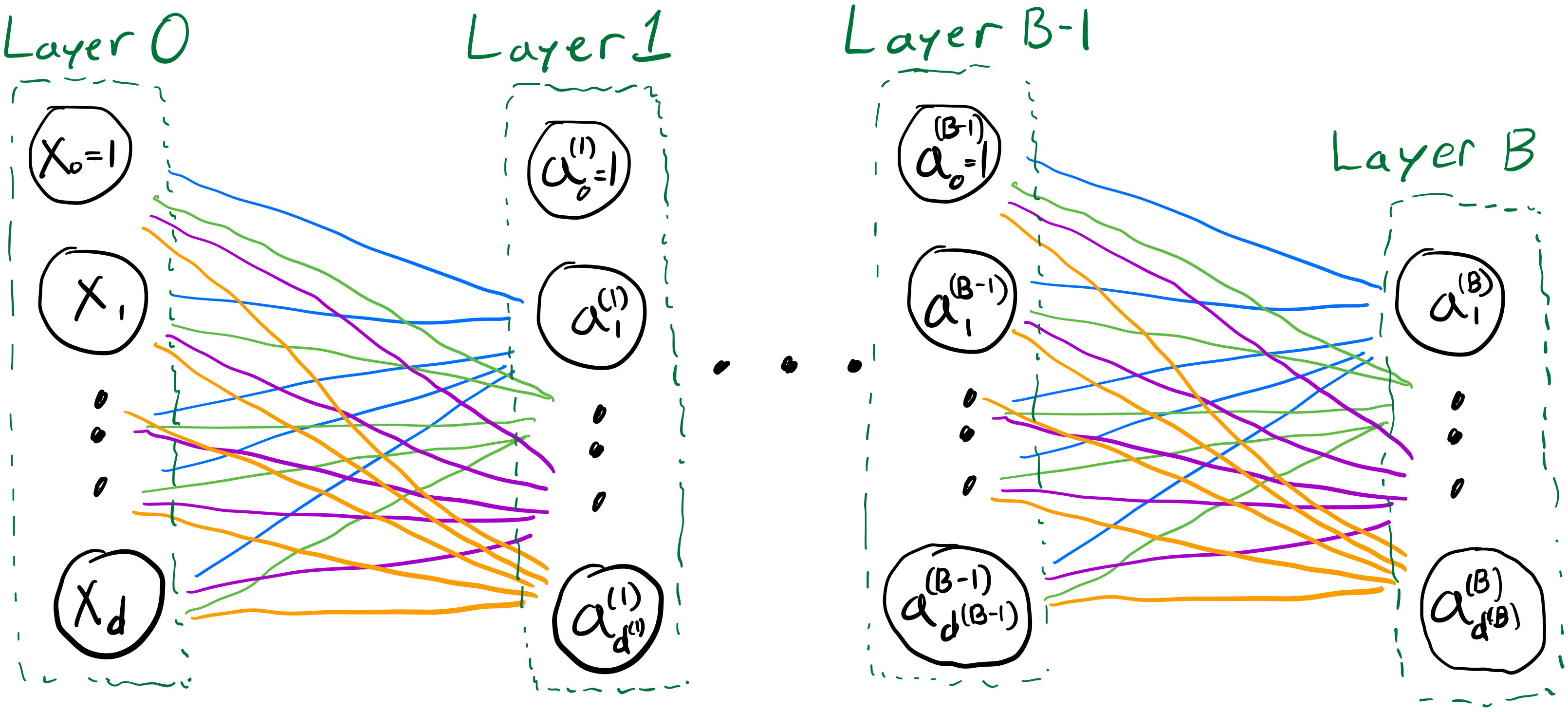
If $K = 2$ and $\mathcal{Y} = \{0, 1\}$

Assume $Y_i \mid \vec{X}_i = \vec{x}_i \sim \text{Categorical}(\alpha_0^*(\vec{x}_i), \alpha_1^*(\vec{x}_i))$

$$= \text{Bernoulli}(\alpha^*(\vec{x}_i) = \alpha_1^*(\vec{x}_i))$$

<u>Goal:</u> Show $\sigma_1(\vec{x}^T \vec{w}_{MLE,0}, \vec{x}^T \vec{w}_{MLE,1}) = \sigma(\vec{x}^T \vec{w}_{MLE})$

Neural Networks (NN)



For layer $b \in \{1, \ldots, B\}$

Weights

$$\vec{w}_1^{(b)}, \ldots, \vec{w}_{d^{(b)}}^{(b)} \in \mathbb{R}^{d^{(b-1)}+1}$$

Pre-activations

$$\vec{z}^{(b)} = (z_1^{(b)}, \ldots, z_{d^{(b)}}^{(b)}) \in \mathbb{R}^{(b)}$$

where $z_j^{(b)} = (\vec{a}^{(b-1)})^T \vec{w}_j$ for $j \in \{1, \ldots, d^{(b)}\}$

Activations

$$\vec{a}^{(0)} = \vec{x} \in \mathbb{R}^{d+1} = \mathbb{R}^{d^{(0)}+1}, \quad d = d^{(0)}$$

$$\vec{a}^{(b)} = (a_0^{(b)} = 1, a_1^{(b)}, \ldots, a_{d^{(b)}}^{(b)}) \in \mathbb{R}^{d^{(b)}+1}$$ except $b = B$

$$\vec{a}^{(B)} = (a_1^{(B)}, \ldots, a_{d^{(B)}}^{(B)}) \in \mathbb{R}^{d^{(B)}}$$

where $a_j^{(b)} = h(z_j^{(b)})$ for $j \in \{1, \ldots, d^{(b)}\}$

Activation function

$h^{(b)}: \mathbb{R} \to \mathbb{R}$

$$\boxed{\text{NN}: f(\vec{x}) = \vec{a}^{(B)}}$$

$$f(\vec{x}) = \vec{a}^{(B)} = h^{(B)}\left(h^{(B-1)}\left(\ldots h^{(2)}\left(h^{(1)}\left((\vec{x})^T W^{(1)}\right) W^{(2)}\right) \ldots W^{(B-1)}\right) W^{(B)}\right)^T$$

$$W^{(b)} = \begin{bmatrix} | & | & & | \\ W_1^{(b)} & W_2^{(b)} & \cdots & W_{d^{(b)}}^{(b)} \\ | & | & & | \end{bmatrix}$$

ERM with Neural Networks

$$\mathcal{D} = ((\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n))$$

$$\mathcal{A}(\mathcal{D}) = \hat{f} = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \; \hat{L}(f) \quad \text{where} \quad \hat{L}(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(\vec{x}_i), y_i)$$

$$\mathcal{F} = \{ f \mid f : \mathcal{X} \to \mathcal{Y} \text{ and } f \text{ is a NN with a specific architecture} \}$$

every $f_{W^{(1)},\ldots,W^{(B)}} \in \mathcal{F}$ is defined by B weight matrices $W^{(1)}, \ldots, W^{(B)}$

$\mathcal{A}(\mathcal{D}) = \hat{f}$ where $\hat{f}(\vec{x}) = \vec{a}^{(B)}$ is defined by

$$\hat{W}^{(1)}, \ldots, \hat{W}^{(B)} = \underset{W^{(1)},\ldots,W^{(B)}}{\operatorname{argmin}} \; \underbrace{\frac{1}{n}\sum_{i=1}^{n} \ell\left(f_{W^{(1)},\ldots,W^{(B)}}(\vec{x}_i), y_i\right)}_{= \hat{L}(W^{(1)}, \ldots, W^{(B)})}$$

Usually no closed form solution

$$\left(\vec{w}_j^{(b)}\right)^{(t+1)} = \left(\vec{w}_j^{(b)}\right)^{(t)} - \eta^{(t)} \nabla_{\vec{w}_j^{(b)}} \hat{L}\left(\left(W^{(1)}\right)^{(t)}, \ldots, \left(W^{(B)}\right)^{(t)}\right)$$

$\hat{L}(W^{(1)}, \dots, W^{(13)})$ is usually not convex

Language Models

Task: generating text

Ex: Input: "Why did the chicken cross the road?"
Output: "To get to the other side."

Auto-regressive: generates output sequentially by using its previous outputs and inputs

Ex: Input: "Why did the chicken cross the road?"
Output: "To"
Input: "Why did the chicken cross the road?
To"

Output: "get"

⋮

Input: "Why did the chicken cross the road?
To get to the other side."

Output: "<EOS>" End Of sequence token

We want a model $f: \mathcal{X} \to \mathcal{Y}$ where

$\vec{x} \in \mathcal{X}$ is a sequence of ~~words~~ tokens

$f(\vec{x}) \in \mathcal{Y}$ is the next ~~word~~ token

$\mathcal{Y} = \{$all words + punctuation + "<EOS>" + "<PAD>"$\}$

$= \{1, \ldots, K\}$ Vocabulary $|\mathcal{Y}| = K$

token $\in \mathcal{Y}$

Predicting discrete labels causes problems with optimization

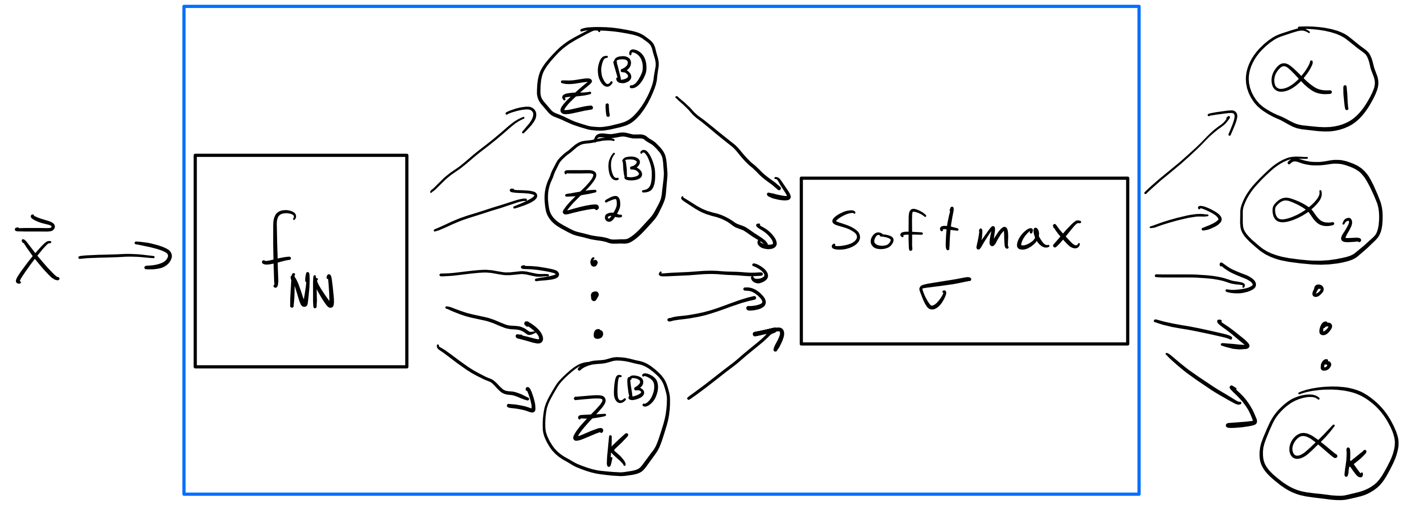
Lets predict the probability of each token and then pick the token with the largest probability

We want a model $f_{prob}: \mathcal{X} \to \mathcal{Y}_{prob}$ where

$\vec{x} \in \mathcal{X}$ is a sequence of tokens

$f_{prob}(\vec{x}) \in \mathcal{Y}_{prob}$ is a vector of probabilities of all possible next tokens

$$\mathcal{Y}_{prob} = [0,1]^K$$



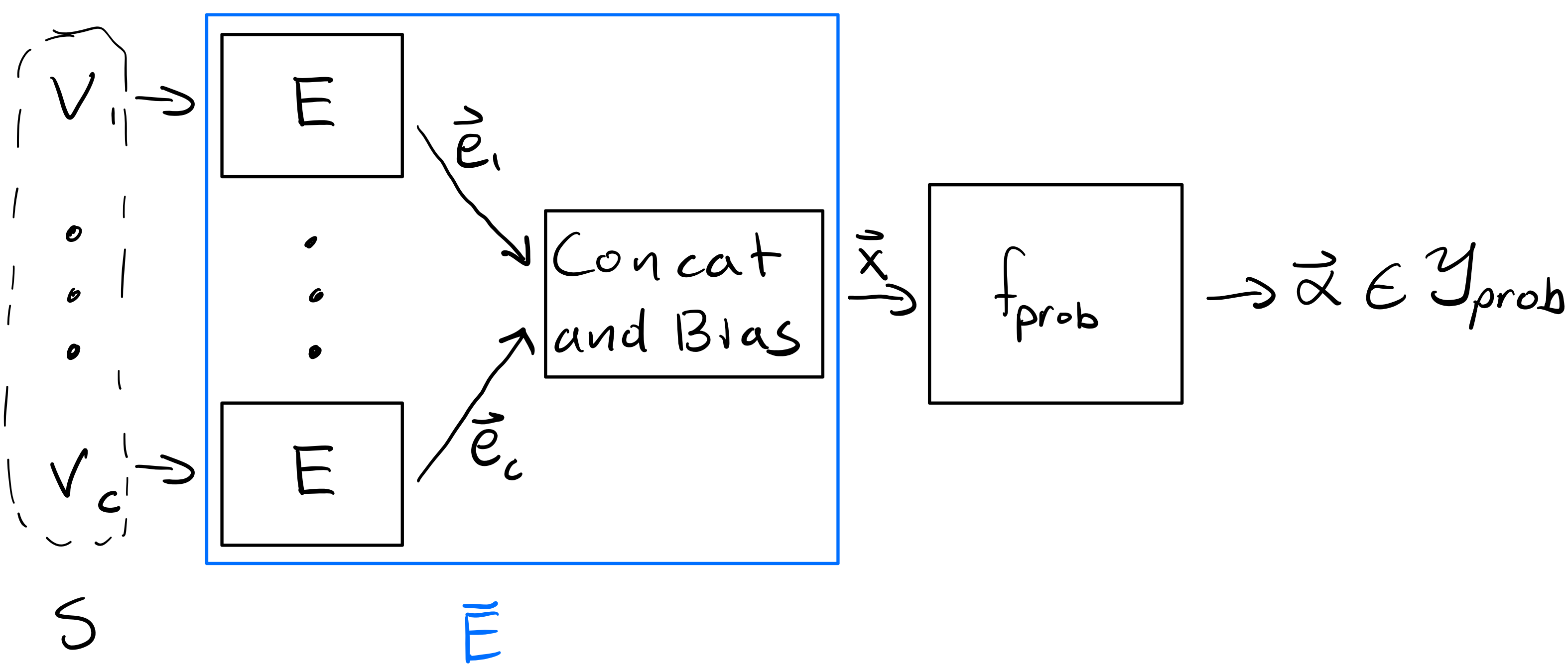
$f_{prob}(\vec{x}) = \sigma(f_{NN}(\vec{x})) \in \mathcal{Y}_{prob}$ $\quad\quad \sum_{q=1}^{K} \alpha_q = 1, \quad \alpha_q \in [0, 1]$

How do we represent a sequence of tokens s as a vector $\vec{x} \in \mathbb{R}^{d+1}$?

$E : \mathcal{Y} \to \mathbb{R}^{d'}$ Embedding function

tokens vectors

assume it is given



$\bar{E}(s) = \vec{x} \in \mathcal{X} = \mathbb{R}^{d+1}$ where $d = cd'$

context length

s is a sequence of at most c tokens

To handle sequences shorter than c words a padding token "<PAD>" is added

Creating a Dataset

$$S = \Big(\underbrace{\overbrace{\underbrace{v_1}_{s_1}}^{s_2}, \underbrace{v_2}_{y_1}, \overbrace{v_3}^{y_2}, \dots, v_c}_{s_c}, \overbrace{\underbrace{v_{c+1}}_{y_c}}^{s_{c+1}}, \overbrace{v_{c+2}}^{y_{c+1}}, \dots, v_a\Big)$$

$\vec{x}_1 = \bar{E}(s_1),\ \vec{x}_2 = \bar{E}(s_2),\ \dots,\ \vec{x}_n = \bar{E}(s_n) \in \mathbb{R}^{d+1}$

$\vec{y}_1 = \text{onehot}(y_1) \in \{0,1\}^K \subset [0,1]^K$

$\vec{y}_2 = \text{onehot}(y_2) \in \{0,1\}^K$

$\vdots$

$\vec{y}_n = \text{onehot}(y_n) \in \{0,1\}^K$

$D = ((\vec{x}_1, \vec{y}_1), \dots, (\vec{x}_n, \vec{y}_n))$

ERM Learner:

$$\mathcal{A}(D) = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \hat{L}(f)$$

$\mathcal{F} = \{ f \mid f : \mathcal{X} \to \mathcal{Y}_{prob}$ where $f = \sigma(f_{NN})$ and f_{NN} is a NN with a fixed architecture $\}$

ℓ is multiclass cross-entropy loss

Embeddings

Ex: $d' = 2$ $\qquad E : \mathcal{Y} \to \mathbb{R}^{d'}$