

Please do course survey



<https://go.blueja.io/eLzPhoC0e0atAYGdS8Mi8Q>

Math Review

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

$\in, \subset, \not\subset, \subseteq, \not\subseteq, \cup, \cap, \setminus, \text{set}^c$

Set builder notation:

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

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

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

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

Functions:

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

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

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

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

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

Summation and Integration:

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

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

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

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

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

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

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

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

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

(Partial) Derivatives:

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

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

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

Common derivatives and properties
on formula sheet

Probability

Outcome space and Events:

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

$\tilde{E} \subseteq \mathcal{X}$ event

$\tilde{E} = \{1, 2, 3\}$

$= \mathcal{X}$

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

Probability Distribution:

P takes as input events and
outputs values in $[0, 1]$

$P(\tilde{E})$ where $\tilde{E} \subseteq \mathcal{X}$ probability of event $\tilde{E}$

Random Variable:

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

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

Common notation: If $\tilde{E}$ contains a single outcome

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

If $\tilde{E}$ is an interval:

$E = [a, b]$ then $P(a \leq X \leq b) \stackrel{\text{def}}{=} P(X \in [a, b])$

$E = [a, \infty)$ then $P(X \geq a) \stackrel{\text{def}}{=} P(X \in [a, \infty))$

$E = (-\infty, b]$ then $P(X \leq b) \stackrel{\text{def}}{=} P(X \in (-\infty, b])$

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

Discrete r.v.:

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

Continuous r.v.:

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

Calculating Probabilities:

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

$$P(X \in E) \stackrel{\text{def}}{=} \sum_{x \in E} p(x) \quad \text{where } E \subseteq \mathcal{X}$$

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

$$P(X \in E) \stackrel{\text{def}}{=} \int_E p(x) dx \quad \text{where } E \subseteq \mathcal{X}$$

commonly used discrete and continuous distributions on formula sheet.

Ex: Bernoulli, Normal, Laplace, Categorical

Multivariate Probability:

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

On formula sheet $\mathcal{E}_x \subseteq \mathcal{X}, \mathcal{E}_y \subseteq \mathcal{Y}$

Joint: $P(X \in \mathcal{E}_x, Y \in \mathcal{E}_y)$

Marginal: $P_x(X \in \mathcal{E}_x), \quad P_y(Y \in \mathcal{E}_y)$

Conditional: $P_{X|Y}(X \in \mathcal{E}_x | Y=y), \quad P_{Y|X}(Y \in \mathcal{E}_y | X=x)$

Product Rule: $p(x, y) = p(y|x) p(x) = p(x|y) p(y)$

Independence:

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

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

Functions of r.v.:

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

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

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

$\bar{X} = g(X_1, \dots, X_n) = \frac{X_1 + \dots + X_n}{n}$ 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: $\mathbb{E}[f(Y) | X = x]$

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

Supervised Learning

Dataset:

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

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

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

$\mathcal{Y}$ Labels or targets

Learner:

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

Predictor:

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

loss function

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

Expected loss

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

Objective:

Define A such that $E[L(A(D))]$ 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 $\{\text{cat}, \text{dog}, \text{bird}\}$

Use 0-1 loss

Learner: ERM input dataset D

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

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

Expectation and Variance:

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

$$\text{Var}[\bar{X}] = \frac{\text{Var}[X]}{n} = \frac{\text{Var}[X_1]}{n} = \dots = \frac{\text{Var}[X_n]}{n}$$

Optimization

$$\min_{w \in W} g(w) = g(w^*) \quad \text{where } w^* = \arg\min_{w \in W} g(w)$$

$$w^* = \arg\min_{w \in W} g(w) = \arg\max_{w \in W} -g(w)$$

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

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 differentiable: 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

$$\vec{w}^* = (w_1^*, \dots, w_d^*)^T = \arg\min_{\vec{w} \in \mathcal{W}} g(\vec{w})$$

by setting w_j^* as the solution to

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

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

Linear Regression (Closed Form)

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

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

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

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

we want

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

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

Learner

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

$$\text{where } \hat{f}(\vec{x}) = \vec{x}^T \hat{\vec{w}} \quad \text{and} \quad \hat{\vec{w}} = \overset{\substack{\text{Depends on } \mathcal{D} \\ \downarrow}}{A^{-1}} \vec{b}$$

Algorithm: Closed form linear regression Learner

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

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

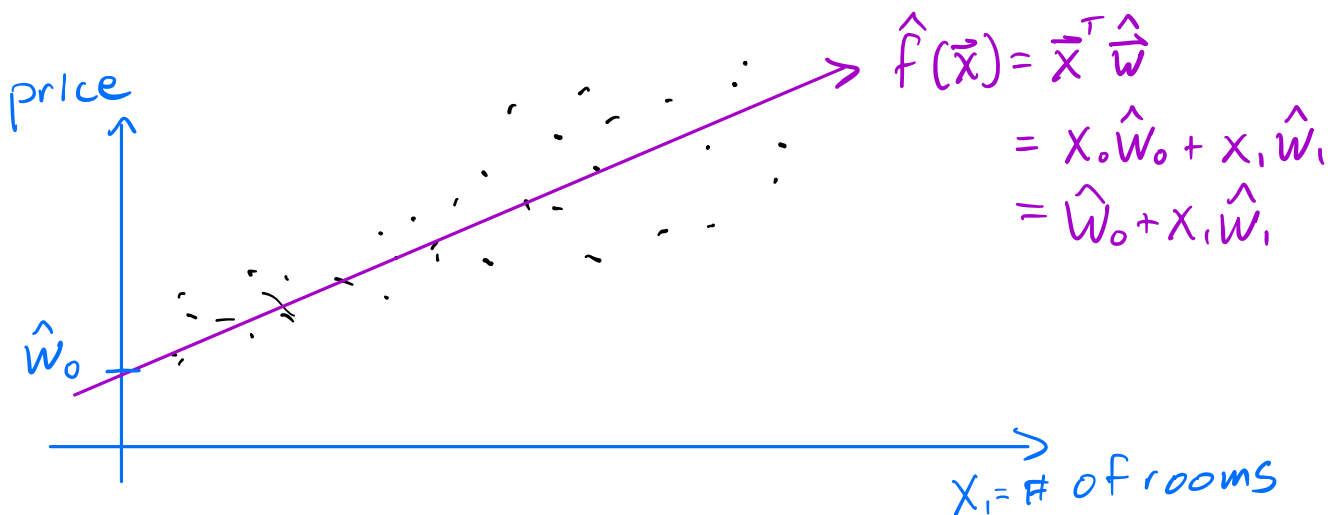
$$b \leftarrow \sum_{i=1}^n \vec{x}_i y_i$$

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

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

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

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



Gradient Descent

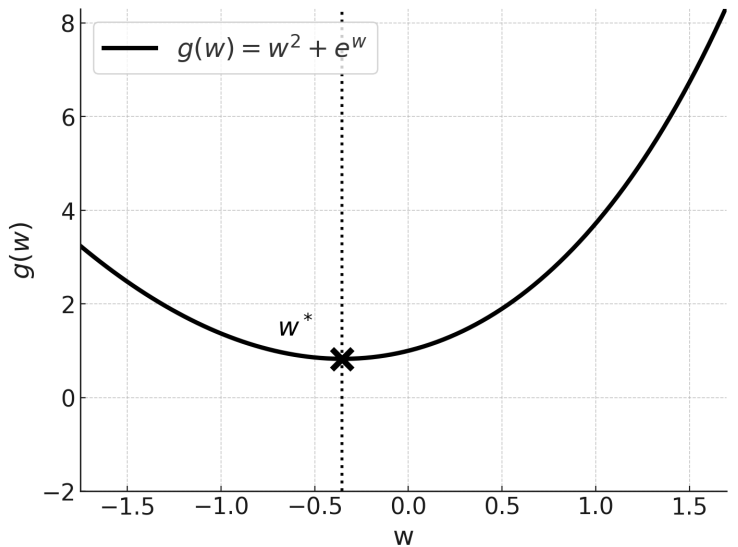
Ex: $g(w) = w^2 + e^w$, $g'(w) = 2w + e^w$, $g''(w) = 2 + e^w \geq 0$
Convex

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

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

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

No closed form
solution



Second-order Gradient Descent:

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

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

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

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

$w^{(t+1)}$ approaches $w^* = \arg \min_{w \in \mathcal{W}} g(w)$
as $t \rightarrow \infty$

First-order Gradient Descent

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

Multivariate Gradient Descent

Objective:

$$\vec{w}^* = (w_1^*, \dots, w_d^*)^T = \arg \min_{\vec{w} \in \mathcal{W}} g(\vec{w})$$

gradient descent update rule:

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

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

Selecting the step size

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

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

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

4. Normalized gradient: $\eta^{(t)} = \frac{\eta}{\epsilon + \|\nabla g(\vec{w}^{(t)})\|}$ where $\eta, \epsilon \in (0, \infty)$
 ϵ is small
(ex: $\epsilon = 10^{-8}$)

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

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

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

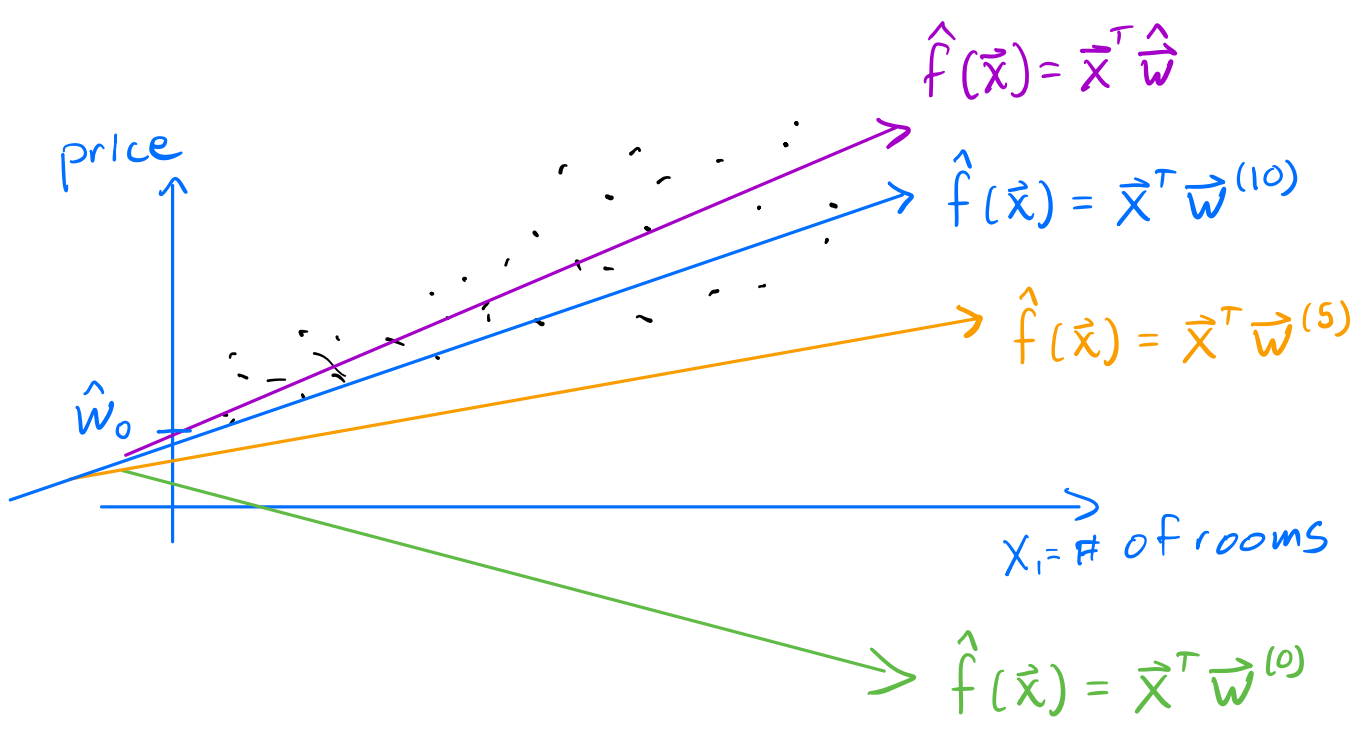
for $t = 1, \dots, T$

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

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

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

$\vec{x}$:



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)

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

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

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

Algorithm: MBbD 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 $\mathbb{R}^{d+1}$

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

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

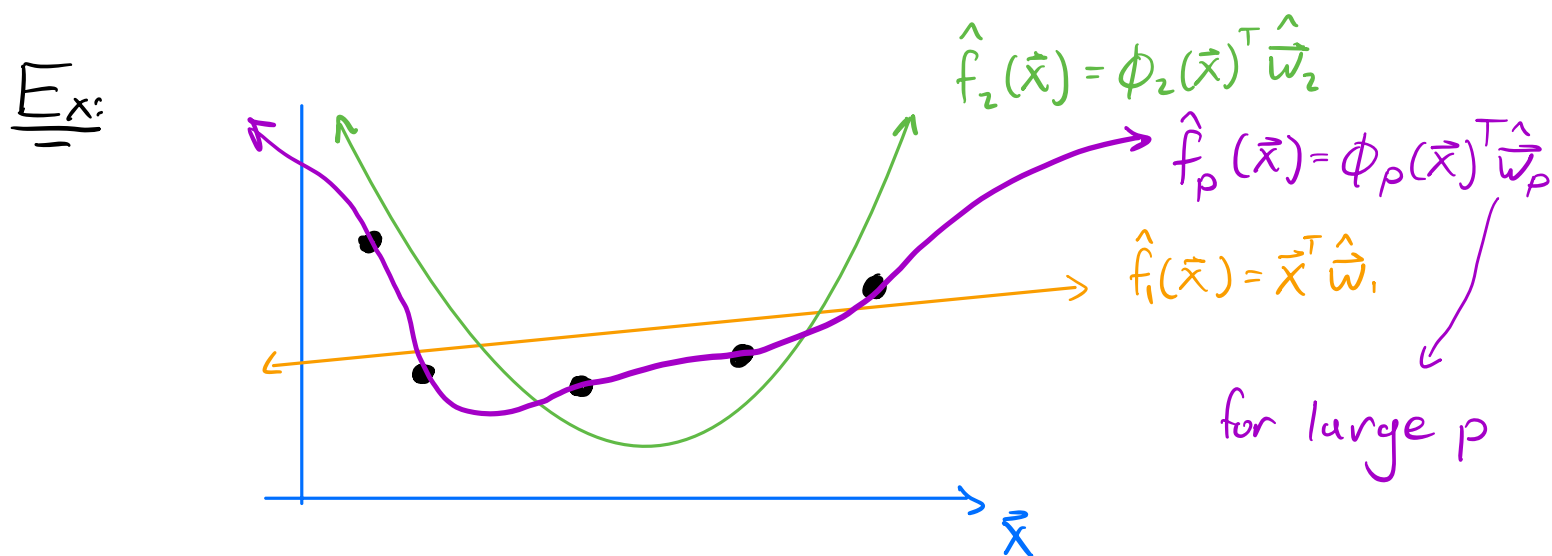
$\phi_p: \mathcal{X} \rightarrow \mathbb{Z}$ is a degree p polynomial "feature map"

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

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

$$\tilde{\mathcal{F}}_1 \subseteq \tilde{\mathcal{F}}_2 \subseteq \dots \subseteq \tilde{\mathcal{F}}_p$$

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



Evaluating Predictors/Models

Objective (formal):

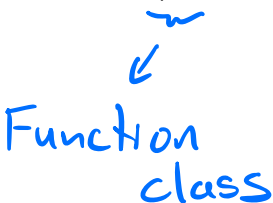
Define a Learner $\mathcal{A}: (\mathcal{X} \times \mathcal{Y})^n \rightarrow \{f \mid f: \mathcal{X} \rightarrow \mathcal{Y}\}$
such that $\mathbb{E}[L(\mathcal{A}(D))]$ is small

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

Estimation:

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

Optimization:

pick $\hat{f}$ to be the $f \in \mathcal{F}$ that minimizes $\hat{L}(f)$

Function class

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 $A(D) = f_0$ depends on D then

$$\mathbb{E}[\hat{L}(\hat{f}_0)] \neq \mathbb{E}[L(\hat{f}_0)]$$

$l(\hat{f}_0(\vec{X}_i), Y_i)$ are not i.i.d.

$\hat{f}_0$ depends on $(\vec{X}_1, Y_1), \dots, (\vec{X}_n, Y_n)$!

It can be shown that:

$$\mathbb{E}[L(\hat{f}_0)] - \mathbb{E}[\hat{L}(\hat{f}_0)]$$

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

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

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

$\mathcal{F}$ any function class

$$f_{\text{Bayes}} = \operatorname{argmin}_{f \in \{f | f: x \rightarrow y\}} L(f)$$

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

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

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

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

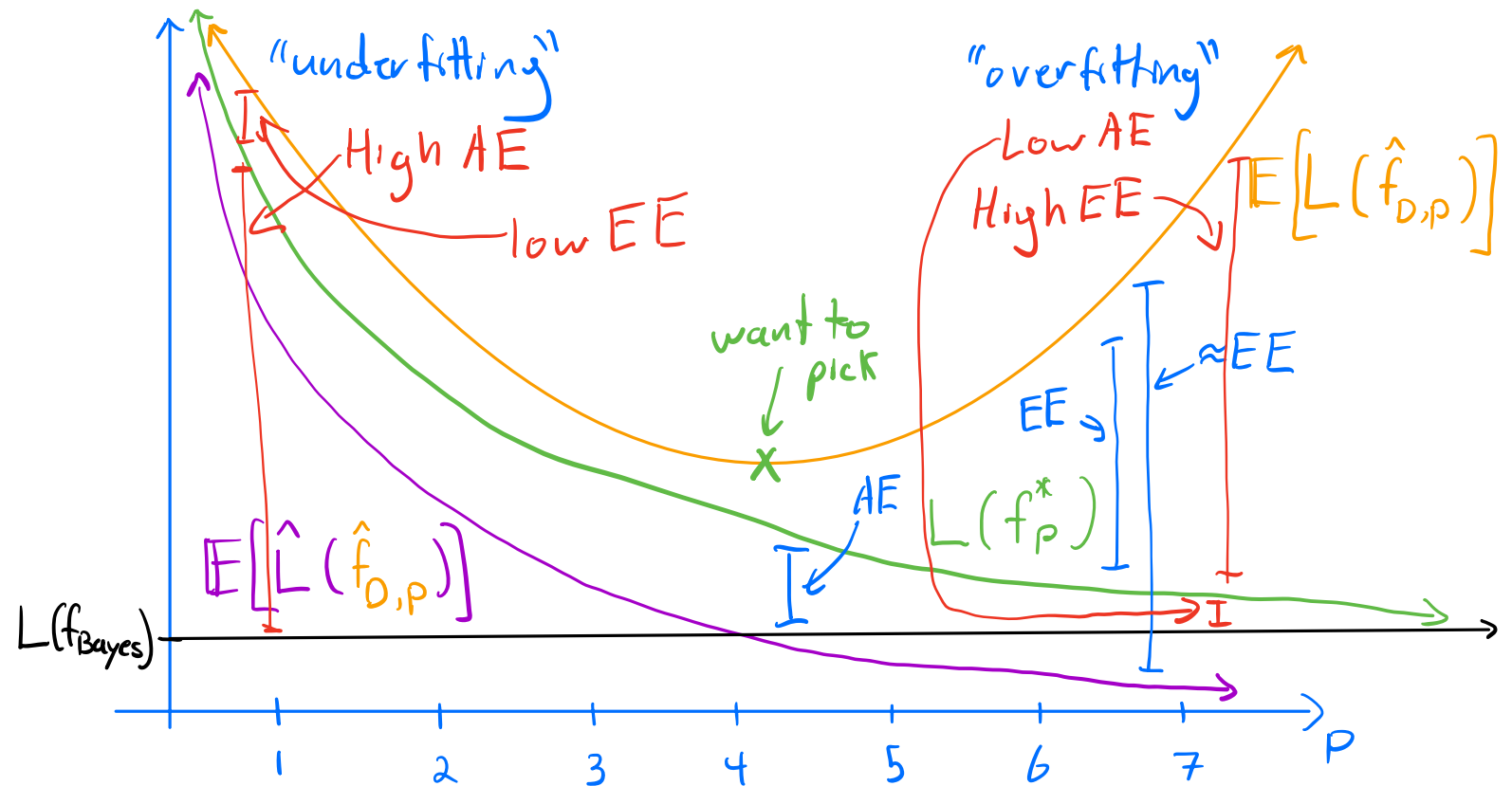
Estimation Error: Due to random dataset D

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

$$\mathcal{A}(D) = \hat{f}_{D,p} = \arg\min_{f \in \tilde{F}_p} \hat{L}(f) \quad \tilde{F}_1 \subset \dots \subset \tilde{F}_p$$

$$E[L(\hat{f}_0)] - E[\hat{L}(\hat{f}_0)]$$

$$E[L(\hat{f}_0)] = \underbrace{E[L(\hat{f}_0)] - L(f^*)}_{\text{Estimation Error (EE)}} + \underbrace{L(f^*) - L(f_{\text{Bayes}})}_{\text{Approximation Error (AE)}} + \underbrace{L(f_{\text{Bayes}})}_{\text{Irreducible Error (IE)}}$$



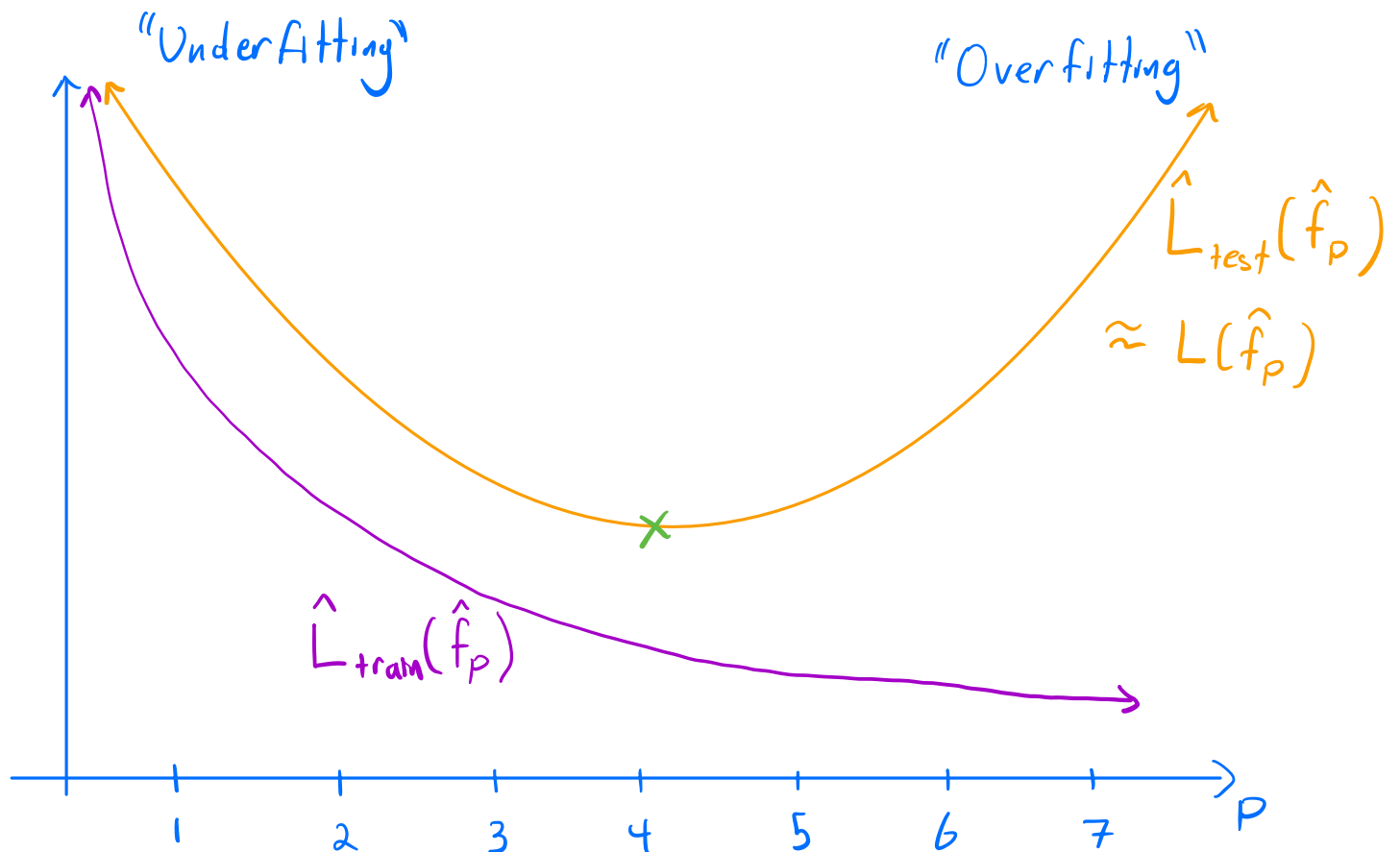
Can't calculate L , so use train, test split

$$D_{\text{train}} = ((\vec{x}_1, y_1), \dots, (\vec{x}_{n-m}, y_{n-m}))$$

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

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

$$\mathcal{A}(D_{\text{train}}) = \hat{f}_p = \arg\min_{f \in \tilde{F}_p} \hat{L}_{\text{train}}(f) \quad \tilde{F}_1 \subset \dots \subset \tilde{F}_p$$



Bias-Variance Tradeoff

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

Let l be squared loss

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

$$\begin{aligned} &= \overbrace{\mathbb{E}[\mathbb{E}[(\hat{f}_D(\vec{x}) - \bar{f}(\vec{x}))^2 | \vec{x}]]}^{= EE} + \overbrace{\mathbb{E}[(\bar{f}(\vec{x}) - f_{\text{Bayes}}(\vec{x}))^2]}^{= AE} + \underbrace{L(f_{\text{Bayes}})}_{IE} \\ &\quad \text{Variance} = \text{Var}[\hat{f}_D(\vec{x}) | \vec{x}] \qquad \text{Bias} \qquad \text{IE} \end{aligned}$$

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

Effects of changing F, n on Bias, Variance

Bias $\downarrow$ if $F \uparrow$

Variance $\downarrow$ if $n \uparrow$

$\uparrow$ if $F \uparrow$

Regularization

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

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

$$= \hat{L}(f_p)$$

$$\hat{L}_\lambda(f_p) = \frac{1}{n} \sum_{i=1}^n \ell(f_p(\vec{x}_i), y_i) + \frac{\lambda}{n} \sum_{j=1}^{\bar{p}-1} w_j^2$$

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

Regularized
estimated
loss

$j=0$ not included "Regularizer"

Bias vs. Variance

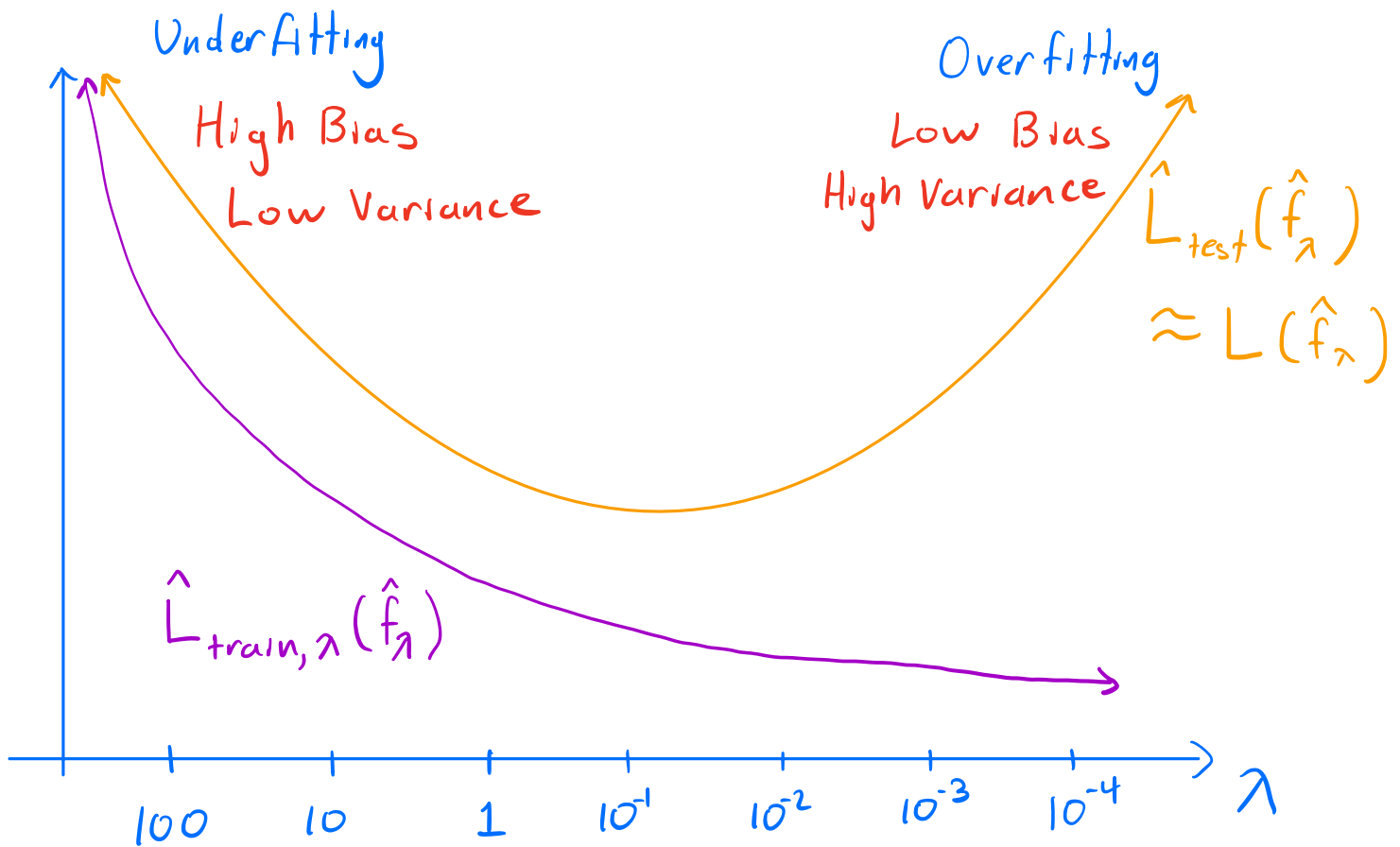
$$\text{Bias: } (\bar{f}_\lambda(\vec{x}) - f_{\text{Bayes}}(\vec{x}))^2$$

- Decreases if λ decreases

$$\text{Variance: } E[(\hat{f}_{0,\lambda}(\vec{x}) - \bar{f}_\lambda(\vec{x}))^2 | \vec{x}]$$

- Increases if λ decreases
- Decreases if n increases

$$\hat{f}_\lambda = \operatorname{argmin}_{f \in \mathcal{F}_0} \hat{L}_\lambda(f)$$



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

Objective:

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

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

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

$$\begin{aligned}\mathbb{E}[L(\hat{f}_D)] &= \underbrace{\mathbb{E}[L(\hat{f}_D)] - L(f^*)}_{\text{Estimation Error (EE)}} + \underbrace{L(f^*) - L(f_{\text{Bayes}})}_{\text{Approximation Error (AE)}} + \underbrace{L(f_{\text{Bayes}})}_{\text{Irreducible Error (IE)}} \\ \mathbb{E}[L(\hat{f}_D)] &= \mathbb{E} \left[\underbrace{\mathbb{E}[(\hat{f}_D(\mathbf{X}) - \bar{f}(\mathbf{X}))^2 | \mathbf{X}]}_{\text{Variance}} \right] + \mathbb{E} \left[\underbrace{(\bar{f}(\mathbf{X}) - f_{\text{Bayes}}(\mathbf{X}))^2}_{\text{Bias}} \right] + \underbrace{L(f_{\text{Bayes}})}_{\text{Irreducible Error}}\end{aligned}$$

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

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

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

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

	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.

MLE

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

assume squared loss and Regression

$$\begin{aligned} f_{\text{Bayes}}(\vec{x}) &= \mathbb{E}[Y \mid \vec{X} = \vec{x}] \\ &= \int_{\mathcal{Y}} y \, p_{Y|\vec{X}}(y|\vec{x}) \, dy \end{aligned}$$

Use 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 given a fixed data $D = (z_1, \dots, z_n)$

$$\begin{aligned} w_{\text{MLE}} &= \underset{w \in \mathcal{W}}{\operatorname{argmax}} \underbrace{p(D|w)}_{\text{likelihood}} \\ &= \underset{w \in \mathcal{W}}{\operatorname{argmin}} \underbrace{-\log(p(D|w))}_{\text{negative log-likelihood}} \end{aligned}$$

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

$$= \arg \min_{w \in \mathcal{H}} -\sum_{i=1}^n \log(p(z_i | w))$$

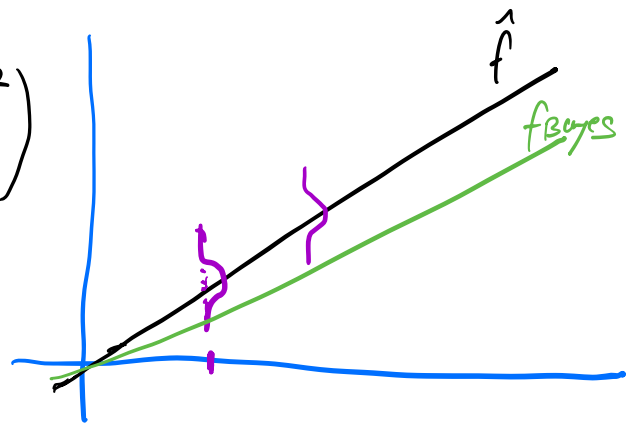
Estimating $P_{Y|\vec{X}}$

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

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

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

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



$$\vec{w}_{MLE} = \arg \min_{\vec{w} \in \mathbb{R}^{d_H}} \sum_{i=1}^n \frac{(Y_i - \vec{x}_i^T \vec{w})^2}{2}$$

$$= \arg \min_{\vec{w} \in \mathbb{R}^{d_H}} \hat{L}(\vec{w})$$

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

MAP

If P_{riX} is a function of some parameter $w \in W$ then we just need to estimate that parameter

$$\text{MLE: } \arg \max_{w \in W} \underbrace{p(D|w)}_{\text{Likelihood}}$$

$$\text{MAP: } \arg \max_{w \in W} \underbrace{p(w|D)}_{\text{Posterior}}$$

$$w_{\text{MAP}} = \arg \max_{w \in W} p(w|D)$$

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

$$= \arg \max_{w \in W} \frac{p(D|w) p(w)}{P(D)}$$

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

$$= \arg \min_{w \in \mathcal{W}} -\log(p(D|w) p(w))$$

$$= \arg \min_{w \in \mathcal{W}} [-\log(p(D|w)) - \log(p(w))]$$

if n large then:

$$w_{\text{MAP}} \approx w_{\text{MLE}}$$

if $\text{Var}(w)$ is large then:

$$w_{\text{MAP}} \approx w_{\text{MLE}}$$

small then:

$$w_{\text{MAP}} \approx \mathbb{E}[w]$$

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

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

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

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 $j \in \{1, \dots, d\}$

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

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

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

$$\vec{w}_{\text{MAP}} = \underset{\vec{w} \in \mathbb{R}^{d+1}}{\text{argmax}} \quad p(\vec{w} | D)$$

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

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

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

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

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

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

$$\vec{w}_{\text{MAP}} = \arg \max_{\vec{w} \in \mathbb{R}^{d+1}} p(\vec{w} | D)$$

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

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

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

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

assume 0-1 Loss

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

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

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

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

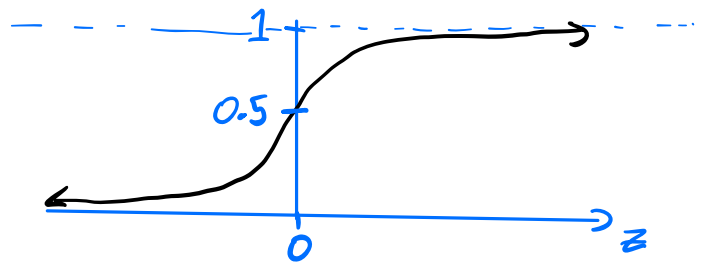
$(\vec{X}_i, Y_i)$ are i.i.d. with $P_{\vec{X}, Y}, P_{\vec{X}, Y}$

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

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

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

"logistic" or "sigmoid" function



$$\vec{w}_{MLE} = \underset{\vec{w} \in \mathbb{R}^{d_H}}{\operatorname{argmax}} \quad p(\mathcal{D} | \vec{w})$$

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

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

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

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

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

$$\sigma(\vec{x}^T \vec{w}^{(T)}) \approx \sigma(\vec{x}^T \vec{w}_{MLE}) \approx \alpha^*(\vec{x})$$

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

Binary Classification Learner:

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

$$f_{\text{Bayes}}(\vec{x}) = \operatorname{argmax}_{y \in \mathcal{Y}} p(y|\vec{x})$$

$$\approx \operatorname{argmax}_{y \in \{0,1\}} p(y|\vec{x}, \vec{w}_{\text{MLE}})$$

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

$$p(y=1|\vec{x}, w_{\text{MLE}}) = f_{\text{MLE}}(\vec{x}) = \sigma(\vec{x}^T \vec{w}_{\text{MLE}})$$

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

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

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

decision boundary

Logistic Regression

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

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

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

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

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

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

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

$$= f_{MLE}$$

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 $P_{\vec{X}, Y}$, $P_{\vec{X}, Y}$ $\vec{\alpha}^*(\vec{x}_i)$

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

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

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

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

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

$$= \arg \min_{\vec{w}_0, \dots, \vec{w}_{K-1} \in \mathbb{R}^{d_H}} - \sum_{i=1}^n \log(p(y_i | x_i, \vec{w}_0, \dots, \vec{w}_{K-1}))$$

$$= \arg \min_{\vec{w}_0, \dots, \vec{w}_{K-1} \in \mathbb{R}^{d_H}} - \sum_{i=1}^n \left[\vec{x}_i^T \vec{w}_{y_i} - \log \left(\sum_{q=0}^{K-1} \exp(\vec{x}_i^T \vec{w}_q) \right) \right] \quad \begin{matrix} = g(\vec{w}_0, \dots, \vec{w}_{K-1}) \\ \text{Convex} \end{matrix}$$

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

$$\nabla_{\vec{w}_y} g(\vec{w}_0, \dots, \vec{w}_{K-1}) = \left(\frac{\partial g}{\partial \vec{w}_{y_0}}, \dots, \frac{\partial g}{\partial \vec{w}_{y_d}} \right)^T \in \mathbb{R}^{d+1} \quad \text{for } y \in \mathcal{Y}$$

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

Multiclass Classification Learner:

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

$$f_{\text{Bayes}}(\vec{x}) = \operatorname{argmax}_{y \in \mathcal{Y}} p(y|\vec{x})$$

$$\approx \operatorname{argmax}_{y \in \mathcal{Y}} p(y|\vec{x}, \vec{w}_{\text{MLE},0}, \dots, \vec{w}_{\text{MLE},K-1})$$

$$= \operatorname{argmax}_{y \in \mathcal{Y}} \sigma_y(\vec{x}_i^T \vec{w}_{\text{MLE},0}, \dots, \vec{x}_i^T \vec{w}_{\text{MLE},K-1})$$

$$= \hat{f}_{\text{Mul}}$$

Softmax Regression

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

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

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

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

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

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

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

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

$$= f_{\text{MLE}}$$

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

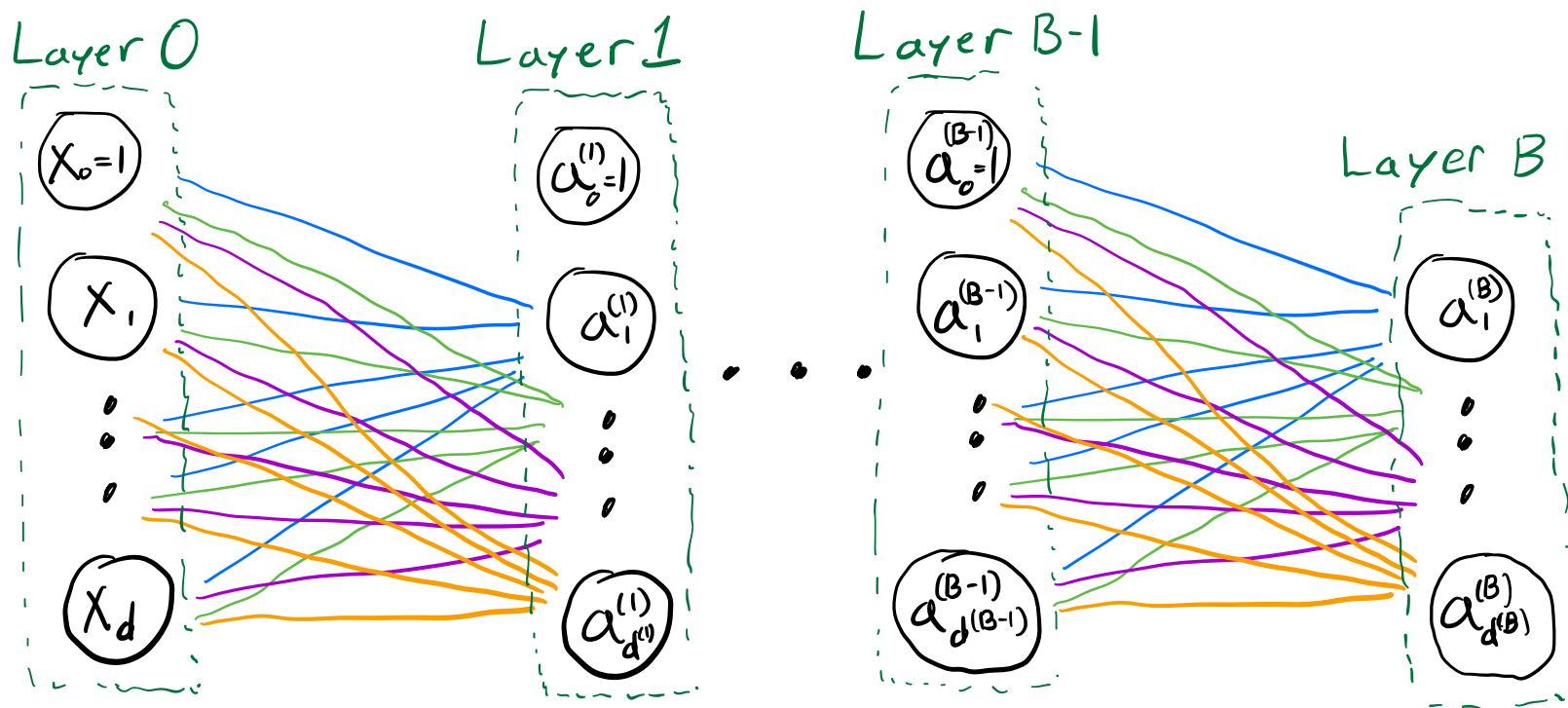
Comparison to Logistic Regression

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

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

Can show $\sigma_i(\vec{x}^T \vec{w}_{MLE,0}, \vec{x}^T \vec{w}_{MLE,1}) = \sigma(\vec{x}^T \vec{w}_{MLE})$

Neural Networks (NN)



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

Weights

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

Pre-activations

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

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

Activations

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

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

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

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

Activation function

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

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

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

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

ERM With Neural Networks

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

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

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

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

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

$$\hat{W}^{(1)}, \dots, \hat{W}^{(B)} = \arg \min_{W^{(1)}, \dots, W^{(B)}} \underbrace{\frac{1}{n} \sum_{i=1}^n \ell(f_{W^{(1)}, \dots, W^{(B)}}(\vec{x}_i), y_i)}_{= \hat{L}(W^{(1)}, \dots, W^{(B)})}$$

Usually no closed form solution

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

$\hat{L}(W^{(1)}, \dots, W^{(B)})$ 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 $f: \mathcal{X} \rightarrow \mathcal{Y}$ where

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

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

$\mathcal{Y} = \{ \text{all words + punctuation} + \langle \text{EOS} \rangle + \langle \text{PAD} \rangle \}$

$= \{1, \dots, K\}$

Vocabulary

$|\mathcal{Y}| = K$

token $\in \mathcal{Y}$

Predicting discrete labels causes problems with optimization

Lets predict the probability of each token

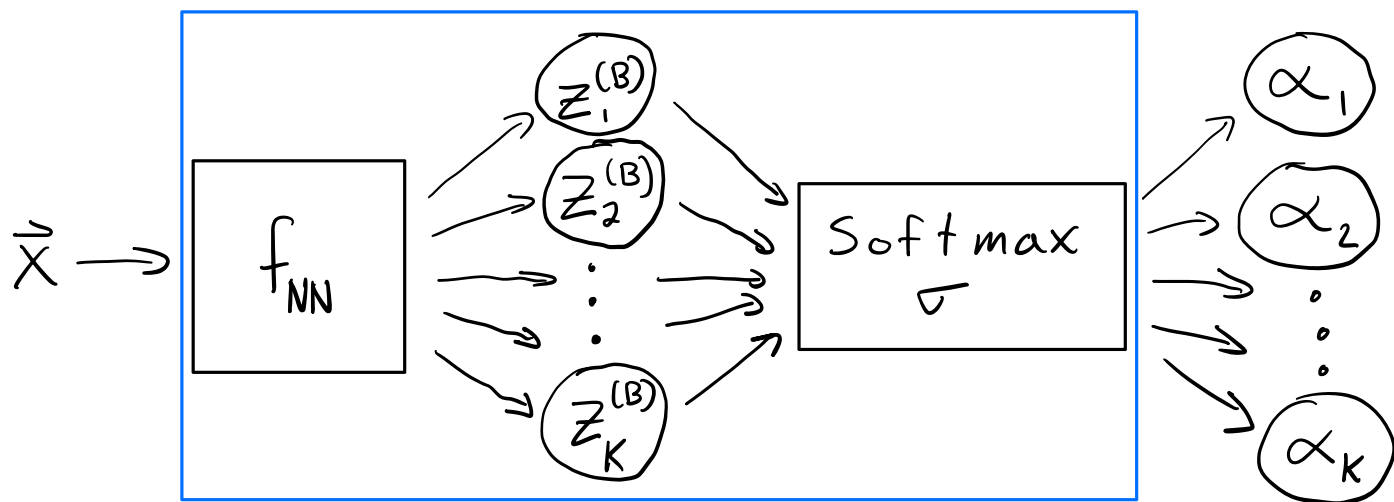
and then pick the token with the largest probability

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

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

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

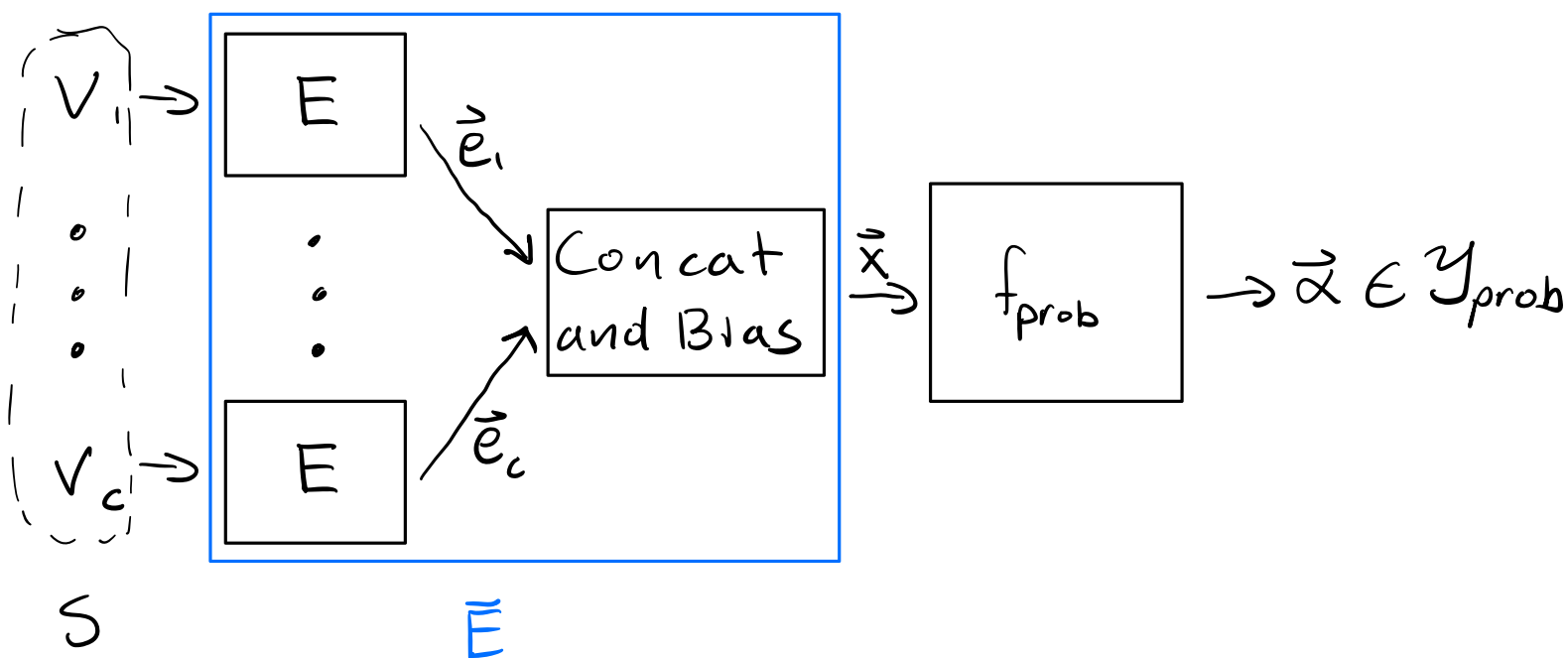
$$\mathcal{Y}_{\text{prob}} = [0, 1]^K$$



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

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

$E: \mathcal{Y} \rightarrow \mathbb{R}^{d'}$ Embedding function
 tokens vectors
 assume it is given



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

s is a sequence of
at most c tokens

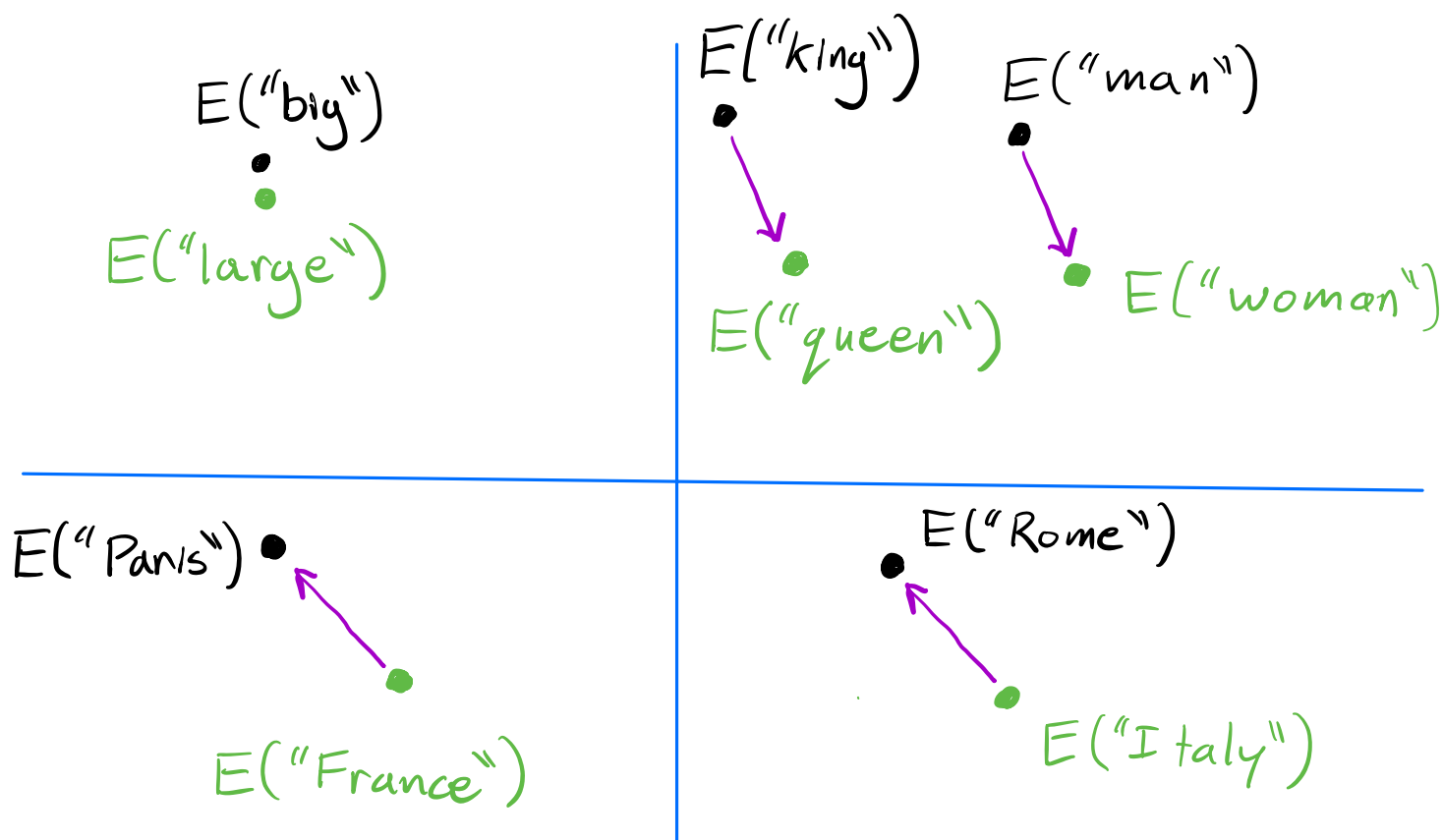
context length

If a sequence is longer than c tokens
take the last c tokens

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

Embeddings

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



Creating a Dataset

$$S = (V_1, V_2, V_3, \dots, V_c, V_{c+1}, \dots, V_{n+1})$$

$$S_1 = (\overbrace{\langle \text{PAD} \rangle, \dots, \langle \text{PAD} \rangle}^{c-1}, V_1), \quad y_1 = V_2$$

$$S_2 = (\overbrace{\langle \text{PAD} \rangle, \dots, \langle \text{PAD} \rangle}^{c-2}, V_1, V_2), \quad y_2 = V_3$$

$\vdots$

$$S_c = (V_1, \dots, V_c), \quad y_c = V_{c+1}$$

$\vdots$

$$S_n = (V_{n-c+1}, \dots, V_n), \quad y_n = V_{n+1}$$

Ex: $C=2, S = (\text{"Why"}, \text{"did"}, \text{"the"})$

$$S_1 = (\langle \text{PAD} \rangle, \text{"Why"}), \quad y_1 = \text{"did"}$$

$$S_2 = (\text{"Why"}, \text{"did"}), \quad y_2 = \text{"the"}$$

$$\vec{X}_1 = \bar{E}(S_1), \quad \vec{X}_2 = \bar{E}(S_2), \quad \dots, \quad \vec{X}_n = \bar{E}(S_n) \in \mathbb{R}^{d+1}$$

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

$\vdots$

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

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

ERM Learner:

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

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

ℓ is multiclass cross-entropy loss

For a new sequence $s = (v_1, \dots, v_c) \in \mathcal{Y}^c$

The Language model is:

$$\hat{f}_{\text{LM}}(s) = \arg \max_{q \in \{1, \dots, k\}} \hat{y}_q$$

$$\hat{\tilde{y}} = \hat{f}_{\text{prob}}(\tilde{E}(s))$$