Midterm Exam 2 Review

Linear Regression (Closed form)

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

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

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

$\hat{L}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell\left(f(\vec{x}_i), y_i\right)$ 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 $\mathcal{F} = \{f \mid f : \mathbb{R}^{d+1} \to \mathbb{R}$ and $f(\vec{x}) = \vec{x}^T \vec{w}$ where $\vec{w} \in \mathbb{R}^{d+1}\}$

Learner

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

Depends on D

where $\hat{f}(\vec{x}) = \vec{x}^T \hat{\vec{w}}$ and $\hat{\vec{w}} = A^{-1} \vec{b}$

Algorithm: Closed form linear regression Learner

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

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

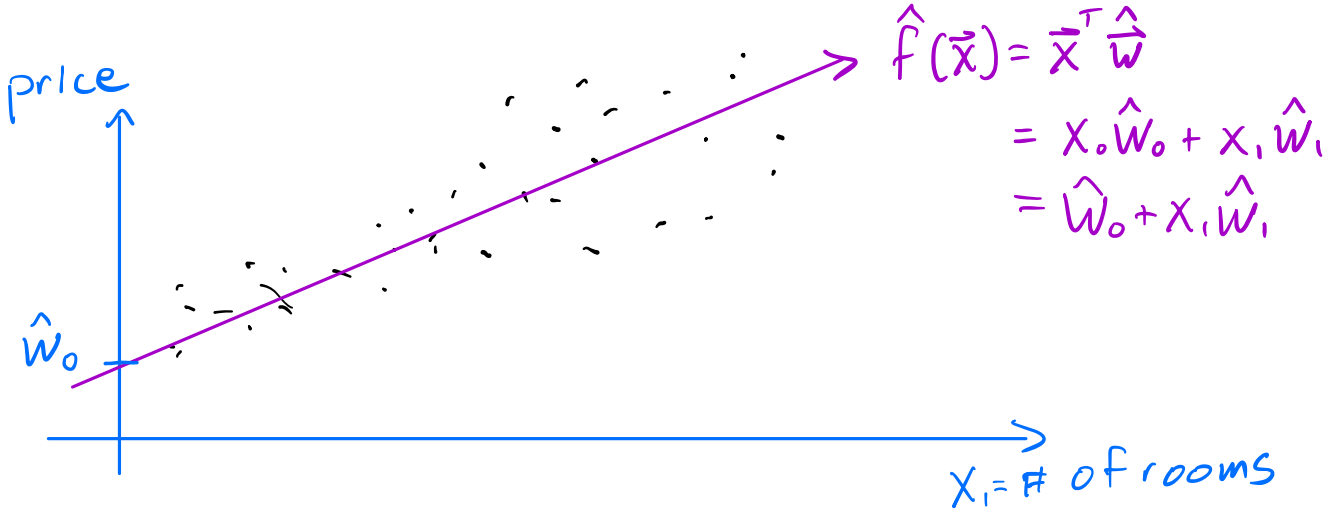
$b \leftarrow \sum_{i=1}^{n} \vec{x}_i y_i$

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

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

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

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



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

$= x_0 \hat{w}_0 + x_1 \hat{w}_1$

$= \hat{w}_0 + x_1 \hat{w}_1$

Gradient Descent

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

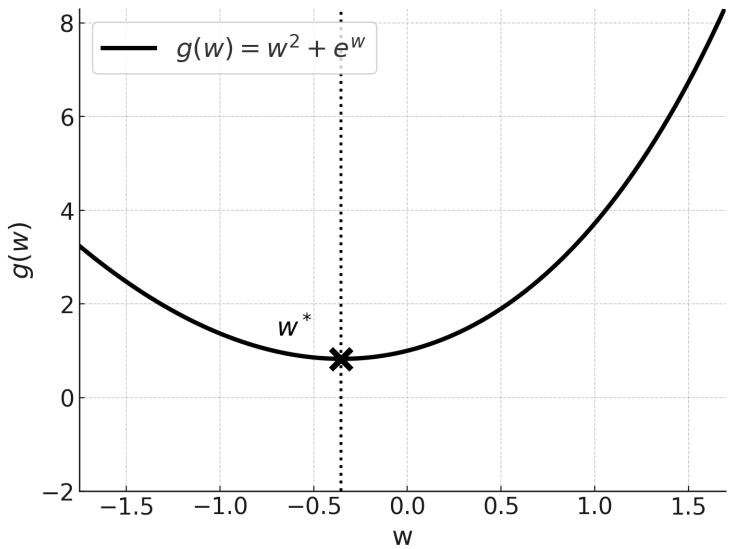
Convex

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

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

$$2w = -e^w$$

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 $t \to \infty$

First-order Gradient Descent

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

Multivariate Gradient Descent

Objective:

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

gradient descent update rule:

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

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: $\eta^{(t)} = \frac{\eta}{1+\lambda t}$ 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: $\eta^{(t)} = \frac{\eta}{\varepsilon + \|\nabla g(\vec{w}^{(t)})\|}$ 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 $\mathbb{R}^{d+1}$

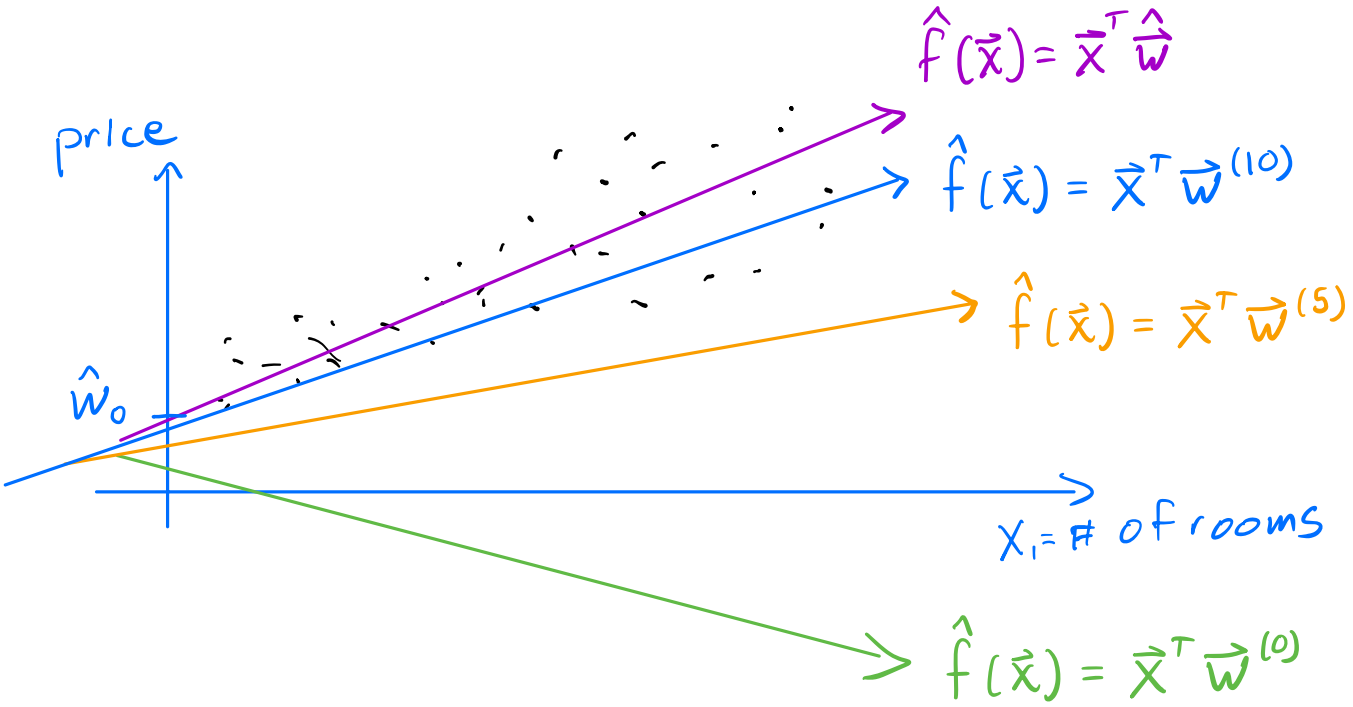
for $t = 1, \ldots, T$

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

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

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

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

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

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

---

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

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

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

for $t = 1, \ldots, T$

  Randomly shuffle D

  for $m = 1, \ldots, M$

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

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

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"

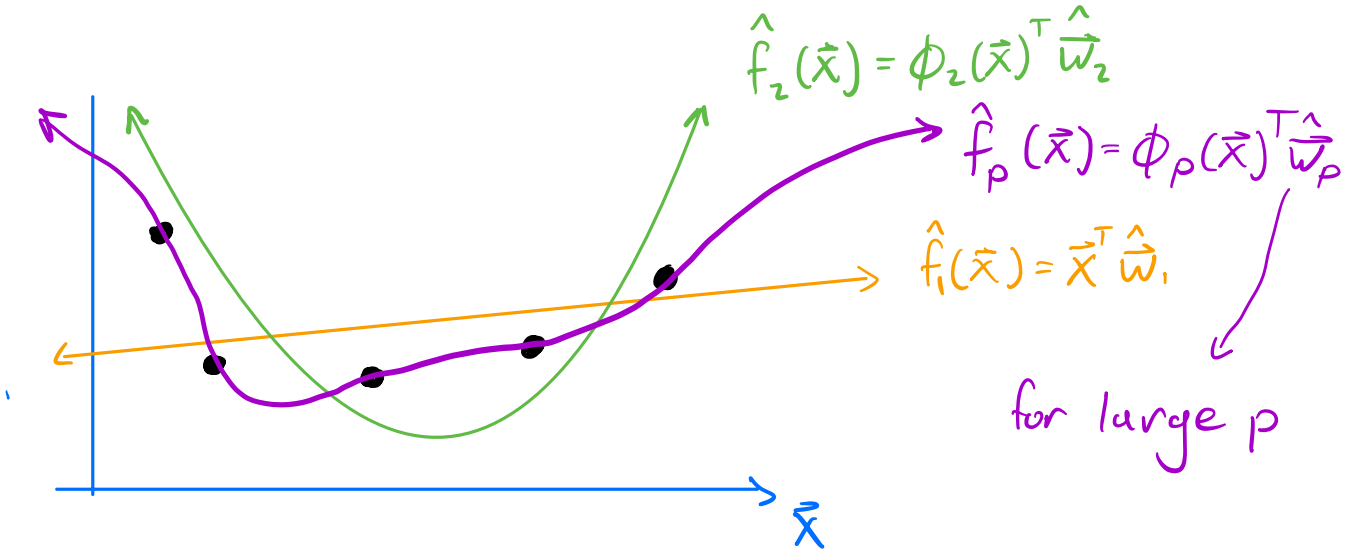
For $\mathcal{X} = \mathbb{R}^{d+1}$, $\mathcal{Z} = \mathbb{R}^{\bar{p}}$ where $\bar{p} = \binom{(d+1)+p-1}{p} = \binom{d+p}{p}$

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

$$\mathcal{F}_1 \subsetneq \mathcal{F}_2 \subsetneq \cdots \subsetneq \mathcal{F}_p$$

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

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 $f \in \mathcal{F}$ that minimizes $\hat{L}(f)$

$\mathcal{F}$: 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 $\mathcal{A}(D) = f_D$ 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 $\mathbb{E}[L(\mathcal{A}(D))]$

$$f_{Bayes} = \underset{f \in \{f \mid f: x \to y\}}{\operatorname{argmin}} L(f)$$

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

$\mathcal{F}$ any function class

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

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

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

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

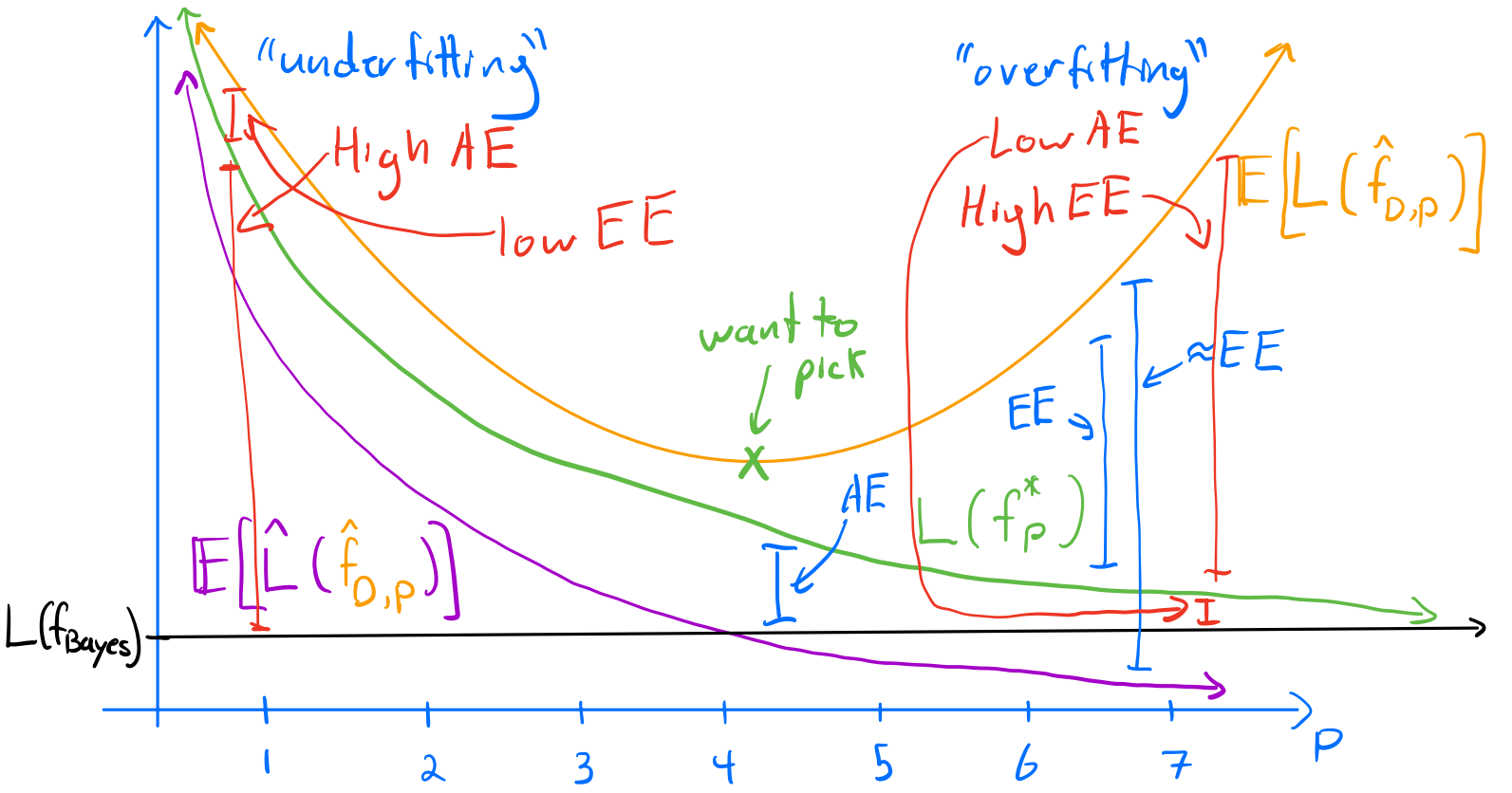
Estimation Error: Due to random dataset D

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

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

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

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



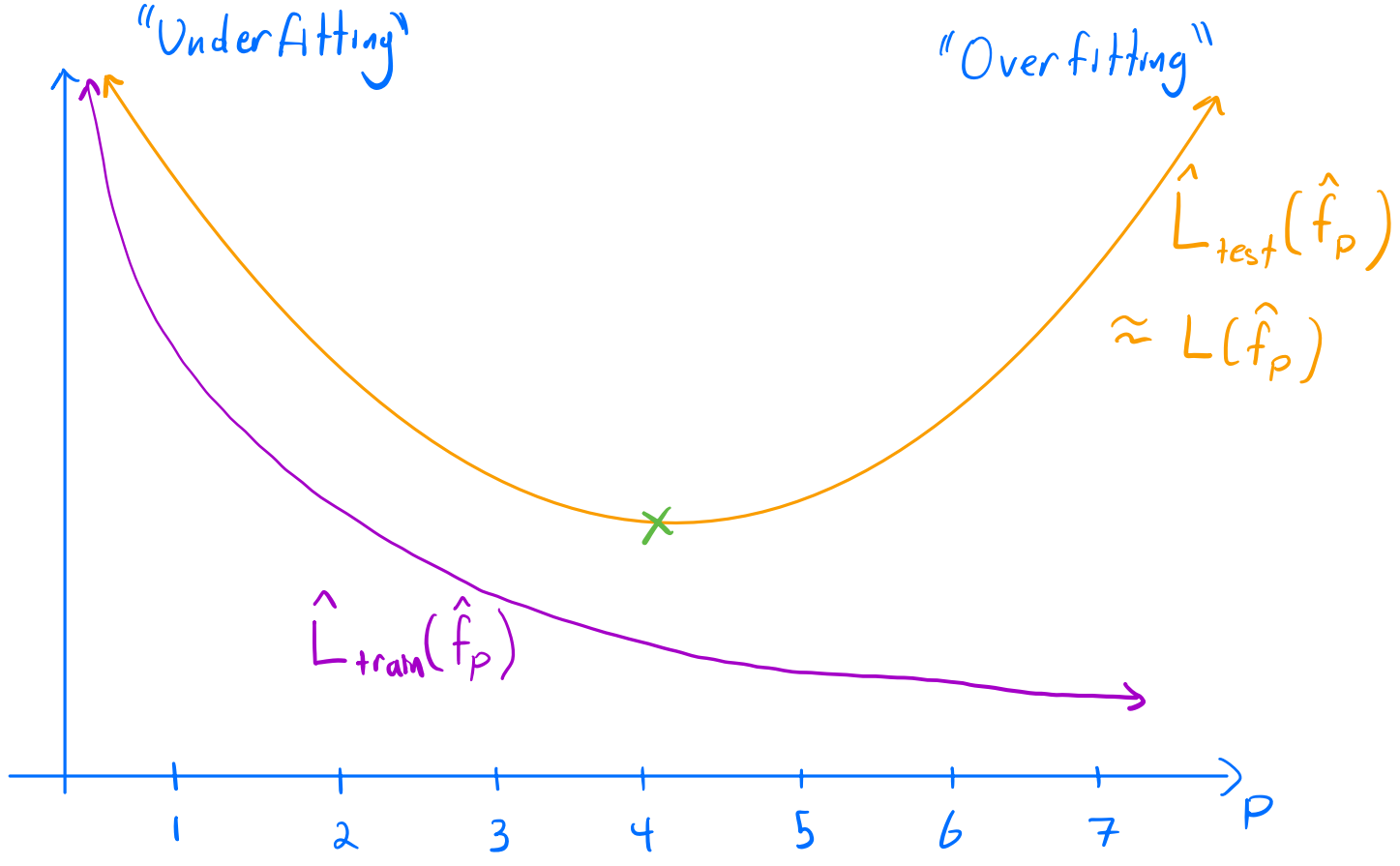
Can't calculate L, so use train, test split

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

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

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

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



Bias-Variance Tradeoff

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

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

$$= \underbrace{\mathbb{E}\left[\mathbb{E}\left[\left(\hat{f}_D(\vec{x}) - \bar{f}(\vec{x})\right)^2 \middle| \vec{x}\right]\right]}_{= EE} + \underbrace{\mathbb{E}\left[\left(\bar{f}(\vec{x}) - f_{Bayes}(\vec{x})\right)^2\right]}_{= AE} + \underbrace{L(f_{Bayes})}_{IE}$$

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

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

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

Bias ↓ if $\mathcal{F}$ ↑

Variance ↓ if n ↑

↑ if $\mathcal{F}$ ↑

Regularization

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

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

Regularization: penalize large weights

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

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

- $\hat{L}_\lambda(f_p)$: Regularized estimated loss
- λ: Regularization parameter, $\lambda \in [0, \infty)$
- $\frac{\lambda}{n}\sum_{j=1}^{\bar{p}-1} w_j^2$: "Regularizer"
- $j=0$ not included

Bias vs. Variance

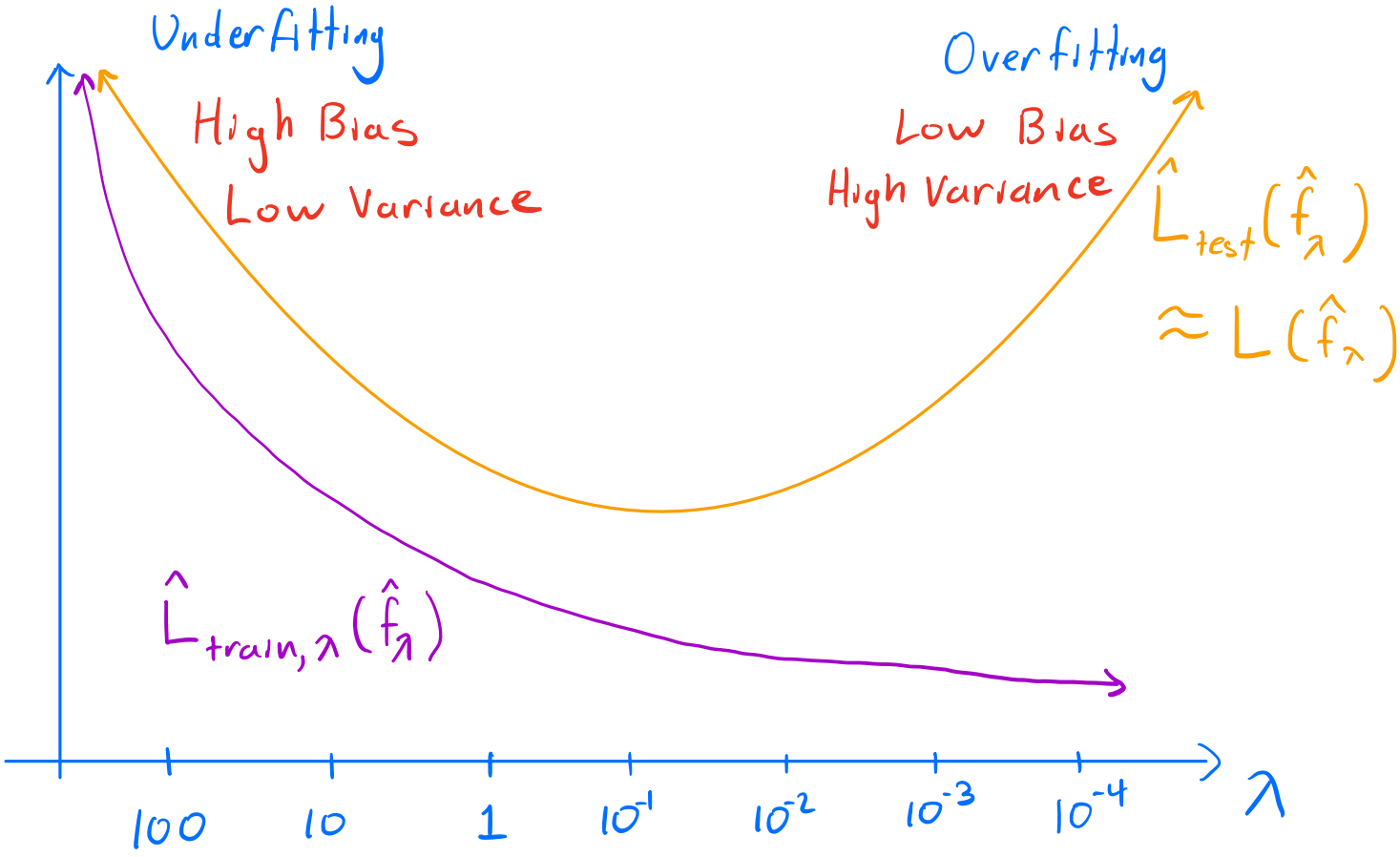
Bias: $\left(\bar{f}_\lambda(\vec{x}) - f_{Bayes}(\vec{x})\right)^2$

- Decreases if λ decreases

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

- Increases if λ decreases
- Decreases if n increases

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



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

Objective:

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

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

BGD:

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

MLE

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

assume squared loss and Regression

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

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

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

MLE Basics

$D = (Z_1, \ldots, Z_n)$ 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 $D = (z_1, \ldots, z_n)$

$$w_{MLE} = \underset{w \in \mathcal{W}}{\operatorname{argmax}} \overbrace{p(D|w)}^{\text{likelihood}}$$

$$= \underset{w \in \mathcal{W}}{\operatorname{argmin}} \overbrace{-\log(p(D|w))}^{\text{negative log-likelihood}}$$

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

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

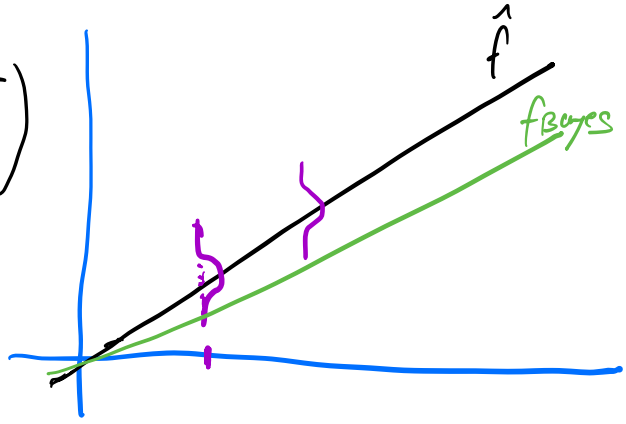
Estimating $P_{Y|\vec{X}}$

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

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

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

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



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

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

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

MAP

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

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

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

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

Bayes' Rule

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Lasso regression

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

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

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

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