ML COURSERA WEEK 6

LEO ROBINOVITCH

What to Try Next?

eg. implemented regularized lin. regression to predict housing prices – makes large errors on new set of houses! $\left(J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]\right)$

↳ POSSIBILITIES:
- Get more training data
- try smaller feature sets, additional features, polynomial features
- decrease / increase λ (play w/ under / overfitting)

⇒ SIMPLE TECHNIQUES: ML DIAGNOSTICS

↳ gain insight into what is / isn't working

↳ time costly, but helpful

Evaluating a Hypothesis:

⇒ low training error ≠ good $h_\theta(x)$! (overfitting)

⇒ split data into train + test (eg. 70/30)

$(x^{(i)}, y^{(i)})$ $(x_{test}^{(i)}, y_{test}^{(i)})$ ⇒ send randomly, "hold out" method

⇒ process: ① get $h_\theta(x)$ from training set

② compute test set error $J_{test}(\theta) = \frac{1}{2m_{test}}\sum_{i=1}^{m_{test}}(h_\theta(x_{test}^{(i)}) - y_{test}^{(i)})^2$

↳ for logistic regression, $J_{test}(\theta) = -\frac{1}{m_{test}}\left[\sum_{i=1}^{m_{test}} y_{test}^{(i)}\log(h_\theta(x_{test}^{(i)})) + (1-y_{test}^{(i)})\log(h_\theta(x_{test}^{(i)}))\right]$

↳ could also compute "misclassification error" for logistic reg.

$$err(h_\theta(x), y) = \begin{cases} 1 & \text{if } h_\theta(x) \geq 0.5,\ y=0 \\ & \text{or } h_\theta(x) < 0.5,\ y=1 \\ 0 & \text{otherwise} \end{cases} \Big\} \text{ error!}$$

$$\text{test error} = \frac{1}{m_{test}}\sum_{i=1}^{m_{test}} err\left[h_\theta(x_{test}^{(i)}), y_{test}^{(i)}\right)$$

Choosing λ: Model Selection Problems:

Choosing b/n: $h_\theta(x) = \theta_0 + \theta_1 x$ deg = 1

$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$ deg = 2

$\vdots$

deg = 10

=> what is best?

=> $\Theta^{(1)}$ is [2×1] (degree = 1)

$\Theta^{(2)}$ is [3×1] (degree = 2)

=> could compare $J(\theta^d)$'s! (test set error)

=> BAD comparison → NOT predictive for fitting to new models!

=> Better Method: ~60% data = training set

~20% data = cross validation set } split data

~20% data = test sit

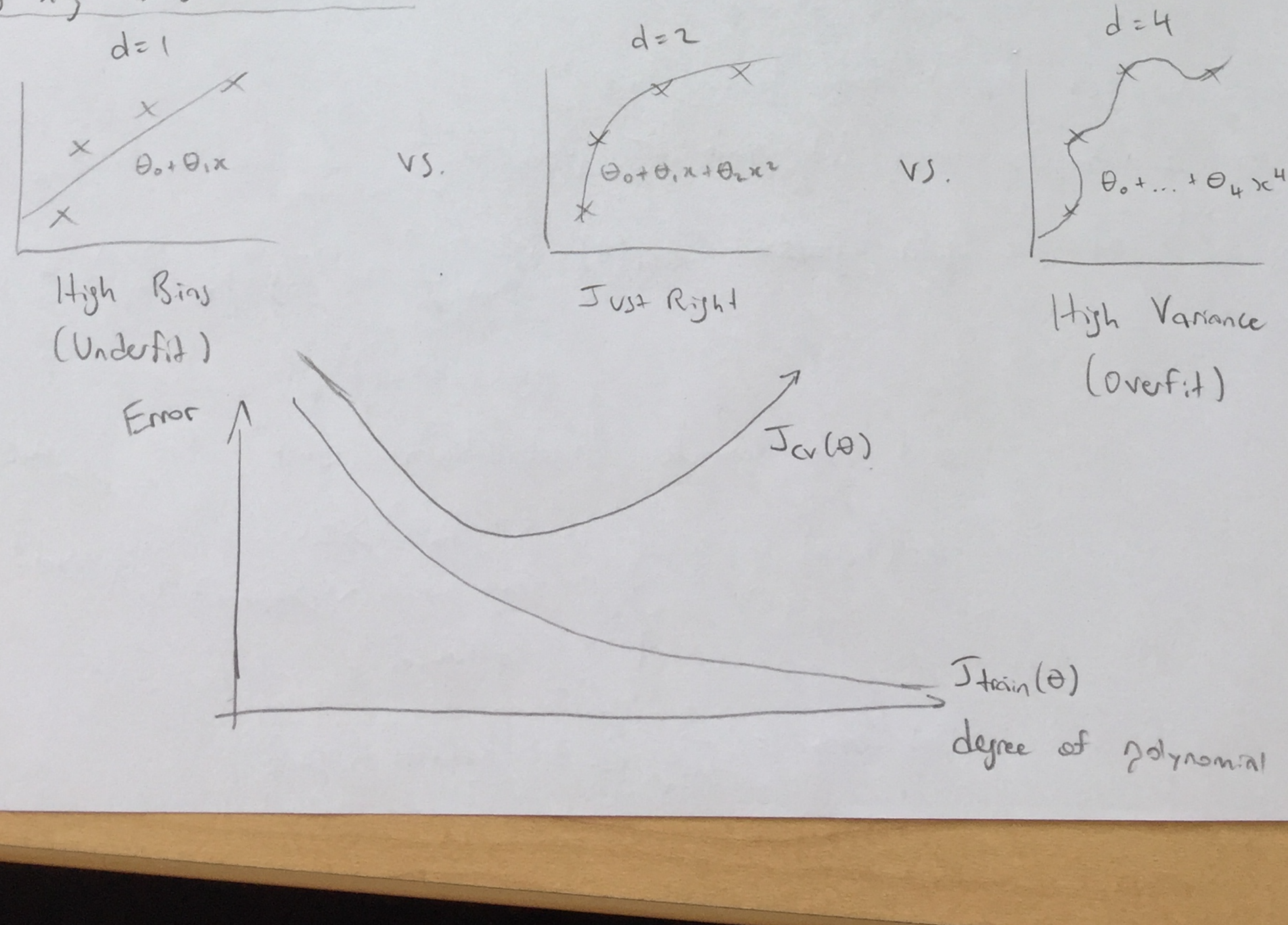
=> m_{cv} examples of $(x^{(i)}_{cv}, y^{(i)}_{cv})$, etc.

=> $J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$ & likewise w/ $J_{cv}(\theta)$, $J_{test}(\theta)$

Now can compare $J_{cv}(\theta^{(d)})$ → choose deg w/ lowest $J(\theta^{(d)})$

↳ report out general error using $J_{test}(\theta^d)$ (test set)

Diagnosing Bias & Variance:



=> if $J_{cv}(\theta)$ is high, are we over (variance) or under (bias) fitting?

↳ HIGH BIAS: J_{train} & J_{cv} both high ($J_{cv} \simeq J_{train}$)

HIGH VARIANCE: J_{train} low but J_{cv} high ($J_{cv} >> J_{train}$)

Regularization & Bias / Variance:

$$h_\theta(x) = \theta_0 + \dots \theta_d x^d$$

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m} \sum_{j=1}^{m} \theta_j^2$$

↳ ↑ λ → high bias (underfit)

mid λ → just right

↓ λ → high variance (overfit)

Note J_{train}, J_{cv}, J_{test} all same, just use m_{train}, m_{cv}, m_{test}

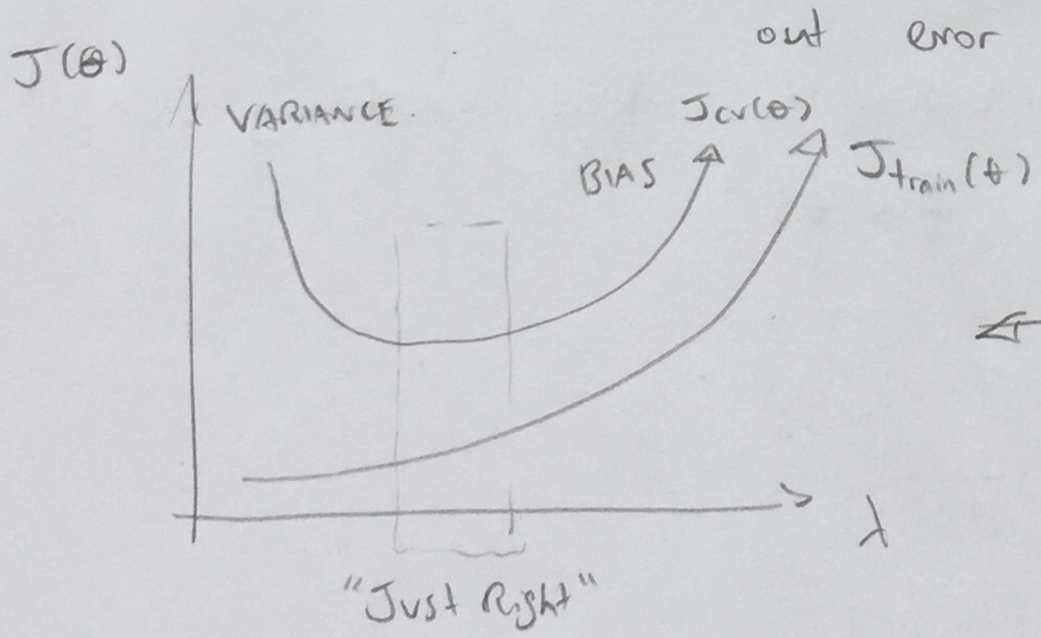
=> to choose λ; cycle through λ values (eg. 0, 0.02, 0.04 → 10.0)

$\lambda = 0$ → $J(\theta)$, set $\theta^{(1)}$ ← # of λ, not degree

$\lambda = 0.02$ → $J(\theta)$, set $\theta^{(2)}$

etc.

↳ pick λ that gives lowest $J_{cv}(\theta^{(\lambda)})$, report out error using $J_{test}(\theta^{(\lambda)})$



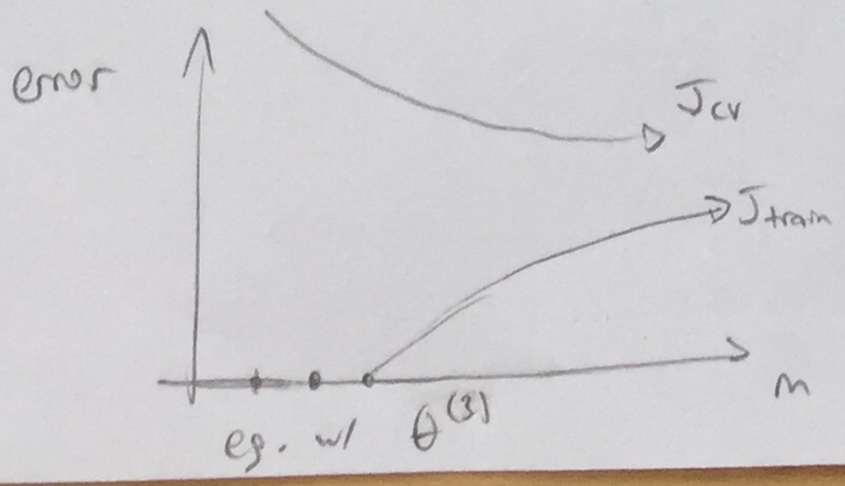
← Idealized Visualization.

Learning Curves:

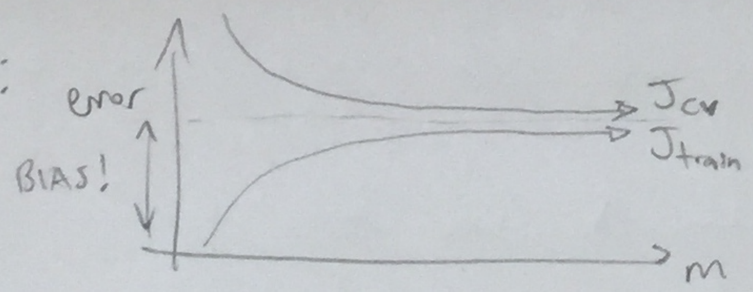
limit to m to get plot

$$J_{train}(\theta) = \frac{1}{2m_{train}} \sum_{i=1}^{m_{train}} [h_\theta(x^{(i)}) - y^{(i)}]^2$$

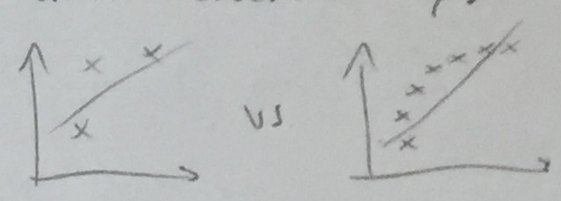
$$J_{cv} = \text{" (w/ } m_{cv})$$



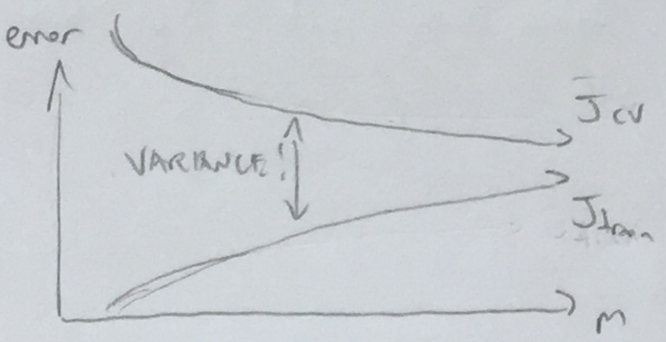
With high BIAS:



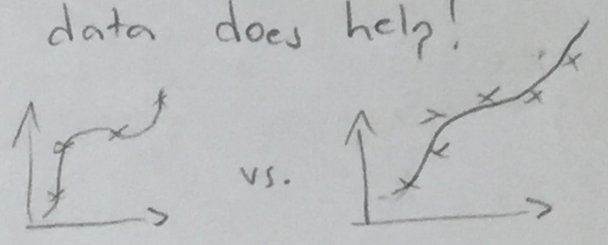
more data doesn't help!



high VARIANCE:



more data does help!



Summary:

- More training examples → helps for high variance (look for $J_{cv} > J_{train}$)
- Smaller feature set → fixes high variance (lower order d)
- Add features → fixes bias (increase order)
- Changing λ → $\downarrow\lambda$ for bias, $\uparrow\lambda$ for variance fix
- For NN's: ⇒ low hidden units, low hidden layers → "lower features"
  - ↳ prone to underfitting (bias) but computationally ↓$
  - ⇒ more units / layers → "higher features"
    - ↳ prone to overfitting (variance)
    - ↳ can use regularization to address overfitting (λ)!
  - ⇒ can compare NN's $J_{cv}(\theta)$!

Recommended Approach to Building Algorithms:

⇒ Start w/ simple algorithm → implement & test on CV data

⇒ Plot learning curves (train + test error vs. # training examples / # features) to see what may help

⇒ Manually examine examples that algorithm makes error on → see if pattern

⇒ Words: → stemming software (N.L.P.) helps distinguish roots/meanings
eg. discount, discounted, etc.
↳ can compare classification errors w/ or w/o stemming

→ upper/lowercase also compared

⇒ Always better to be quick n' dirty in initial implementation
↳ helps decide where to spend time later

⇒ **Skewed Classes:**

⇒ logistic/classification problem → find 1% test error
↳ what if only 0.5% of patients have cancer?! → $y=0$ does better than your algorithm!
↳ hard to tell if good model just by error...

⇒ Precision / Recall Metric:

predicted \ actual	1	0
1	True +	False +
0	False −	True −

$$\text{Precision} = \frac{\#\text{ true pos}}{\#\text{ pred. pos (true \& false +)}}$$

$$\text{Recall} = \frac{\#\text{ true pos}}{\#\text{ actual pos (true + \& false −)}}$$

⇒ Can't "cheat" by predicting only $y=0$, $y=1$
↳ algorithm w/ good precision & recall → good algorithm

Trading off Precision & Recall

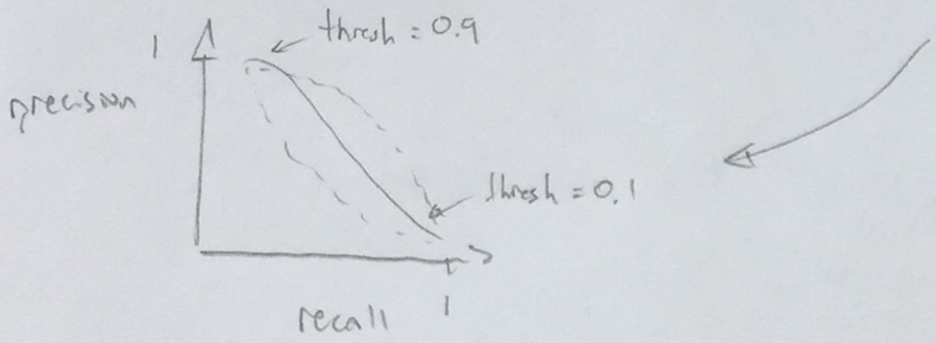
⇒ say want to predict $y=1$ (e.g. cancerous) only if very confident

↳ change $h_\theta(x)_{\text{log}}$ threshold from $0.5 \rightarrow 0.7$ (or 0.9)

⇒ HIGH PRECISION, LOW RECALL

⇒ vice versa, if want to be overcautious → $h_\theta(x)_{\text{log}}$ threshold $0.5 \rightarrow 0.3$

⇒ HIGH RECALL, LOW PRECISION



⇒ How to compare algorithms given their prec./recall? (2 #'s)

	precision:	recall:
Alg 1	0.5	0.4
Alg 2	0.7	0.1
Alg 3	0.02	1.0 ← predict $y=1$ all the time equivalent

↳ to get one comparison metric:

F_1 Score: $F_1 = 2\frac{PR}{P+R}$ $\quad F_1 \in [0,1]$

lower than either P or R

↳ compare F_1 scores of P & R on CV. set!

Large Data Rationale: (better to have more data assumption)

↳ many parameters → J_{train} small ┐
↳ large training set → $J_{train} \simeq J_{cv}$ ┘→ J_{cv} small!

⇒ features x must contain enough info to correlate/predict y,
regardless of # parameters!