CSE 446
Bias-Variance & Naïve Bayes

Administrative

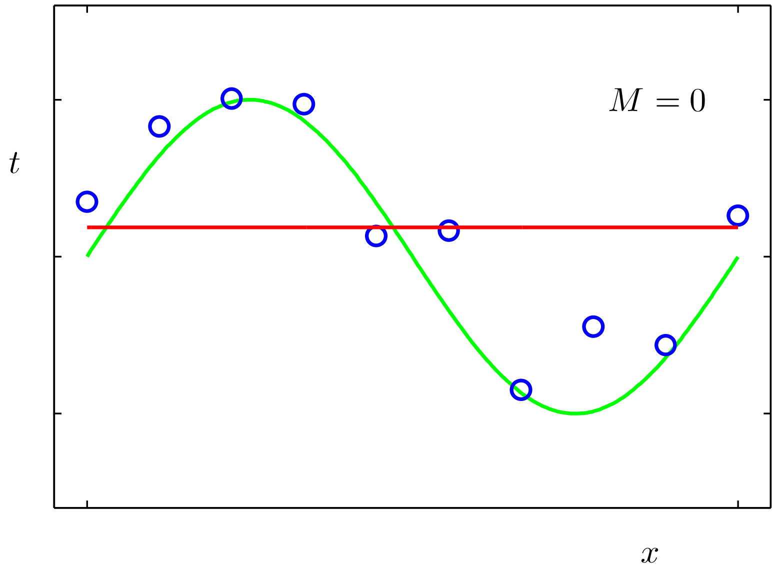
- Homework 1 due next week on **Friday**
  - Good to finish early
- Homework 2 is out on **Monday**
  - Check the course calendar
  - Start early (midterm is right before Homework 2 is due!)

Today

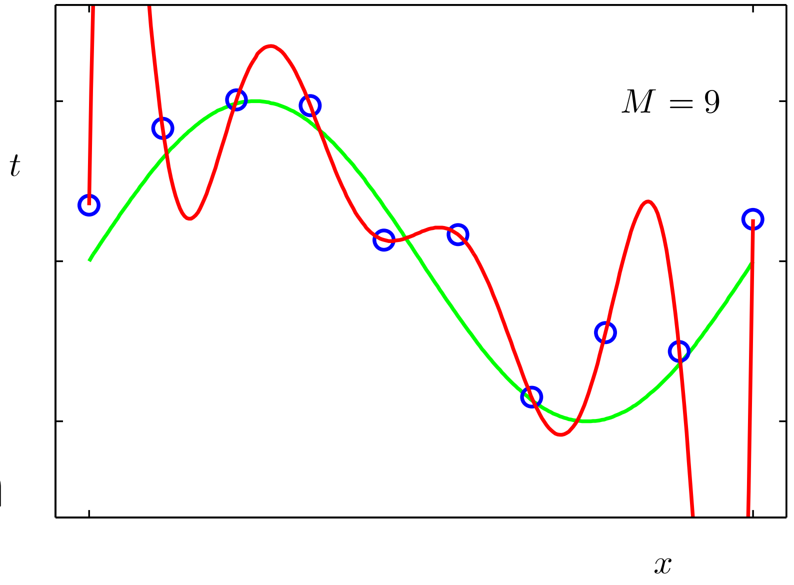
- Finish linear regression: discuss bias & variance tradeoff
  - Relevant to other ML problems, but will discuss for linear regression in particular
- Start on Naïve Bayes
  - Probabilistic classification method

Bias-Variance tradeoff – Intuition

- **Model too simple:** does not fit the data well
  - A *biased* solution
  - Simple = fewer features
  - Simple = more regularization

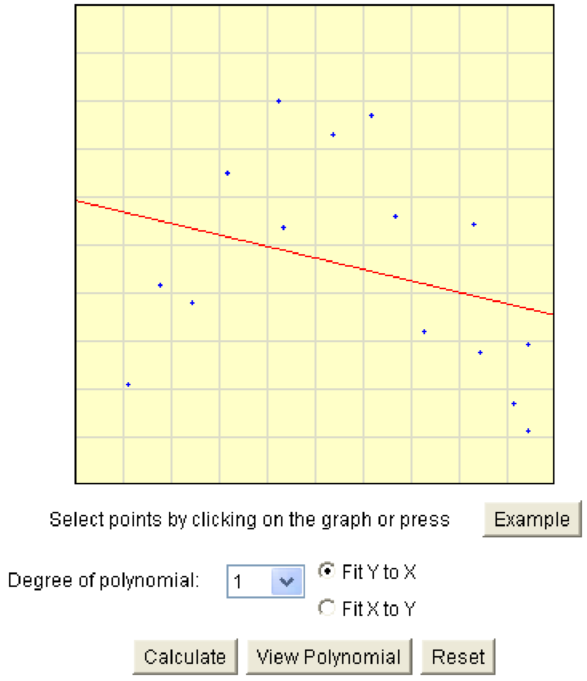


- **Model too complex:** small changes to the data, solution changes a lot
  - A *high-variance* solution
  - Complex = more features
  - Complex = less regularization



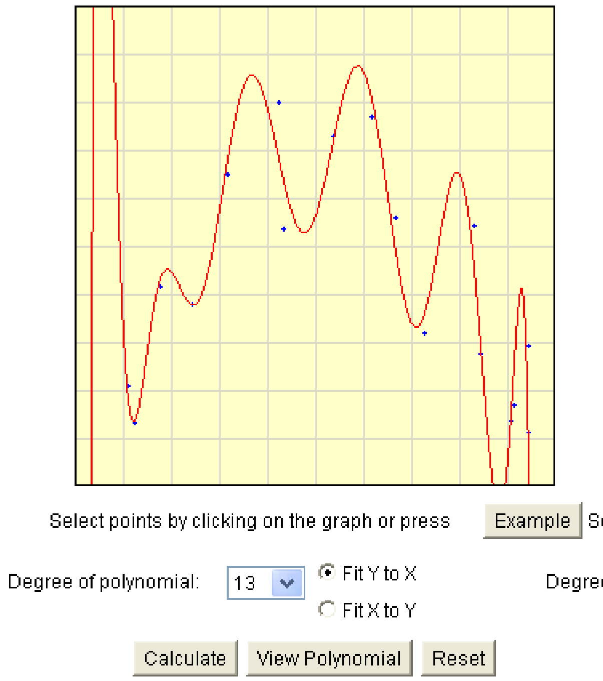
Bias-Variance Tradeoff

- Choice of hypothesis class introduces learning bias
  - More complex class → less bias
  - More complex class → more variance


Select points by clicking on the graph or press

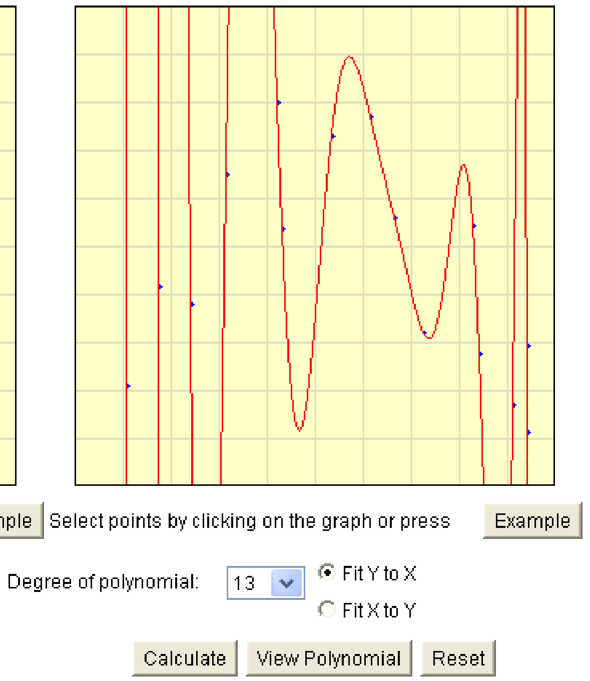
Degree of polynomial: 1

[✓ Fit Y to X] Fit X to Y


Select points by clicking on the graph or press

Degree of polynomial: 13

[✓ Fit Y to X] Fit X to Y


Select points by clicking on the graph or press

Degree of polynomial: 13

[✓ Fit Y to X] Fit X to Y

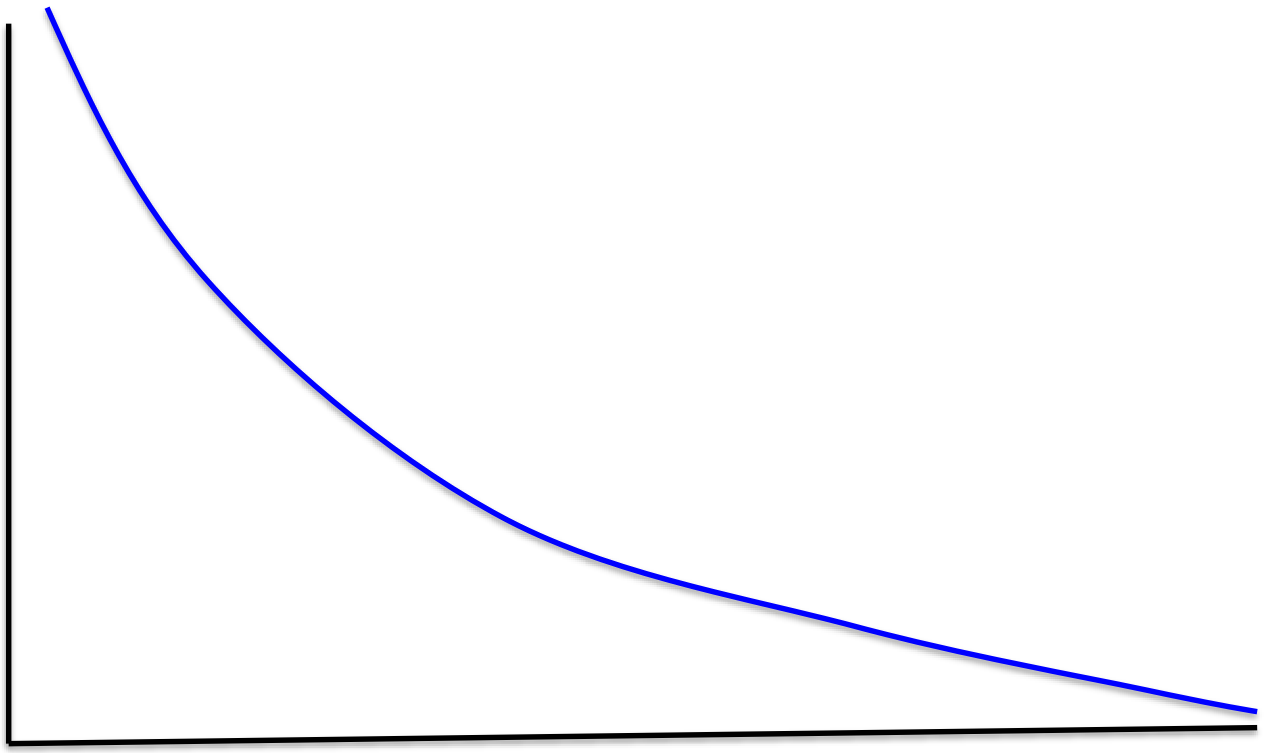
Training set error

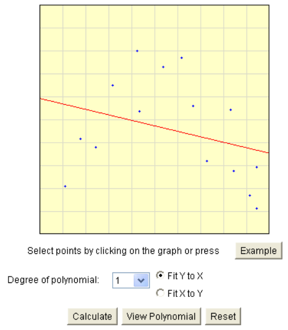
- Given a dataset (Training data)
- Choose a loss function
  - e.g., squared error (L_2) for regression
- Training error: For a particular set of parameters, loss function on training data:

$$\mathcal{E}_{\text{train}}(w) = \frac{1}{N_{\text{train}}} \sum_{i=1}^{N_{\text{train}}} (x_i \cdot w - y_i)^2$$

Training error as a function of model complexity

$$\mathcal{E}_{\text{train}}(w) = \frac{1}{N_{\text{train}}} \sum_{i=1}^{N_{\text{train}}} (x_i \cdot w - y_i)^2$$

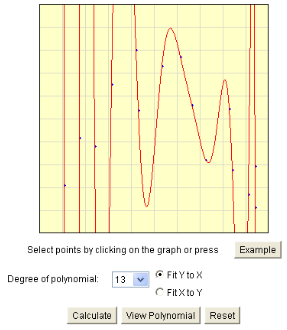




Select points by clicking on the graph or press

Degree of polynomial: 1

[✓ Fit Y to X] Fit X to Y

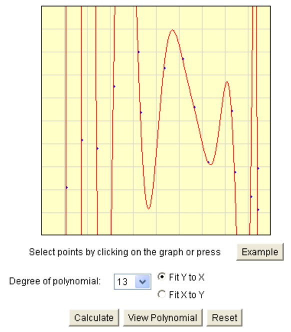


Select points by clicking on the graph or press

Degree of polynomial: 13

[✓ Fit Y to X] Fit X to Y

Prediction error



Degree of polynomial: 13

[✓ Fit Y to X] Fit X to Y

- Training set error can be poor measure of "quality" of solution
- Prediction error (true error): We really care about error over all possibilities:

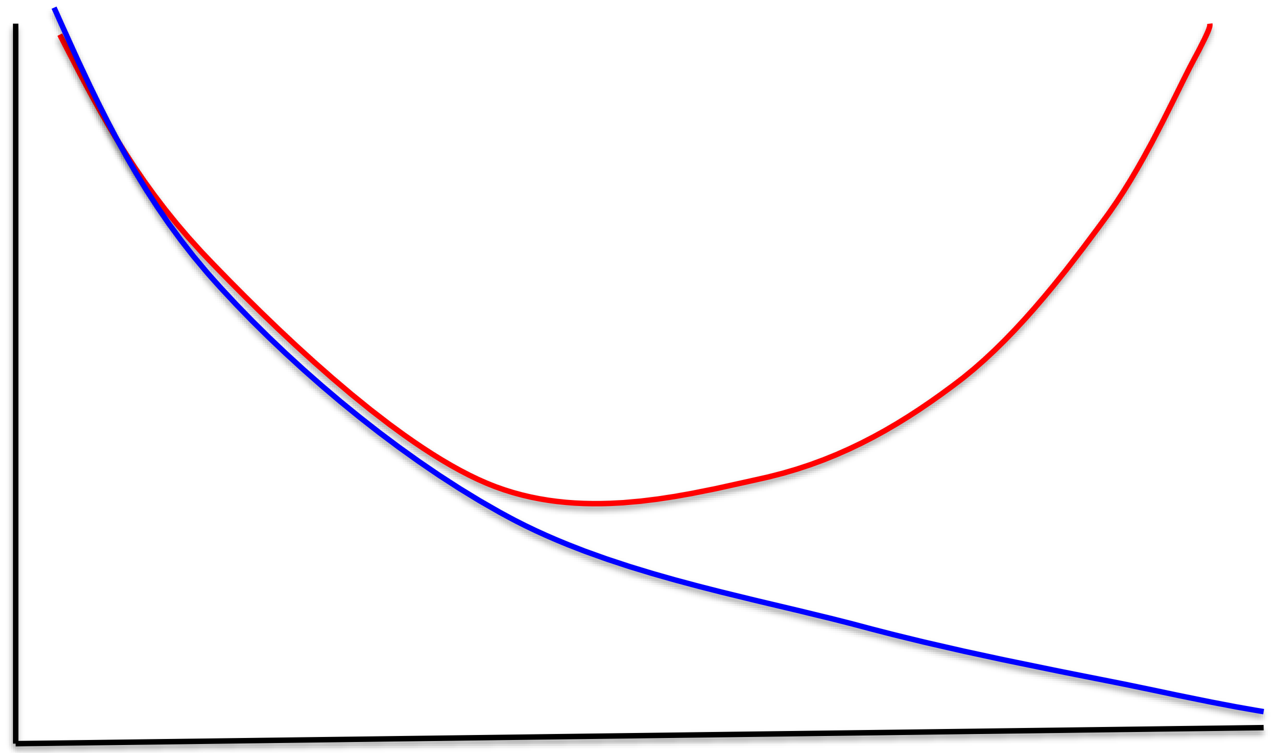
$$\mathcal{E}_{\text{true}}(w) = E_{p(x)}\left[(x_i \cdot w - y_i)^2\right]$$

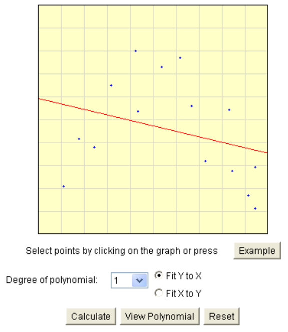
$$= \int p(x)\,(x_i \cdot w - y_i)^2\,dx$$

Prediction error as a function of model complexity

$$\mathcal{E}_{\text{train}}(w) = \frac{1}{N_{\text{train}}}\sum_{i=1}^{N_{\text{train}}} (x_i \cdot w - y_i)^2$$

$$\mathcal{E}_{\text{true}}(w) = E_{p(x)}\left[(x_i \cdot w - y_i)^2\right]$$

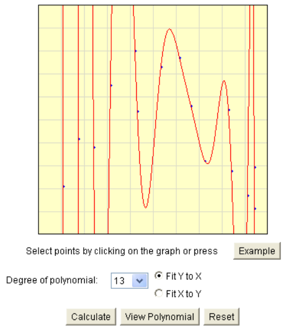




Select points by clicking on the graph or press Example

Degree of polynomial: 1 [✓ Fit Y to X] Fit X to Y

Calculate View Polynomial Reset



Select points by clicking on the graph or press Example

Degree of polynomial: 13 [✓ Fit Y to X] Fit X to Y

Calculate View Polynomial Reset

Computing prediction error

- To correctly predict error
  - Hard integral!
  - May not know y for every **x,** may not know p(x)

$$\mathcal{E}_{\text{true}}(w) = \int p(x)\left(x_i \cdot w - y_i\right)^2 dx$$

- Monte Carlo integration (sampling approximation)
  - Sample a set of i.i.d. points {$\mathbf{x}_1,\ldots,\mathbf{x}_M$} from p(**x**)
  - Approximate integral with sample average

$$\mathcal{E}_{\text{true}}(w) \approx \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \left(x_i \cdot w - y_i\right)^2$$

Why training set error doesn't approximate prediction error?

- Sampling approximation of prediction error:

$$\mathcal{E}_{\text{true}}(w) \approx \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} (x_i \cdot w - y_i)^2$$

- Training error :

$$\mathcal{E}_{\text{train}}(w) = \frac{1}{N_{\text{train}}} \sum_{i=1}^{N_{\text{train}}} (x_i \cdot w - y_i)^2$$

- Very similar equations
  - Why is training set a bad measure of prediction error?

Why training set error doesn't approximate prediction error?

- Sampling approximation of prediction error:

$$\mathcal{E}_{\text{true}}(w) \approx \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} (x_i \cdot w - y_i)^2$$

- Training error :

$$\mathcal{E}_{\text{train}}(w) = \frac{1}{N_{\text{train}}} \sum^{N_{\text{train}}} (x_i \cdot w - y_i)^2$$

- Very
  - W

w was optimized with respect to the training error!

Training error is a (optimistically) biased estimate of prediction error

Test set error

- Given a dataset, **randomly** split it into two parts:
  - Training data – {$\mathbf{x}_1$,…, $\mathbf{x}_{\text{Ntrain}}$}
  - Test data – {$\mathbf{x}_1$,…, $\mathbf{x}_{\text{Ntest}}$}
- Use training data to optimize parameters **w**
- Test set error: For the ***final solution*** **w***, evaluate the error using:

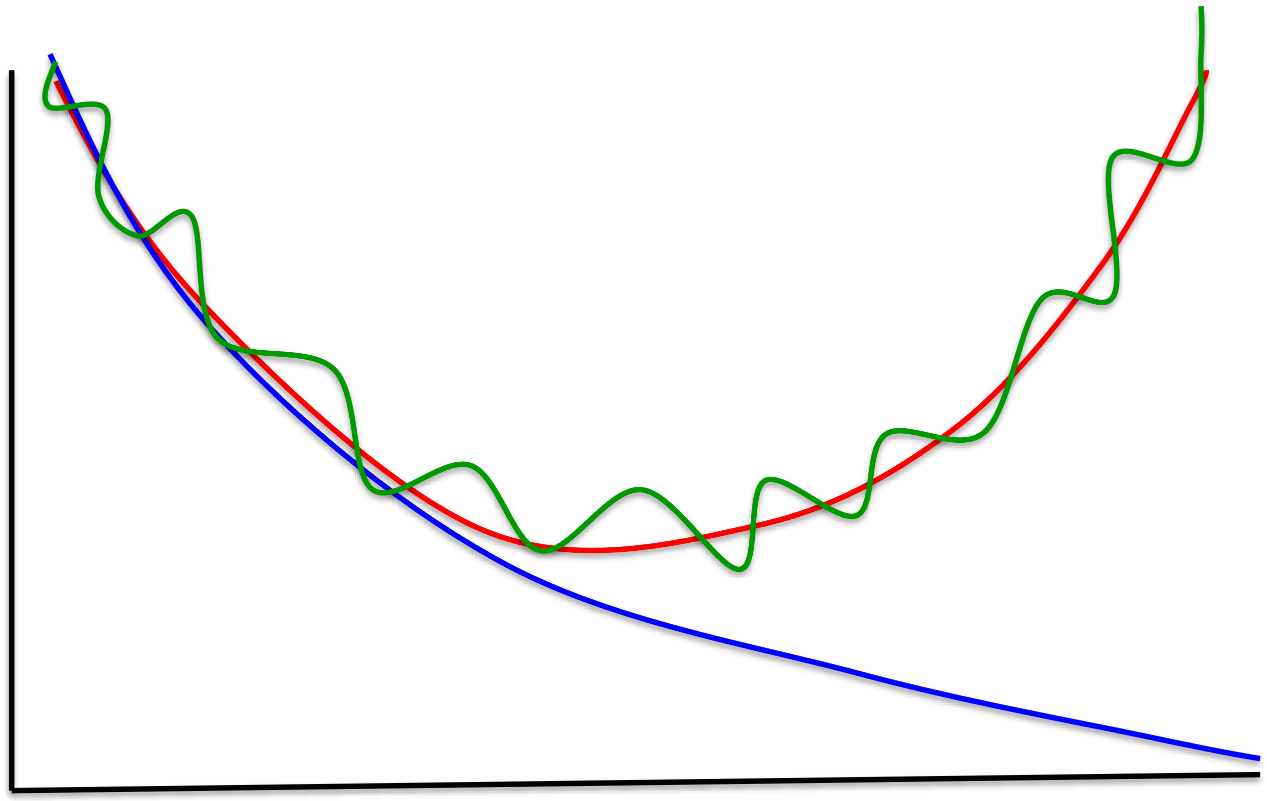
$$\mathcal{E}_{\text{test}}(w) = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} (x_i \cdot w - y_i)^2$$

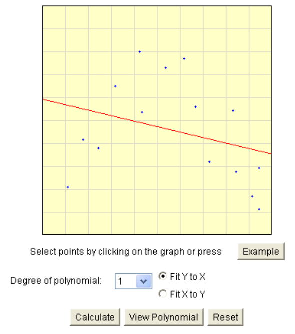
Test set error as a function of model complexity

$$\boxed{\mathcal{E}_{\text{train}}(w) = \frac{1}{N_{\text{train}}}\sum_{i=1}^{N_{\text{train}}}(x_i\cdot w - y_i)^2}$$

$$\boxed{\mathcal{E}_{\text{true}}(w) = E_{p(x)}\left[(x_i\cdot w - y_i)^2\right]}$$

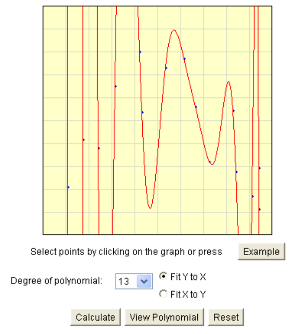
$$\boxed{\mathcal{E}_{\text{test}}(w) = \frac{1}{N_{\text{test}}}\sum_{i=1}^{N_{\text{test}}}(x_i\cdot w - y_i)^2}$$




Select points by clicking on the graph or press Example

Degree of polynomial: 1 [✓ Fit Y to X] Fit X to Y

Calculate | View Polynomial | Reset


Select points by clicking on the graph or press Example

Degree of polynomial: 13 [✓ Fit Y to X] Fit X to Y

Calculate | View Polynomial | Reset

Overfitting (again)

- Assume:
  - Data generated from distribution $D(X,Y)$
  - A hypothesis space H
- Define: errors for hypothesis $h \in H$
  - Training error: $error_{train}(h)$
  - Data (true) error: $error_{true}(h)$

- We say h ***overfits*** the training data if there exists an $h' \in H$ such that:

$$error_{train}(h) < error_{train}(h')$$

and

$$error_{true}(h) > error_{true}(h')$$

Summary: error estimators

- Gold Standard:

$$\mathcal{E}_{\text{true}}(w) = \int p(x)\,(x_i \cdot w - y_i)^2\, dx$$

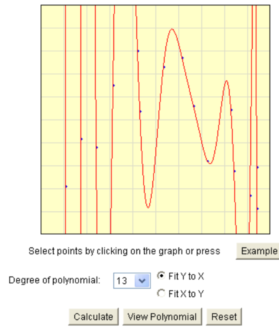
- Training: optimistically biased

$$\mathcal{E}_{\text{train}}(w) = \frac{1}{N_{\text{train}}} \sum_{i=1}^{N_{\text{train}}} (x_i \cdot w - y_i)^2$$

- Test: our final measure

$$\mathcal{E}_{\text{test}}(w) = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} (x_i \cdot w - y_i)^2$$

Error as a function of number of training examples for a fixed model complexity


Select points by clicking on the graph or press Example

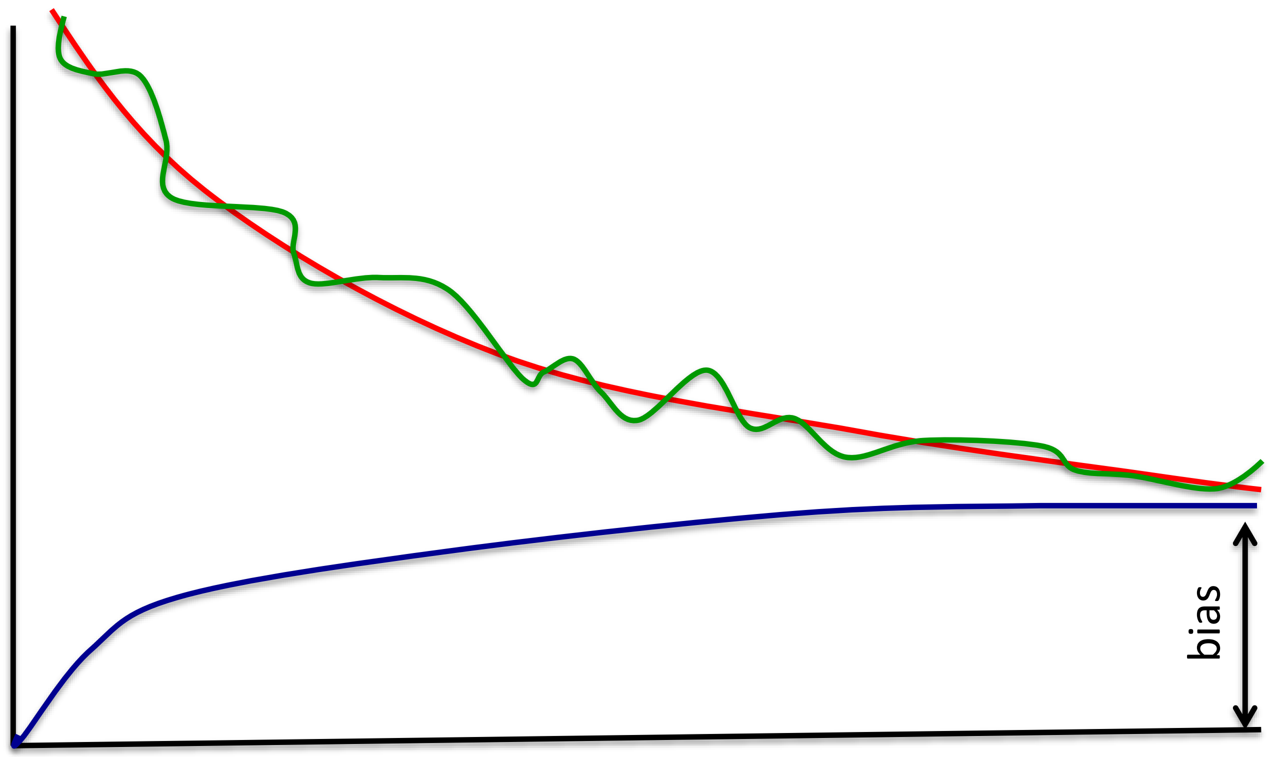
Degree of polynomial: 13

[✓ Fit Y to X] Fit X to Y

$$\mathcal{E}_{\text{train}}(w) = \frac{1}{N_{\text{train}}}\sum_{i=1}^{N_{\text{train}}}(x_i \cdot w - y_i)^2$$

$$\mathcal{E}_{\text{true}}(w) = E_{p(x)}\left[(x_i \cdot w - y_i)^2\right]$$

$$\mathcal{E}_{\text{test}}(w) = \frac{1}{N_{\text{test}}}\sum_{i=1}^{N_{\text{test}}}(x_i \cdot w - y_i)^2$$



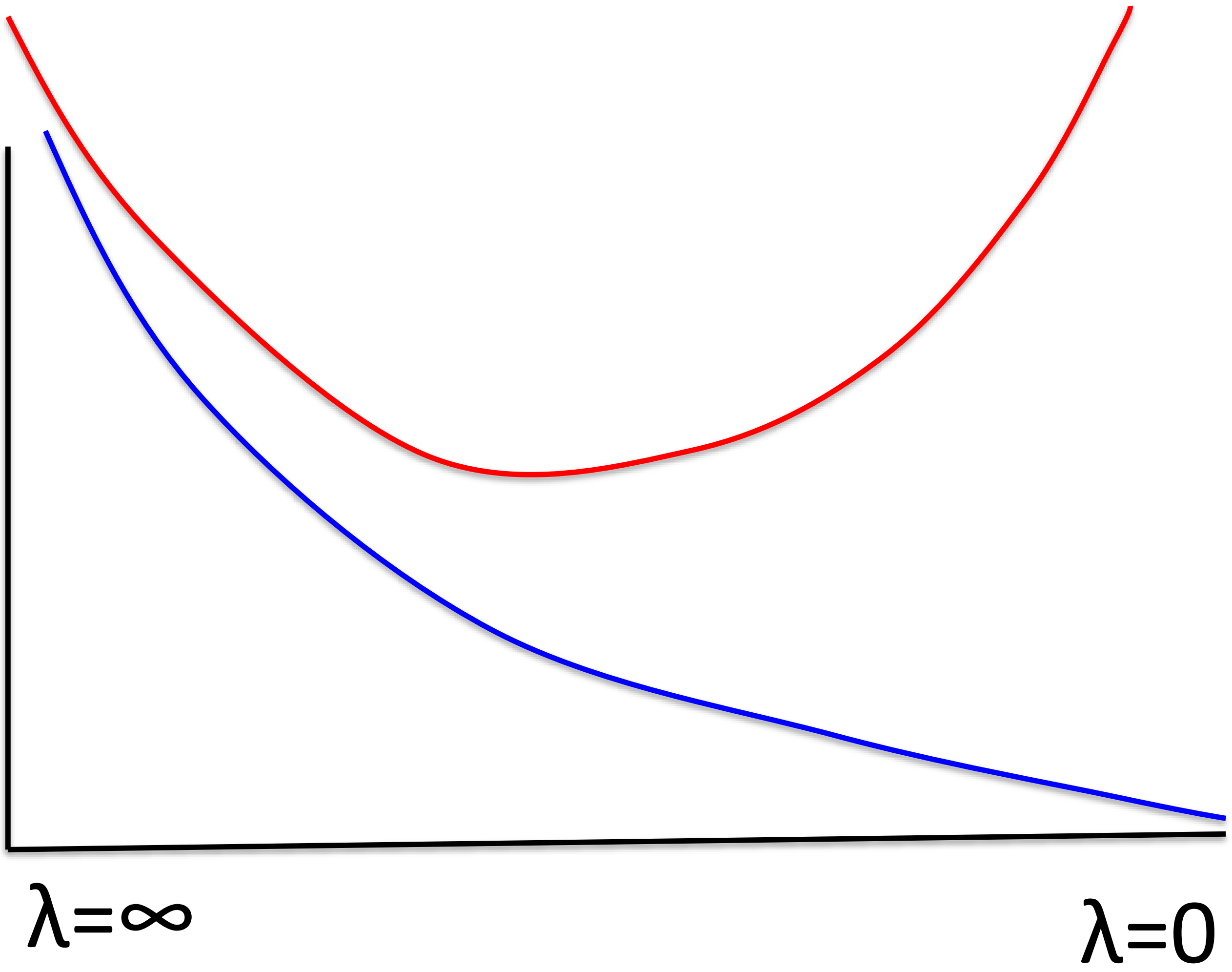
little data

infinite data

Error as function of regularization parameter, fixed model complexity

$$\mathcal{E}_{\text{train}}(w) = \frac{1}{N_{\text{train}}}\sum_{i=1}^{N_{\text{train}}}(x_i \cdot w - y_i)^2$$

$$\mathcal{E}_{\text{true}}(w) = E_{p(x)}\left[(x_i \cdot w - y_i)^2\right]$$



Summary: error estimators

- Gold Standard:
- Trai
- Test:

Be careful

Test set only unbiased if you never do any learning on the test data

If you need to select a hyperparameter, or the model, or anything at all, use the validation set (also called a holdout set, development set, etc.)

$$\mathcal{E}_{\text{test}}(w) = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} (x_i \cdot w - y_i)^2$$

What you need to know (linear regression)

- Regression
  - Basis function/features
  - Optimizing sum squared error
  - Relationship between regression and Gaussians
- Regularization
  - Ridge regression math & derivation as MAP
  - LASSO formulation
  - How to set lambda (hold-out, K-fold)
- Bias-Variance trade-off

Back to Classification

- Given: Training set $\{(x_i, y_i) \mid i = 1 \ldots n\}$
- Find: A good approximation to $f : X \rightarrow Y$

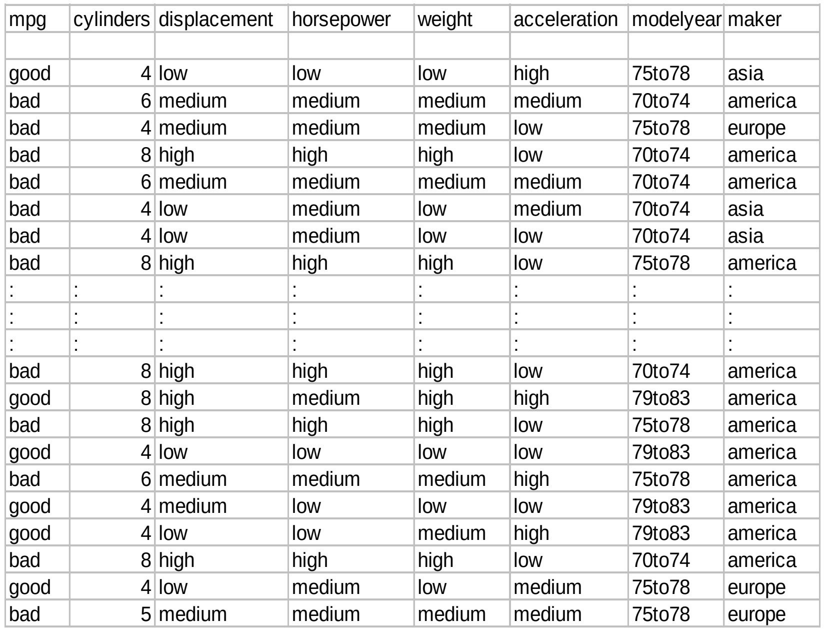
Examples: what are X and Y ?

- Spam Detection
  - Map email to {Spam,Ham}
- Digit recognition
  - Map pixels to {0,1,2,3,4,5,6,7,8,9}
- Stock Prediction
  - Map new, historic prices, etc. to Â (the real numbers)

Classification

Can we Frame Classification as MLE?

- In linear regression, we learn the conditional P(Y|X)
- Decision trees also model P(Y|X)
- P(Y|X) is complex (hence decision trees cannot be built optimally, but only greedily)
- What if we instead model P(X|Y)?
- [see lecture notes]



mpg	cylinders	displacement	horsepower	weight	acceleration	modelyear	maker
good	4	low	low	low	high	75to78	asia
bad	6	medium	medium	medium	medium	70to74	america
bad	4	medium	medium	medium	low	75to78	europe
bad	8	high	high	high	low	70to74	america
bad	6	medium	medium	medium	medium	70to74	america
bad	4	low	medium	low	medium	70to74	asia
bad	4	low	medium	low	low	70to74	asia
bad	8	high	high	high	low	75to78	america
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
bad	8	high	high	high	low	70to74	america
good	8	high	medium	high	high	79to83	america
bad	8	high	high	high	low	75to78	america
good	4	low	low	low	low	79to83	america
bad	6	medium	medium	medium	high	75to78	america
good	4	medium	low	low	low	79to83	america
good	4	low	low	medium	high	79to83	america
bad	8	high	high	high	low	70to74	america
good	4	low	medium	low	medium	75to78	europe
bad	5	medium	medium	medium	medium	75to78	europe

MLE for the parameters of NB

- Given dataset
  - Count(A=a,B=b): number of examples with A=a and B=b
- MLE for discrete NB, simply:
  - Prior:

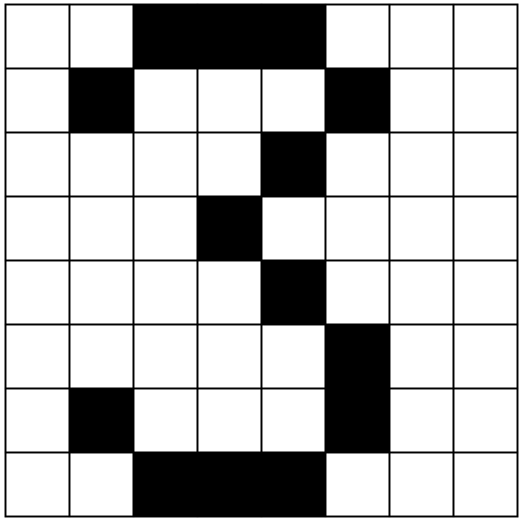
$$p(y=j) = \frac{\text{Count}(y=j)}{\sum_{j'} \text{Count}(y=j')}$$

  - Likelihood:

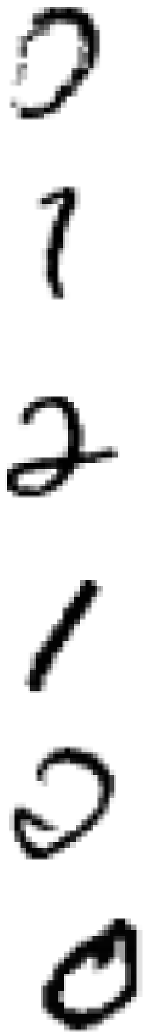
$$p(x_k=\ell|y=j) = \frac{\text{Count}(x_k=\ell \text{ and } y=j)}{\sum_{\ell'} \text{Count}(x_k=\ell' \text{ and } y=j')}$$

A Digit Recognizer

- Input: pixel grids



- Output: a digit 0-9



Naïve Bayes for Digits (Binary Inputs)

- Simple version:
  - One feature F_{ij} for each grid position <i,j>
  - Possible feature values are on / off, based on whether intensity is more or less than 0.5 in underlying image
  - Each input maps to a feature vector, e.g.

$$\rightarrow \langle F_{0,0} = 0 \;\; F_{0,1} = 0 \;\; F_{0,2} = 1 \;\; F_{0,3} = 1 \;\; F_{0,4} = 0 \;\; \ldots F_{15,15} = 0 \rangle$$

  - Here: lots of features, each is binary valued
- Naïve Bayes model:

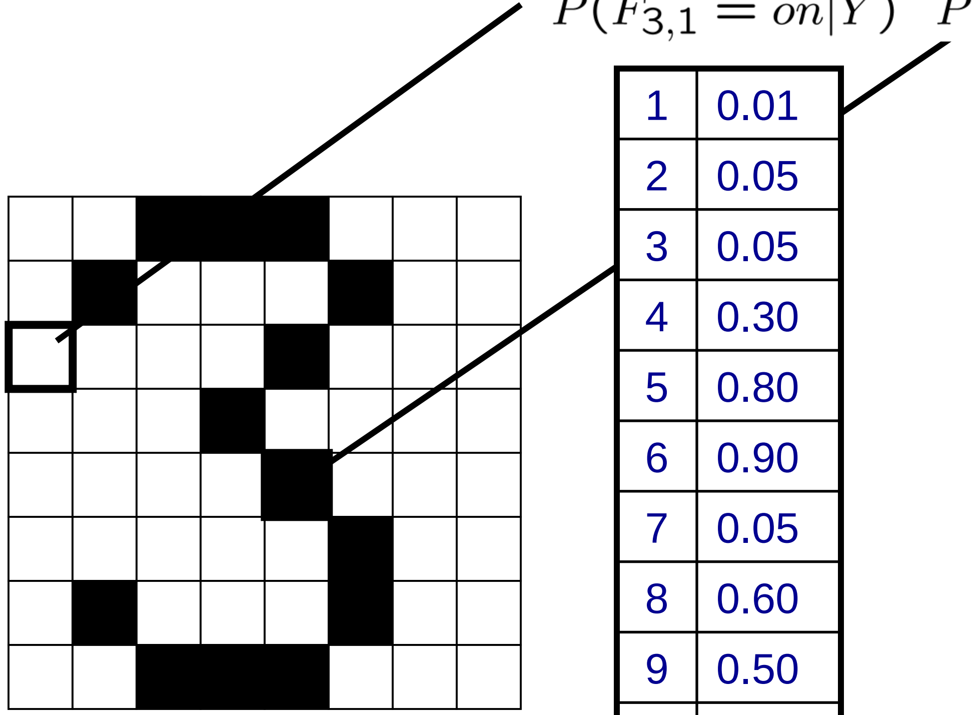
$$P(Y|F_{0,0} \ldots F_{15,15}) \propto P(Y) \prod_{i,j} P(F_{i,j}|Y)$$

- Are the features independent given class?
- What do we need to learn?

Example Distributions

$P(Y)$

1	0.1
2	0.1
3	0.1
4	0.1
5	0.1
6	0.1
7	0.1
8	0.1
9	0.1
0	0.1



$P(F_{3,1} = on|Y)$

1	0.01
2	0.05
3	0.05
4	0.30
5	0.80
6	0.90
7	0.05
8	0.60
9	0.50
0	0.80

$P(F_{5,5} = on|Y)$

1	0.05
2	0.01
3	0.90
4	0.80
5	0.90
6	0.90
7	0.25
8	0.85
9	0.60
0	0.80