CSE546: Linear Regression Bias / Variance Tradeoff

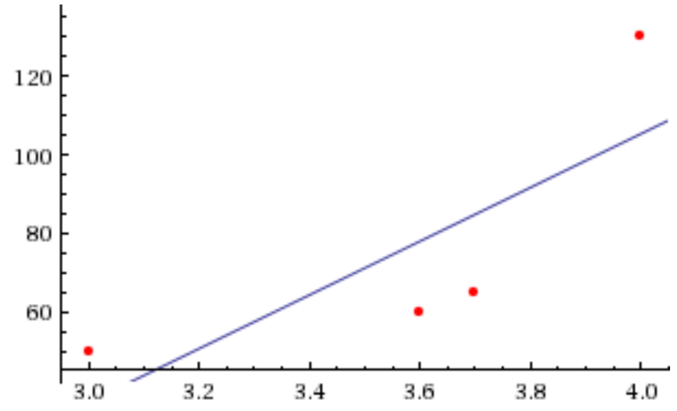
Winter 2012

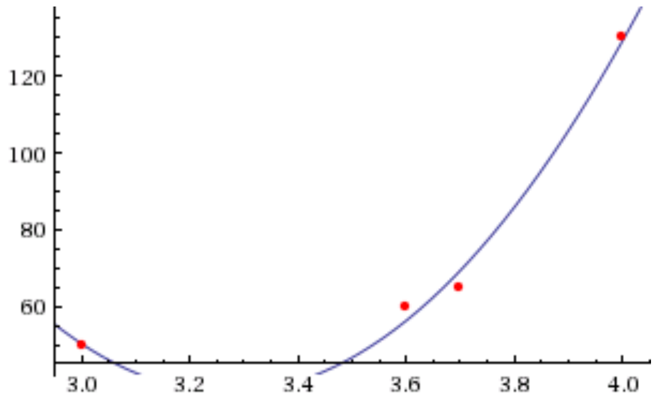
Luke Zettlemoyer

Slides adapted from Carlos Guestrin

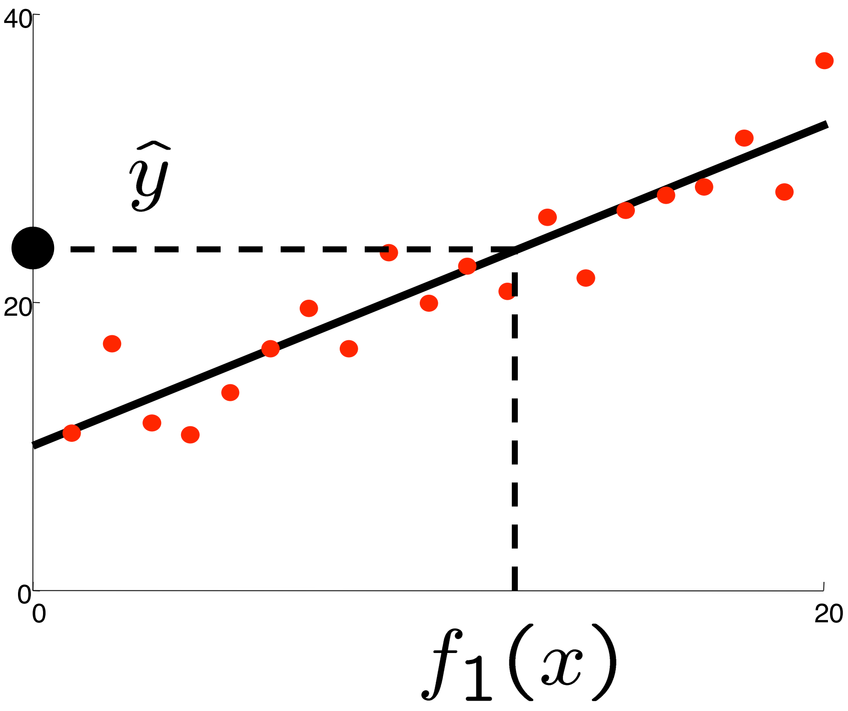
Prediction of continuous variables

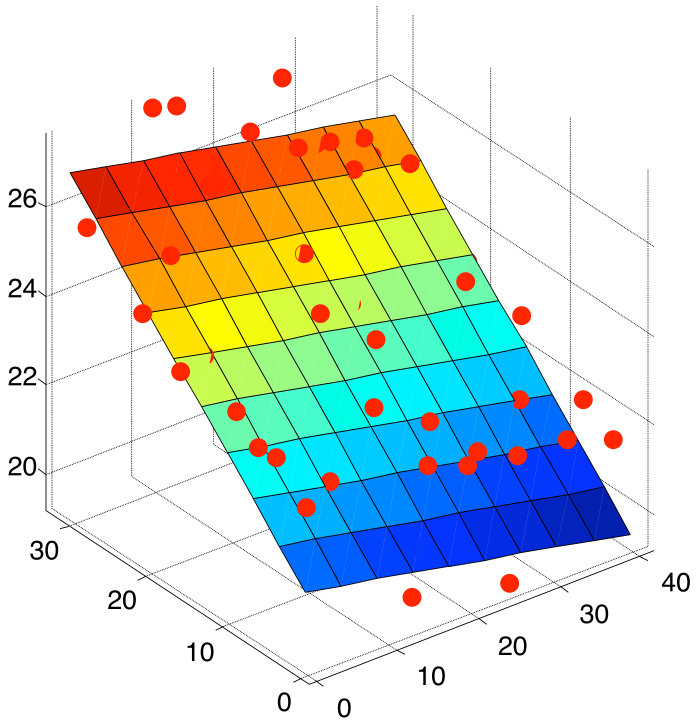
- Billionaire says: Wait, that’s not what I meant!
- You say: Chill out, dude.
- He says: I want to predict a continuous variable for continuous inputs: I want to predict salaries from GPA.
- You say: **I can regress that…**





Linear Regression





Prediction

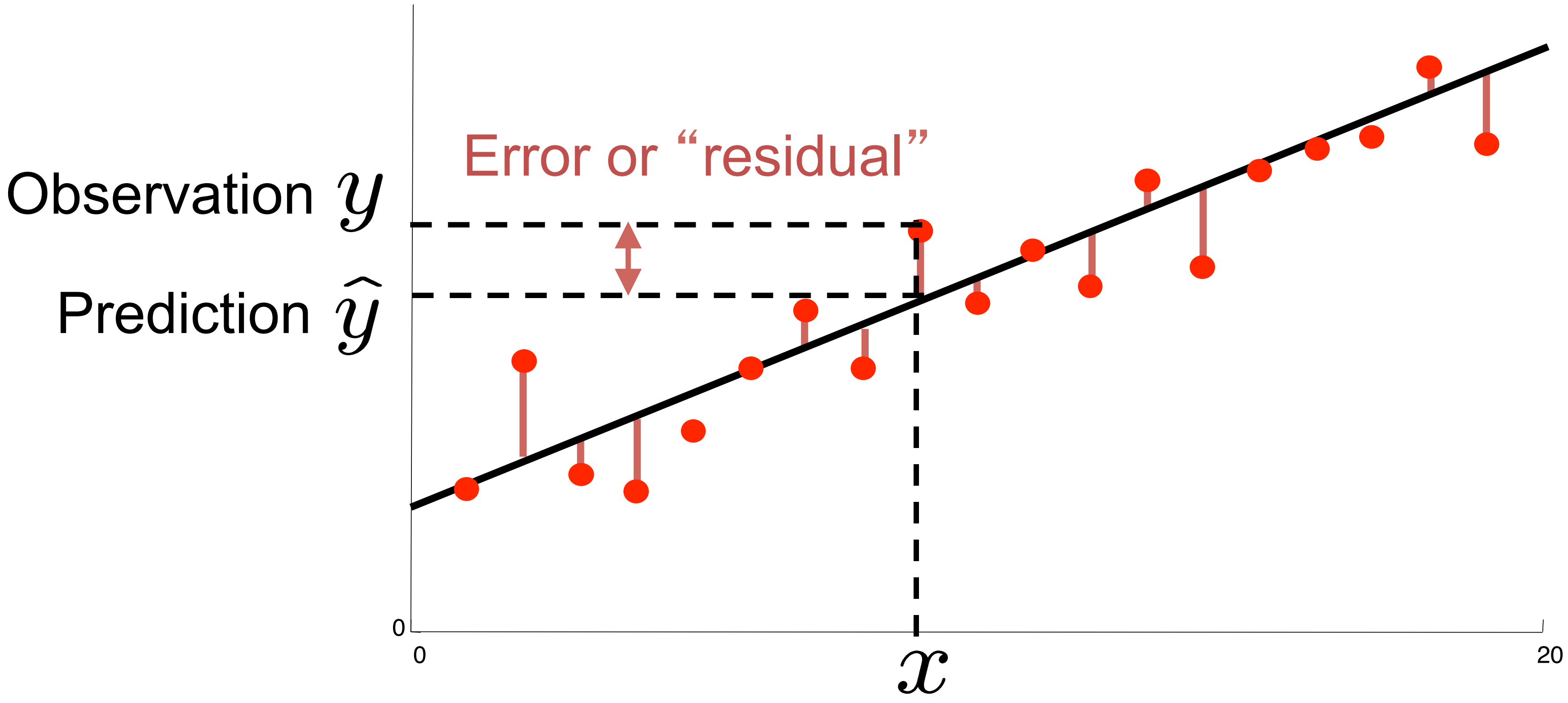
$$\hat{y} = w_0 + w_1 f_1(x)$$

Prediction

$$\hat{y}_i = w_0 + w_1 f_1(x) + w_2 f_2(x)$$

Ordinary Least Squares (OLS)

$$\text{total error} = \sum_i (y_i - \widehat{y}_i)^2 = \sum_i \left(y_i - \sum_k w_k f_k(x_i) \right)^2$$



The regression problem

- Instances: <$\mathbf{x}_j$, t_j>
- Learn: Mapping from x to t(**x**)
- Hypothesis space:
  - Given, basis functions $\{h_1, \ldots, h_k\}$
  - Find coeffs **w**=$\{w_1, \ldots, w_k\}$
  - Why is this usually called *linear regression*?
    - model is linear in the parameters
    - Can we estimate functions that are not lines???

$$H = \{h_1, \ldots, h_K\}$$

$$\underbrace{t(\mathbf{x})}_{\text{data}} \approx \widehat{f}(\mathbf{x}) = \textstyle\sum_i w_i h_i(\mathbf{x})$$

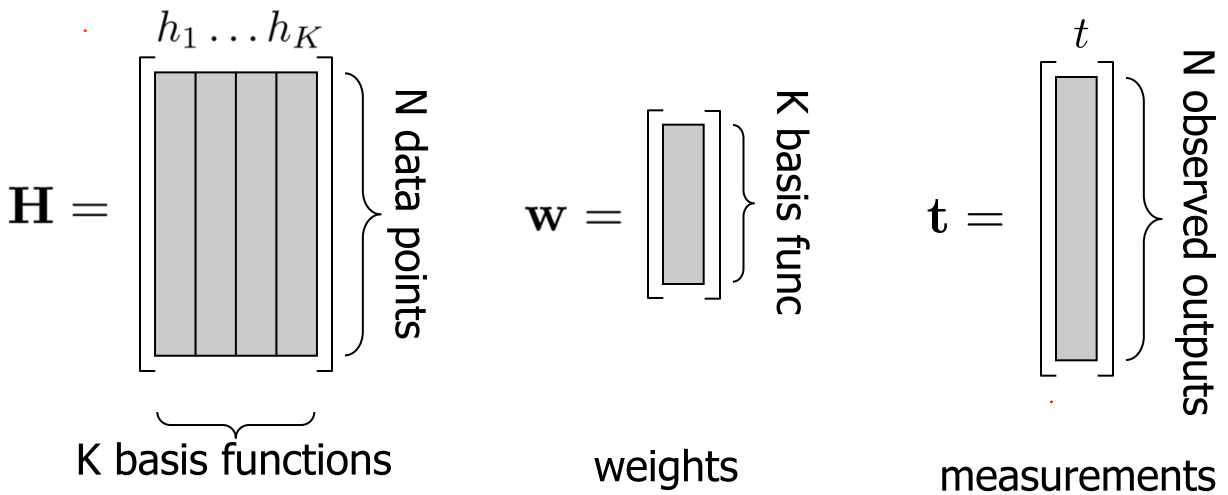
- Precisely, minimize the residual squared error:

$$\mathbf{w}^* \quad = \quad \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$

Regression: matrix notation

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \underbrace{(\mathbf{H}\mathbf{w} - \mathbf{t})^T (\mathbf{H}\mathbf{w} - \mathbf{t})}_{\text{residual error}}$$



K basis functions

weights

measurements

Regression solution: simple matrix math

$$\mathbf{w}^* \quad = \quad \arg\min_{\mathbf{w}} \underbrace{(\mathbf{H}\mathbf{w}-\mathbf{t})^T(\mathbf{H}\mathbf{w}-\mathbf{t})}_{\text{residual error}}$$

$$\text{solution: } \mathbf{w}^* = \underbrace{\left(\mathbf{H}^{\mathrm{T}}\mathbf{H}\right)^{-1}}_{\mathbf{A}^{-1}} \underbrace{\mathbf{H}^{\mathrm{T}}\mathbf{t}}_{\mathbf{b}} = \mathbf{A}^{-1}\mathbf{b}$$

$$\text{where } \mathbf{A} = \mathbf{H}^{\mathrm{T}}\mathbf{H} = \underbrace{\begin{bmatrix} \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{bmatrix}}_{\substack{\text{k×k matrix} \\ \text{for k basis functions}}} \qquad \mathbf{b} = \mathbf{H}^{\mathrm{T}}\mathbf{t} = \underbrace{\begin{bmatrix} \square \\ \square \\ \square \\ \square \end{bmatrix}}_{\text{k×1 vector}}$$

But, why?

- Billionaire (again) says: Why sum squared error???
- You say: Gaussians, Dr. Gateson, Gaussians…
- Model: prediction is linear function plus Gaussian noise
  - $t(\mathbf{x}) = \sum_i w_i h_i(\mathbf{x}) + \varepsilon$
- Learn **w** using MLE:

$$P(t \mid \mathbf{x}, \mathbf{w}, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-\left[t - \sum_i w_i h_i(\mathbf{x})\right]^2}{2\sigma^2}}$$

Maximizing log-likelihood

Maximize wrt w:

$$\ln P(\mathcal{D} \mid \mathbf{w}, \sigma) = \ln \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^N \prod_{j=1}^{N} e^{\frac{-[t_j - \sum_i w_i h_i(\mathbf{x}_j)]^2}{2\sigma^2}}$$

$$\arg\max_w \ln \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^N + \sum_{j=1}^{N} \frac{-[t_j - \sum_i w_i h_i(x_j)]^2}{2\sigma^2}$$

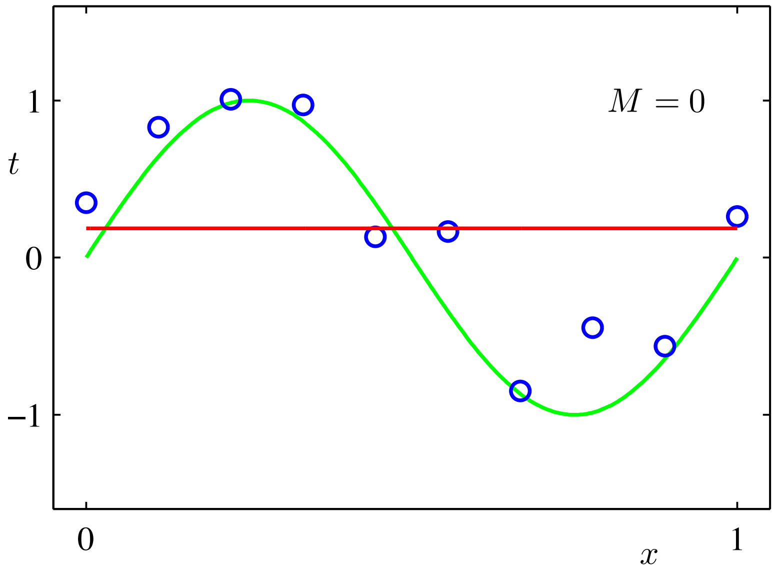
$$= \arg\max_w \sum_{j=1}^{N} \frac{-[t_j - \sum_i w_i h_i(x_j)]^2}{2\sigma^2}$$

$$= \arg\min_w \sum_{j=1}^{N} [t_j - \sum_i w_i h_i(x_j)]^2$$

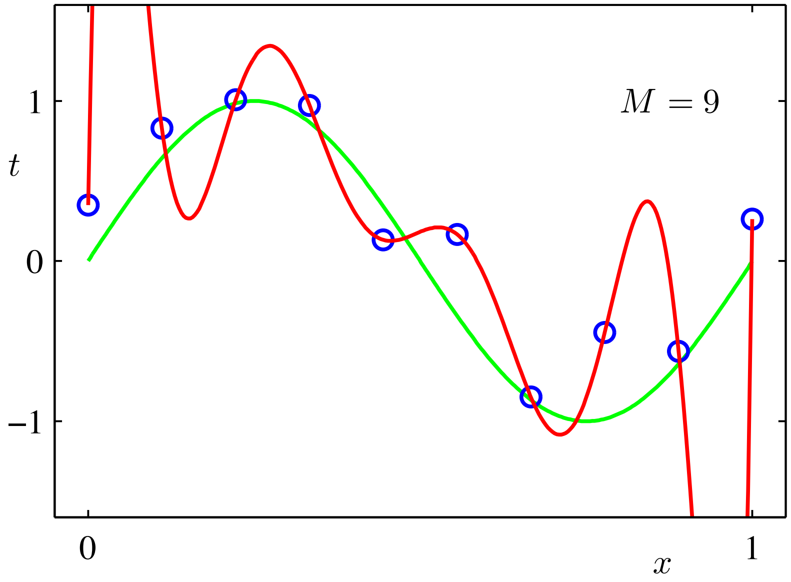
Least-squares Linear Regression is MLE for Gaussians!!!

Bias-Variance tradeoff – Intuition

- Model too simple: does not fit the data well
  - A *biased* solution



- Model too complex: small changes to the data, solution changes a lot
  - A *high-variance* solution



(Squared) Bias of learner

- Given: dataset *D* with *m* samples
- Learn: for different datasets *D*, you will get different functions h(x)
- Expected prediction (averaged over hypotheses): $E_D[h(x)]$
- Bias: difference between expected prediction and truth
  - Measures how well you expect to represent true solution
  - Decreases with more complex model

$$bias^2 = \int_x \{E_D[h(x)] - t(x)\}^2 \, p(x)dx$$

Variance of learner

- Given: dataset *D* with *m* samples
- Learn: for different datasets *D*, you will get different functions h(x)
- Expected prediction (averaged over hypotheses): $E_D[h(x)]$
- Variance: difference between what you expect to learn and what you learn from a from a particular dataset
  - Measures how sensitive learner is to specific dataset
  - Decreases with simpler model

$$\begin{aligned} \bar{h}(x) &= E_D[h(x)] \\ variance &= \int E_D[(h(x) - \bar{h}(x))^2]p(x)dx \end{aligned}$$

Bias–Variance decomposition of error

- Consider simple regression problem f:X→T

$$f(x) = g(x) + \varepsilon$$

noise ~ N(0,σ)

deterministic

- Collect some data, and learn a function h(x)
- What are sources of prediction error?

$$E_D\left[\int_x \int_t (h(x) - t)^2 p(t|x) p(x) dt dx\right]$$

Sources of error 1 – noise

$f(x) = g(x) + \varepsilon$

- What if we have perfect learner, infinite data?
  - If our learning solution h(x) satisfies h(x)=g(x)
  - Still have remaining, *unavoidable error* of σ^2 due to noise ε

$$error(h) = \int_x \int_t (h(x) - t)^2 p(f(x) = t|x) p(x) dt dx$$

Sources of error 2 – Finite data

$f(x) = g(x) + \varepsilon$

- What if we have imperfect learner, or only *m* training examples?
- What is our expected squared error per example?
  - Expectation taken over random training sets *D* of size *m*, drawn from distribution P(X,T)

$$E_D\left[\int_x \int_t \{h(x) - t\}^2 p(f(x) = t|x)p(x)dtdx\right]$$

Bias-Variance Decomposition of Error

Bishop Chapter 3   Assume target function: t(x) = g(x) + ε

- Then expected squared error over fixed size training sets *D* drawn from P(X,T) can be expressed as sum of three components:

$$E_D\left[\int_x \int_t (h(x) - t)^2 p(t|x) p(x) dt dx\right]$$

$$= unavoidableError + bias^2 + variance$$

Where:

$$unavoidableError = \sigma^2$$

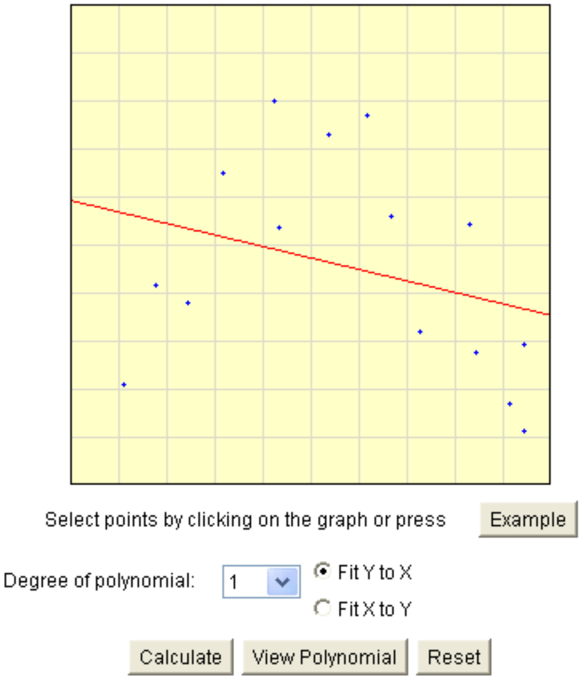
$$bias^2 = \int (E_D[h(x)] - g(x))^2 p(x) dx$$

$$\bar{h}(x) = E_D[h(x)]$$

$$variance = \int E_D[(h(x) - \bar{h}(x))^2] p(x) dx$$

Bias-Variance Tradeoff

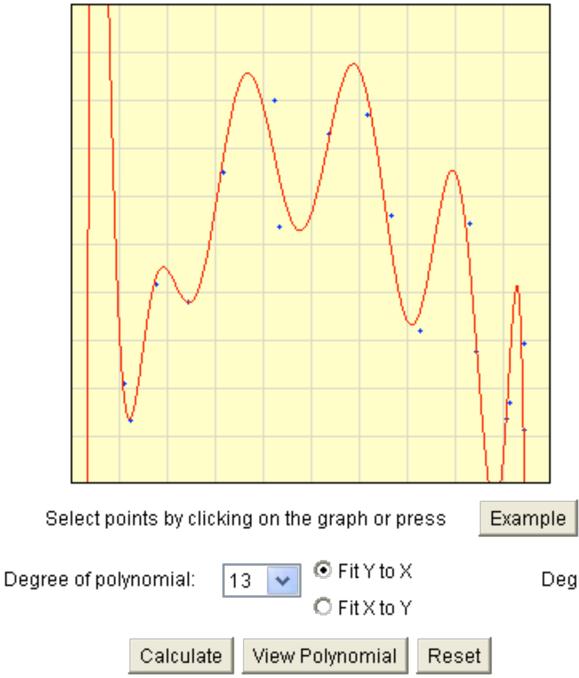
- Choice of hypothesis class introduces learning bias
  - More complex class → less bias
  - More complex class → more variance



Select points by clicking on the graph or press

Degree of polynomial: 1

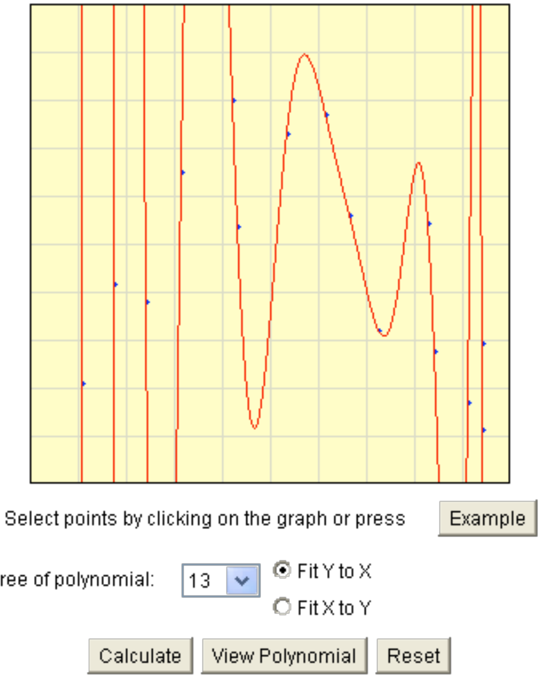
● Fit Y to X
○ Fit X to Y



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

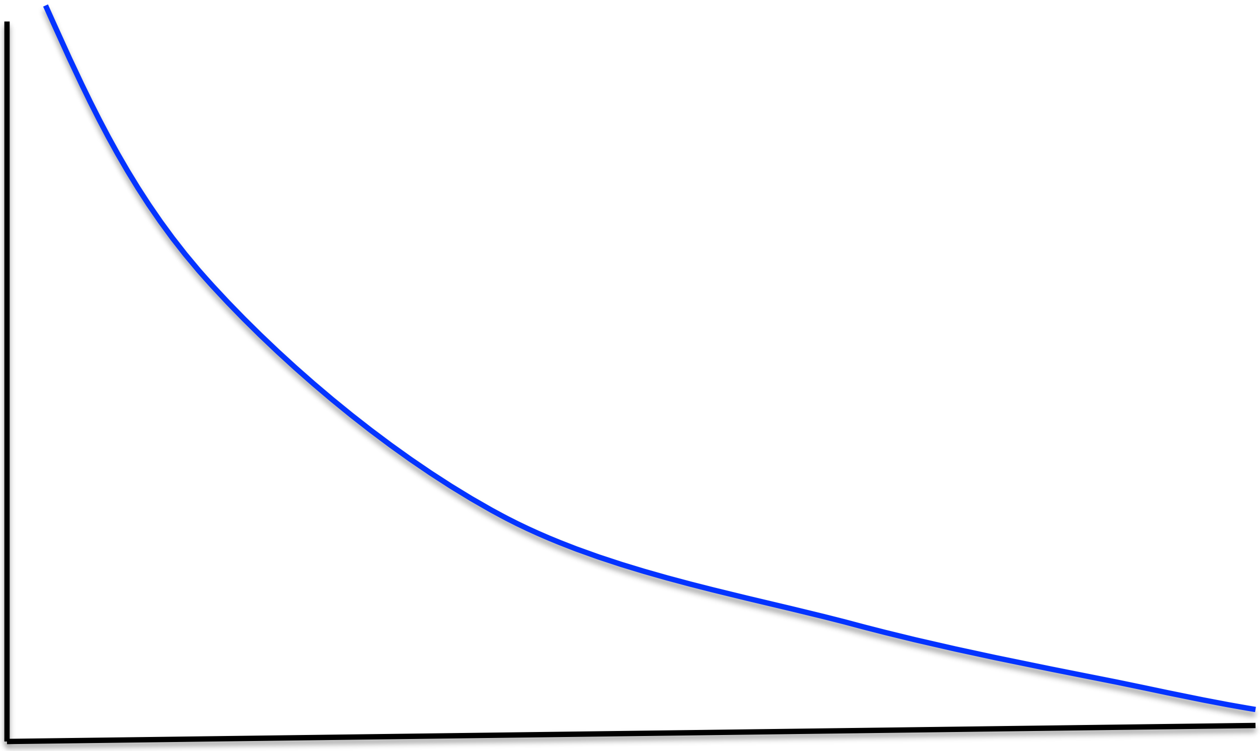
$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$

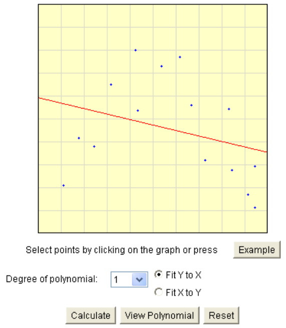
- Given a dataset (Training data)
- Choose a loss function
  - e.g., squared error (L_2) for regression
- Training error: For a particular set of parameters, loss function on training data:

$$error_{train}(\mathbf{w}) = \frac{1}{N_{train}} \sum_{j=1}^{N_{train}} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$

Training error as a function of model complexity

$$error_{train}(\mathbf{w}) = \frac{1}{N_{train}} \sum_{j=1}^{N_{train}} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$



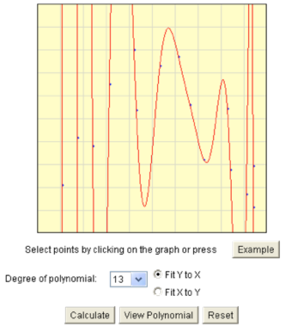


Select points by clicking on the graph or press

Degree of polynomial: 1

● Fit Y to X

○ Fit X to Y



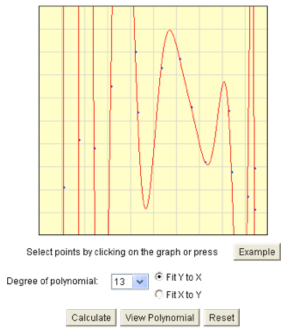
Select points by clicking on the graph or press

Degree of polynomial: 13

● Fit Y to X

○ Fit X to Y

Prediction error



Select points by clicking on the graph or press

Degree of polynomial: 13

● Fit Y to X

○ Fit X to Y

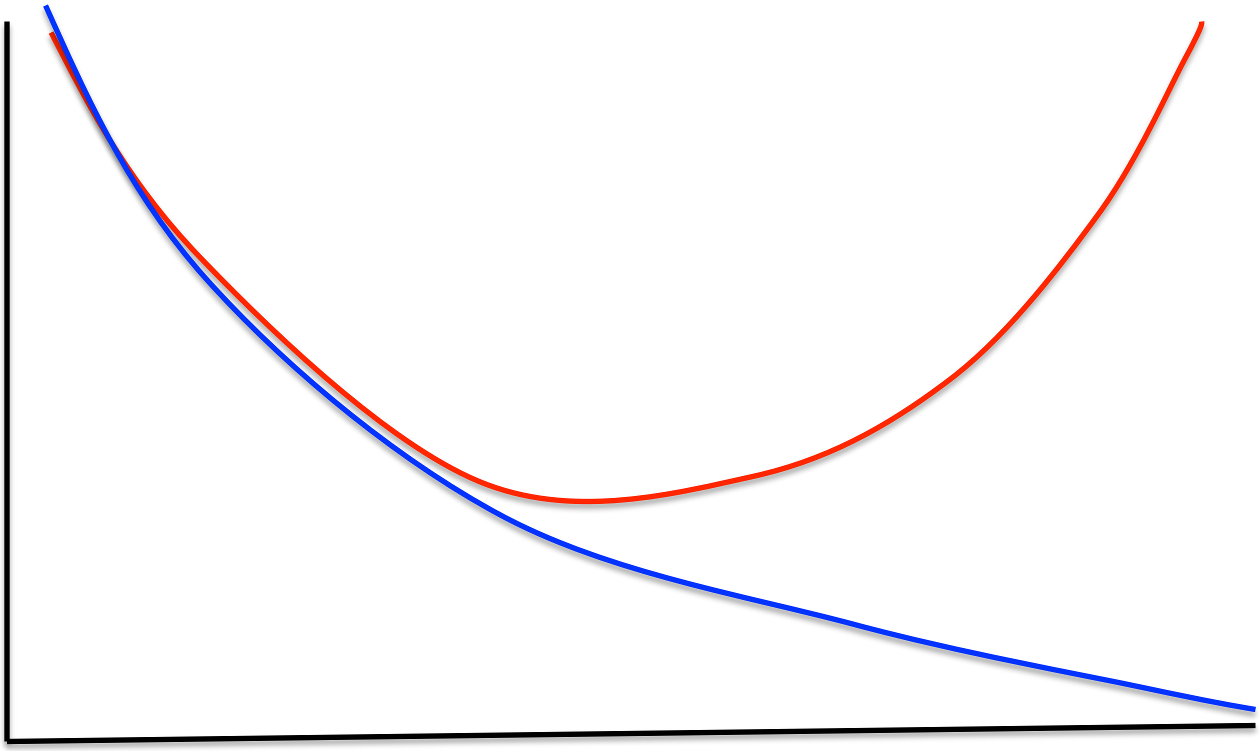
- Training set error can be poor measure of "quality" of solution
- Prediction error (true error): We really care about error over all possibilities:

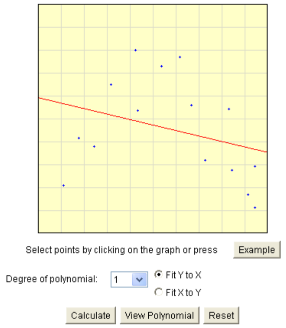
$$
\begin{aligned}
error_{true}(\mathbf{w}) &= E_{\mathbf{x}}\left[\left(t(\mathbf{x}) - \sum_i w_i h_i(\mathbf{x})\right)^2\right] \\
&= \int_{\mathbf{x}} \left(t(\mathbf{x}) - \sum_i w_i h_i(\mathbf{x})\right)^2 p(\mathbf{x}) d\mathbf{x}
\end{aligned}
$$

Prediction error as a function of model complexity

$$error_{train}(\mathbf{w}) = \frac{1}{N_{train}} \sum_{j=1}^{N_{train}} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$

$$error_{true}(\mathbf{w}) = \int_{\mathbf{x}} \left(t(\mathbf{x}) - \sum_i w_i h_i(\mathbf{x}) \right)^2 p(\mathbf{x}) d\mathbf{x}$$





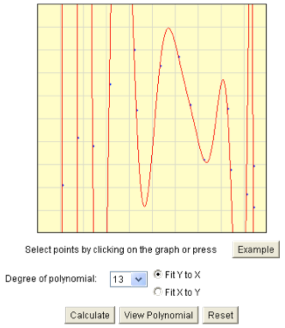
Select points by clicking on the graph or press [Example]

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 t(**x**) for every **x**, may not know p(x)

$$error_{true}(\mathbf{w}) \quad = \quad \int_{\mathbf{x}} \left(t(\mathbf{x}) - \sum_i w_i h_i(\mathbf{x}) \right)^2 p(\mathbf{x}) d\mathbf{x}$$

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

$$error_{true}(\mathbf{w}) \quad \approx \quad \frac{1}{M} \sum_{j=1}^{M} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$

Why training set error doesn't approximate prediction error?

- Sampling approximation of prediction error:

$$error_{true}(\mathbf{w}) \approx \frac{1}{M}\sum_{j=1}^{M}\left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j)\right)^2$$

- Training error :

$$error_{train}(\mathbf{w}) = \frac{1}{N_{train}}\sum_{j=1}^{N_{train}}\left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j)\right)^2$$

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

Why training set error doesn't approximate prediction error?

- Sa

Because you cheated!!!

Training error good estimate for a single **w,**
But you optimized **w** with respect to the training error,
and found **w** that is good for this set of samples

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

- Tr

er

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

Test set error

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$

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

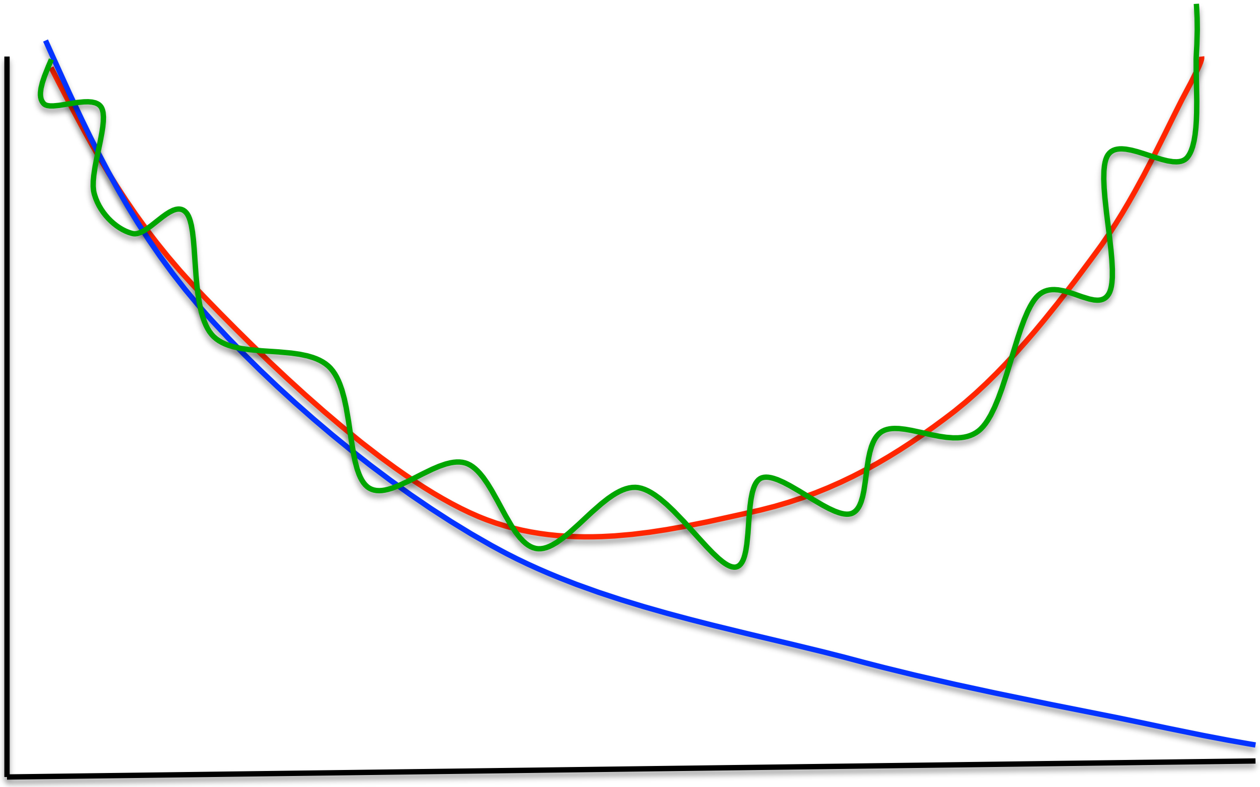
$$error_{test}(\mathbf{w}) = \frac{1}{N_{test}} \sum_{j=1}^{N_{test}} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$

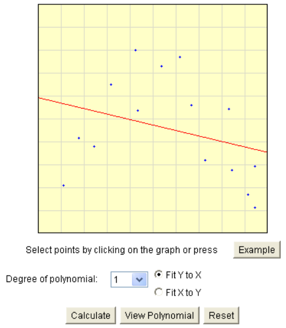
Test set error as a function of model complexity

$$error_{train}(\mathbf{w}) = \frac{1}{N_{train}} \sum_{j=1}^{N_{train}} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$

$$error_{true}(\mathbf{w}) = \int_{\mathbf{x}} \left(t(\mathbf{x}) - \sum_i w_i h_i(\mathbf{x}) \right)^2 p(\mathbf{x}) d\mathbf{x}$$

$$error_{test}(\mathbf{w}) = \frac{1}{N_{test}} \sum_{j=1}^{N_{test}} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$





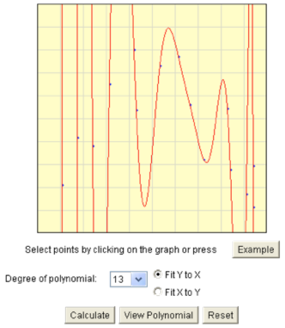
Select points by clicking on the graph or press Example

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: this slide is so important we are looking at it 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:

$$error_{true}(\mathbf{w}) \quad = \quad \int_{\mathbf{x}} \left(t(\mathbf{x}) - \sum_i w_i h_i(\mathbf{x}) \right)^2 p(\mathbf{x}) d\mathbf{x}$$

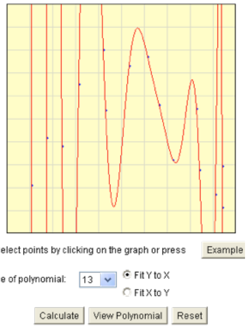
- Training: optimistically biased

$$error_{train}(\mathbf{w}) = \qquad \frac{1}{N_{train}} \sum_{j=1}^{N_{train}} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$

- Test: our final meaure, unbiased?

$$error_{test}(\mathbf{w}) \quad = \quad \frac{1}{N_{test}} \sum_{j=1}^{N_{test}} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$

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


Select points by clicking on the graph or press

Degree of polynomial: 13

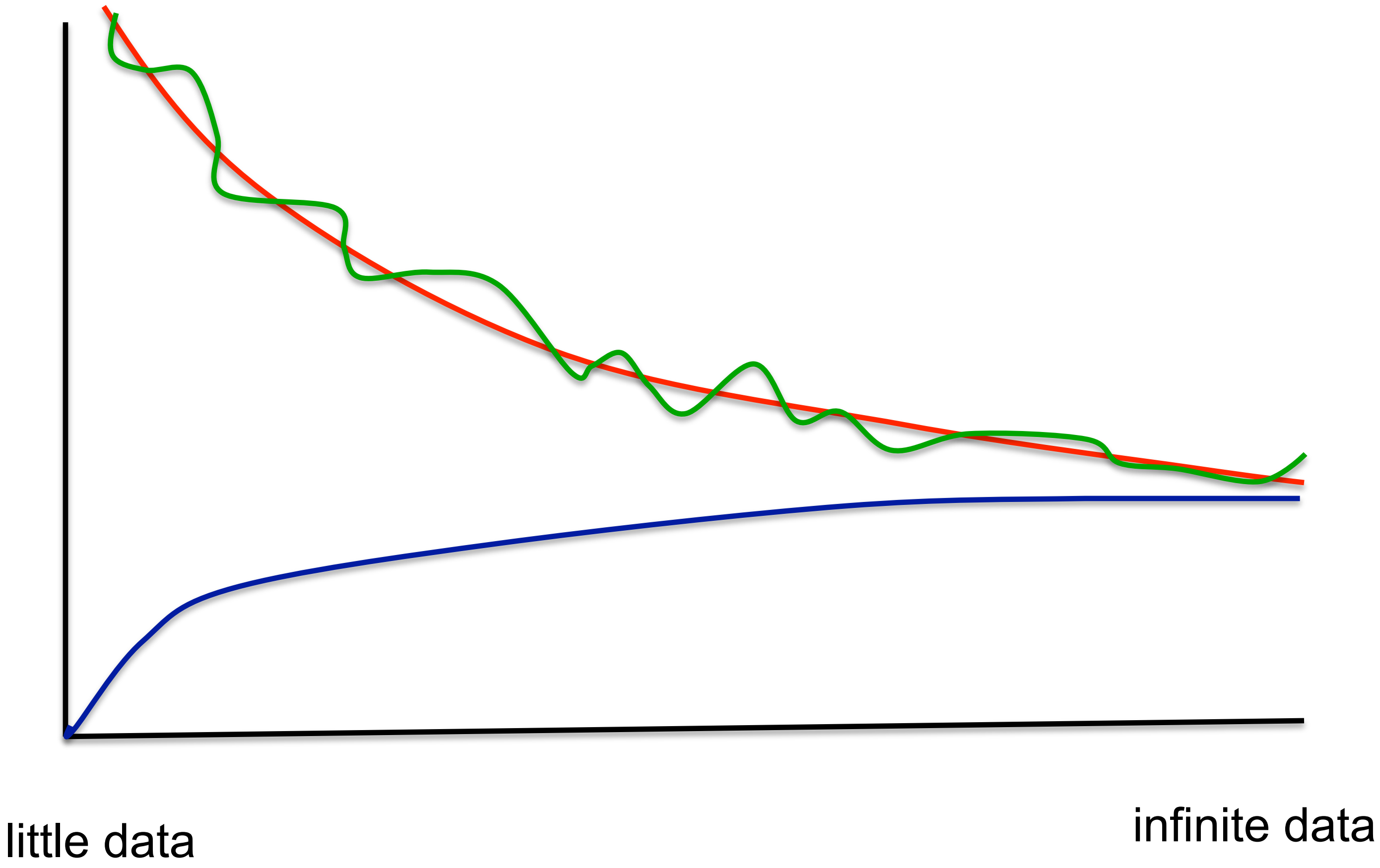
◉ Fit Y to X

○ Fit X to Y

$$error_{train}(\mathbf{w}) = \frac{1}{N_{train}} \sum_{j=1}^{N_{train}} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$

$$error_{true}(\mathbf{w}) = \int_{\mathbf{x}} \left(t(\mathbf{x}) - \sum_i w_i h_i(\mathbf{x}) \right)^2 p(\mathbf{x}) d\mathbf{x}$$

$$error_{test}(\mathbf{w}) = \frac{1}{N_{test}} \sum_{j=1}^{N_{test}} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$



Summary: error estimators

- G
- T

Be careful!!!

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

For example, if you use the test set to select the degree of the polynomial… no longer unbiased!!! (We will address this problem later in the semester)

- Test: our final meaure, unbiased?

$$error_{test}(\mathbf{w}) = \frac{1}{N_{test}} \sum_{j=1}^{N_{test}} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$

What you need to know

- Regression
  - Basis function = features
  - Optimizing sum squared error
  - Relationship between regression and Gaussians
- Bias-Variance trade-off
- Play with Applet
  - http://mste.illinois.edu/users/exner/java.f/leastsquares/
- True error, training error, test error
  - Never learn on the test data
- Overfitting