

CSE446: Linear Regression

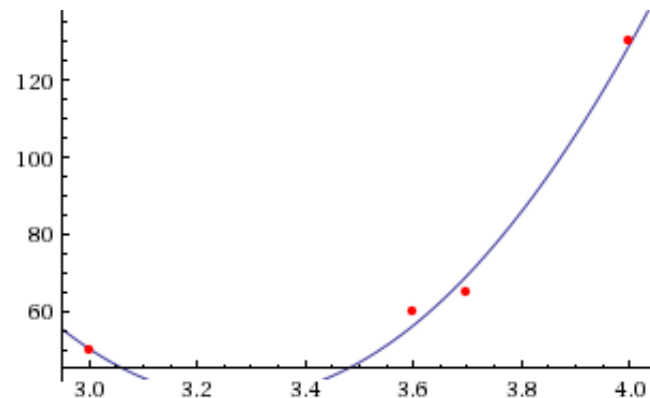
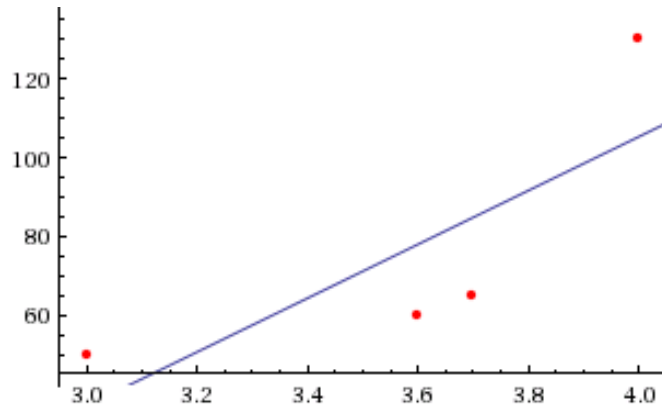
Spring 2017

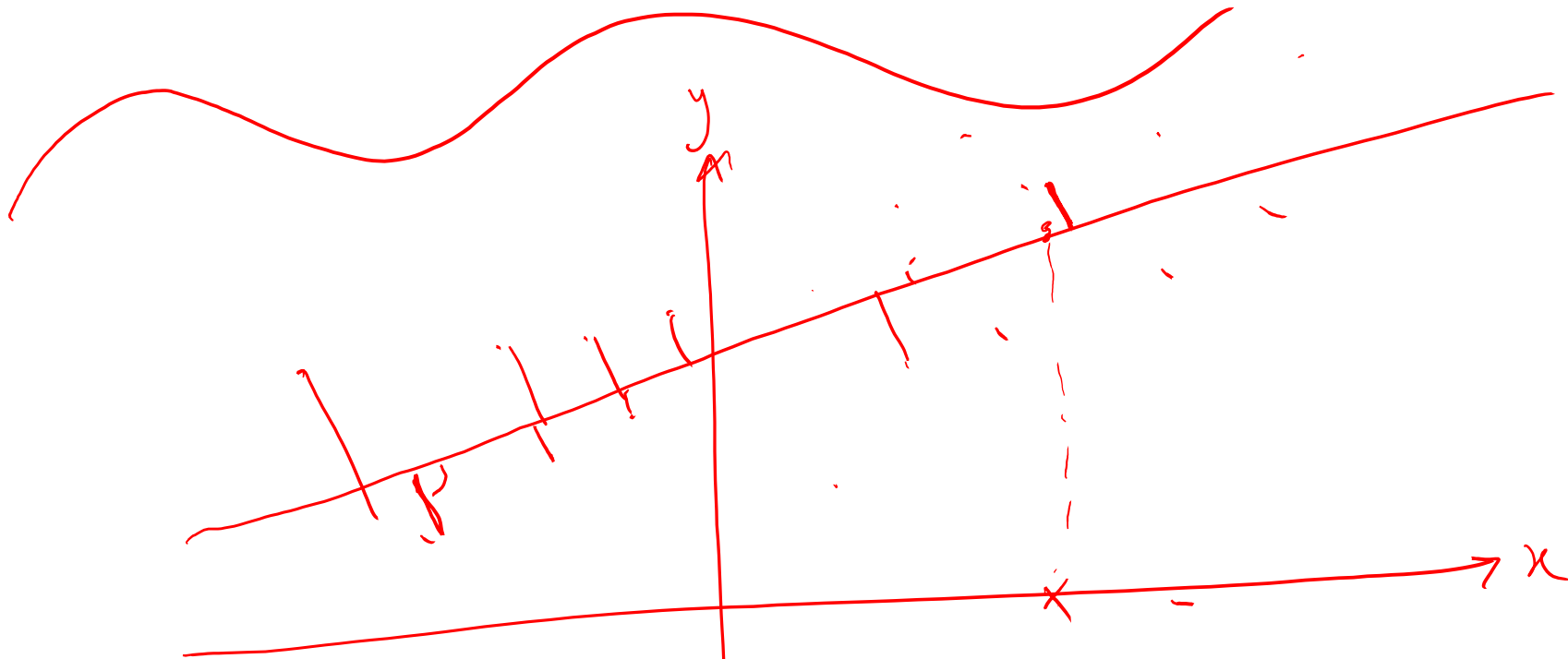
Ali Farhadi

Slides adapted from Carlos Guestrin and Luke Zettlemoyer

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...**





$$y = wx + b$$

$$\sum (y_i - \hat{y}_i)^2$$

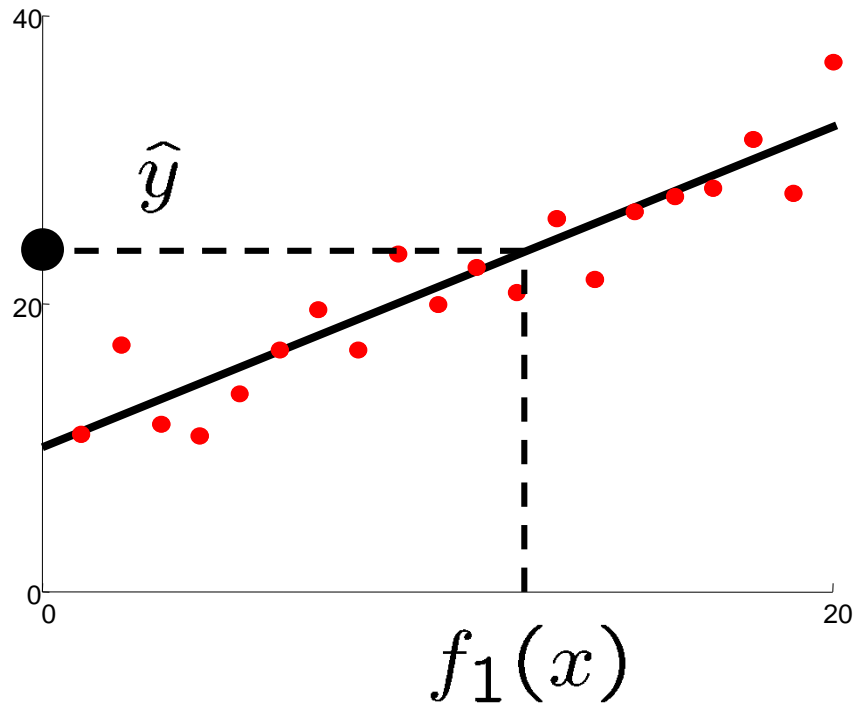
↓

$$\frac{1}{N} \sum (wx + b - y_i)^2$$

Linear Regression

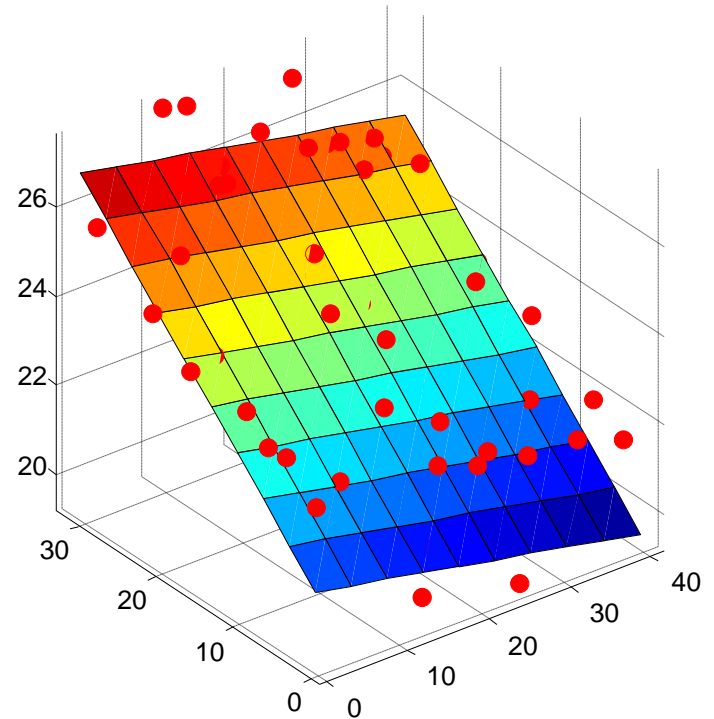
Prediction

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



Prediction

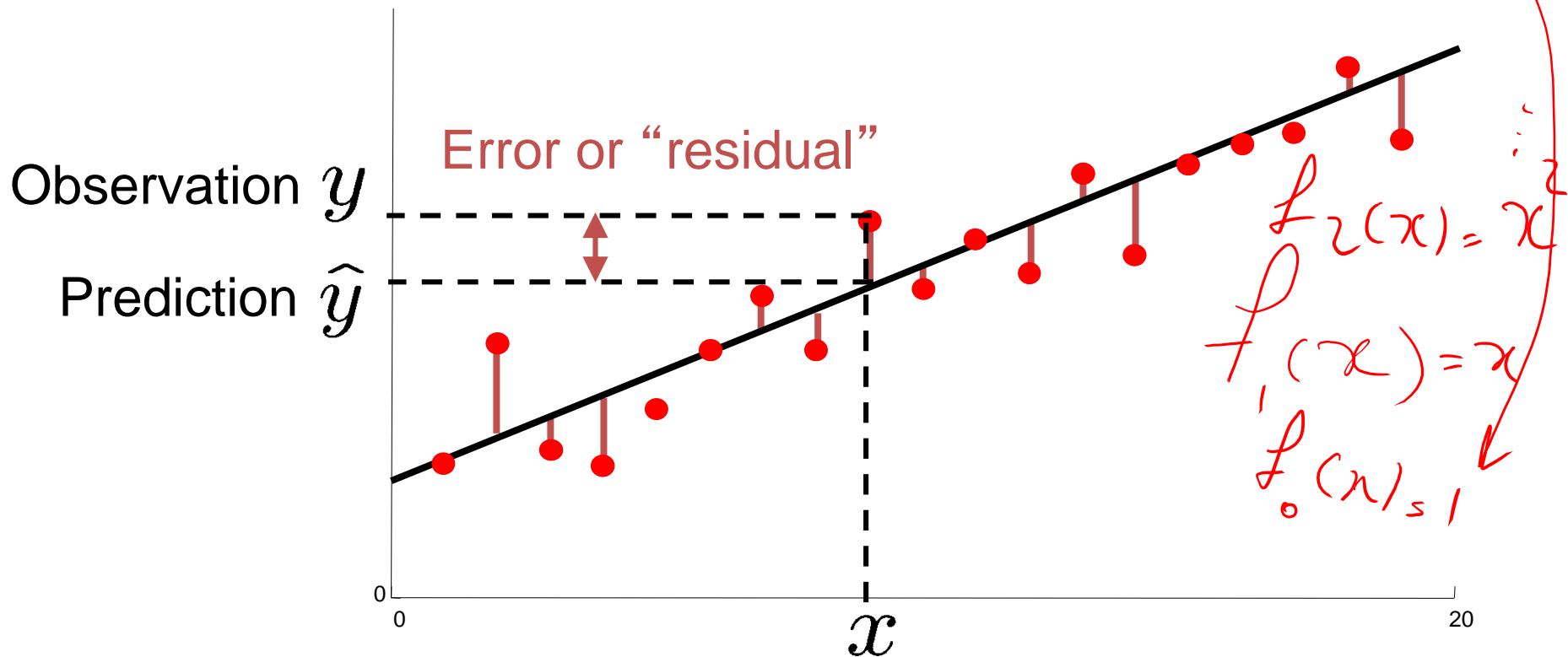
$$\hat{y}_i = w_0 + w_1 f_1(x) + w_2 f_2(x)$$



Ordinary Least Squares (OLS)

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

Handwritten notes: $f_1(x) = 1$ (circled in red), $f_2(x) = x$, $f_1(x) = x$, $f_0(x) = 1$.



The regression problem

- Instances: $\langle \mathbf{x}_j, t_j \rangle$
- Learn: Mapping from \mathbf{x} to $t(\mathbf{x})$

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

- Hypothesis space:

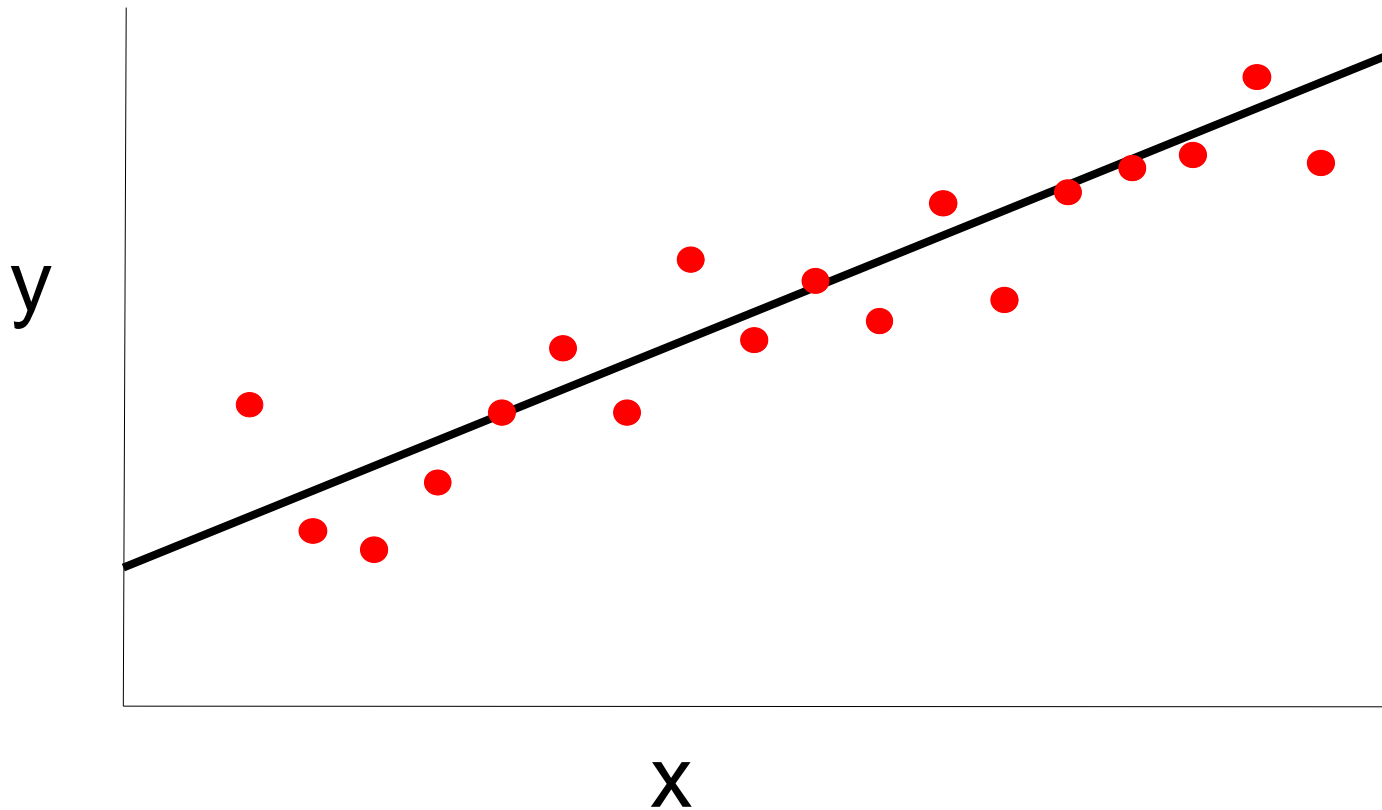
- Given, basis functions $\{h_1, \dots, h_k\}$
- $h_i(\mathbf{x}) \in \mathbb{R}$
- Find coeffs $\mathbf{w} = \{w_1, \dots, w_k\}$

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

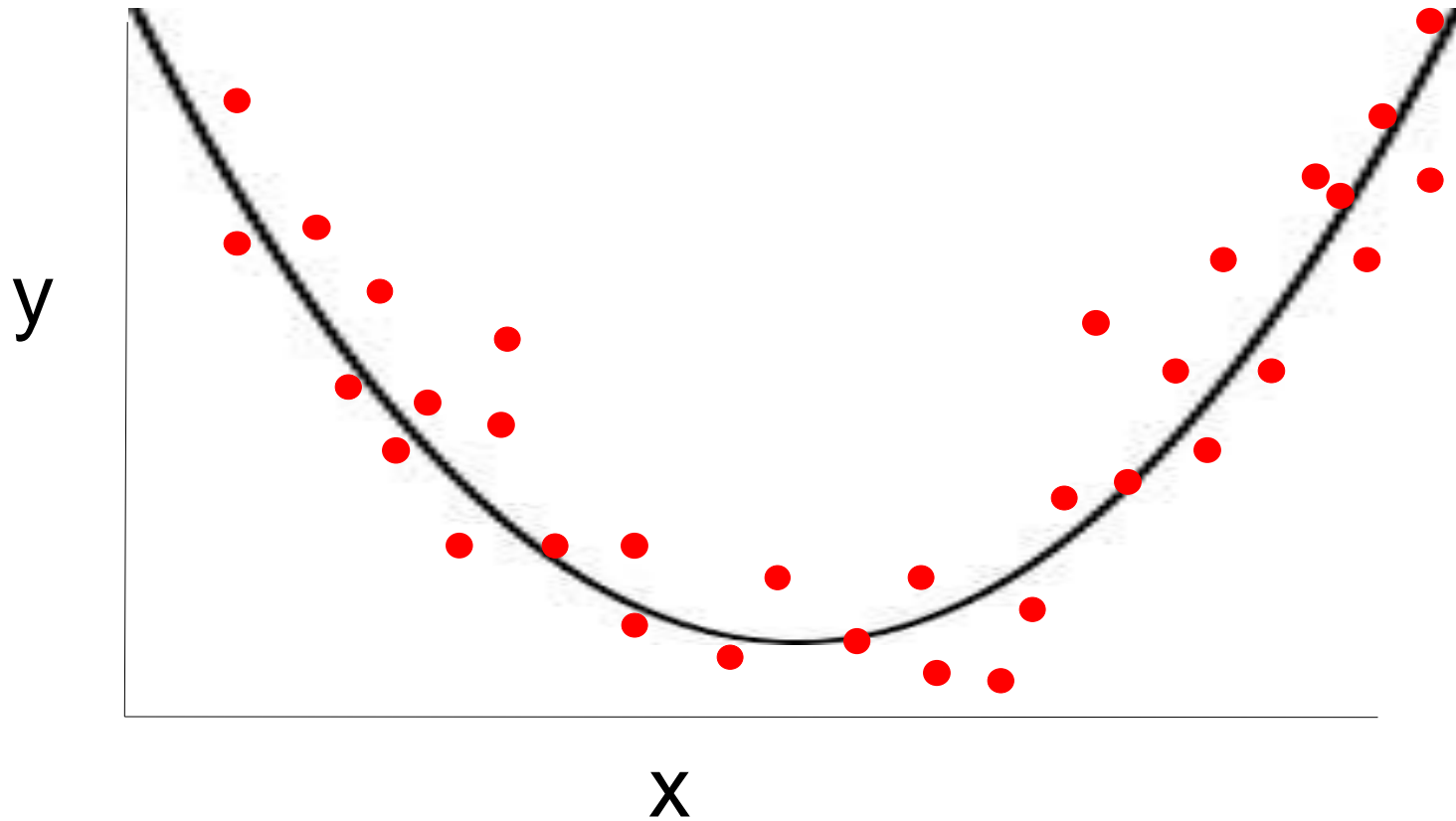
- Why is this usually called *linear regression*?
 - model is linear in the parameters
 - Can we estimate functions that are not lines???

Linear Basis: 1D input

Need a bias term: $\{h_1(x) = x, h_2(x)=1\}$



- Parabola: $\{h_1(x) = x^2, h_2(x)=x, h_3(x)=1\}$



- 2D: $\{h_1(\mathbf{x}) = x_1^2, h_2(\mathbf{x})= x_2^2, h_3(\mathbf{x})=x_1x_2, \dots\}$
- Can define any basis functions $h_i(\mathbf{x})$ for n-dimensional input $\mathbf{x}=\langle x_1, \dots, x_n \rangle$

The regression problem

- Instances: $\langle \mathbf{x}_j, t_j \rangle$

- Learn: Mapping from \mathbf{x} to $t(\mathbf{x})$

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

- Hypothesis space:

- Given, basis functions $\{h_1, \dots, h_k\}$
- $h_i(\mathbf{x}) \in \mathbb{R}$
- Find coeffs $\mathbf{w} = \{w_1, \dots, w_k\}$

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

$$\sum_i w_i^* h_i(\mathbf{x}_j)$$

- Why is this usually called *linear regression*?
 - model is linear in the parameters
 - Can we estimate functions that are not lines???

- Precisely, minimize the **residual squared error**:

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

$$h_1(x) = 1$$

$$h_2(x) = x$$

$$h_3(x) = x^2$$

$$\sum_{i=1}^3 \sum_{j=1}^{10} w_i * h_i(x_j)$$

$$\frac{(H^T H)^{-1} H^T t}{K \times K} \rightarrow z^+$$

$$K \times N$$

$$K \times 1$$

$$H * W$$

1	1	1
1	2	4
1	3	9
1	4	16
1	...	25
1
1	10	100

- w₁
- w₂
- w₃

$$\min_w \frac{d}{dw} (F(w; t))^T$$

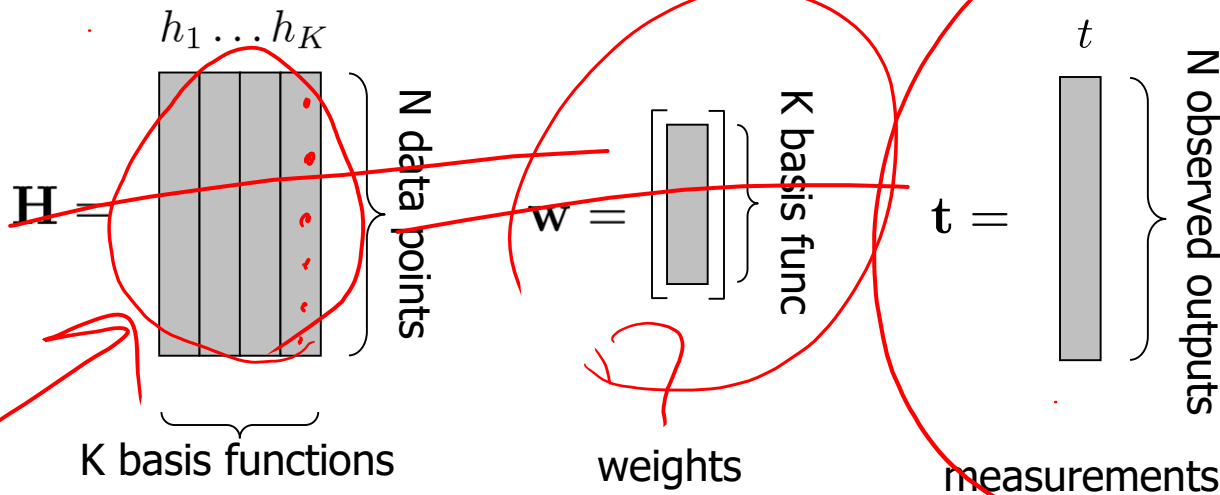
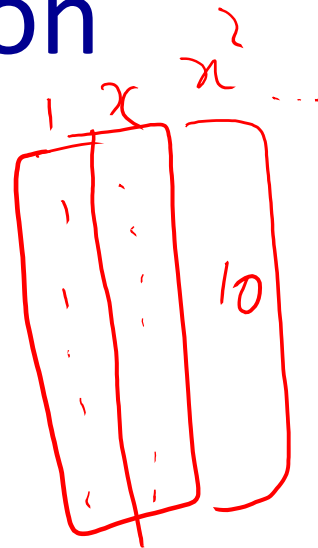
$$(Hw - t) = 0$$

H |

Regression: matrix notation

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

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

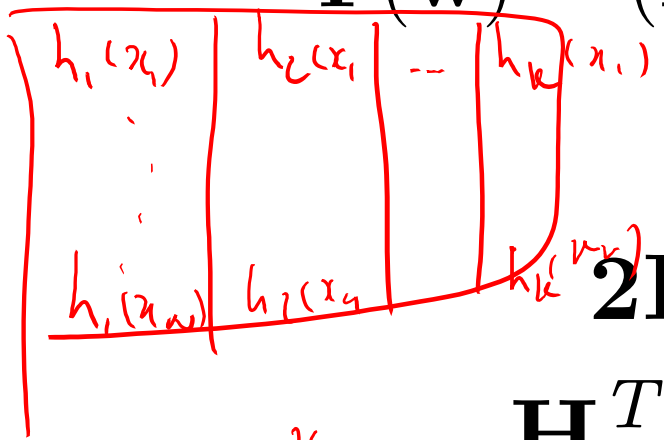


Regression: closed form solution

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

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} (\mathbf{H}\mathbf{w} - \mathbf{t})^T (\mathbf{H}\mathbf{w} - \mathbf{t})$$

$$\mathbf{F}(\mathbf{w}) = (\mathbf{H}\mathbf{w} - \mathbf{t})^T (\mathbf{H}\mathbf{w} - \mathbf{t})$$

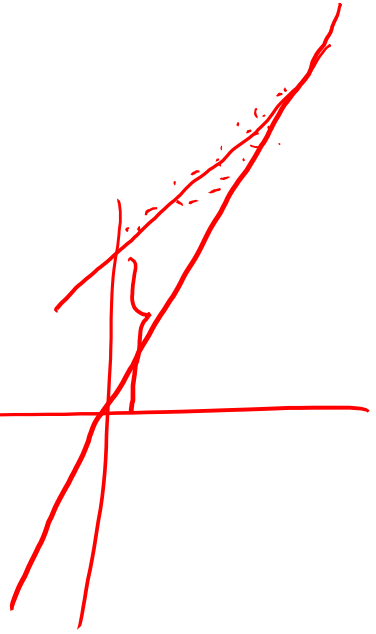


$$\nabla_{\mathbf{w}} \mathbf{F}(\mathbf{w}) = \mathbf{0}$$

$$2\mathbf{H}^T (\mathbf{H}\mathbf{w} - \mathbf{t}) = \mathbf{0}$$

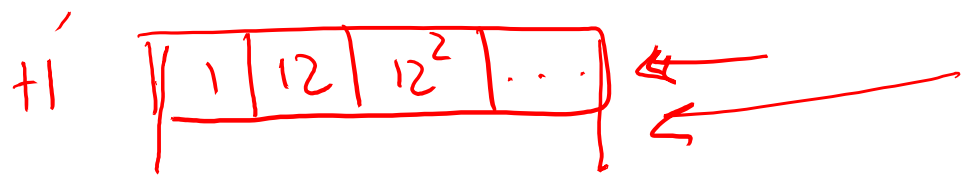
$$\mathbf{H}^T \mathbf{H}\mathbf{w} - \mathbf{H}^T \mathbf{t} = \mathbf{0}$$

$$\mathbf{w}^* = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{t}$$



$$y = wx$$

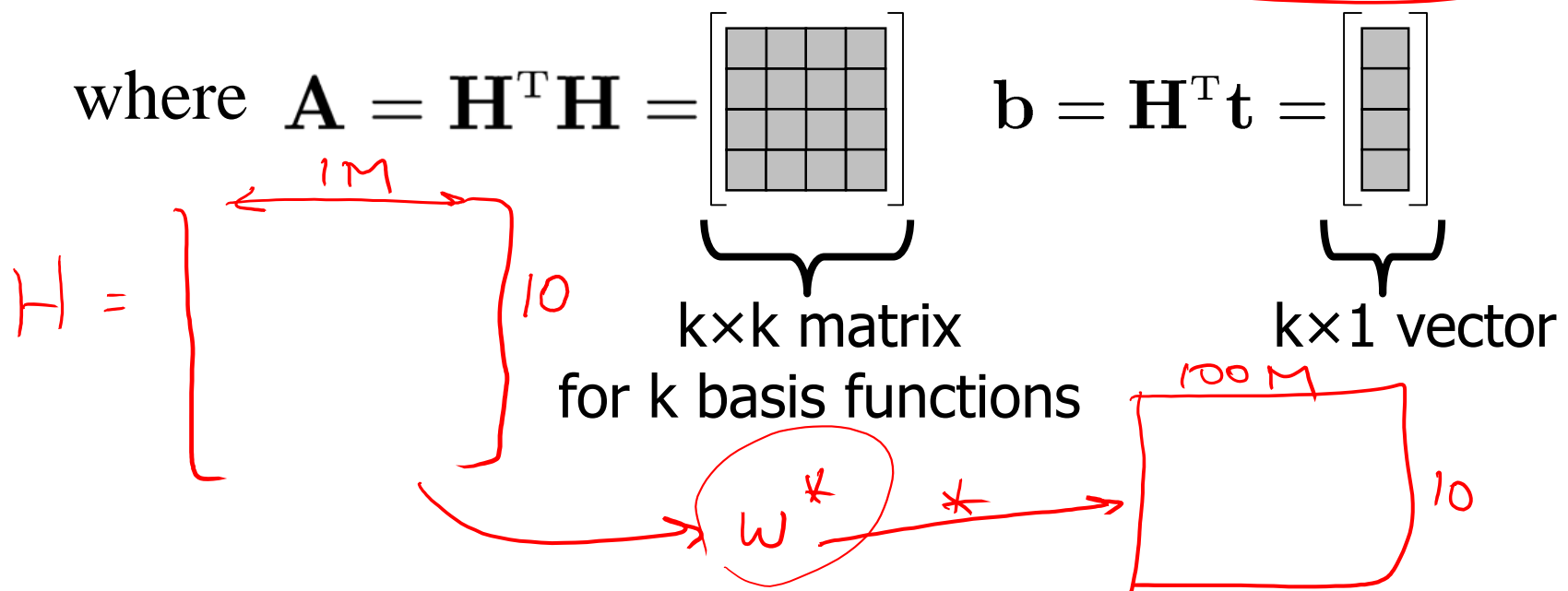
$$y = wx + b$$

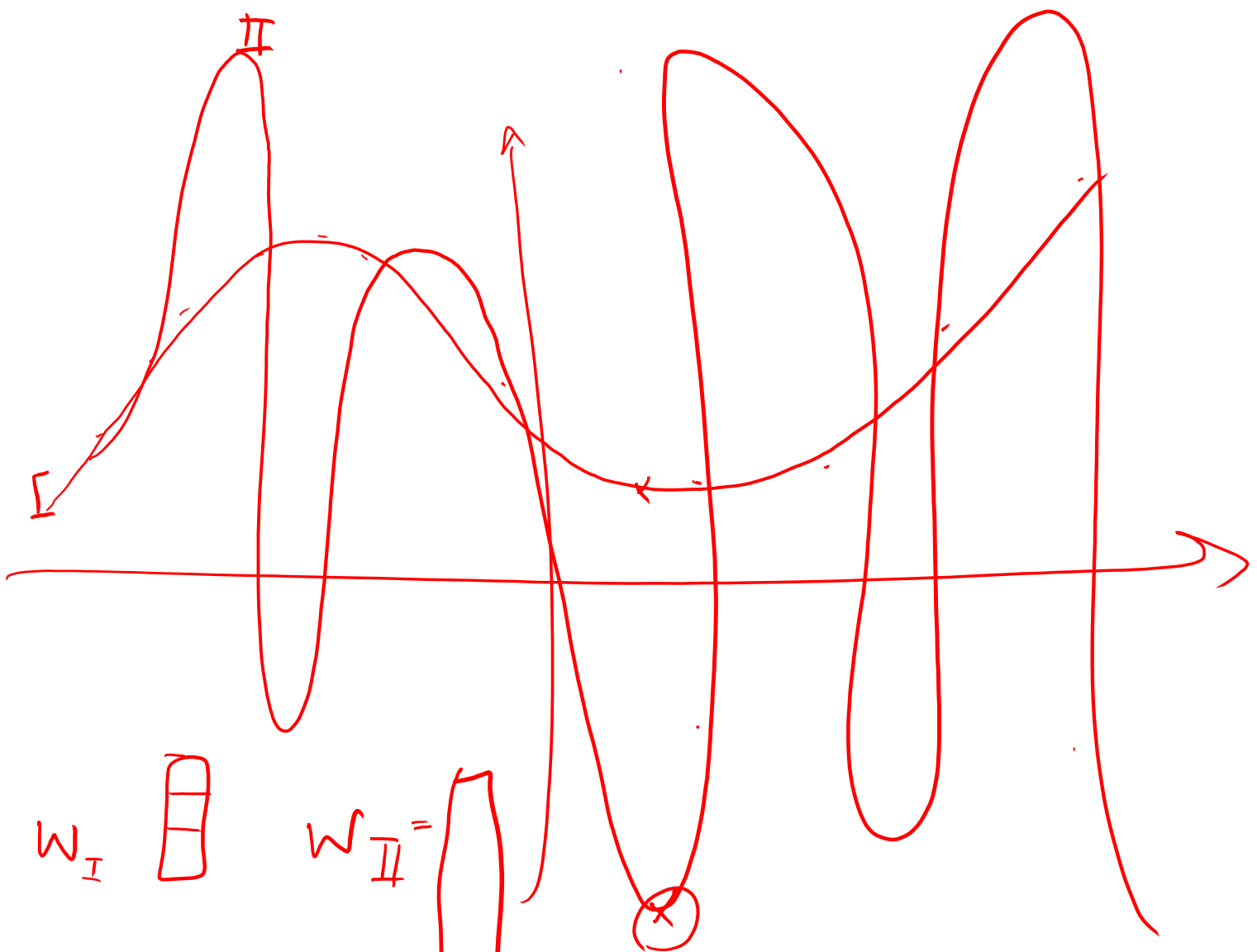


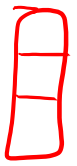
Regression solution: simple matrix math

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \underbrace{(\mathbf{H}\mathbf{w} - \mathbf{t})^T (\mathbf{H}\mathbf{w} - \mathbf{t})}_{\text{residual error}} \quad \begin{array}{l} \rightarrow Ax = b \\ x = A^{-1}b \end{array}$$

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





I → w_I 

$w_{II} =$ 

\bar{II} → II

$$\sqrt{w_1^2 + w_2^2} \rightarrow w^*$$

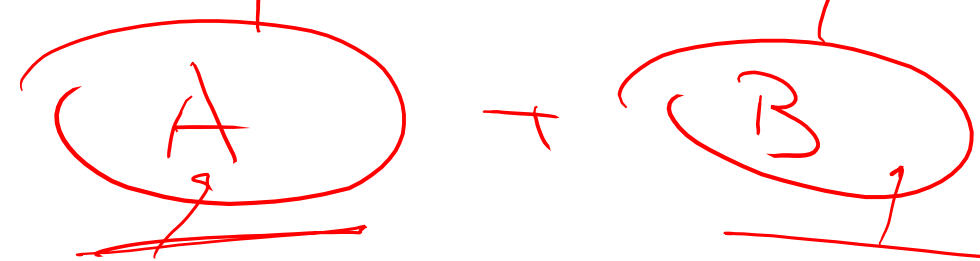
$$\text{argmin}_w (\hat{y} - y)^2 \quad \text{Loss}$$

$$= \text{argmin}_w (HW - t)^T (HW - t)$$

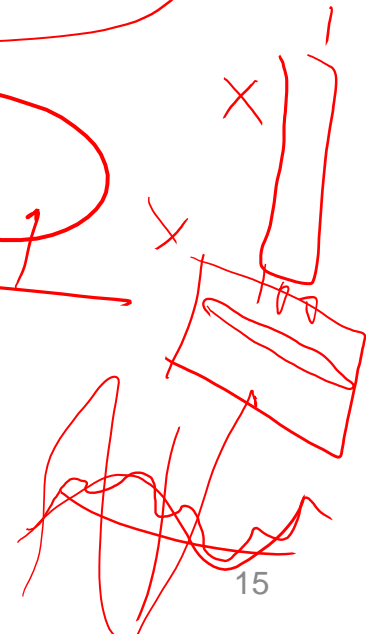
$$\cdot \|w\|_2$$

$$(w \cdot b - \hat{y})^2$$

$$w^* = \text{argmin}_w (HW - t)^T (HW - t) + \|w\|_2$$



LOSS + Regu.



$$\langle x, t \rangle \quad F(x) = \sum_{i=1}^n w_i h_i(x)$$

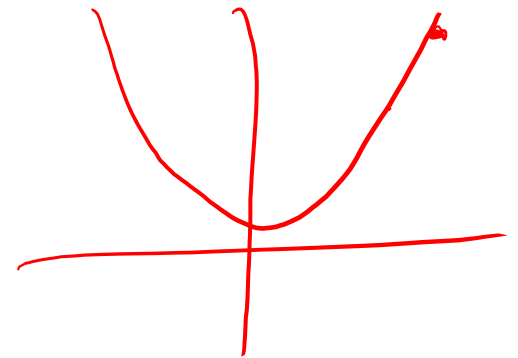
$$w^* = \underset{w^*}{\operatorname{argmin}} \underbrace{\left(\frac{1}{T} \sum_{i=1}^T (t_i - h_i(x_i))^2 \right)}_{\text{Loss}}$$

$$= \underset{w^*}{\operatorname{argmin}} \left(\frac{1}{T} \sum_{i=1}^T (t_i - h_i(x_i))^2 \right) + \lambda \|w\|^2$$

Regr \uparrow

$$w^* = (H^T H)^{-1} H^T t$$

$$w^*$$



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 \mathbf{w} using MLE:

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

Maximizing log-likelihood

Maximize wrt w :

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

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

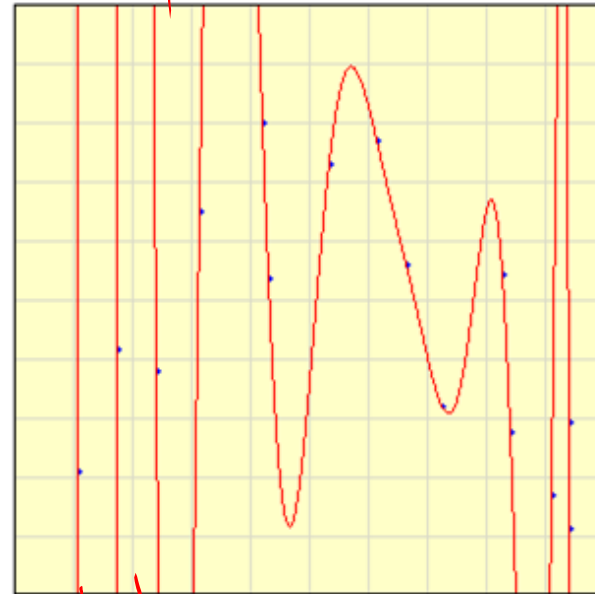
$$= \arg \max_w \sum_{j=1}^N \frac{-[t_j - \sum_i w_i h_i(x_j)]^2}{2\sigma^2}$$

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

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

Regularization in Linear Regression

- One sign of overfitting: large parameter values!



- *Regularized or penalized* regressions modified learning object to penalize large parameters

$\|w\|_2$



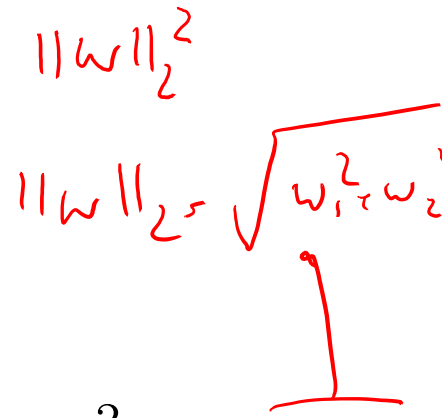
Ridge Regression

- Introduce a new objective function:

$$\hat{w}_{ridge} = \arg \min_w \sum_{j=1}^N \left(t(x_j) - \left(w_0 + \sum_{i=1}^k w_i h_i(x_j) \right) \right)^2$$

- Prefer low error but also add a squared penalize for large weights
- λ is hyperparameter that balances tradeoff
- Explicitly writing out bias feature (essentially $h_0=1$), which is not penalized

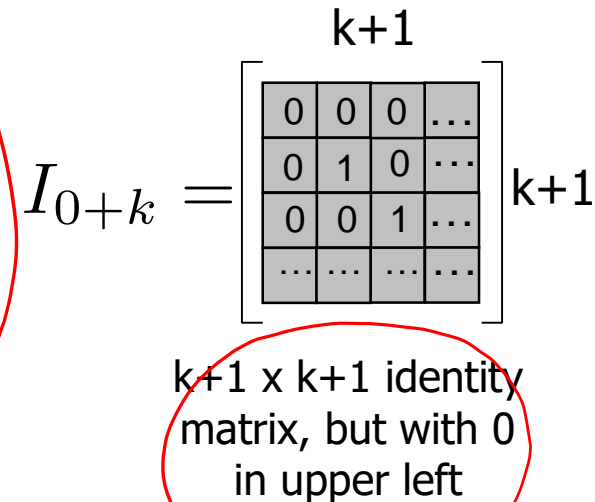
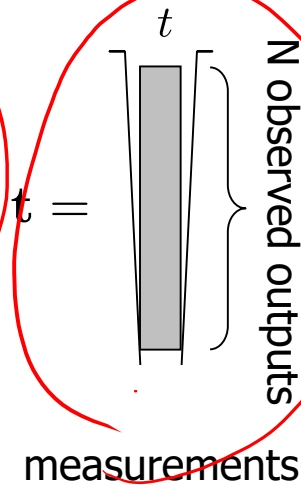
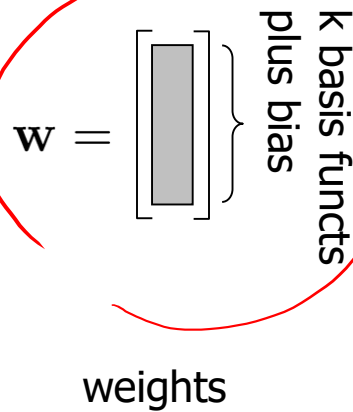
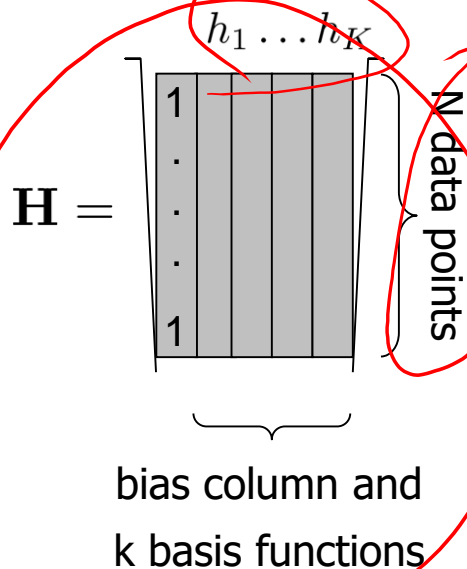
$\|w\|_2^2$
 $\|w\|_2 = \sqrt{w_1^2 + w_2^2}$



Ridge Regression: matrix notation

$$\hat{w}_{ridge} = \arg \min_w \sum_{j=1}^N \left(t(x_j) - \left(w_0 + \sum_{i=1}^k w_i h_i(x_j) \right) \right)^2 + \lambda \sum_{i=1}^k w_i^2$$

$$= \arg \min_w \underbrace{(\mathbf{H}\mathbf{w} - \mathbf{t})^T (\mathbf{H}\mathbf{w} - \mathbf{t})}_{\text{residual error}} + \lambda \mathbf{w}^T \mathbf{I}_{0+k} \mathbf{w}$$



Ridge
Regression:
closed form
solution

$$\begin{aligned}\hat{w}_{ridge} &= \arg \min_w \sum_{j=1}^N \left(t(x_j) - (w_0 + \sum_{i=1}^k w_i h_i(x_j)) \right)^2 + \lambda \sum_{i=1}^k w_i^2 \\ &= \arg \min_{\mathbf{w}} \underbrace{(\mathbf{H}\mathbf{w} - \mathbf{t})^T (\mathbf{H}\mathbf{w} - \mathbf{t})}_{\text{residual error}} + \lambda \mathbf{w}^T I_{0+k} \mathbf{w}\end{aligned}$$

$$\mathbf{F}(\mathbf{w}) = (\mathbf{H}\mathbf{w} - \mathbf{t})^T (\mathbf{H}\mathbf{w} - \mathbf{t}) + \lambda \mathbf{w}^T I_{0+k} \mathbf{w}$$

$$\nabla_{\mathbf{w}} \mathbf{F}(\mathbf{w}) = \mathbf{0}$$

$$2\mathbf{H}^T (\mathbf{H}\mathbf{w} - \mathbf{t}) + 2\lambda I_{0+k} \mathbf{w} = \mathbf{0}$$

$$w_{ridge}^* = (\mathbf{H}^T \mathbf{H} + \lambda I_{0+k})^{-1} \mathbf{H}^T \mathbf{t}$$

Regression solution: simple matrix math

$$\begin{aligned}\hat{w}_{ridge} &= \arg \min_w \sum_{j=1}^N \left(t(x_j) - \left(w_0 + \sum_{i=1}^k w_i h_i(x_j) \right) \right)^2 + \lambda \sum_{i=1}^k w_i^2 \\ &= \arg \min_{\mathbf{w}} \underbrace{(\mathbf{H}\mathbf{w} - \mathbf{t})^T (\mathbf{H}\mathbf{w} - \mathbf{t})}_{\text{residual error}} + \lambda \mathbf{w}^T I_{0+k} \mathbf{w}\end{aligned}$$

$$w_{ridge}^* = (\mathbf{H}^T \mathbf{H} + \lambda I_{0+k})^{-1} \mathbf{H}^T \mathbf{t}$$

Compare to un-regularized regression:

$$\mathbf{w}^* = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{t}$$

Ridge Regression

How does varying lambda change w?

$$\hat{w}_{ridge} = \arg \min_w \sum_{j=1}^N \left(t(x_j) - \left(w_0 + \sum_{i=1}^k w_i h_i(x_j) \right) \right)^2 + \lambda \sum_{i=1}^k w_i^2$$

– Larger λ ? Smaller λ ?

– As $\lambda \rightarrow 0$?

- Becomes same as MLE, unregularized

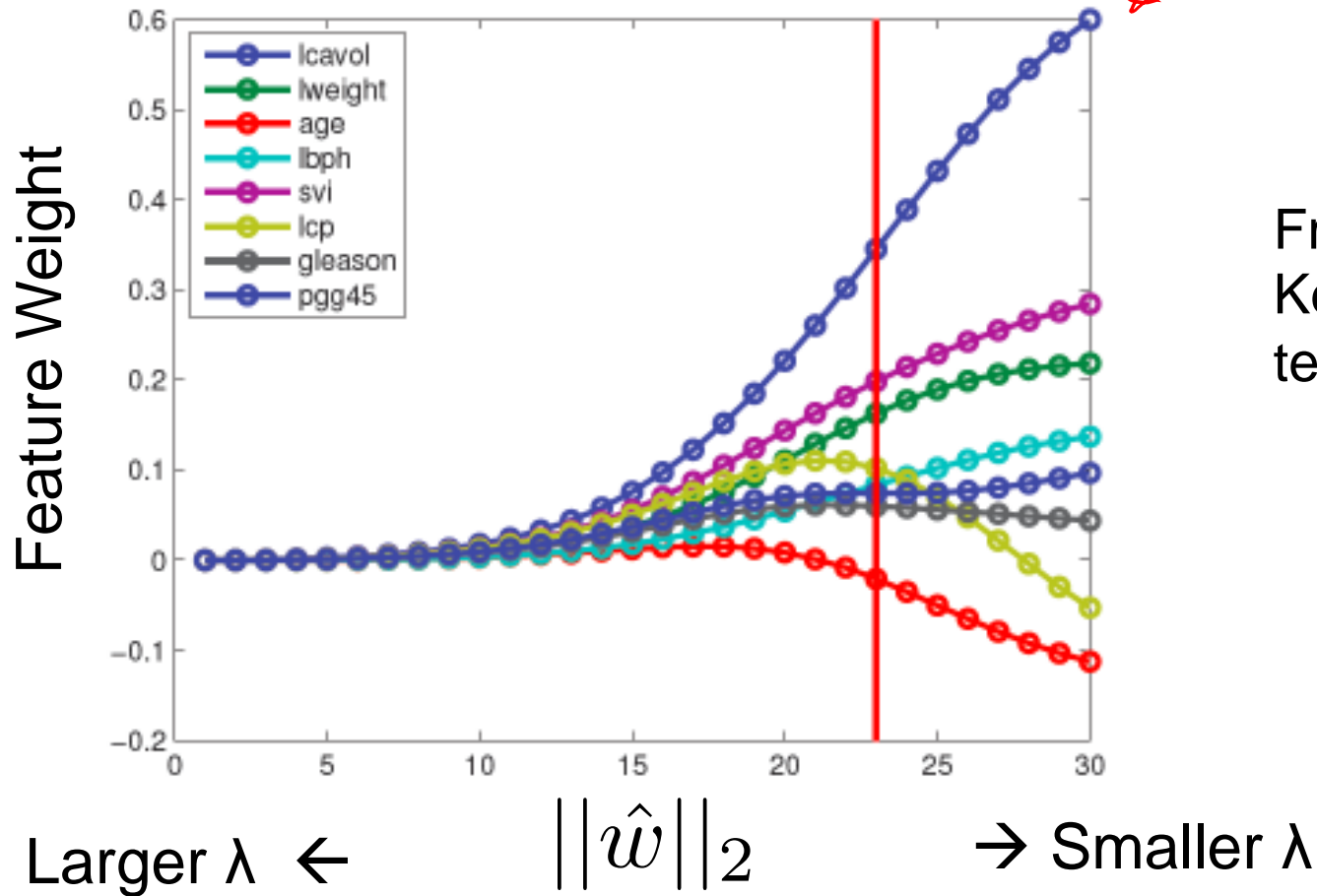
– As $\lambda \rightarrow \infty$?

- All weights will be 0!

$w = \vec{0}$

—

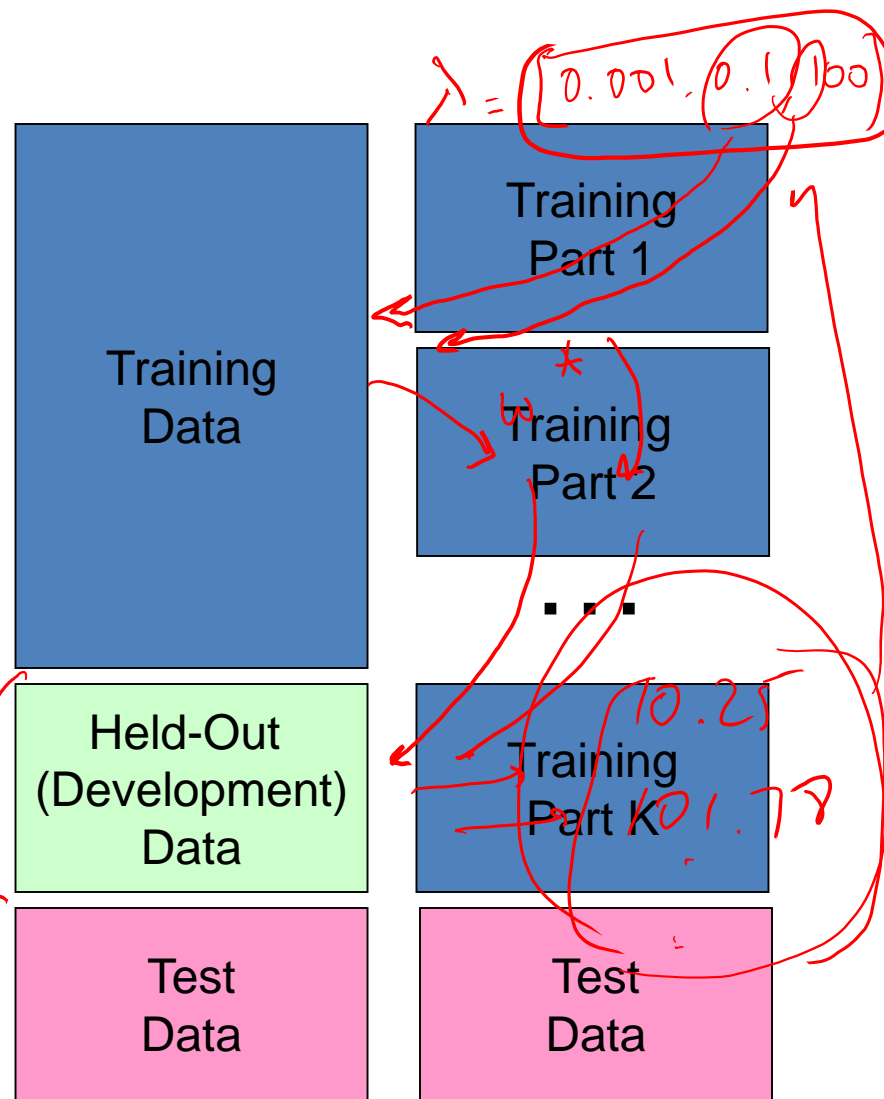
Ridge Coefficient Path



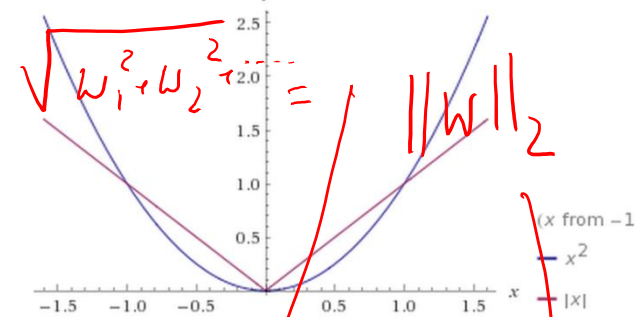
From
Kevin Murphy
textbook

How to pick lambda?

- **Experimentation cycle**
 - Select a hypothesis f to best match training set
 - Tune hyperparameters on held-out set
 - Try many different values of lambda, pick best one
- Or, can do **k-fold cross validation**
 - No held-out set
 - Divide training set into k subsets
 - Repeatedly train on $k-1$ and test on remaining one
 - Average the results



Why squared regularization?



- Ridge:

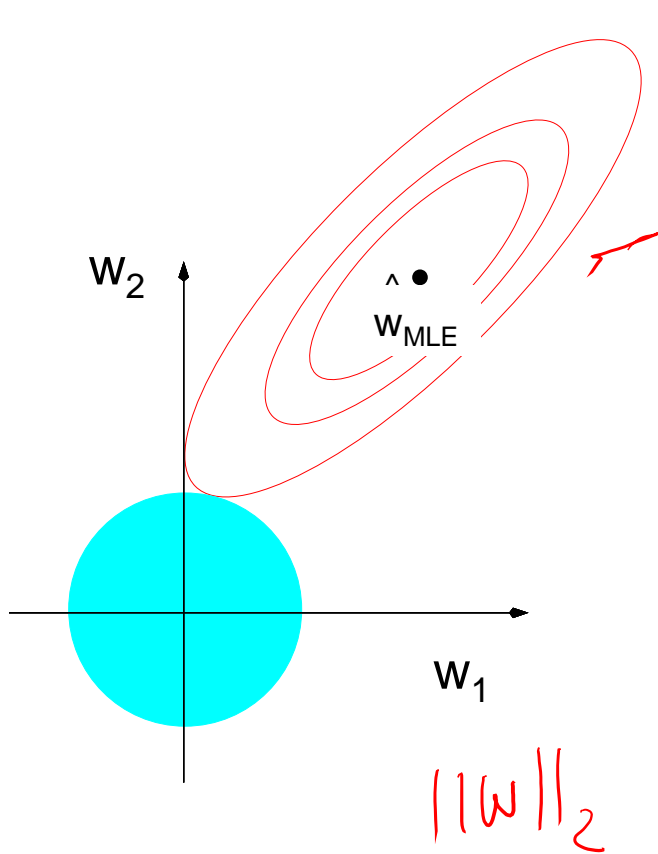
$$\hat{w}_{ridge} = \arg \min_w \sum_{j=1}^N \left(t(x_j) - (w_0 + \sum_{i=1}^k w_i h_i(x_j)) \right)^2 + \lambda \sum_{i=1}^k w_i^2$$

- LASSO:

$$\hat{w}_{LASSO} = \arg \min_w \sum_{j=1}^N \left(t(x_j) - (w_0 + \sum_{i=1}^k w_i h_i(x_j)) \right)^2 + \lambda \sum_{i=1}^k |w_i|$$

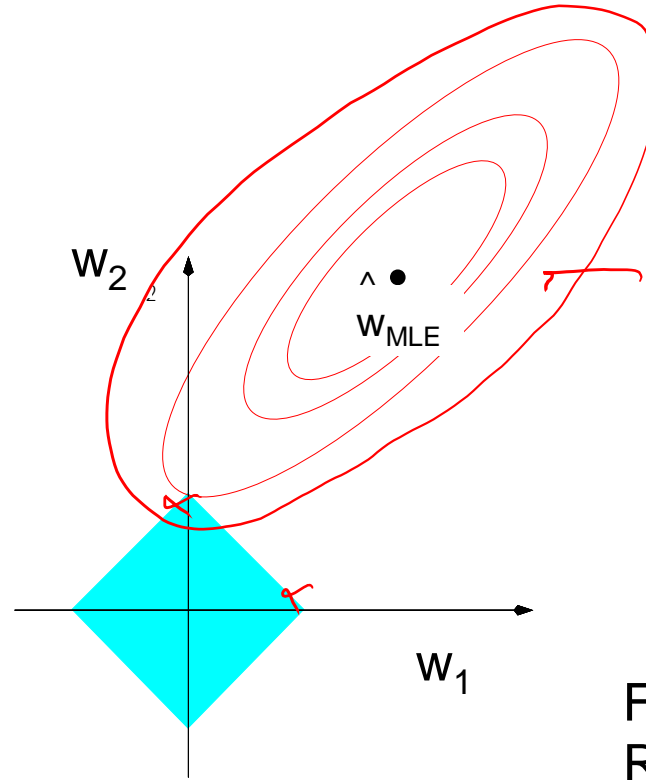
- Linear penalty pushes more weights to zero
- Allows for a type of *feature selection*
- But, not differentiable and no closed form solution....

Geometric Intuition



Ridge Regression

$$\|w\|_2$$

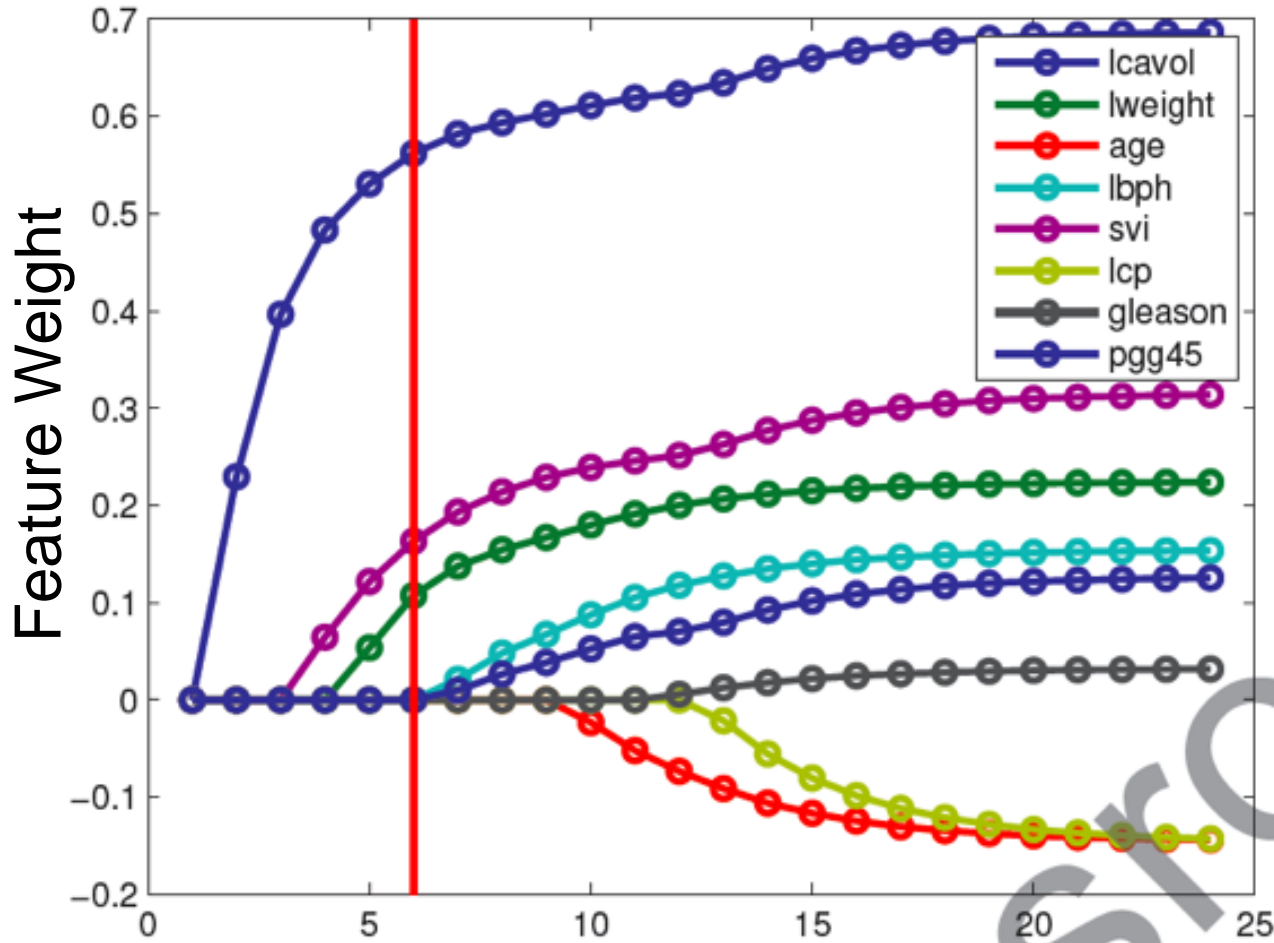


Lasso

$$\|w\|_1$$

From
Rob
Tibshirani
slides

LASSO Coefficient Path

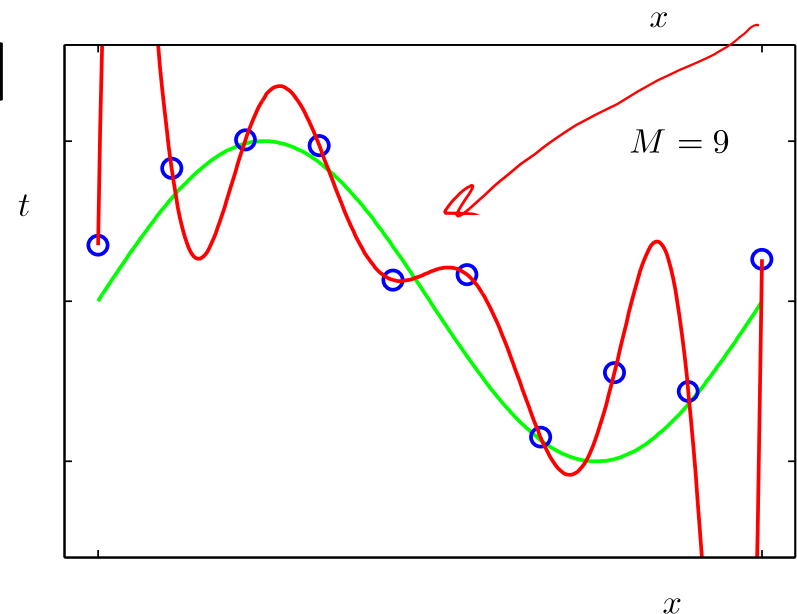
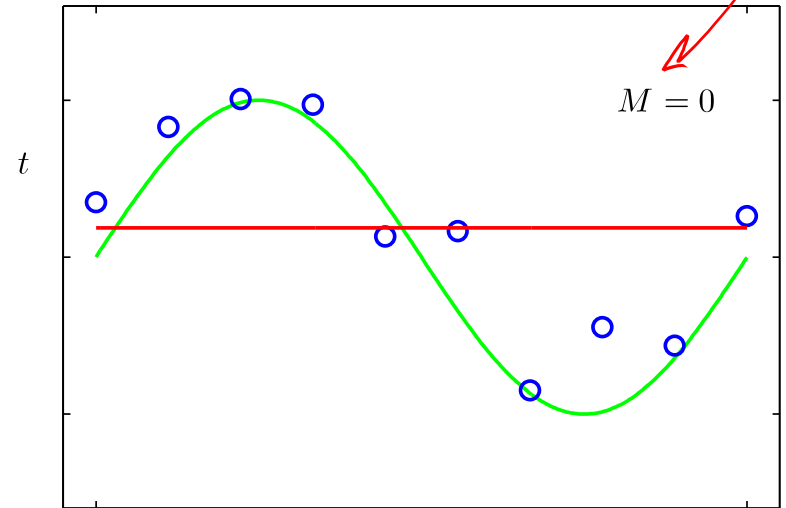


From
Kevin Murphy
textbook

Larger λ \leftarrow $\left\| \hat{w} \right\|_1$ \rightarrow Smaller λ

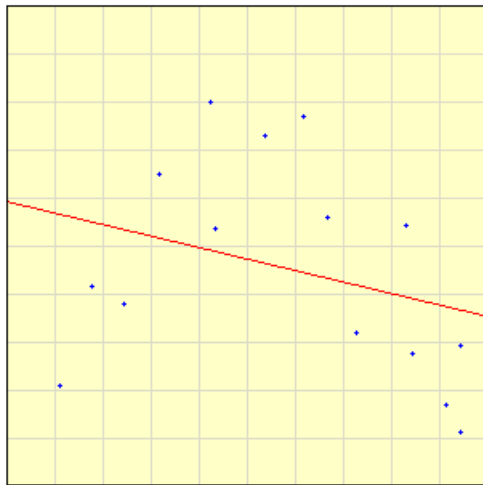
Bias-Variance tradeoff – Intuition

- **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



Bias-Variance Tradeoff

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

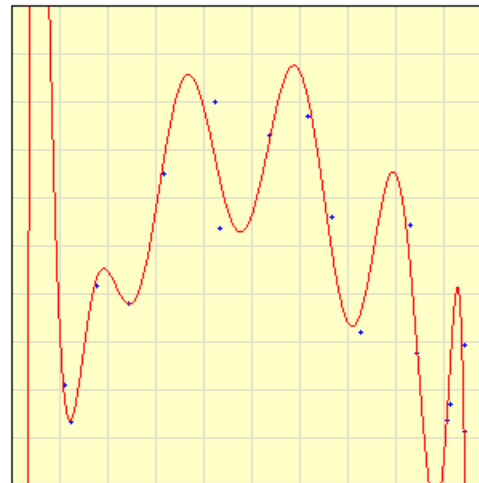


Select points by clicking on the graph or press

Example

Degree of polynomial: Fit Y to X
 Fit X to Y

Calculate View Polynomial Reset

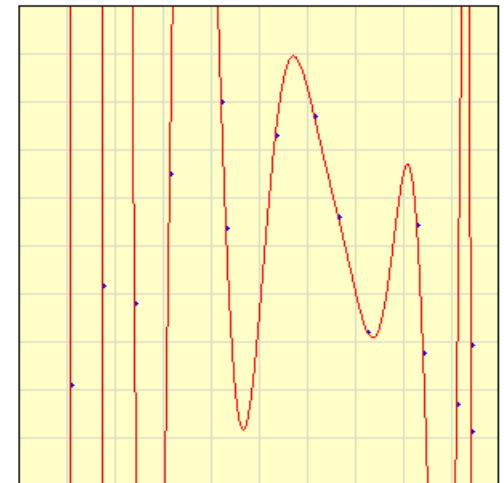


Select points by clicking on the graph or press

Example

Degree of polynomial: 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: Fit Y to X
 Fit X to Y

Calculate View Polynomial Reset

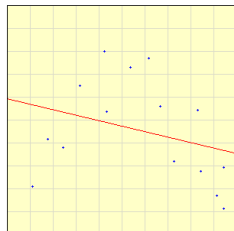
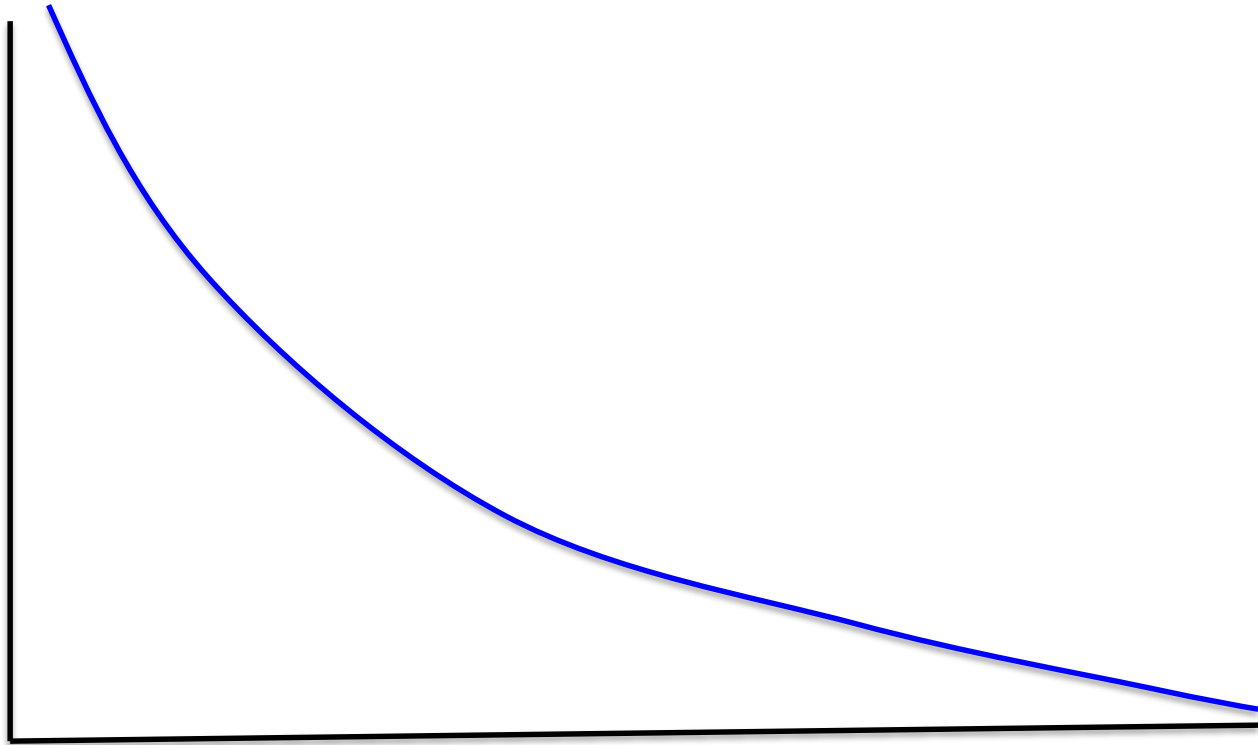
Training set error $\mathbf{w}^* = \arg \min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$

- 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

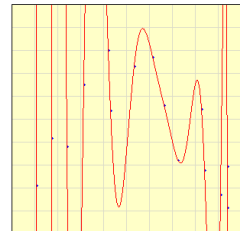
$$\text{error}_{\text{train}}(\mathbf{w}) = \frac{1}{N_{\text{train}}} \sum_{j=1}^{N_{\text{train}}} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$



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

Degree of polynomial: Fit Y to X
 Fit X to Y

[Calculate](#) [View Polynomial](#) [Reset](#)

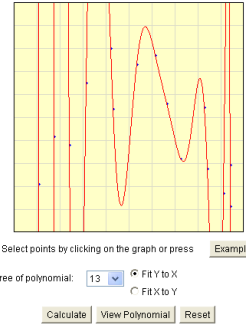


Select points by clicking on the graph or press [Example](#)

Degree of polynomial: Fit Y to X
 Fit X to Y

[Calculate](#) [View Polynomial](#) [Reset](#)

Prediction error



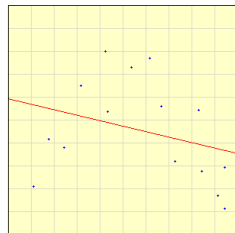
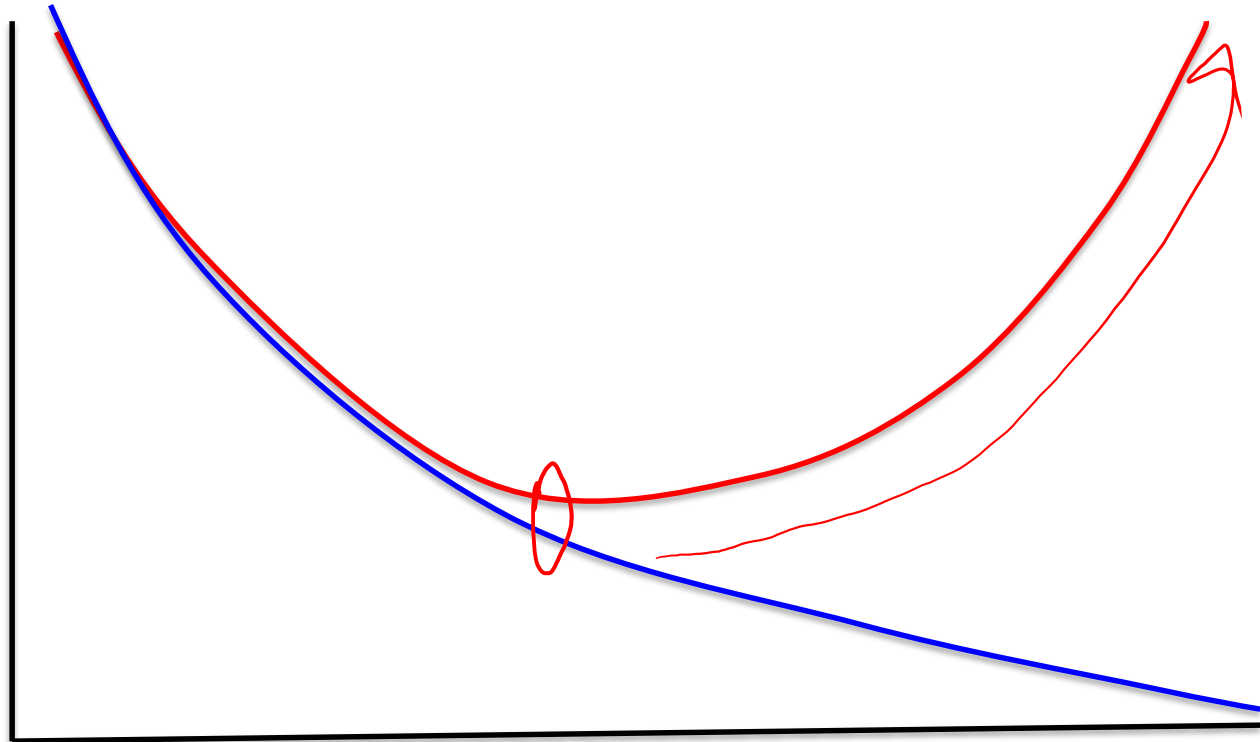
- Training set error can be poor measure of “quality” of solution
- Prediction error (true error): We really care about error over all possibilities:

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

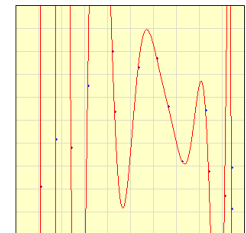
Prediction error as a function of model complexity

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

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



Select points by clicking on the graph or press [Example](#)
 Degree of polynomial: Fit Y to X Fit X to Y



Select points by clicking on the graph or press [Example](#)
 Degree of polynomial: Fit Y to X Fit X to Y

Computing prediction error

- To correctly predict error

- Hard integral!
- May not know $t(\mathbf{x})$ for every \mathbf{x} , may not know $p(\mathbf{x})$

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

- Monte Carlo integration (sampling approximation)

- Sample a set of i.i.d. points $\{\mathbf{x}_1, \dots, \mathbf{x}_M\}$ from $p(\mathbf{x})$
- Approximate integral with sample average

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

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?

- Sample error
Because you cheated!!!
Training error good estimate for a single \mathbf{w} ,
But you optimized \mathbf{w} with respect to the training error,
and found \mathbf{w} that is good for this set of samples
- Training error
Training error is a (optimistically) biased estimate of prediction error
- 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 – $\{\mathbf{x}_1, \dots, \mathbf{x}_{N_{\text{train}}}\}$
 - Test data – $\{\mathbf{x}_1, \dots, \mathbf{x}_{N_{\text{test}}}\}$
- Use training data to optimize parameters \mathbf{w}
- **Test set error:** For the *final solution* \mathbf{w}^* , evaluate the error using:

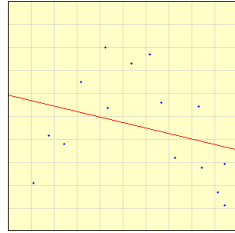
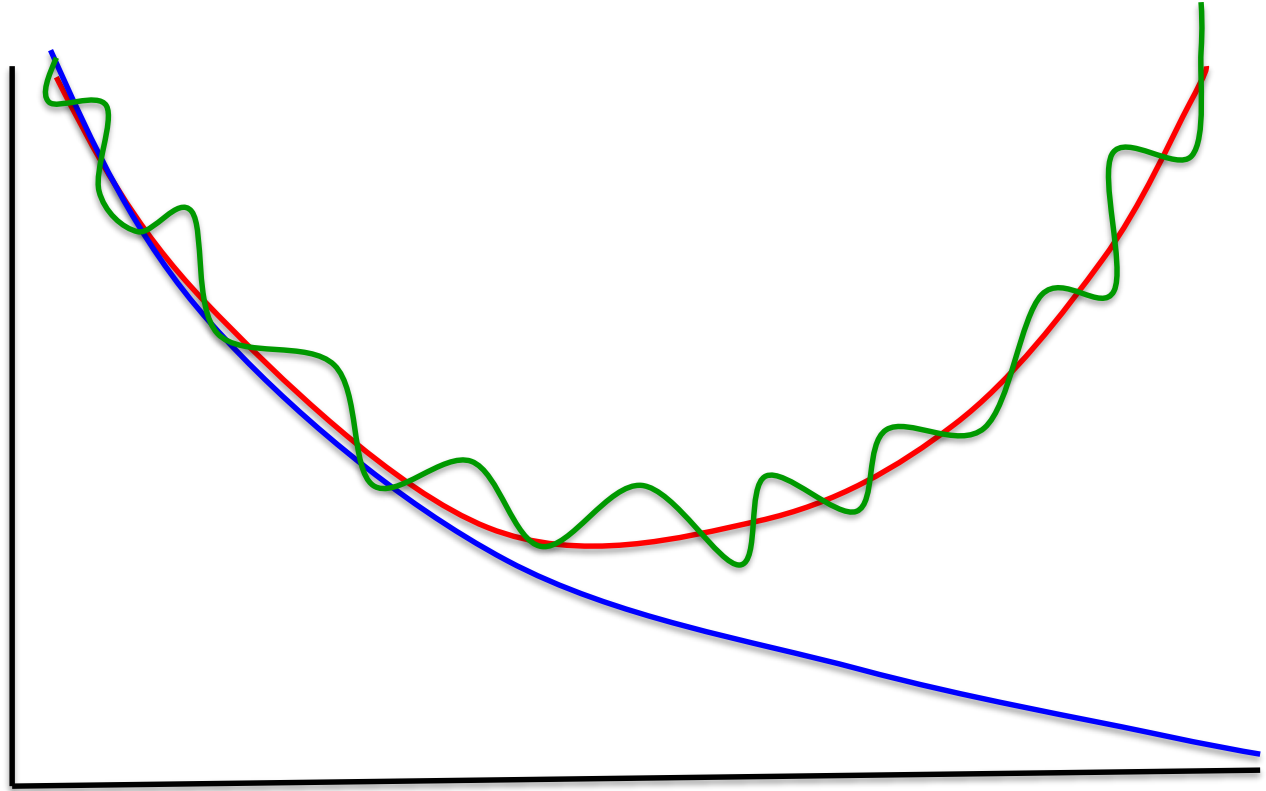
$$error_{test}(\mathbf{w}) = \frac{1}{N_{test}} \sum_{j=1}^{N_{test}} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$

Test set error as a function of model complexity

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

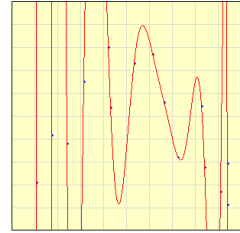
$$error_{true}(\mathbf{w}) = \int_{\mathbf{x}} \left(t(\mathbf{x}) - \sum_i w_i h_i(\mathbf{x}) \right)^2 p(\mathbf{x}) d\mathbf{x}$$

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



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

Degree of polynomial: FIT to X FIT to Y



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

Degree of polynomial: FIT to X FIT to Y

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:

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


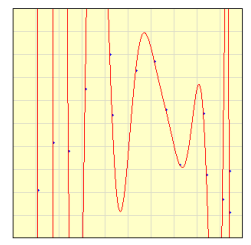
- Training: optimistically biased

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


- Test: our final measure

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

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

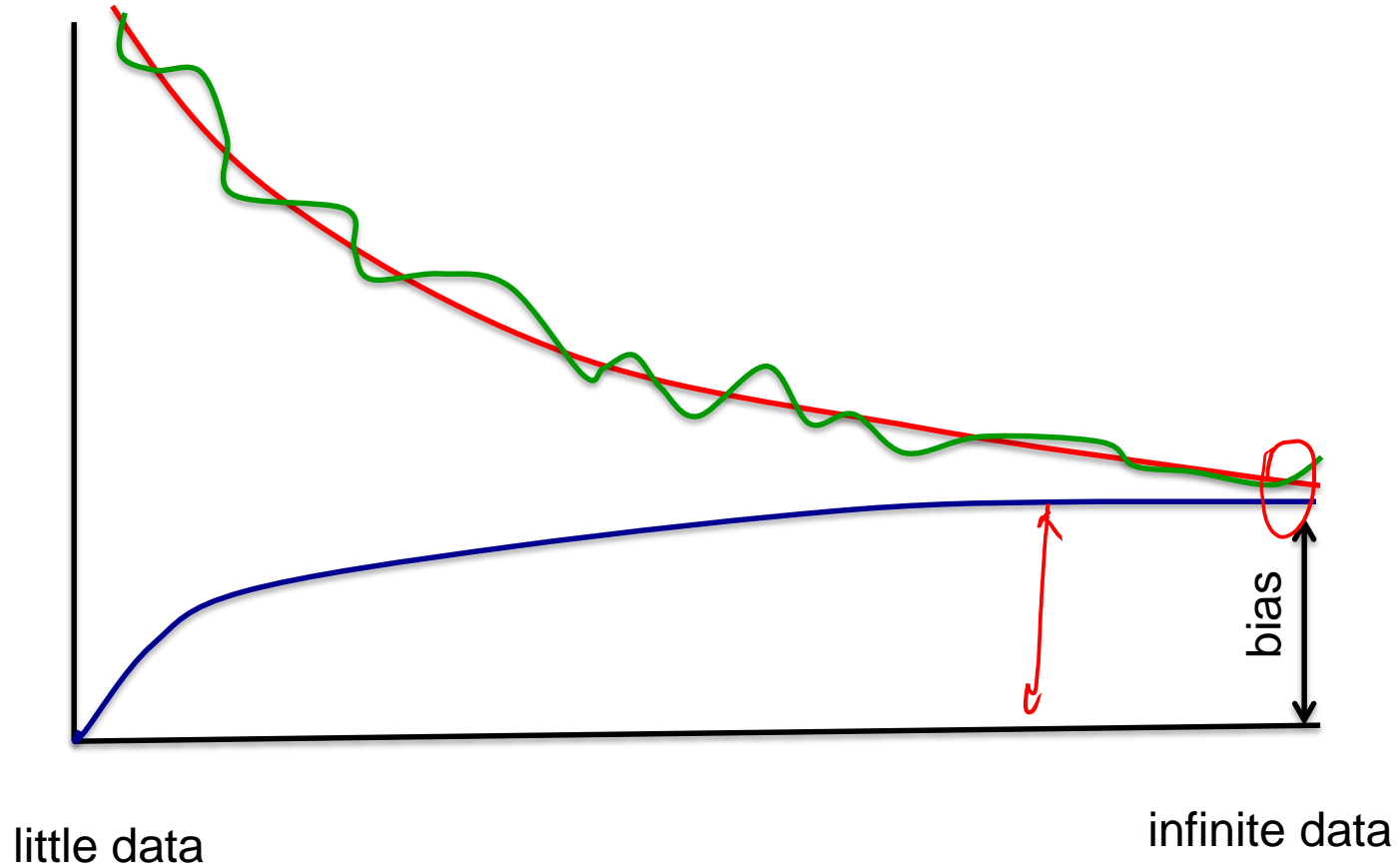


Select points by clicking on the graph or press [Example](#)
 Degree of polynomial: Fit Y to X Fit X to Y

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

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

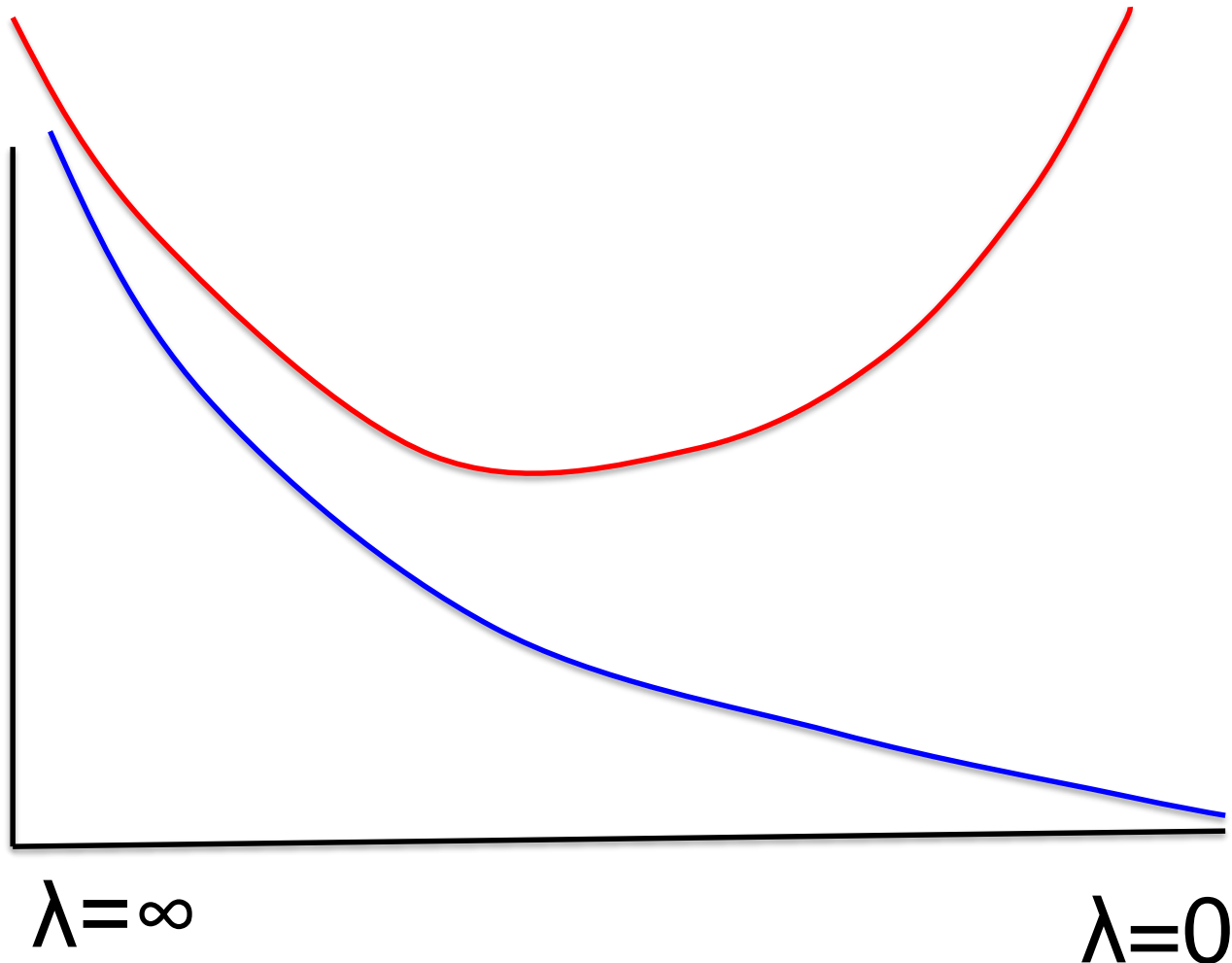
$$error_{test}(\mathbf{w}) = \frac{1}{N_{test}} \sum_{j=1}^{N_{test}} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$



Error as function of regularization parameter, fixed model complexity

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

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



Summary: error estimators

- G

Be careful!!!

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

- T

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 quarter)

- Test: our final measure

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

What you need to know

- Regression

- Basis function = features

- Optimizing sum squared error

- Relationship between regression and Gaussians

- Regularization

- Ridge regression math

- LASSO Formulation

- How to set lambda

- Bias-Variance trade-off

$\|w\|_2^2$

$\|w\|_1$

→ feature select