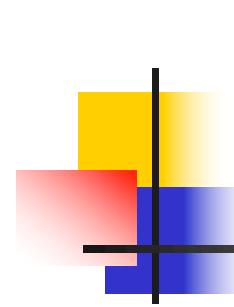


# CSCI 4520 - Introduction to Machine Learning

Mehdi Allahyari  
Georgia Southern University

## Linear Regression

(most of the slides borrowed from Emily Fox, Tom Mitchell and Ali Farhadi



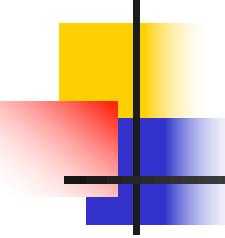
# Regression

---

So far, we've been interested in learning  $P(Y|X)$  where  $Y$  has discrete values (called 'classification')

What if  $Y$  is continuous? (called 'regression')

- predict weight from gender, height, age, ...
- predict Google stock price today from Google, Yahoo, MSFT prices yesterday
- predict each pixel intensity in robot's current camera image, from previous image and previous action



# Regression

---

Wish to learn  $f: X \rightarrow Y$ , where  $Y$  is real, given  $\{<x^1, y^1> \dots <x^n, y^n>\}$

Approach:

1. choose some parameterized form for  $P(Y|X; \theta)$   
( $\theta$  is the vector of parameters)
2. derive learning algorithm as MLE or MAP estimate for  $\theta$

We'll get back to this slide!

# How much is my house worth?



# How much is my house worth?



## Data



*input*      *output*  
 $(x_1 = \text{sq.ft.}, y_1 = \$)$



$(x_2 = \text{sq.ft.}, y_2 = \$)$



$(x_3 = \text{sq.ft.}, y_3 = \$)$



$(x_4 = \text{sq.ft.}, y_4 = \$)$



$(x_5 = \text{sq.ft.}, y_5 = \$)$

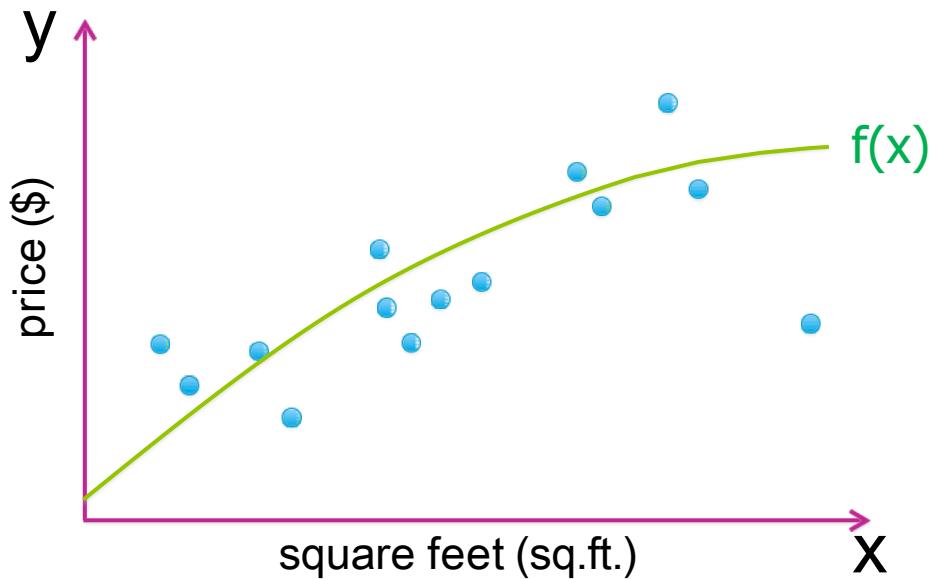
⋮

### Input vs. Output:

- $y$  is the quantity of interest
- assume  $y$  can be predicted from  $x$

# Model

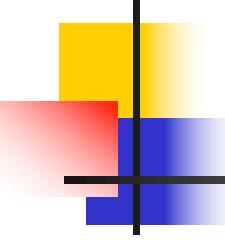
How we *assume* the world works



Regression model:

$$y_i = f(x_i) + \epsilon_i$$

where  $\epsilon \sim N(0, \sigma)$

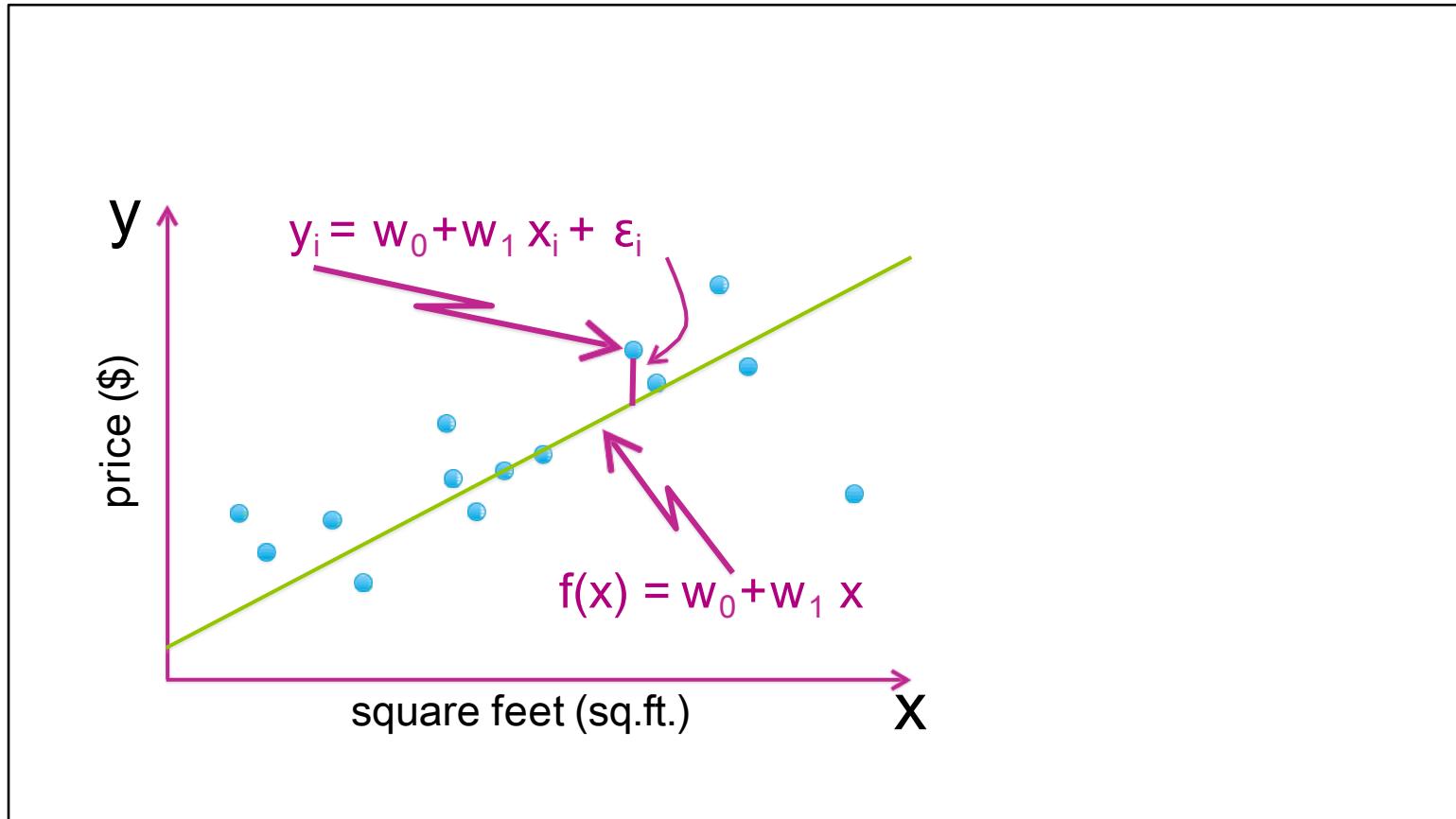


# Noise

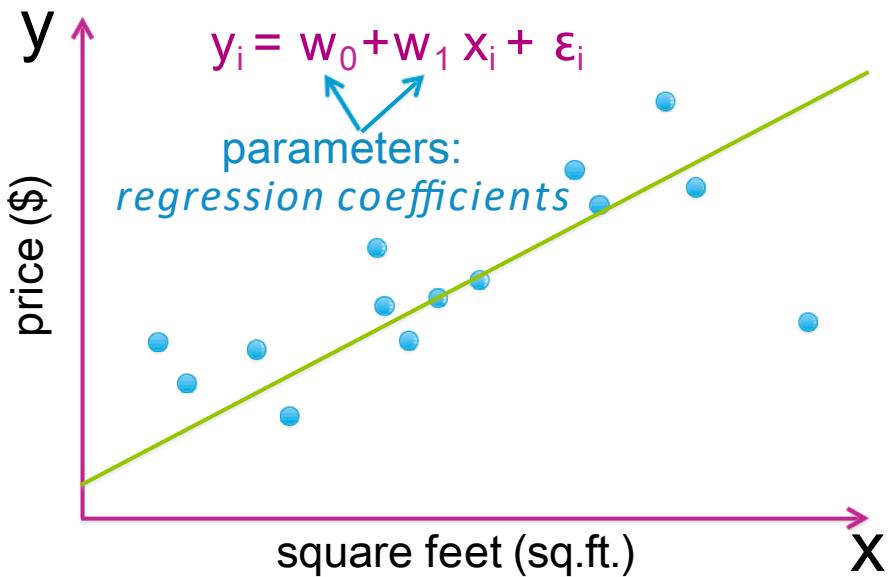
---

- A simple model typically does not exactly fit the data – lack of fit can be considered **noise**
- Sources of noise:
  - Imprecision in data attributes (input noise)
  - Errors in data targets (mislabeling)
  - Additional attributes not taken into account by data attributes, affect target values (latent variables)
  - Model may be too simple to account for data targets

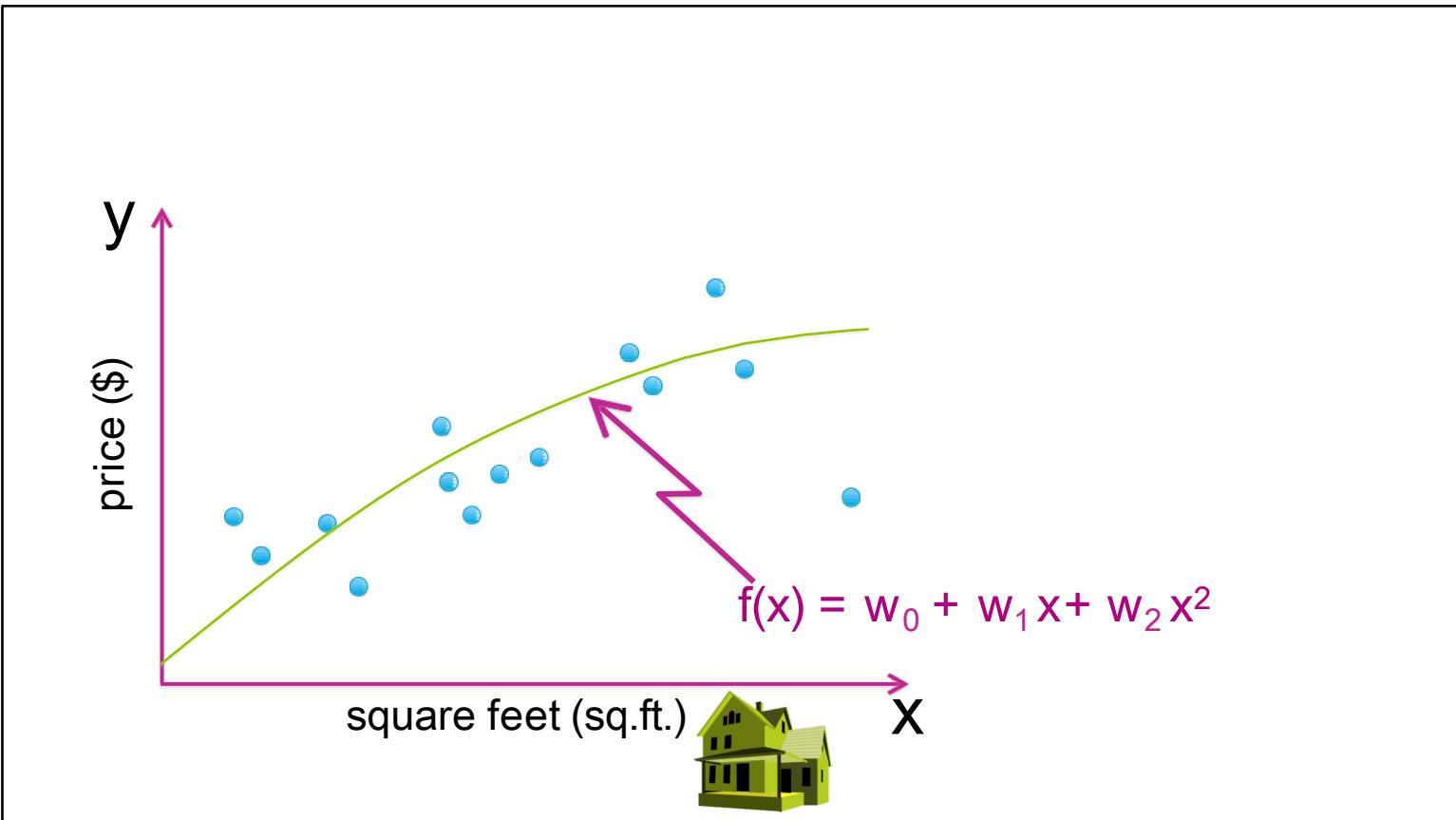
# Simple linear regression model



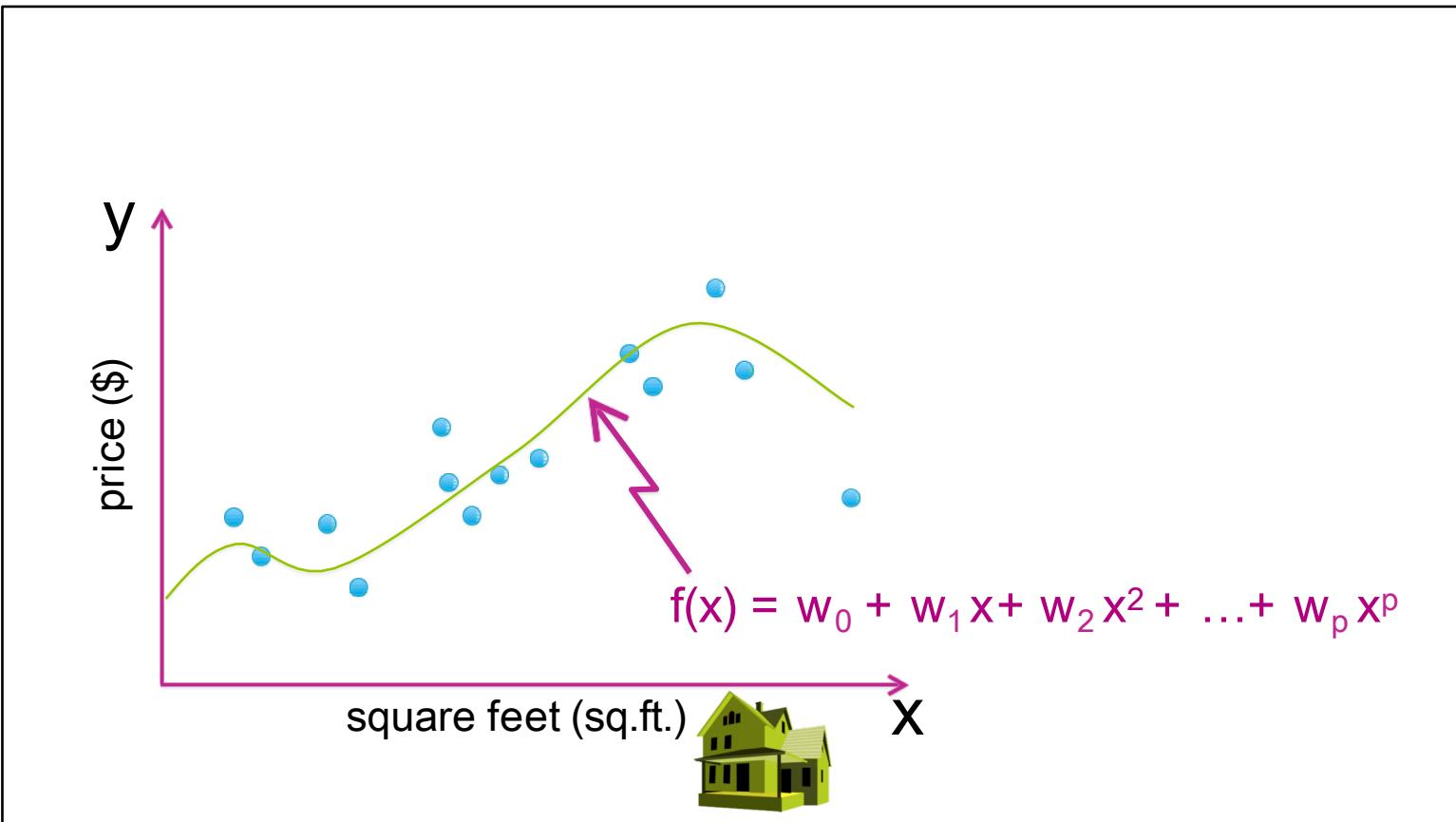
# Simple linear regression model



# What about a quadratic function?



# Even higher order polynomial



# Polynomial regression

Model:

$$y_i = w_0 + w_1 x_i + w_2 x_i^2 + \dots + w_p x_i^p + \varepsilon_i$$

treat as different features

feature 1 = 1 (constant) parameter 1 =  $w_0$

feature 2 =  $x$  parameter 2 =  $w_1$

feature 3 =  $x^2$  parameter 3 =  $w_2$

...

...

feature  $p+1 = x^p$  parameter  $p+1 = w_p$

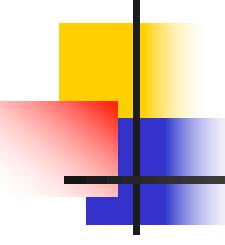
# Generic basis expansion

Model:

$$\begin{aligned}y_i &= w_0 h_0(x_i) + w_1 h_1(x_i) + \dots + w_D h_D(x_i) + \varepsilon_i \\&= \sum_{j=0}^D w_j h_j(x_i) + \varepsilon_i\end{aligned}$$

*j<sup>th</sup> regression coefficient  
or weight*

*j<sup>th</sup> feature*



# Generic basis expansion

Model:

$$\begin{aligned}y_i &= w_0 h_0(x_i) + w_1 h_1(x_i) + \dots + w_D h_D(x_i) + \epsilon_i \\&= \sum_{j=0}^D w_j h_j(x_i) + \epsilon_i\end{aligned}$$

*feature 1 =  $h_0(x)$ ... often 1 (constant)*

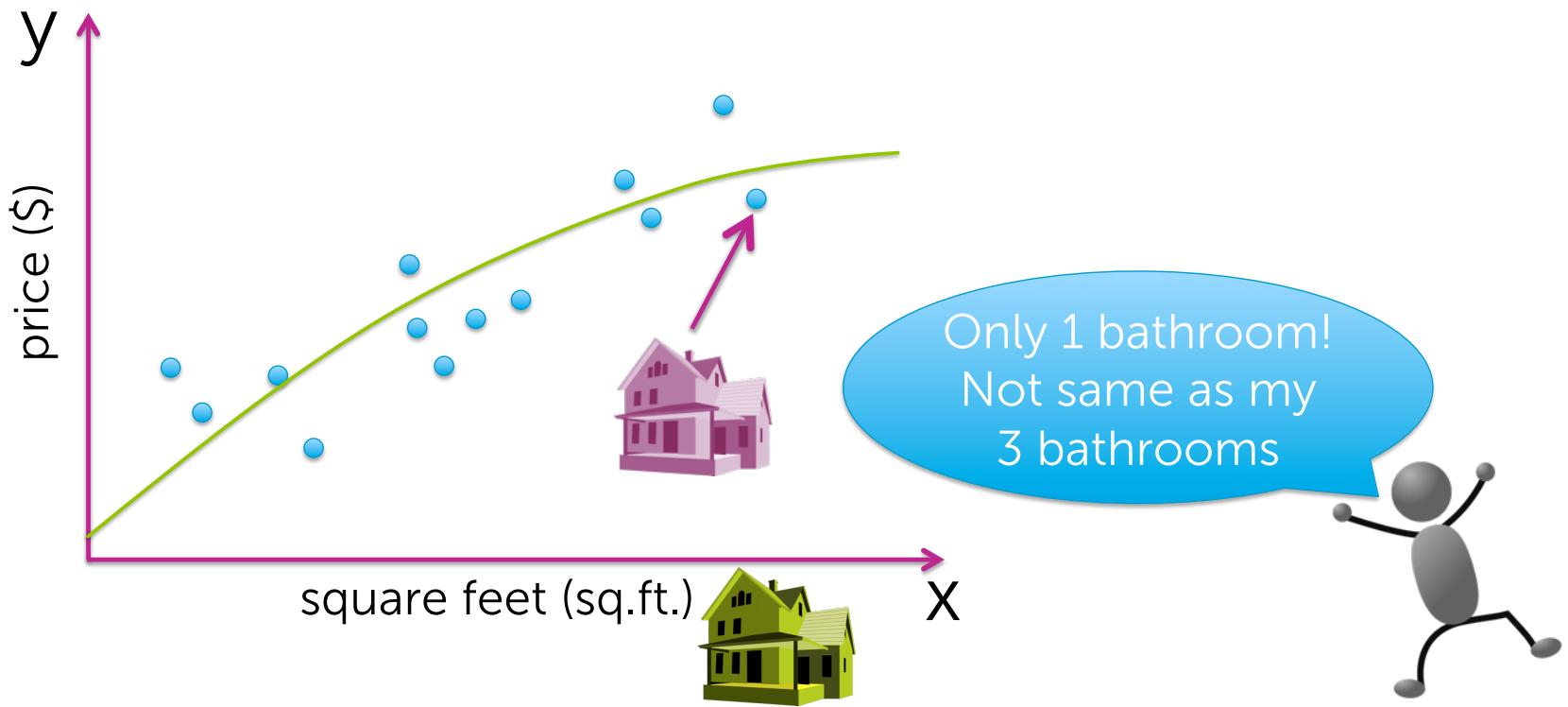
*feature 2 =  $h_1(x)$ ... e.g.,  $x$*

*feature 3 =  $h_2(x)$ ... e.g.,  $x^2$  or  $\sin(2\pi x/12)$*

...

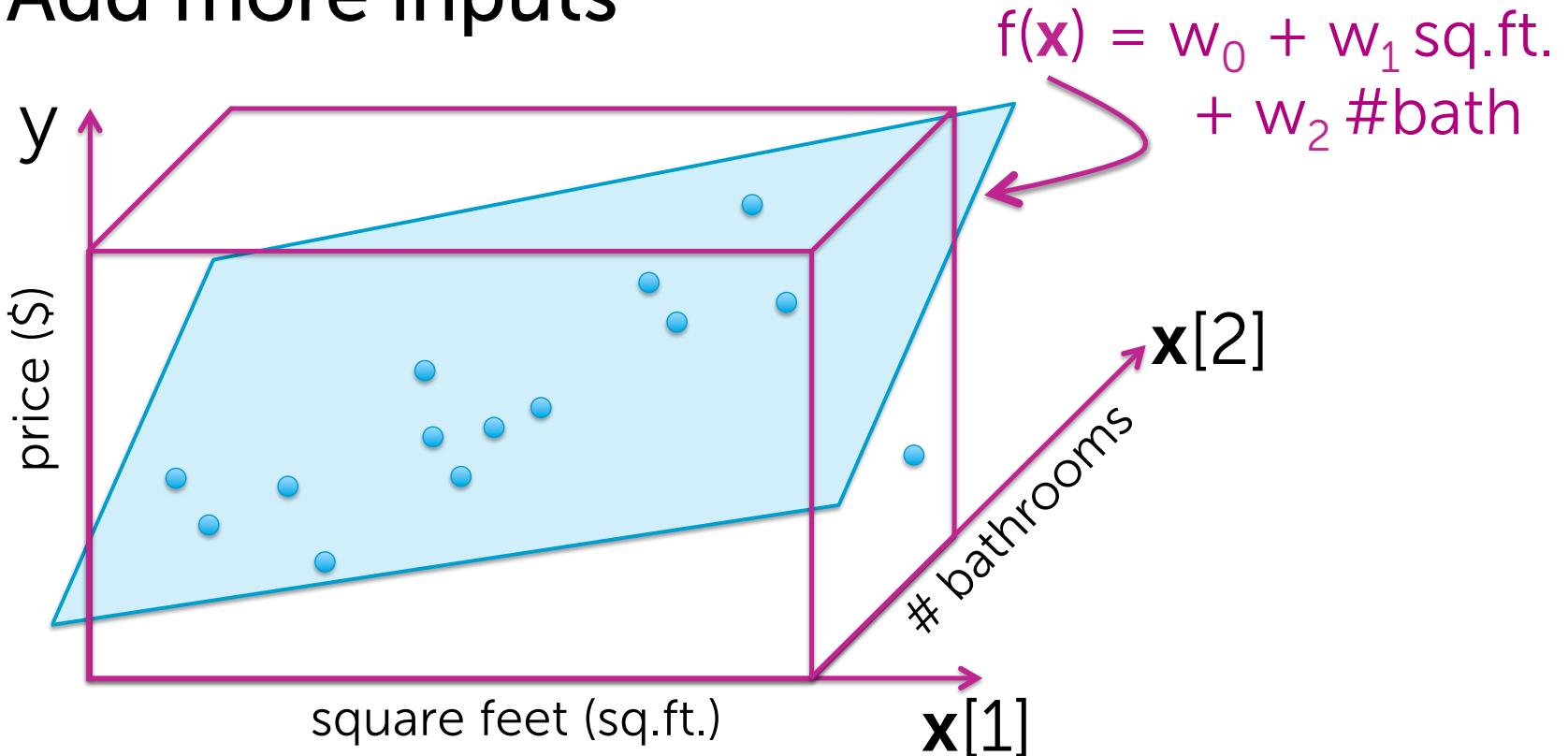
*feature  $D+1 = h_D(x)$ ... e.g.,  $x^p$*

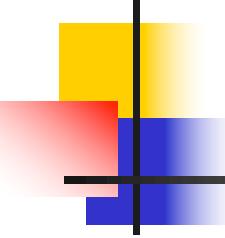
# Predictions just based on house size



# Add more inputs

## Add more inputs

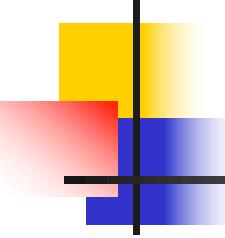




# Many possible inputs

---

- Square feet
- # bathrooms
- # bedrooms
- Lot size
- Year built
- ...



# The Regression Problem

- Instances:  $\langle \mathbf{x}_j, t_j \rangle$
- Learn: Mapping from  $\mathbf{x}$  to  $t(\mathbf{x})$
- Hypothesis space:
  - Given, basis functions  $\{h_1, \dots, h_k\}$
  - $h_i(\mathbf{x}) \in \mathbb{R}$
  - Find coeffs  $\mathbf{w} = \{w_1, \dots, w_k\}$
- 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**:

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

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

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

# General Notation

Output:  $y \leftarrow$  scalar

Inputs:  $\mathbf{x} = (\mathbf{x}[1], \mathbf{x}[2], \dots, \mathbf{x}[d])$

$\uparrow$   
d-dim vector

Notational conventions:

$\mathbf{x}[j] = j^{\text{th}}$  input (scalar)

$h_j(\mathbf{x}) = j^{\text{th}}$  feature (scalar)

$\mathbf{x}_i =$  input of  $i^{\text{th}}$  data point (vector)

$\mathbf{x}_i[j] = j^{\text{th}}$  input of  $i^{\text{th}}$  data point (scalar)

# Generic Linear Regression Model

Model:

$$\begin{aligned}y_i &= w_0 h_0(\mathbf{x}_i) + w_1 h_1(\mathbf{x}_i) + \dots + w_D h_D(\mathbf{x}_i) + \varepsilon_i \\&= \sum_{j=0}^D w_j h_j(\mathbf{x}_i) + \varepsilon_i\end{aligned}$$

*feature 1 =  $h_0(\mathbf{x})$  ... e.g., 1*

*feature 2 =  $h_1(\mathbf{x})$  ... e.g.,  $\mathbf{x}[1]$  = sq. ft.*

*feature 3 =  $h_2(\mathbf{x})$  ... e.g.,  $\mathbf{x}[2]$  = #bath*

*or,  $\log(\mathbf{x}[7])$   $\mathbf{x}[2] = \log(\#bed) \times \#bath$*

...

*feature  $D+1 = h_D(\mathbf{x})$  ... some other function of  $\mathbf{x}[1], \dots, \mathbf{x}[d]$*



# Fitting the Linear Regression Model

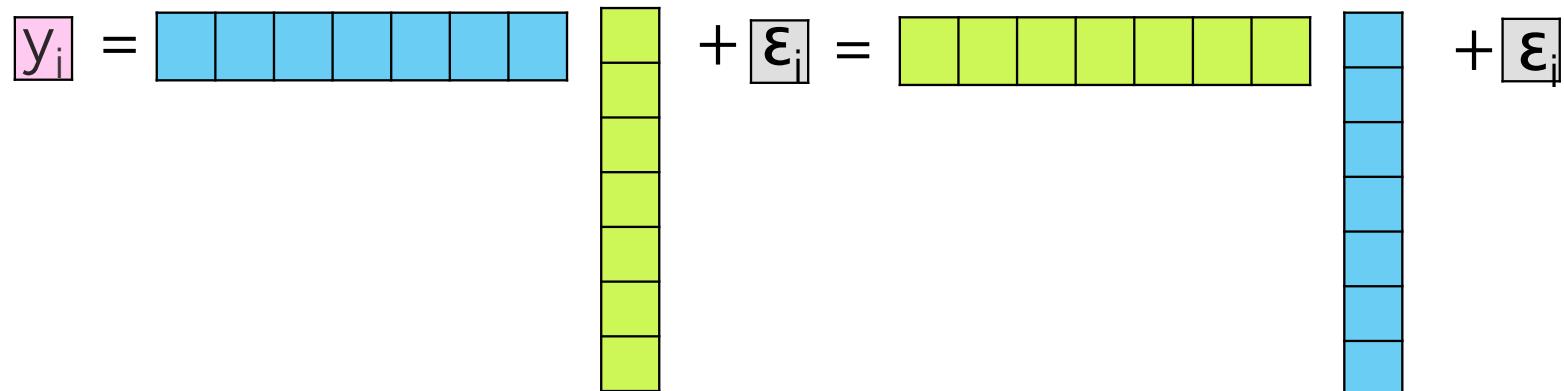


## **Step 1:** Rewrite the regression model

# Rewrite in Matrix Notation

For observation i

$$y_i = \sum_{j=0}^D w_j h_j(\mathbf{x}_i) + \varepsilon_i$$



# Rewrite in Matrix Notation

$$\mathbf{Y} = \mathbf{HW} + \boldsymbol{\epsilon}$$

For all observations together

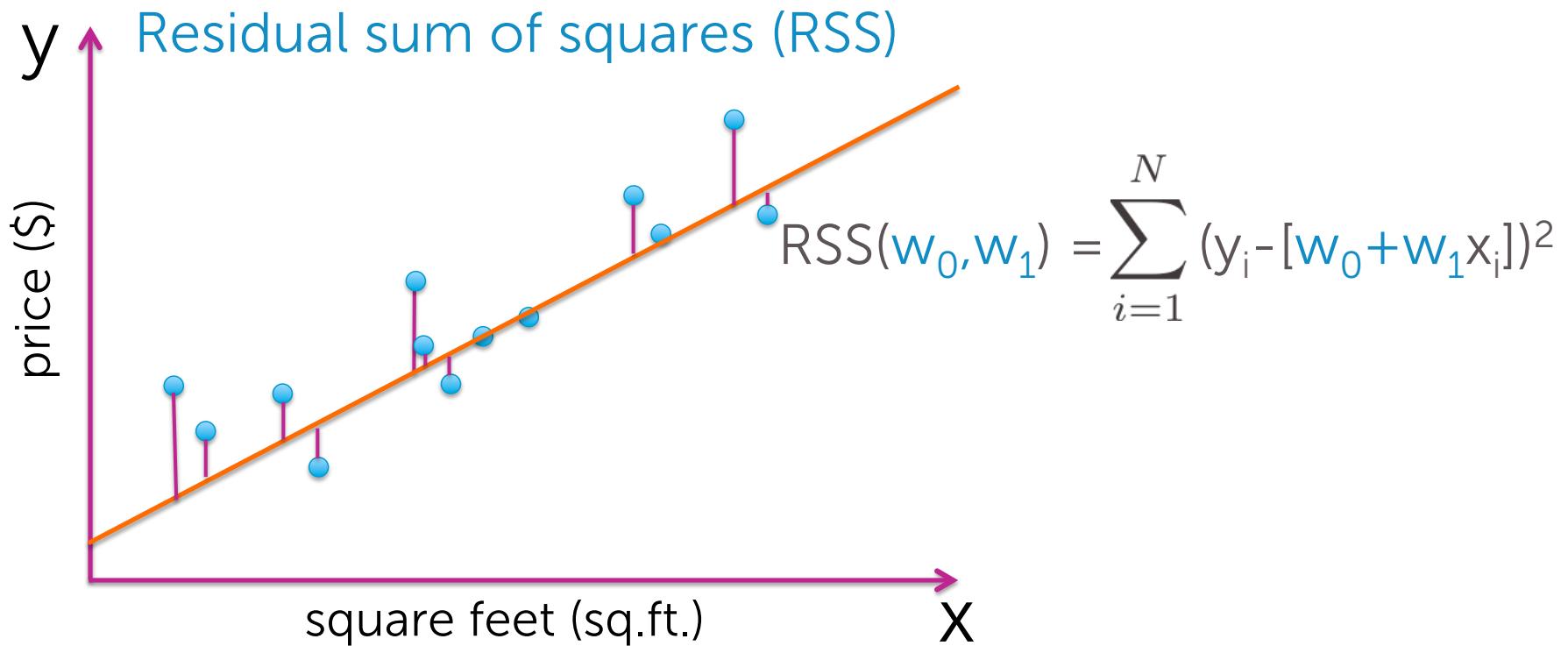
$$\mathbf{Y} = \mathbf{H}\mathbf{W} + \boldsymbol{\epsilon}$$

The diagram illustrates the matrix equation  $\mathbf{Y} = \mathbf{H}\mathbf{W} + \boldsymbol{\epsilon}$ . On the left, a vertical vector  $\mathbf{Y}$  is shown as a stack of 12 pink squares. To its right is an equals sign (=). Next is a 12x12 grid matrix  $\mathbf{H}$ , where the letter 'H' is prominently displayed in the center cell. To the right of  $\mathbf{H}$  is a vertical vector  $\mathbf{W}$  composed of 4 blue squares. Following  $\mathbf{W}$  is a plus sign (+). To the right of the plus sign is a vertical vector  $\boldsymbol{\epsilon}$  composed of 8 grey squares.

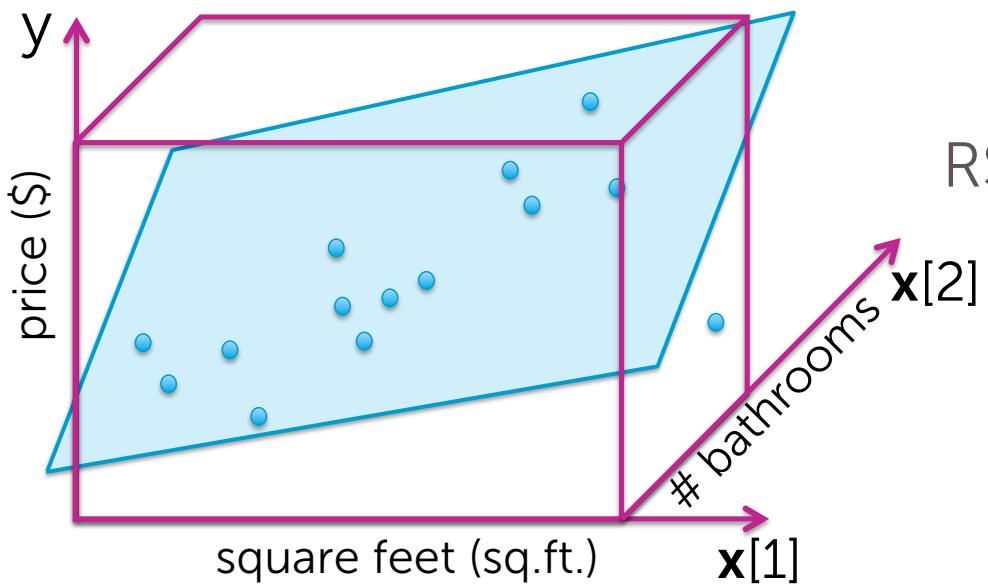


## **Step 2:** Compute the cost

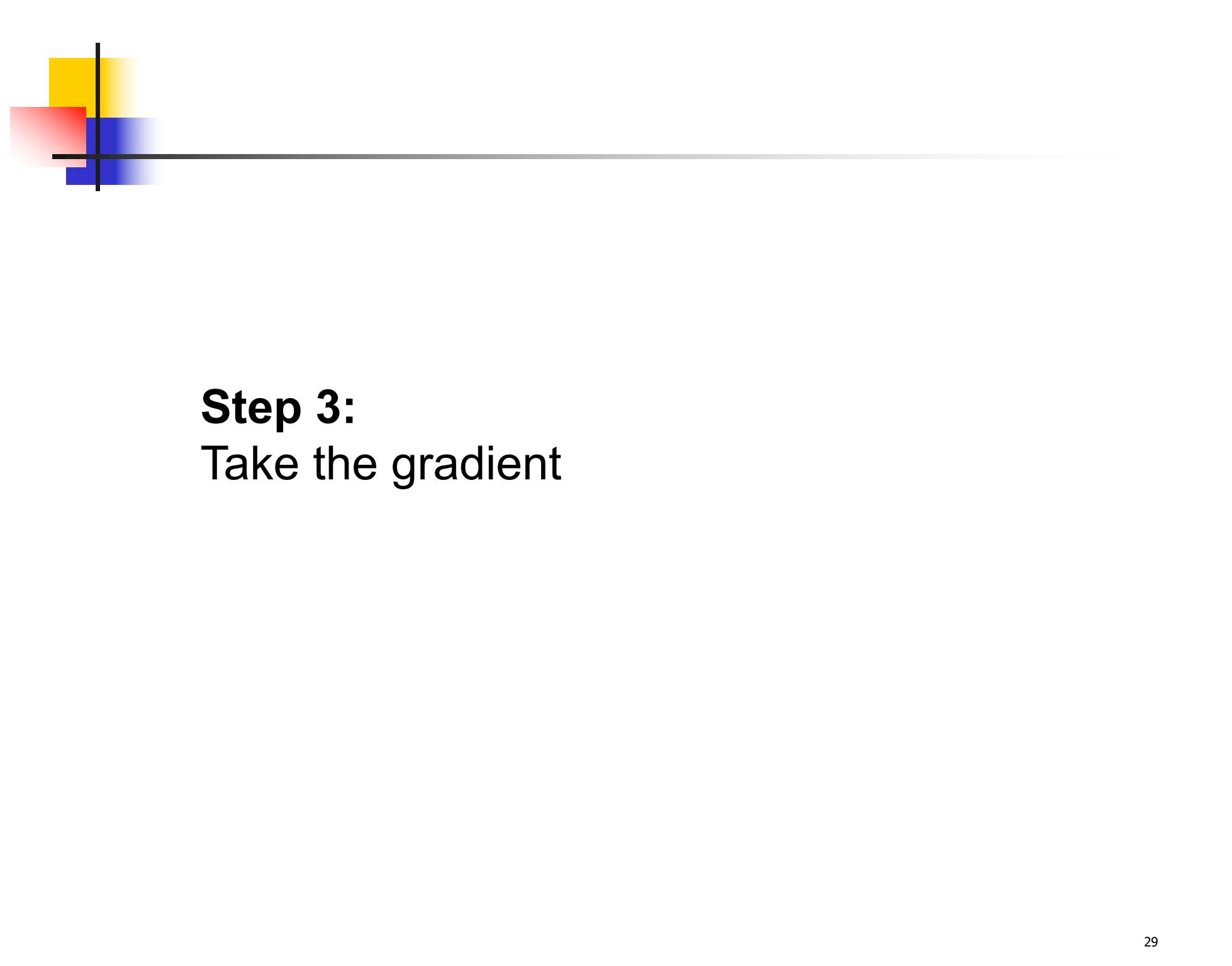
# “Cost” of using a given line



# RSS for multiple regression

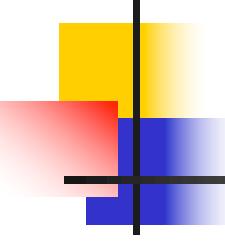


$$\begin{aligned} \text{RSS}(\mathbf{w}) &= \sum_{i=1}^N (y_i - \mathbf{H}^\top(\mathbf{x}_i) \mathbf{w})^2 \\ &= (\mathbf{y} - \mathbf{H}\mathbf{w})^\top(\mathbf{y} - \mathbf{H}\mathbf{w}) \end{aligned}$$



## **Step 3:**

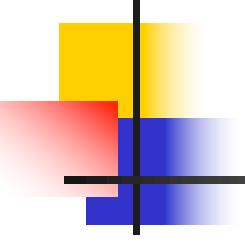
### Take the gradient



## Gradient of RSS

$$\begin{aligned}\nabla \text{RSS}(\mathbf{w}) &= \nabla [(\mathbf{y} - \mathbf{H}\mathbf{w})^\top (\mathbf{y} - \mathbf{H}\mathbf{w})] \\ &= -2\mathbf{H}^\top (\mathbf{y} - \mathbf{H}\mathbf{w})\end{aligned}$$

Why? By analogy to 1D case:

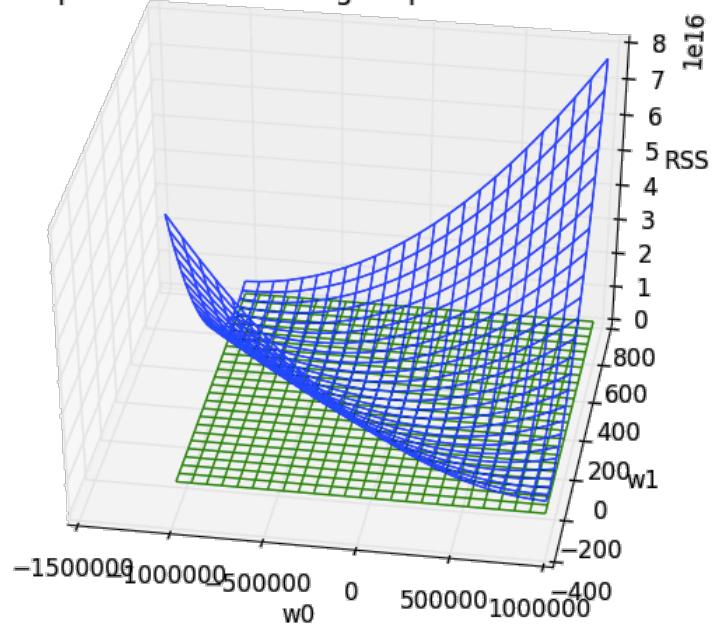


## **Step 4, Approach 1:**

Set the gradient = 0

# Closed-form solution

3D plot of RSS with tangent plane at minimum

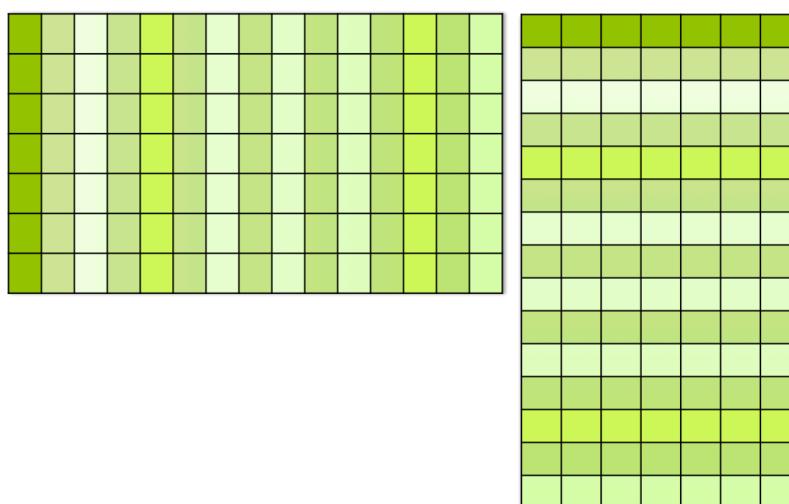


$$\nabla \text{RSS}(\mathbf{w}) = -2\mathbf{H}^T(\mathbf{y} - \mathbf{H}\mathbf{w}) = 0$$

Solve for  $\mathbf{w}$ :

# Closed-form solution

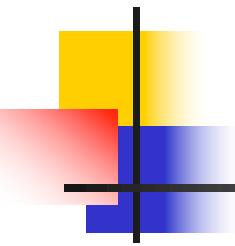
$$\hat{\mathbf{w}} = (\underbrace{\mathbf{H}^T \mathbf{H}}_{\downarrow})^{-1} \mathbf{H}^T \mathbf{y}$$



Invertible if:

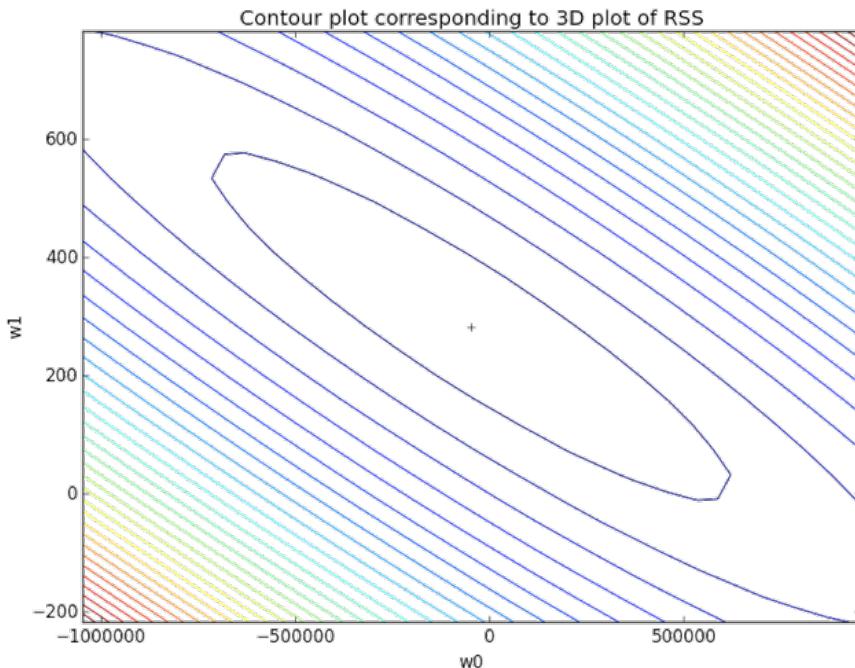
Complexity of inverse:

$$O(D^3)$$



## **Step 4, Approach 2:** Gradient descent

# Gradient descent

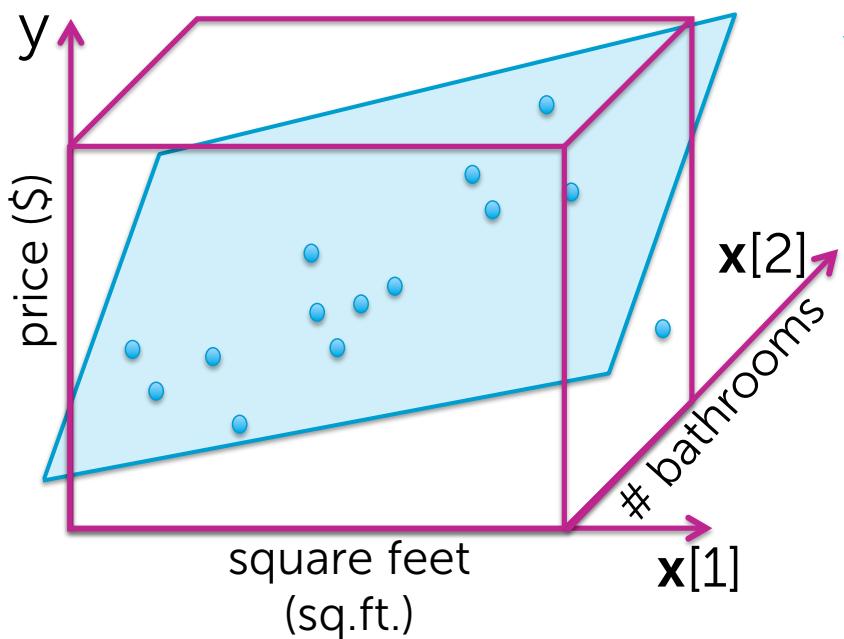


**while** not converged

$$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \eta \nabla \text{RSS}(\mathbf{w}^{(t)})$$

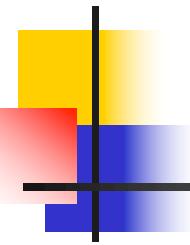
$\underbrace{-2\mathbf{H}^T(\mathbf{y} - \mathbf{H}\mathbf{w})}$

# Interpreting elementwise

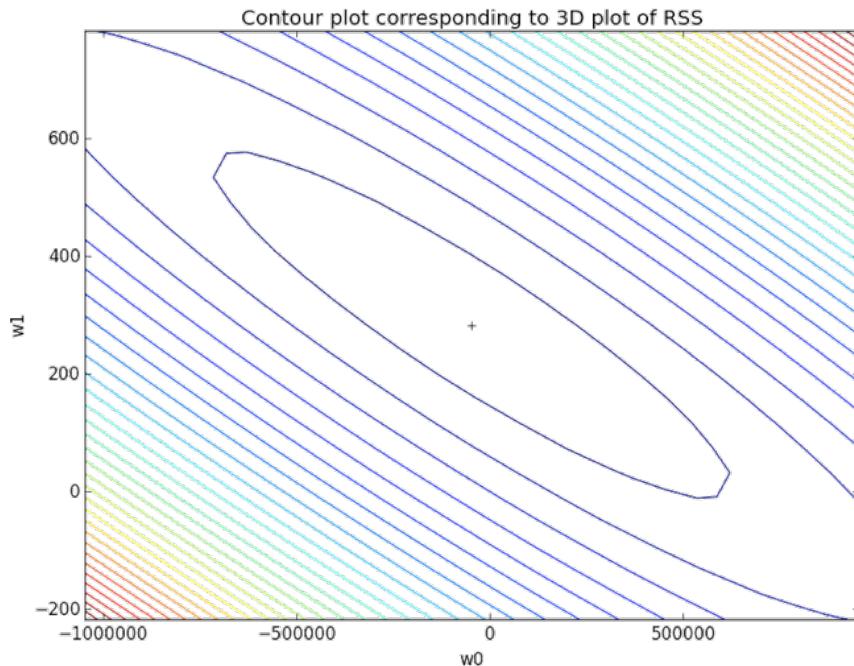


Update to  $j^{\text{th}}$  feature weight:

$$w_j^{(t+1)} \leftarrow w_j^{(t)} + 2\eta \sum_{i=1}^N h_j(\mathbf{x}_i)(y_i - \hat{y}_i(\mathbf{W}^{(t)}))$$



# Summary of gradient descent for multiple regression



```
init  $\mathbf{w}^{(1)} = 0$  (or randomly, or smartly),  $t=1$ 
while  $\|\nabla \text{RSS}(\mathbf{w}^{(t)})\| > \epsilon$ 
    for  $j=0, \dots, D$ 
         $\text{partial}[j] = -2 \sum_{i=1}^N h_j(\mathbf{x}_i)(y_i - \hat{y}_i(\mathbf{w}^{(t)}))$ 
         $\mathbf{w}_j^{(t+1)} \leftarrow \mathbf{w}_j^{(t)} - \eta \text{partial}[j]$ 
     $t \leftarrow t + 1$ 
```

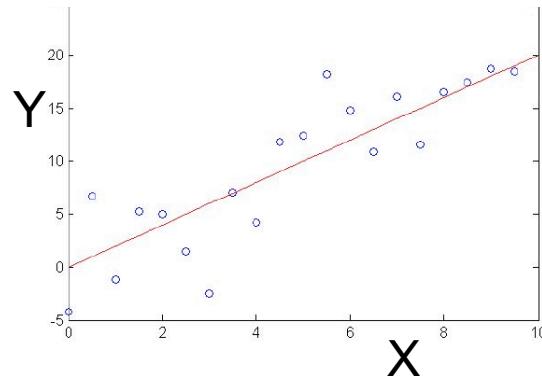


Why min RSS?

# Learn W using MLE

Gaussian method:

1. Choose parameterized form for  $P(Y|X; \theta)$



Assume Y is some deterministic  $f(X)$ , plus random noise

$$y = f(x) + \epsilon \quad \text{where } \epsilon \sim N(0, \sigma)$$

Therefore Y is a random variable that follows the distribution

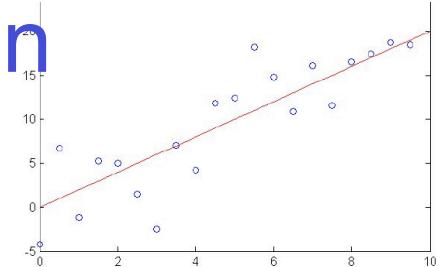
$$p(y|x) = N(f(x), \sigma)$$

and the expected value of y for any given x is  $f(x)$

# Learn $\mathbf{W}$ using MLE

## Consider Linear Regression

$$p(y|x) = N(f(x), \sigma)$$



E.g., assume  $f(x)$  is linear function of  $x$

$$p(y|x) = N(w_0 + w_1x, \sigma)$$

$$E[y|x] = w_0 + w_1x$$

Notation: to make our parameters explicit, let's write

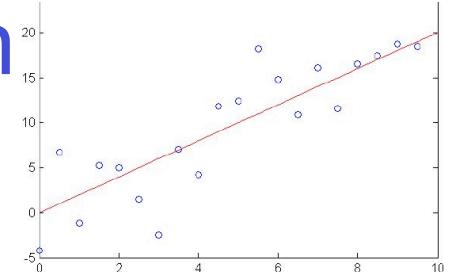
$$\mathbf{W} = \langle w_0, w_1 \rangle$$

$$p(y|x; \mathbf{W}) = N(w_0 + w_1x, \sigma)$$

# Learn $W$ using MLE

## Training Linear Regression

$$p(y|x; W) = N(w_0 + w_1x, \sigma)$$



How can we learn  $W$  from the training data?

Learn Maximum Conditional Likelihood Estimate!

$$W_{MCLE} = \arg \max_W \prod_l p(y^l|x^l, W)$$

$$W_{MCLE} = \arg \max_W \sum_l \ln p(y^l|x^l, W)$$

where

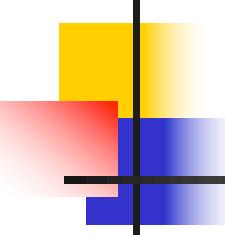
$$p(y|x; W) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}(\frac{y-f(x;W)}{\sigma})^2}$$

# Maximizing log-likelihood

Maximize wrt w:

$$\begin{aligned} \ln P(\mathcal{D} | \mathbf{w}, \sigma) &= \ln \left( \frac{1}{\sigma \sqrt{2\pi}} \right)^N \prod_{j=1}^N e^{-\frac{[t_j - \sum_i w_i h_i(x_j)]^2}{2\sigma^2}} \\ &\arg \max_w \ln \left( \frac{1}{\sigma \sqrt{2\pi}} \right)^N + \sum_{j=1}^N \frac{-[t_j - \sum_i w_i h_i(x_j)]^2}{2\sigma^2} \\ &= \arg \max_w \sum_{j=1}^N \frac{-[t_j - \sum_i w_i h_i(x_j)]^2}{2\sigma^2} \\ &= \arg \min_w \sum_{j=1}^N [t_j - \sum_i w_i h_i(x_j)]^2 \end{aligned}$$

For  $\sigma = 1$  (or some constant) for each input, MLE for Gaussians is equivalent to the least-squares objective for linear regression



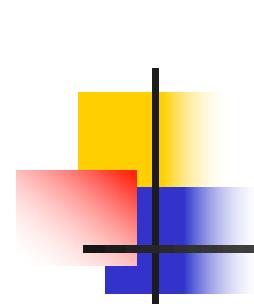
# Regression

---

Wish to learn  $f: X \rightarrow Y$ , where  $Y$  is real, given  $\{<x^1, y^1> \dots <x^n, y^n>\}$

Approach:

1. choose some parameterized form for  $P(Y|X; \theta)$   
( $\theta$  is the vector of parameters)
2. derive learning algorithm as MLE or MAP estimate for  $\theta$



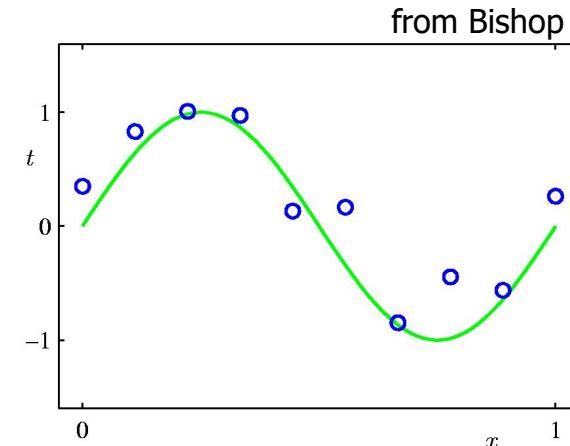
# Regularization in Linear Regression

- **Potential problem:** *Overfitting*
  - especially when data is very high dimensional and training data is sparse.
- One sign of overfitting: large parameter values!
- **Solution:** *Regularization*
  - penalized log likelihood function: penalizes large values of  $W$

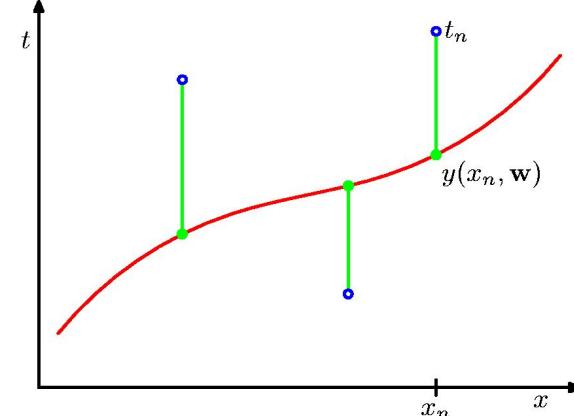
# A simple example: Fitting a polynomial

- The green curve is the true function (which is not a polynomial)
- The data points are uniform in  $x$  but have noise in  $y$ .

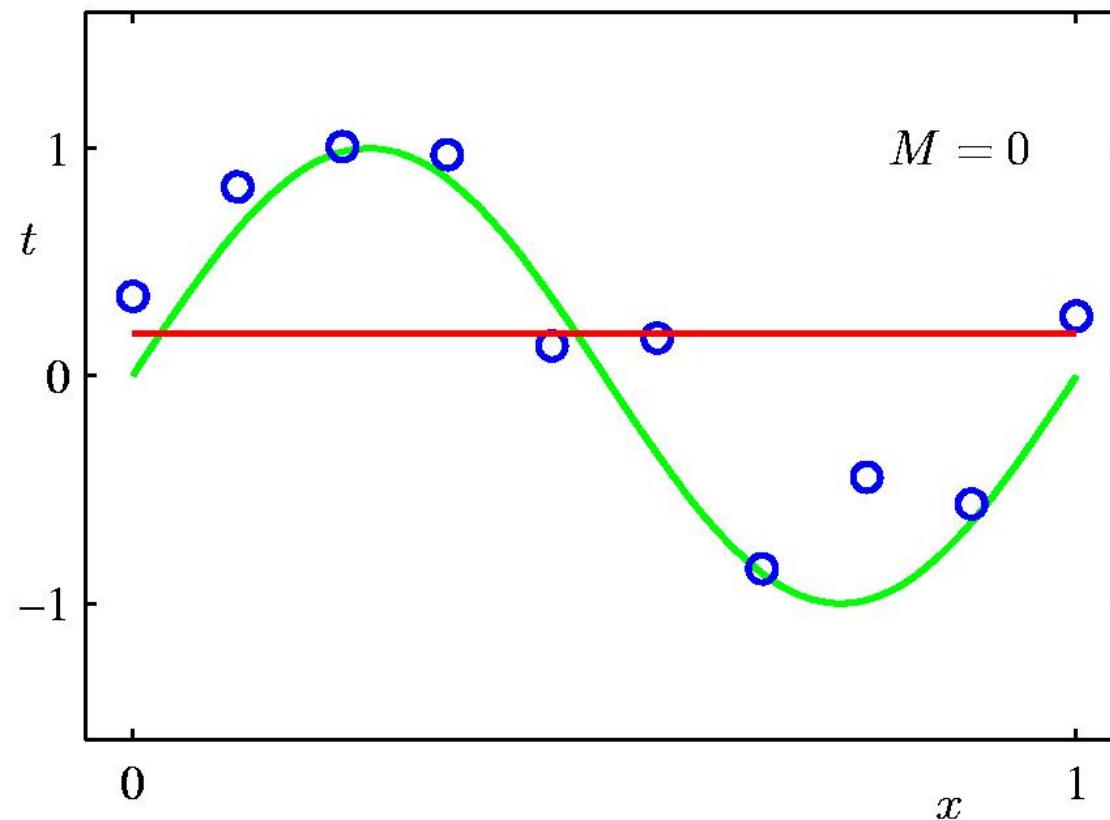
$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \dots + w_M x^M = \sum_{j=0}^M w_j x^j$$



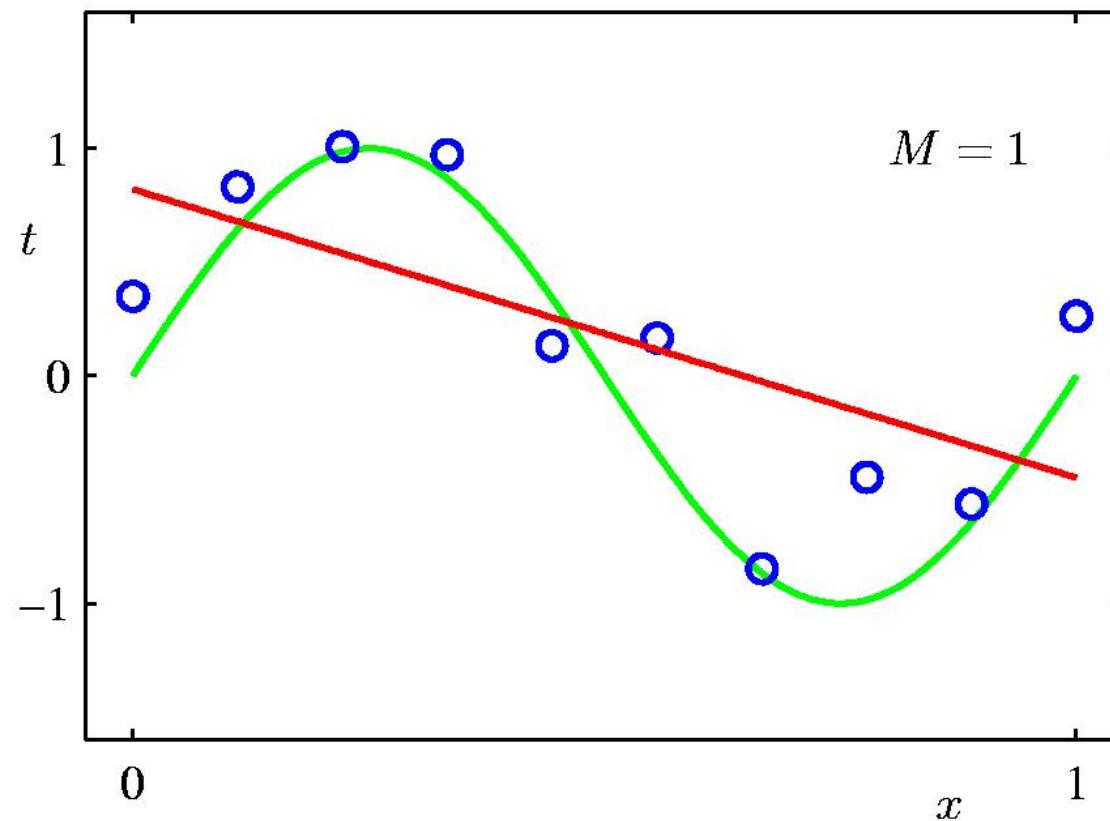
- We will use a loss function that measures the squared error in the prediction of  $y(x)$  from  $x$ . The loss for the red polynomial is the sum of the squared vertical errors.



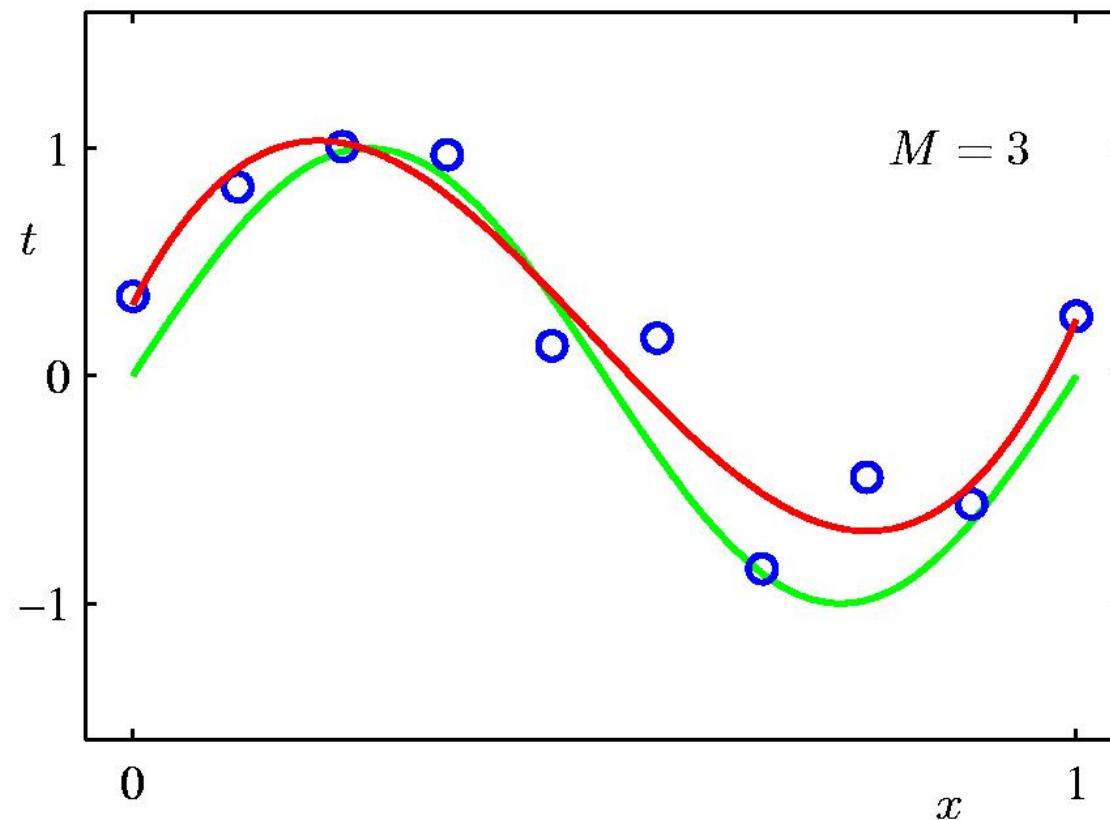
# 0<sup>th</sup> Order Polynomial



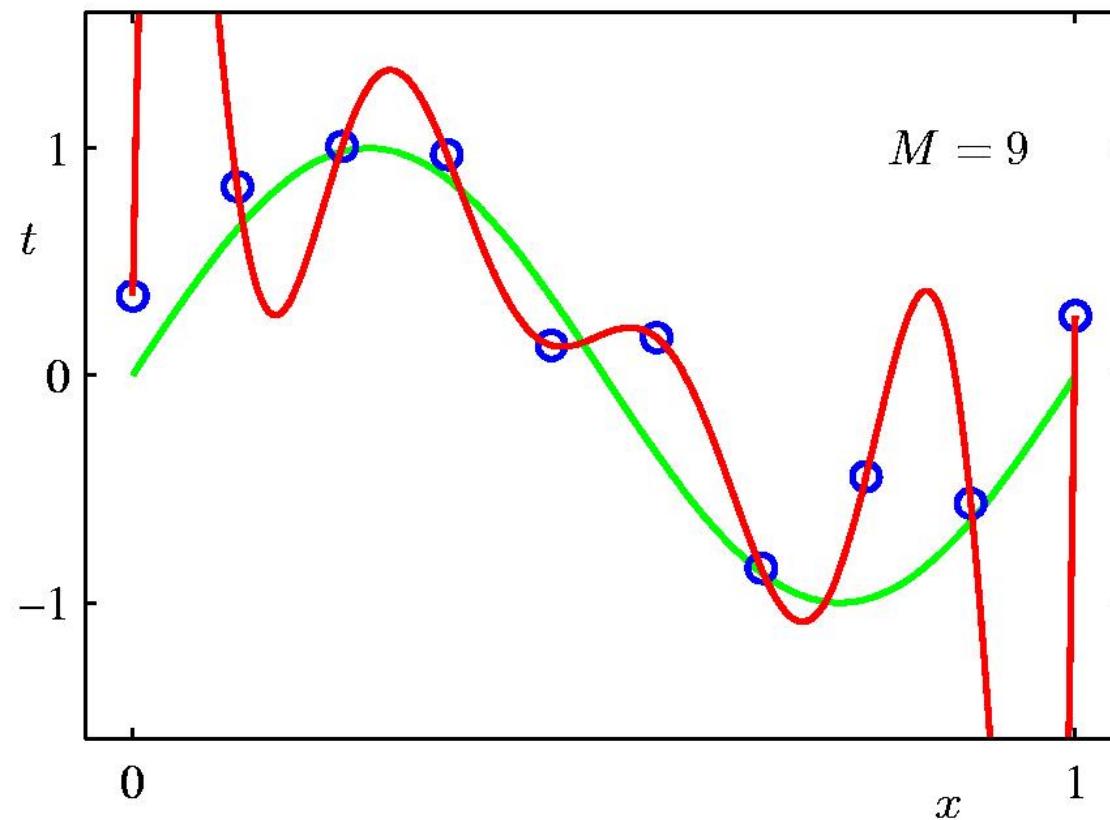
# 1<sup>st</sup> Order Polynomial



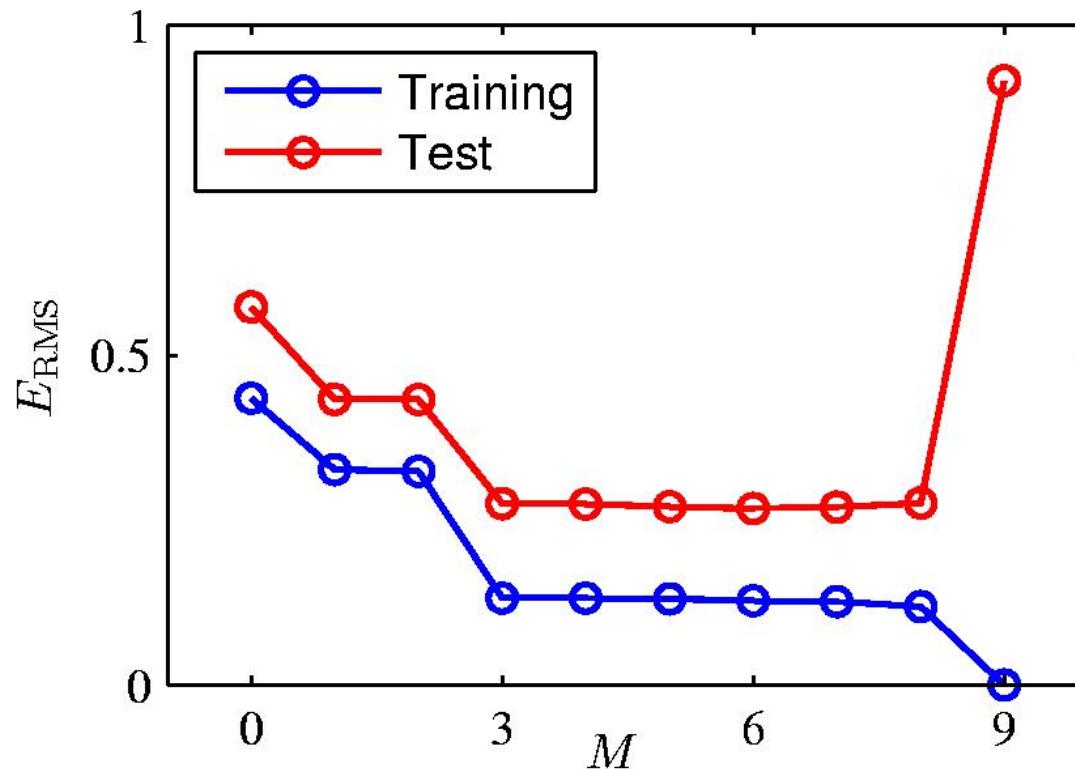
# 3<sup>rd</sup> Order Polynomial



# 9<sup>th</sup> Order Polynomial



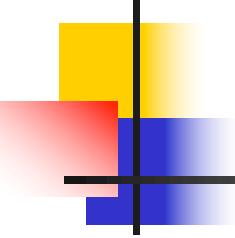
# Over-fitting



Root-Mean-Square (RMS) Error:  $E_{\text{RMS}} = \sqrt{2E(\mathbf{w}^*)/N}$

# Polynomial Coefficients

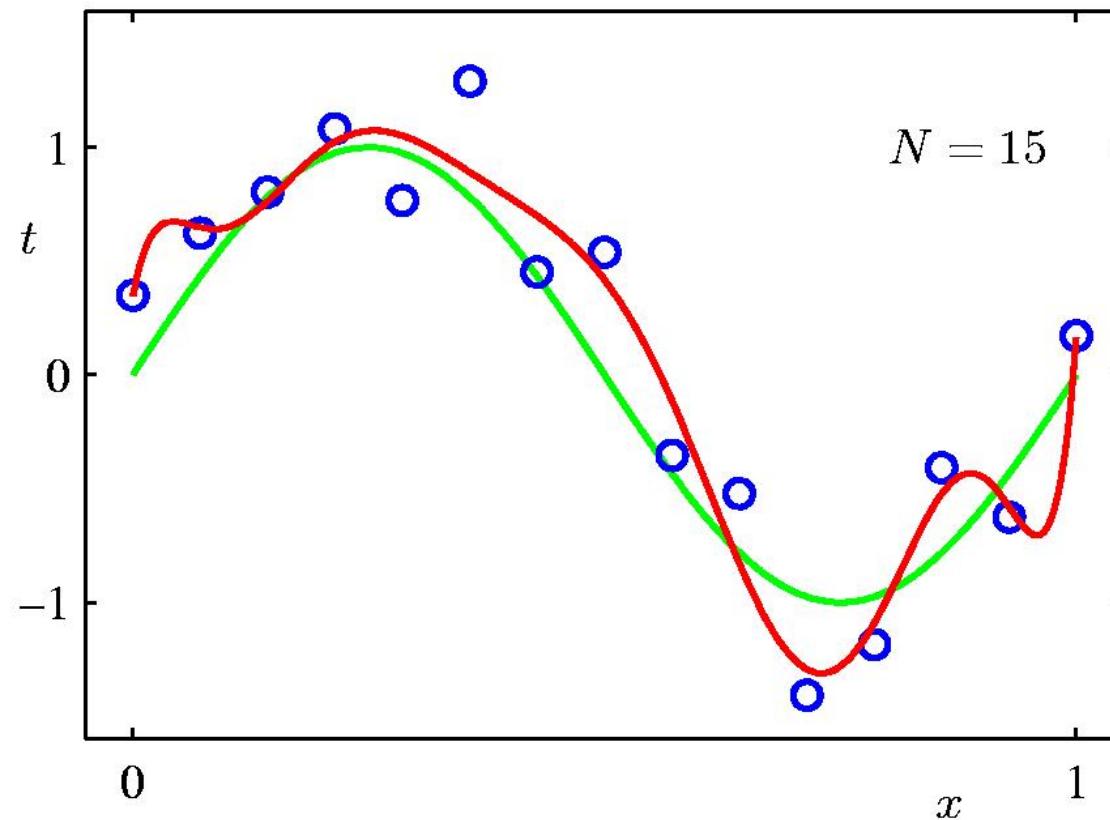
	$M = 0$	$M = 1$	$M = 3$	$M = 9$
$w_0^*$	0.19	0.82	0.31	0.35
$w_1^*$		-1.27	7.99	232.37
$w_2^*$			-25.43	-5321.83
$w_3^*$			17.37	48568.31
$w_4^*$				-231639.30
$w_5^*$				640042.26
$w_6^*$				-1061800.52
$w_7^*$				1042400.18
$w_8^*$				-557682.99
$w_9^*$				125201.43



# Data Set Size:

$$N = 15$$

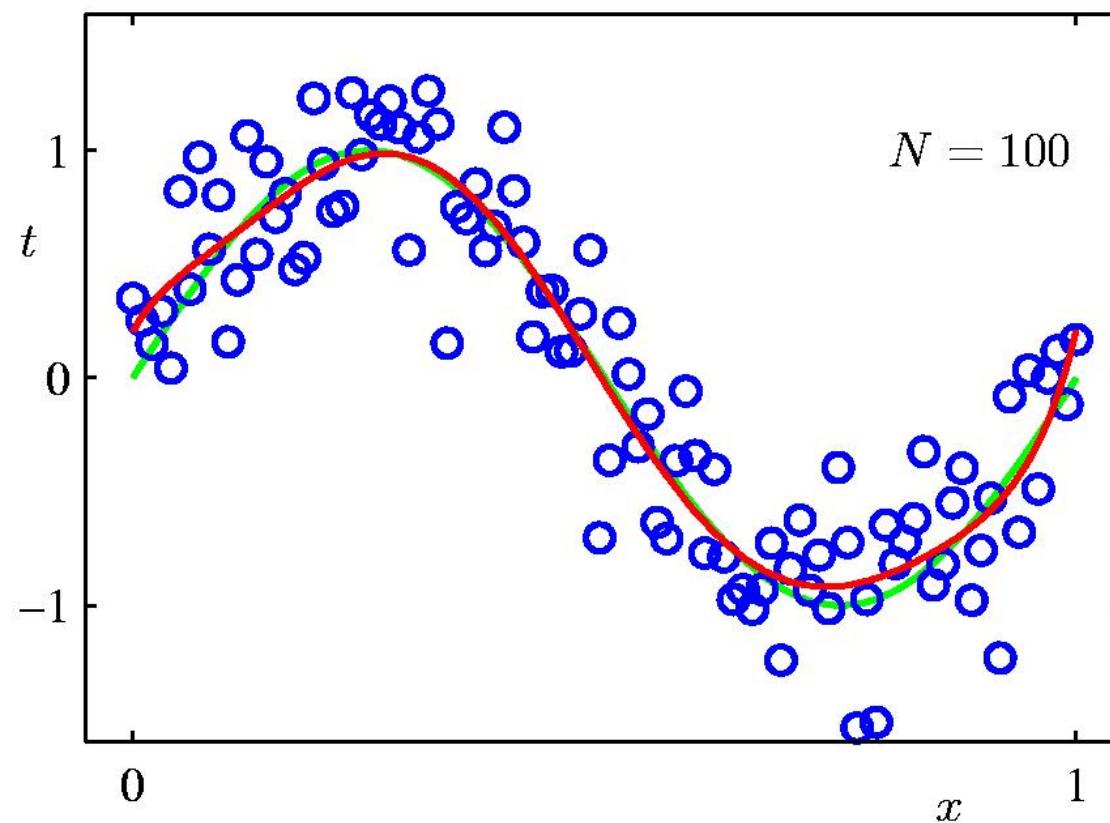
9<sup>th</sup> Order Polynomial

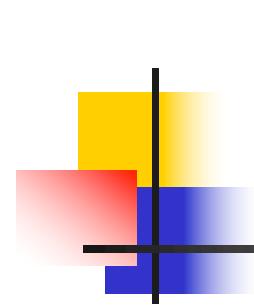


# Data Set Size:

9<sup>th</sup> Order Polynomial

$N = 100$





# Ridge Regularization

---

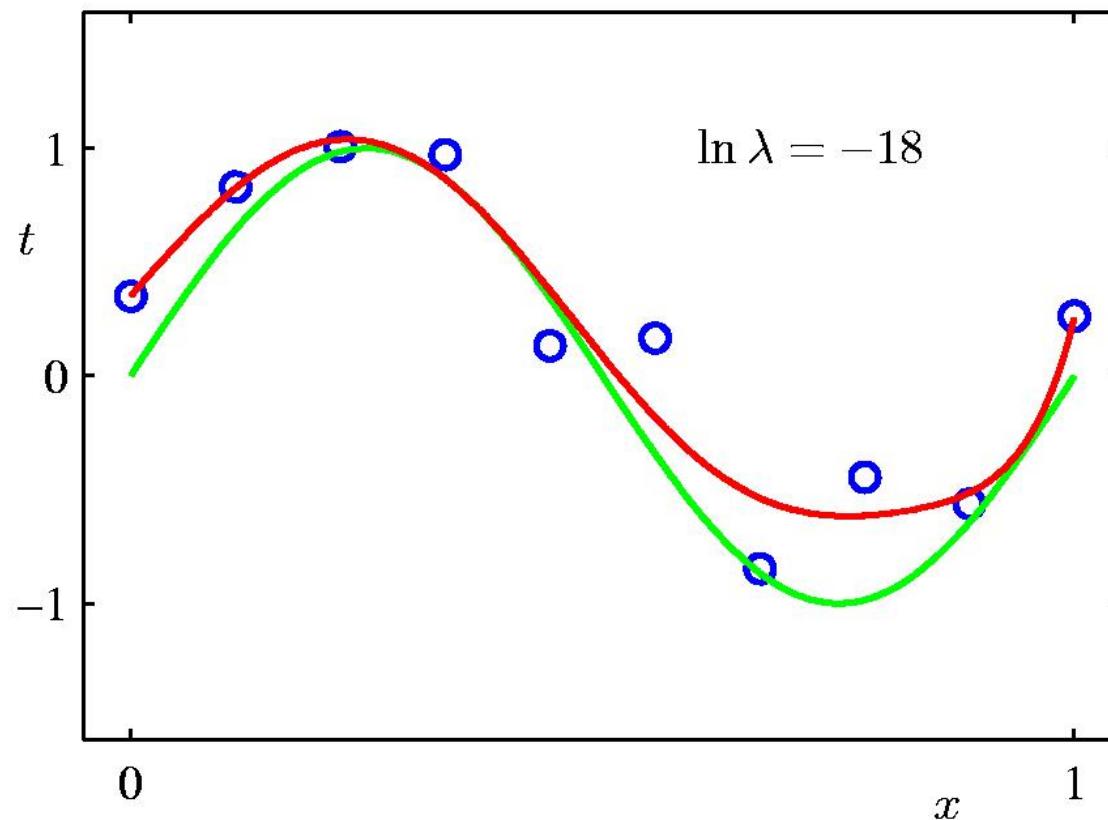
- Penalize large coefficient values

$$\tilde{E}(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

- Prefer low error but also add a square penalize for large weights
- $\lambda$  is hyperparameter that balances tradeoff

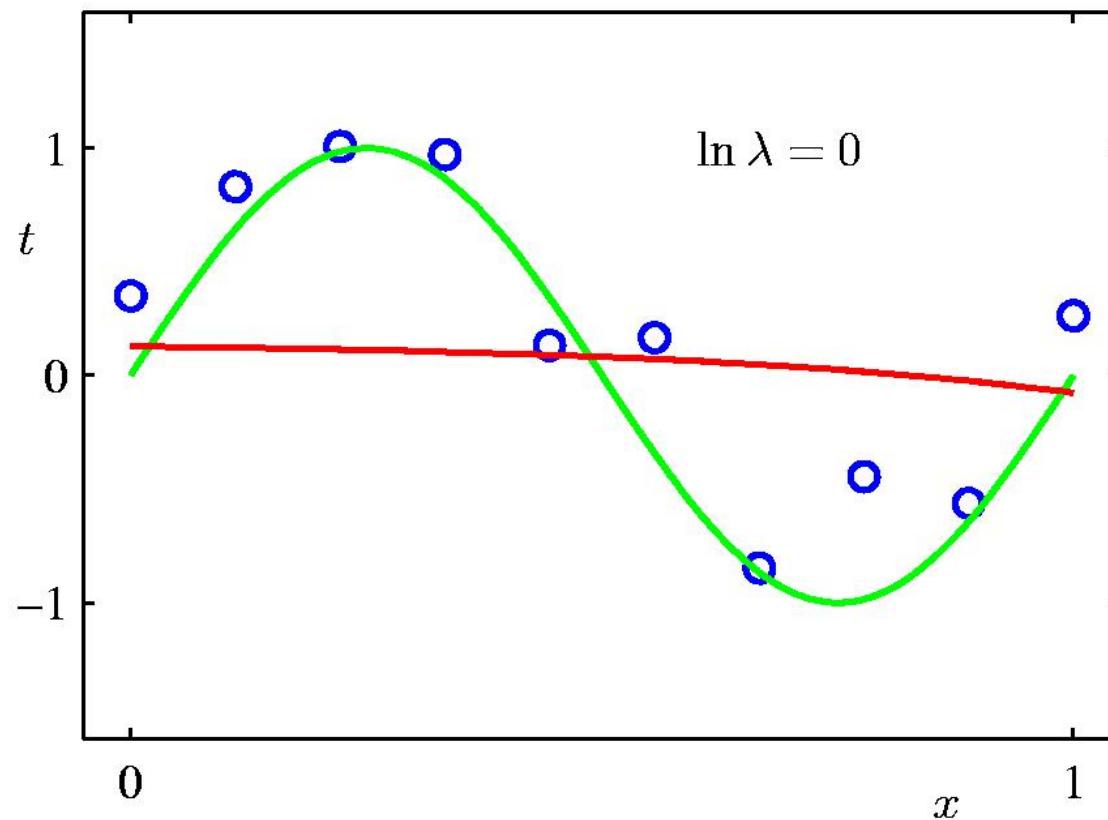
# Regularization:

$$\ln \lambda = -18$$

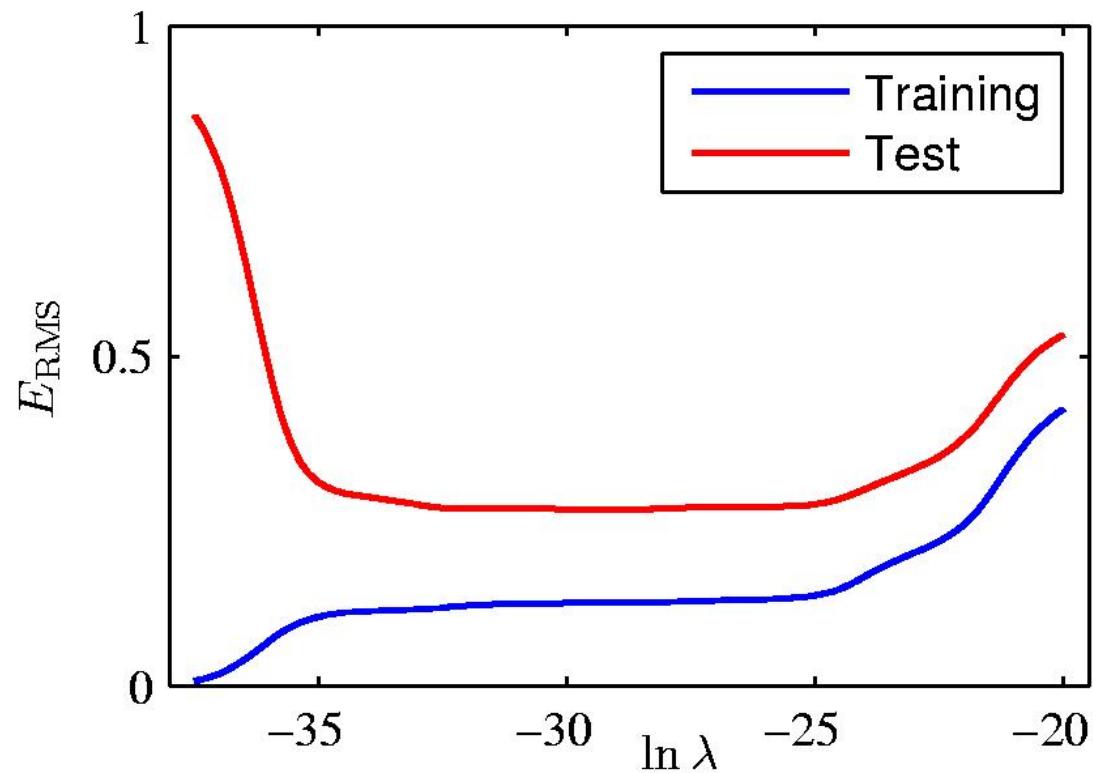


# Regularization:

$$\ln \lambda = 0$$



# Regularization: $E_{\text{RMS}}$ vs. $\ln \lambda$

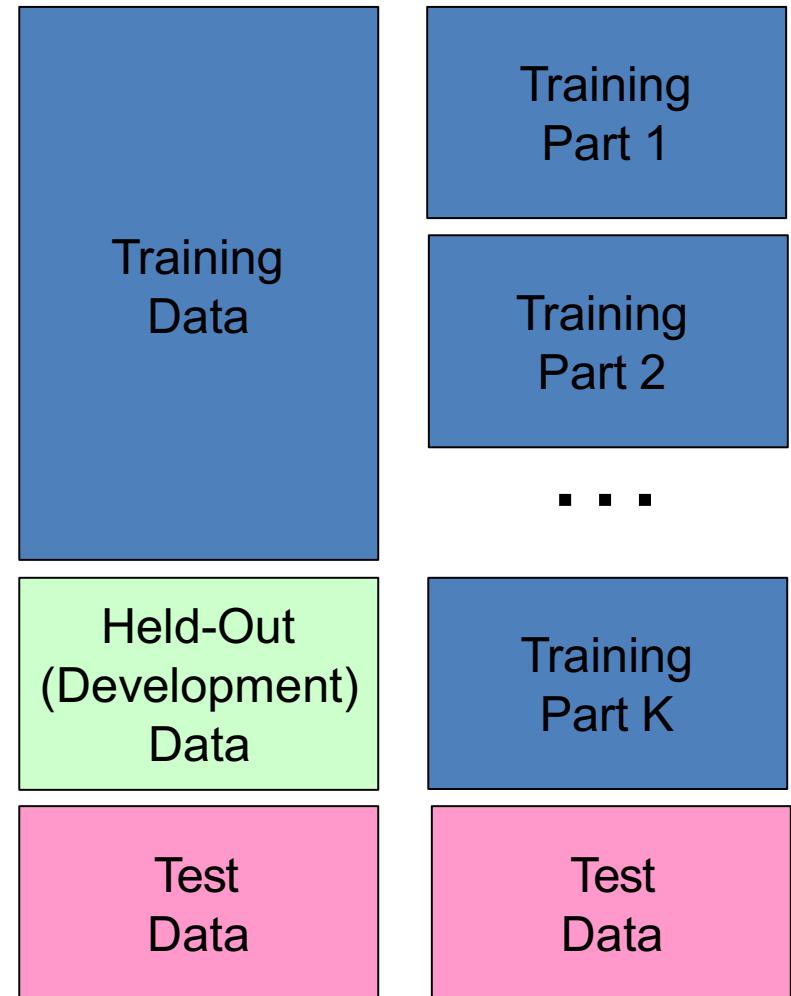


# Polynomial Coefficients

	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$w_0^*$	0.35	0.35	0.13
$w_1^*$	232.37	4.74	-0.05
$w_2^*$	-5321.83	-0.77	-0.06
$w_3^*$	48568.31	-31.97	-0.05
$w_4^*$	-231639.30	-3.89	-0.03
$w_5^*$	640042.26	55.28	-0.02
$w_6^*$	-1061800.52	41.32	-0.01
$w_7^*$	1042400.18	-45.95	-0.00
$w_8^*$	-557682.99	-91.53	0.00
$w_9^*$	125201.43	72.68	0.01

# 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



# Linear Regression: MAP Solution

Assume a Gaussian prior distribution over the weight vector  $\mathbf{w} = [w_1, \dots, w_D]$

$$p(\mathbf{w}) = \mathcal{N}(0, \lambda^{-1} \mathbf{I}_D) = \left( \frac{\lambda}{2\pi} \right)^{D/2} \exp\left(-\frac{\lambda}{2} \mathbf{w}^\top \mathbf{w}\right) = \left( \frac{\lambda}{2\pi} \right)^{D/2} \exp\left(-\frac{\lambda}{2} \|\mathbf{w}\|^2\right)$$

Log posterior probability:

$$\log P(\mathbf{w} \mid \mathcal{D}) = \log \frac{P(\mathbf{w})P(\mathcal{D} \mid \mathbf{w})}{P(\mathcal{D})} = \log P(\mathbf{w}) + \log P(\mathcal{D} \mid \mathbf{w}) - \log P(\mathcal{D})$$

Maximum-a-Posteriori Solution:  $\hat{\mathbf{w}}_{MAP} = \arg \max_{\mathbf{w}} \log P(\mathbf{w} \mid \mathcal{D})$

$$\begin{aligned} &= \arg \max_{\mathbf{w}} \{\log P(\mathbf{w}) + \log P(\mathcal{D} \mid \mathbf{w}) - \log P(\mathcal{D})\} \\ &= \arg \max_{\mathbf{w}} \{\log P(\mathbf{w}) + \log P(\mathcal{D} \mid \mathbf{w})\} \\ &= \arg \max_{\mathbf{w}} \left\{ -\frac{D}{2} \log(2\pi) - \frac{\lambda}{2} \mathbf{w}^\top \mathbf{w} + \sum_{n=1}^N \left\{ -\frac{1}{2} \log(2\pi\sigma^2) - \frac{(y_n - \mathbf{w}^\top \mathbf{x}_n)^2}{2\sigma^2} \right\} \right\} \\ &= \arg \min_{\mathbf{w}} \underbrace{\frac{\beta}{2} \sum_{n=1}^N (y_n - \mathbf{w}^\top \mathbf{x}_n)^2}_{\text{fit to the training data}} + \underbrace{\frac{\lambda}{2} \mathbf{w}^\top \mathbf{w}}_{\text{keep } \mathbf{w} \text{ "simple"}} \end{aligned}$$

For  $\sigma = 1$  (or some constant) for each input, it's equivalent to the regularized least-squares objective

# Linear Regression: MLE vs. MAP

MLE solution:

$$\hat{\mathbf{w}}_{MLE} = \arg \min_{\mathbf{w}} \frac{1}{2\sigma^2} \sum_{n=1}^N (y_n - \mathbf{w}^\top \mathbf{x}_n)^2$$

MAP solution:

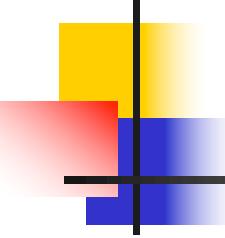
$$\hat{\mathbf{w}}_{MAP} = \arg \min_{\mathbf{w}} \frac{1}{2\sigma^2} \sum_{n=1}^N (y_n - \mathbf{w}^\top \mathbf{x}_n)^2 + \frac{\lambda}{2} \mathbf{w}^\top \mathbf{w}$$

## Take-home messages:

- MLE estimation of a parameter leads to unregularized solutions
- MAP estimation of a parameter leads to regularized solutions
- The prior distribution acts as a regularizer in MAP estimation

Note: For MAP, different prior distributions lead to different regularizers

- Gaussian prior on  $\mathbf{w}$  regularizes the  $\ell_2$  norm of  $\mathbf{w}$
- Laplace prior  $\exp(-C||\mathbf{w}||_1)$  on  $\mathbf{w}$  regularizes the  $\ell_1$  norm of  $\mathbf{w}$



# Regression- What you should know

Under general assumption  $p(y|x; W) = N(f(x; W), \sigma)$

1. MLE corresponds to minimizing sum of squared prediction errors
2. MAP estimate minimizes SSE plus sum of squared weights
3. Again, learning is an optimization problem once we choose our objective function
  - maximize data likelihood
  - maximize posterior probability of  $W$
4. Again, we can use gradient descent as a general learning algorithm
  - as long as our objective function is differentiable wrt  $W$
  - though we might learn local optima ins