CS 1675: Intro to Machine Learning

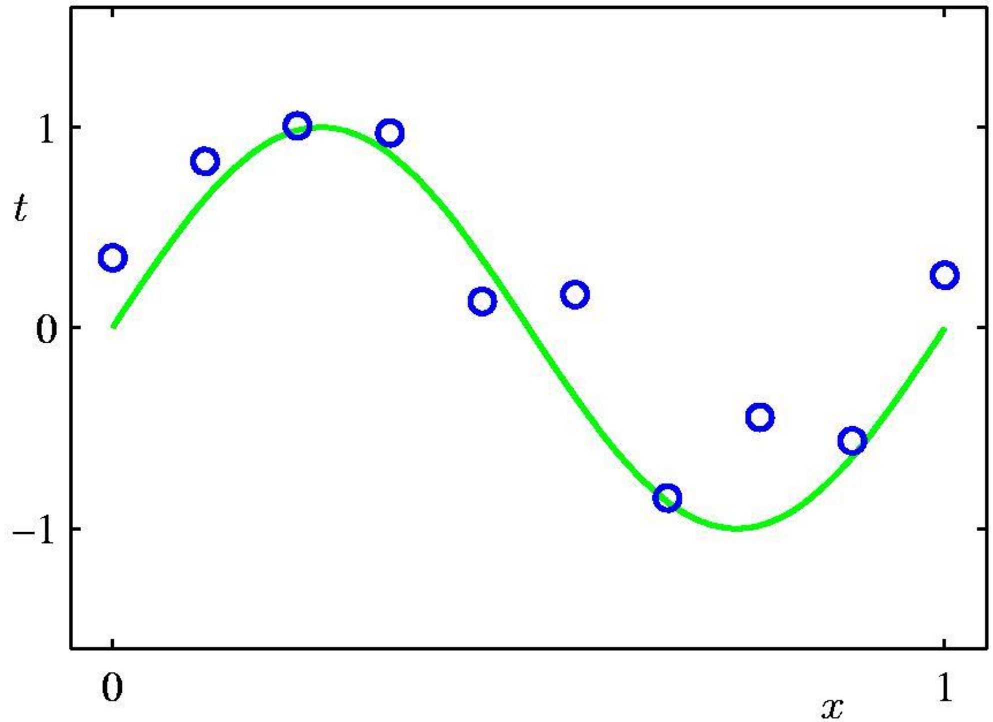
Regression and Overfitting

Prof. Adriana Kovashka
University of Pittsburgh
September 18, 2018

Regression

- Given some features **x**, predict a continuous variable y, e.g.
  - Predict the cost of a house based on square footage, age built, neighborhood, photos
  - Predict temperature tomorrow based on temperature today
  - Predict how far a car will go based on car's speed and environment
- These all involve fitting a function (curve) given training pairs ($\mathbf{x}_i$, y_i)

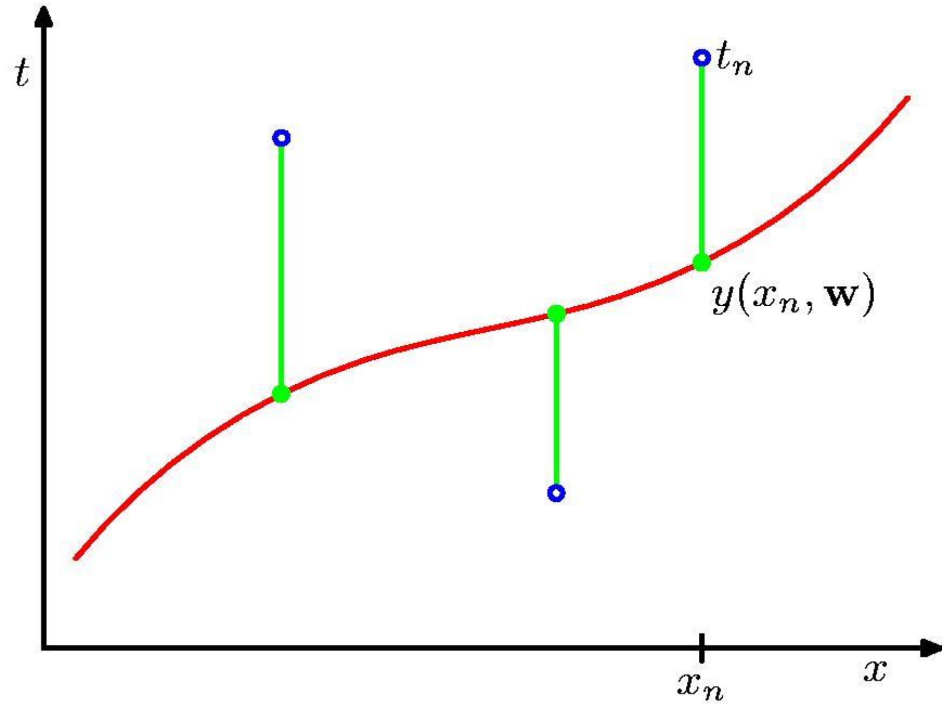
Polynomial Curve Fitting



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

Slide credit: Chris Bishop

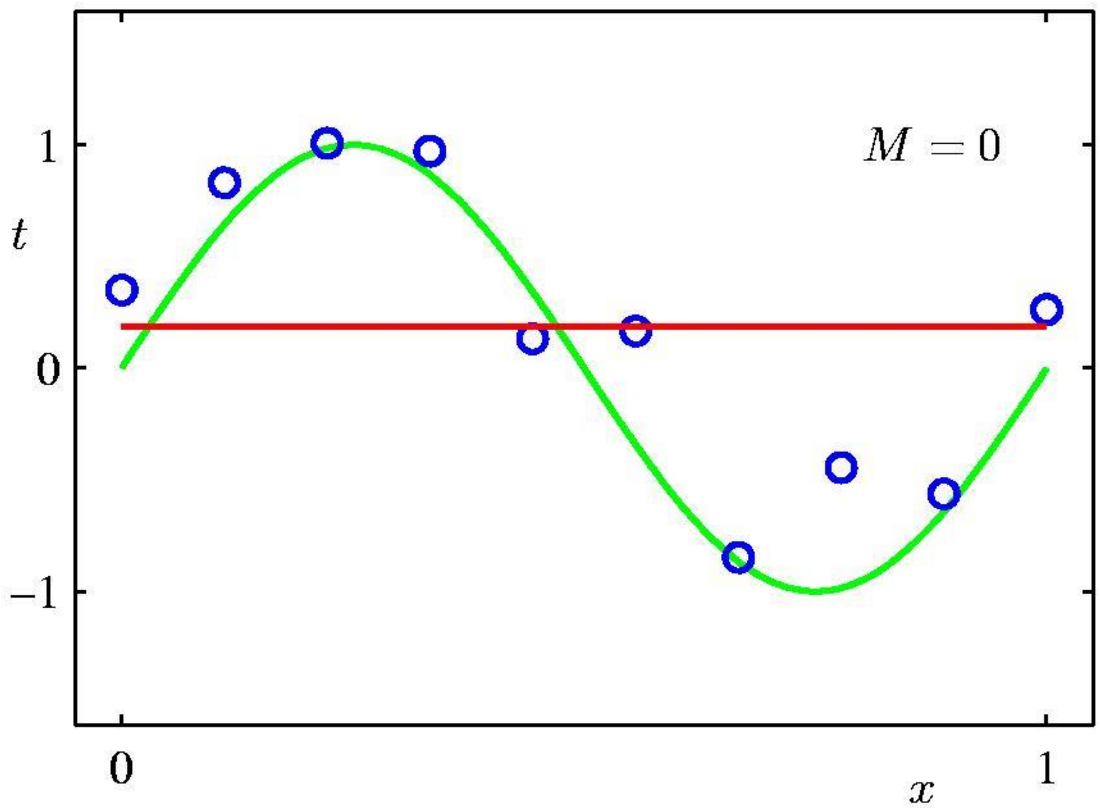
Sum-of-Squares Error Function



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

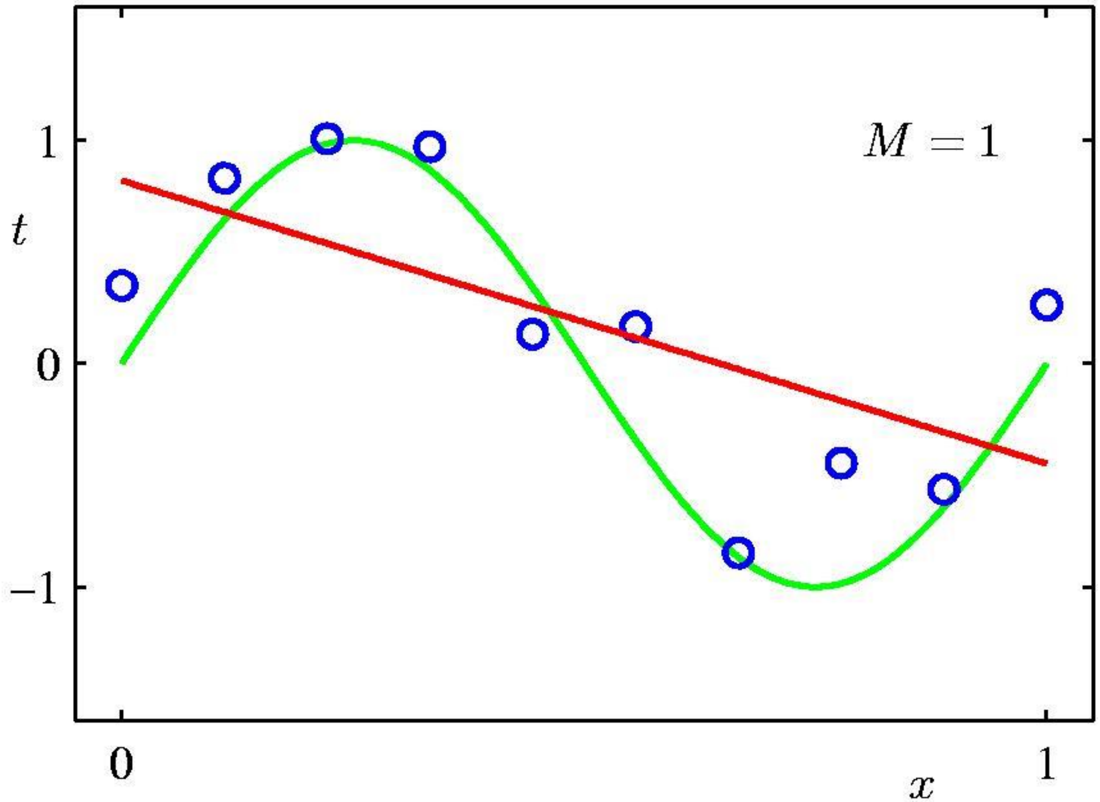
Slide credit: Chris Bishop

0^{th} Order Polynomial



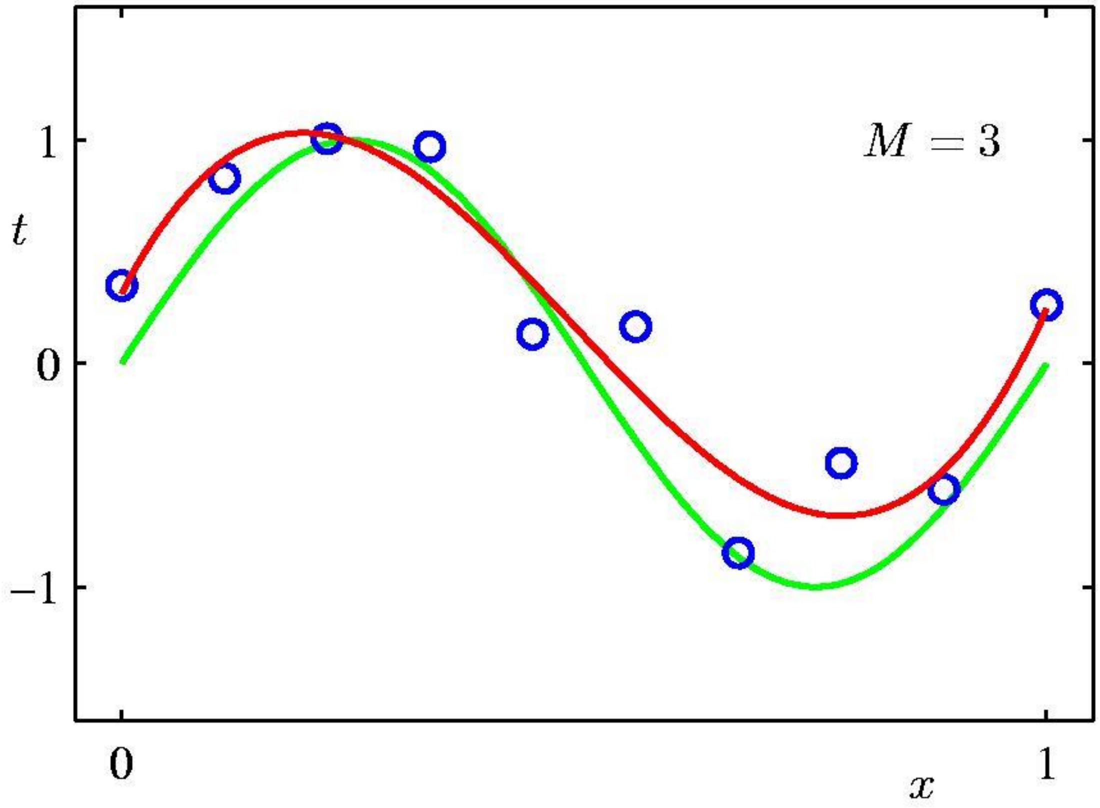
Slide credit: Chris Bishop

1^{st} Order Polynomial



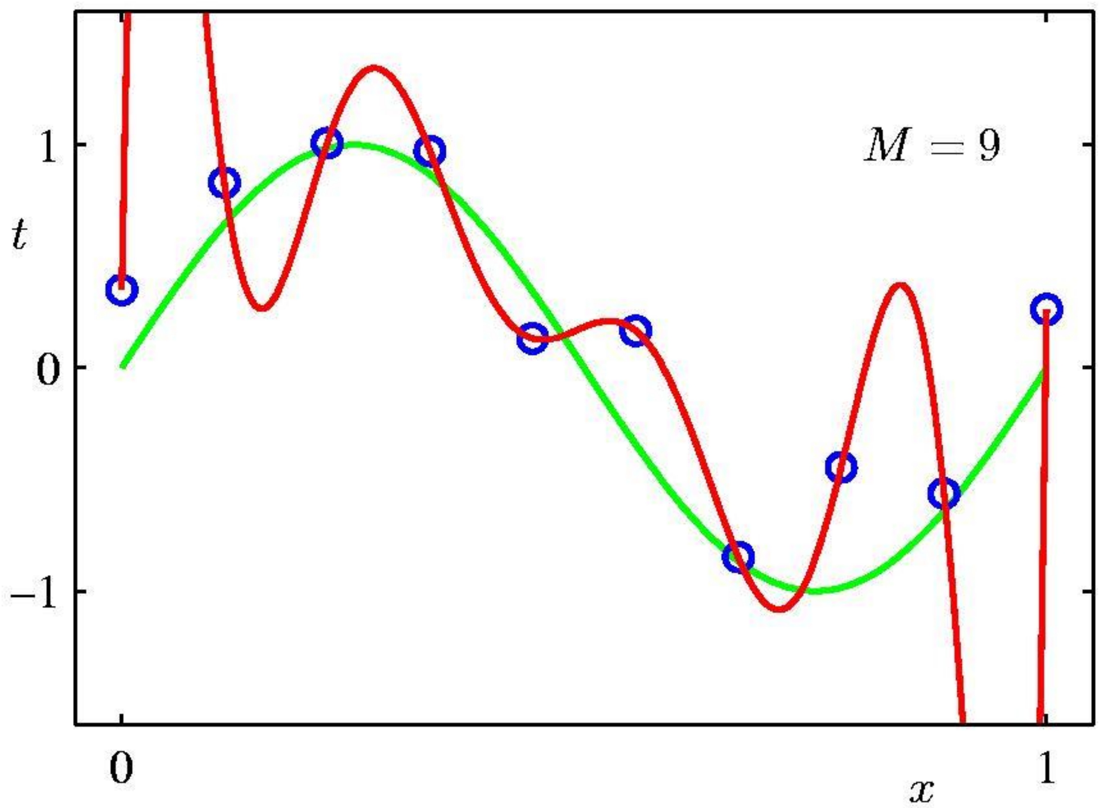
Slide credit: Chris Bishop

3rd Order Polynomial



Slide credit: Chris Bishop

9th Order Polynomial

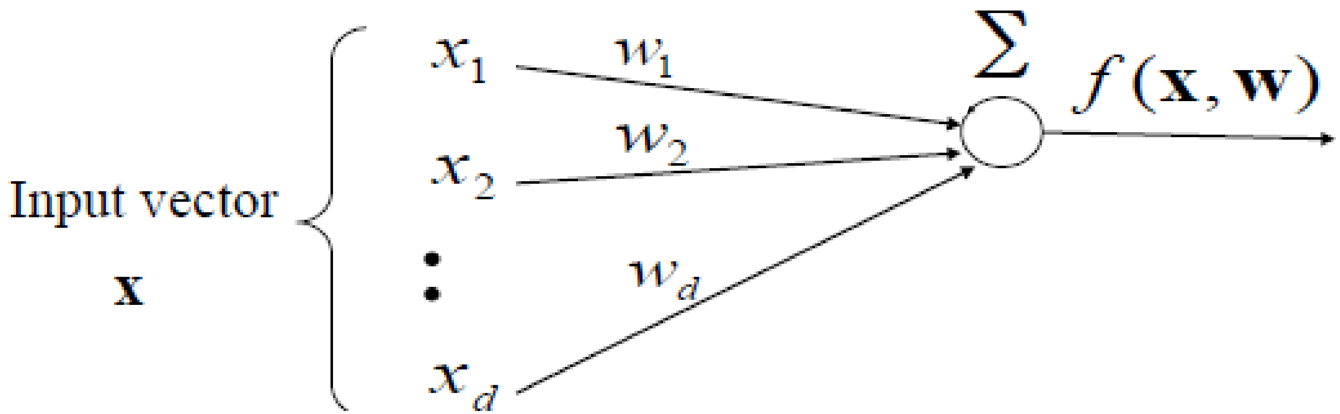


Slide credit: Chris Bishop

Plan for Today

- Linear regression
- Closed-form solution via least squares
- Solution via gradient descent
- Dealing with outliers
- Generalization: bias and variance
- Regularization

Linear Models



- Represent label *y* as a linear combination of the features ***x*** (i.e. weighted average of the features)
- Sometimes want to add a *bias term:* can add 1 as x_0 such that $x = (1, x_1, \ldots, x_d)$

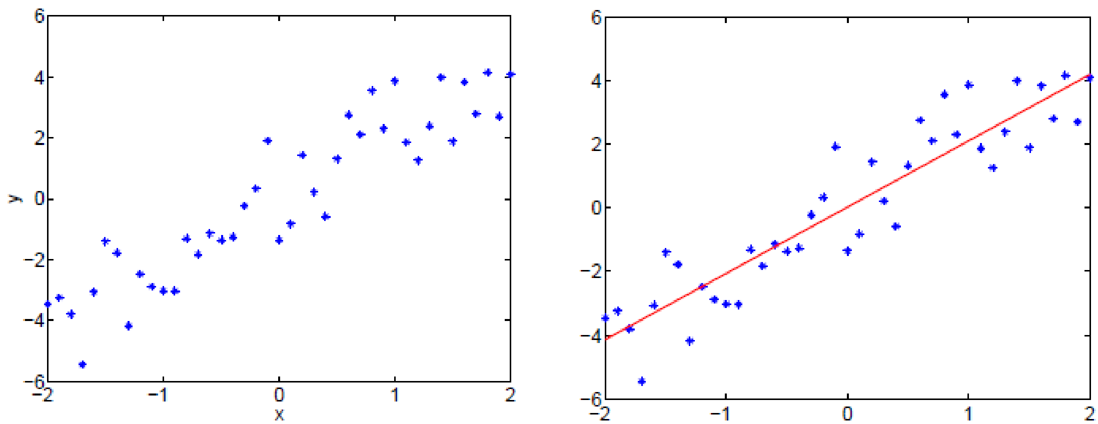
Figure from Milos Hauskrecht

Regression

$$f(x, w): \mathbf{x} \rightarrow y$$

- At training time: Use given $\{(\mathbf{x}_1,y_1), \ldots, (\mathbf{x}_N,y_N)\}$, to estimate mapping function f
  - Objective: minimize $(y_i - f(w, \mathbf{x}_i))^2$, for all i = 1, …, N
  - $\mathbf{x}_i$ are the input features (*d*-dimensional)
  - y_i is the target *continuous* output label (given by human/oracle)
- At test time: Use f to make prediction for a new sample $\mathbf{x}_{test}$ (not in the training set)

Linear regression



- We begin by considering linear regression (easy to extend to more complex predictions later on)

$$f : \mathcal{R} \rightarrow \mathcal{R} \qquad f(x; \mathbf{w}) = w_0 + w_1 x \qquad = b + mx$$

1-d example

- Fit line to points
- Use parameters of line to predict the *y*-coordinate of a new data point x_{new}

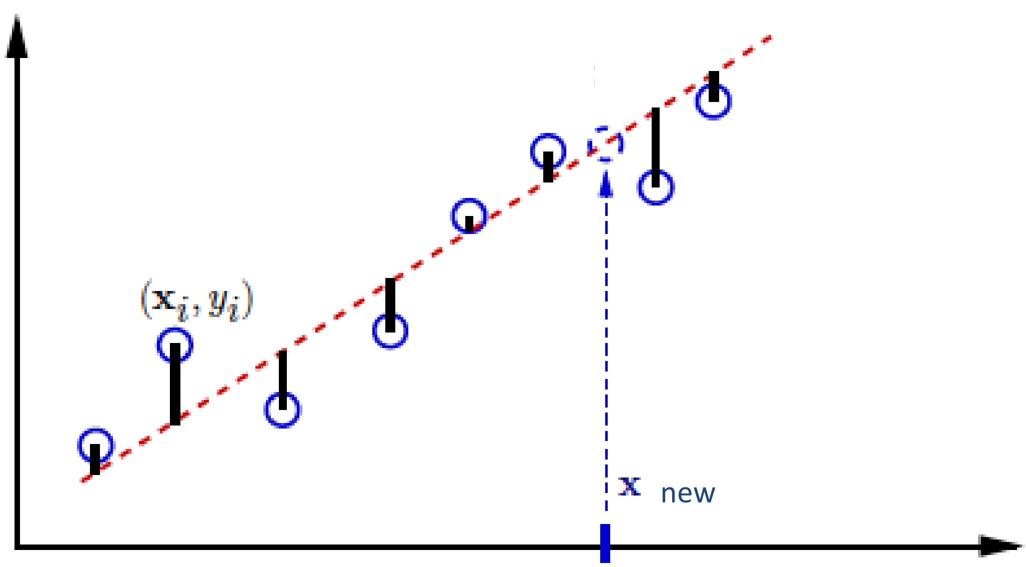


Figure from Greg Shakhnarovich

2-d example

- Find parameters of plane

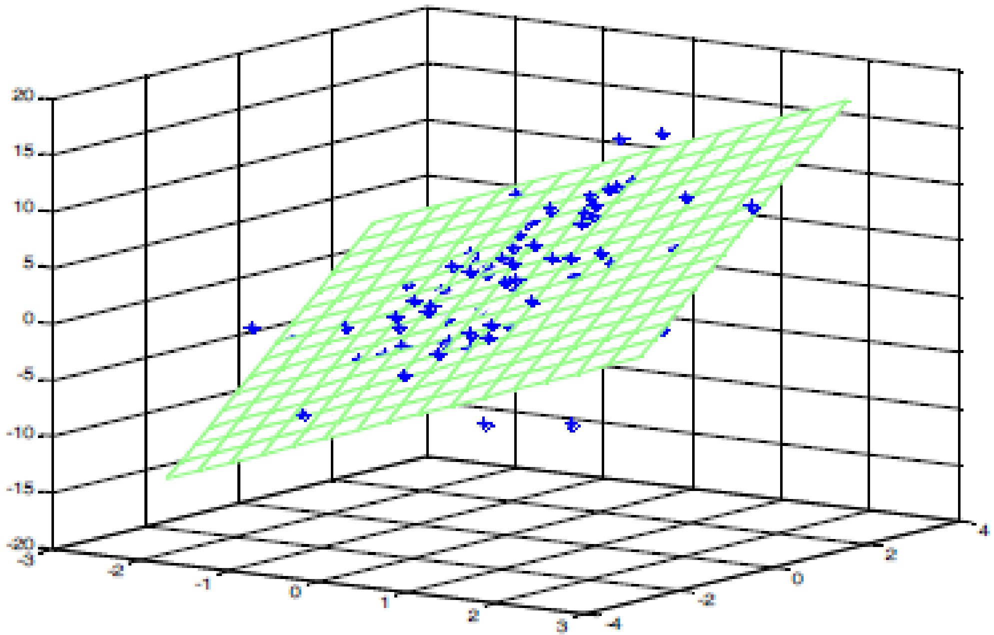


Figure from Milos Hauskrecht

Least squares solution

(Derivation on board)

$$\frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}) = -\frac{2}{N} \left(\mathbf{X}^T \mathbf{y} - \mathbf{X}^T \mathbf{X} \mathbf{w}\right) = 0$$

$$\mathbf{X}^T \mathbf{y} = \mathbf{X}^T \mathbf{X} \mathbf{w} \Rightarrow \mathbf{w}^* = \left(\mathbf{X}^T \mathbf{X}\right)^{-1} \mathbf{X}^T \mathbf{y}$$

$$\hat{y} = \mathbf{w}^{*T} \begin{bmatrix} 1 \\ \mathbf{x}_{\text{test}} \end{bmatrix} = \mathbf{y}^T \mathbf{X}^{\dagger T} \begin{bmatrix} 1 \\ \mathbf{x}_{\text{test}} \end{bmatrix}$$

Slide credit: Greg Shakhnarovich

Challenges

- Computing the pseudoinverse might be slow for large matrices
  - *Cubic* in number of features D
  - Linear in number of samples N
- We might want to adjust solution as new examples come in, without recomputing the pseudoinverse for each new sample that comes in
- Another solution: Gradient descent
  - Cost: linear in both D and N
  - If D > 10,000, use gradient descent

Gradient descent

Want to minimize a loss function

- Loss function: squared error, true/predicted y

- How to minimize?
  - Model is a function of the weights ***w***, hence loss is a function of the weights
  - Find derivative of loss with respect to weights, set to 0 (minimum will be an extremum and if function is convex, there will be only one)

Derivative of the loss function

In 1-dimension, the derivative of a function:

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

In multiple dimensions, the derivative is called a **gradient,** which is the vector of partial derivatives with respect to each dimension of the input.

current W:

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]

loss 1.25347

gradient dW:

[?,
?,
?,
?,
?,
?,
?,
?,
?,…]

current W:

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]
loss 1.25347

W + h (first dim)**:**

[0.34 + **0.0001**,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]
loss 1.25322

gradient dW:

[?,
?,
?,
?,
?,
?,
?,
?,
?,…]

current W:	**W + h** (first dim)**:**	**gradient dW:**
[0.34,	[0.34 + **0.0001**,	[**-2.5**,
-1.11,	-1.11,	?,
0.78,	0.78,	?,
0.12,	0.12,	(1.25322 - 1.25347)/0.0001 = -2.5
0.55,	0.55,	$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$
2.81,	2.81,	
-3.1,	-3.1,	
-1.5,	-1.5,	?,
0.33,…]	0.33,…]	?,…]
loss 1.25347	**loss 1.25322**	

current W:	**W + h** (second dim)**:**	**gradient dW:**
[0.34,	[0.34,	[-2.5,
-1.11,	-1.11 + **0.0001**,	?,
0.78,	0.78,	?,
0.12,	0.12,	?,
0.55,	0.55,	?,
2.81,	2.81,	?,
-3.1,	-3.1,	?,
-1.5,	-1.5,	?,
0.33,…]	0.33,…]	?,…]
loss 1.25347	**loss 1.25353**	

current W:

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]
loss 1.25347

W + h (second dim)**:**

[0.34,
-1.11 + **0.0001**,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]
loss 1.25353

gradient dW:

[-2.5,
0.6,
?,
?,

(1.25353 - 1.25347)/0.0001
= 0.6

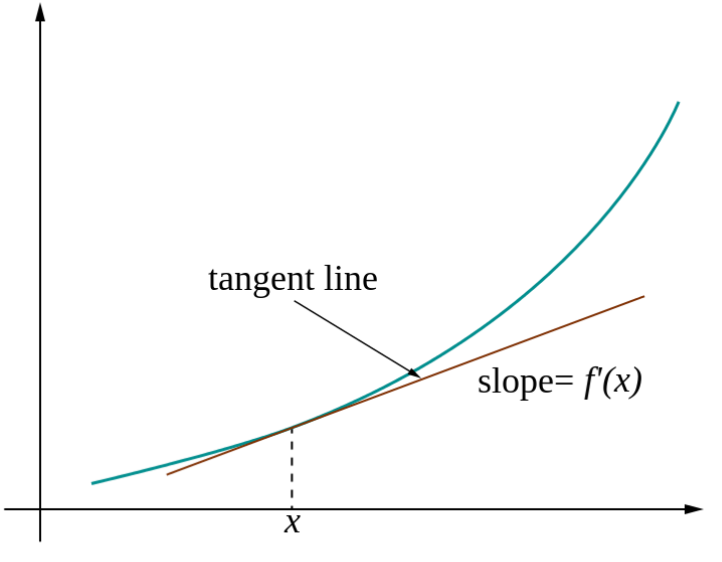
$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

?,…]

current W:	**W + h** (third dim)**:**	**gradient dW:**
[0.34,	[0.34,	[-2.5,
-1.11,	-1.11,	0.6,
0.78,	0.78 + **0.0001**,	?,
0.12,	0.12,	?,
0.55,	0.55,	?,
2.81,	2.81,	?,
-3.1,	-3.1,	?,
-1.5,	-1.5,	?,
0.33,…]	0.33,…]	?,…]
loss 1.25347	**loss 1.25347**	

Loss gradients

- Denoted as (diff notations): $\frac{\partial E_n}{\partial w_{ji}^{(1)}}$ $\nabla_W L$
- i.e. how loss changes as function of weights
- Change the weights in such a way that makes the loss decrease as fast as possible



Gradient descent

- Update weights: move opposite to gradient

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w}^{(\tau)})$$

Time

Learning rate

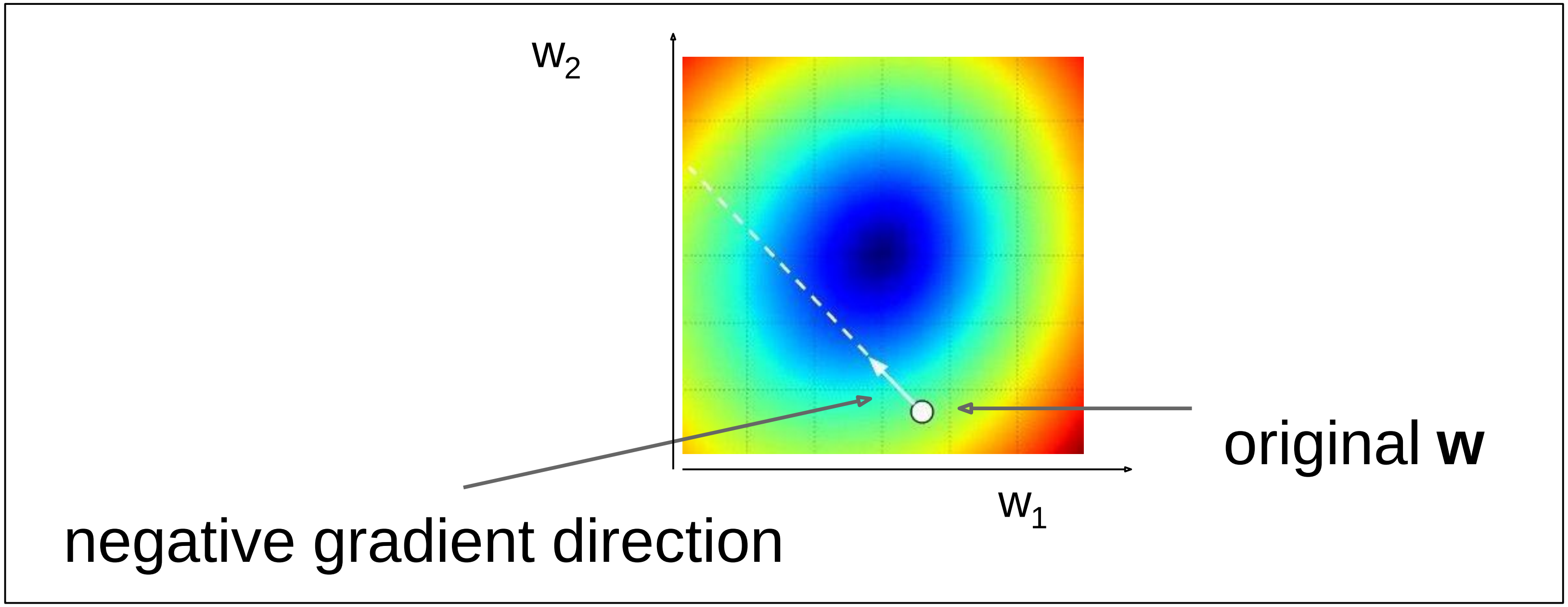


Figure from Andrej Karpathy

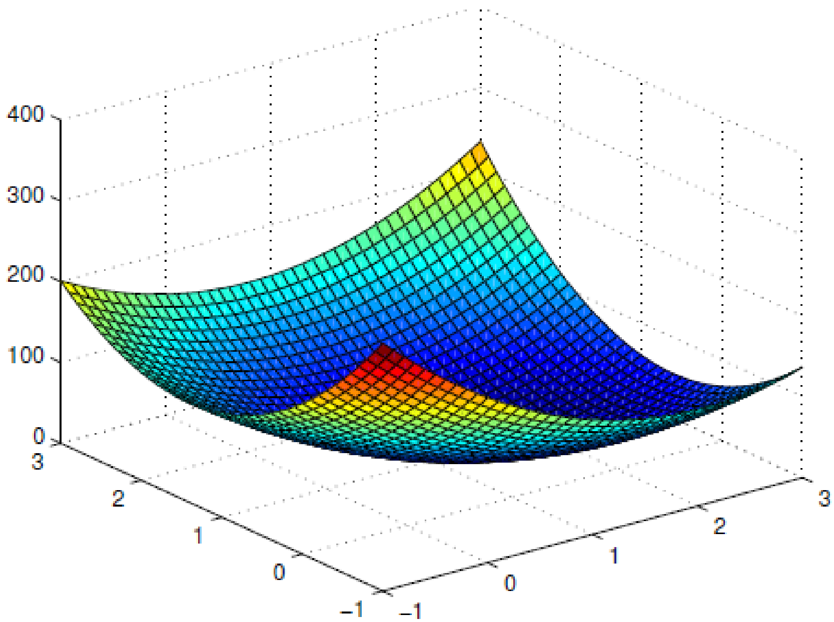
Gradient descent

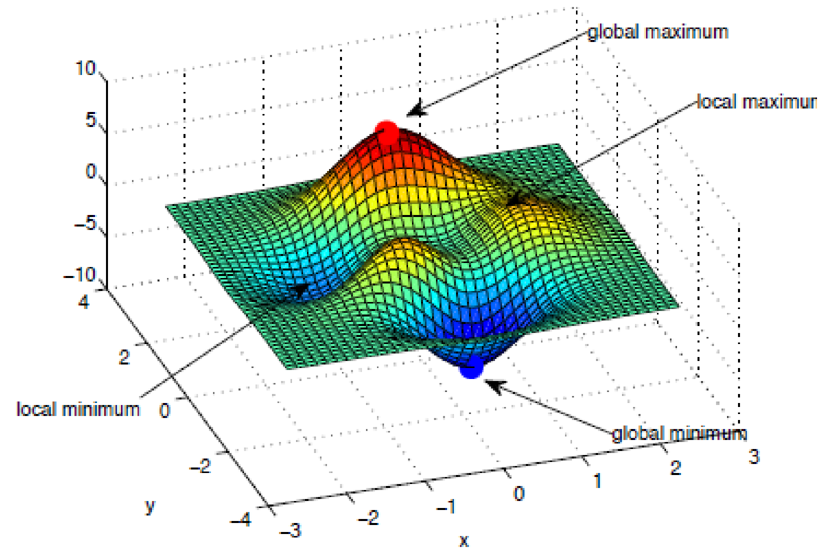
- Iteratively *subtract* the gradient with respect to the model parameters (**w**)
- I.e. we're moving in a direction opposite to the gradient of the loss
- I.e. we're moving towards *smaller* loss

Classic vs stochastic gradient descent

- In classic gradient descent, we compute the gradient from the loss for all training examples
- Could also only use *some* of the data for each gradient update
- We cycle through all the training examples multiple times
- Each time we’ve cycled through all of them once is called an ‘epoch’
- Allows faster training (e.g. on GPUs), parallelization

Solution optimality





- Global vs local minima
- Fortunately, least squares is convex

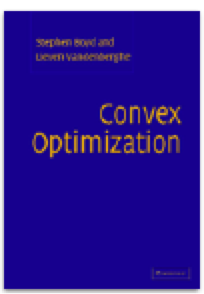
Adapted from Erik Sudderth

Solution optimality

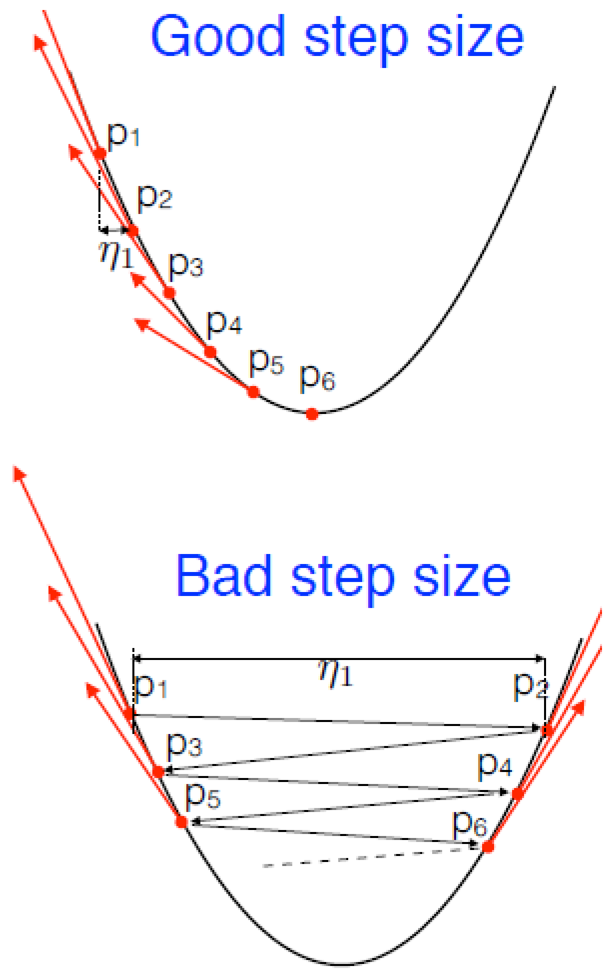
- The step size is important –
  - too small: slow convergence
  - too large: no convergence
- A strategy is to use large step sizes initially and small step sizes later:

$$\eta_t \leftarrow \eta_0/(t_0 + t)$$

- There are methods that converge faster by adapting step size to the curvature of the function
  - Field of convex optimization



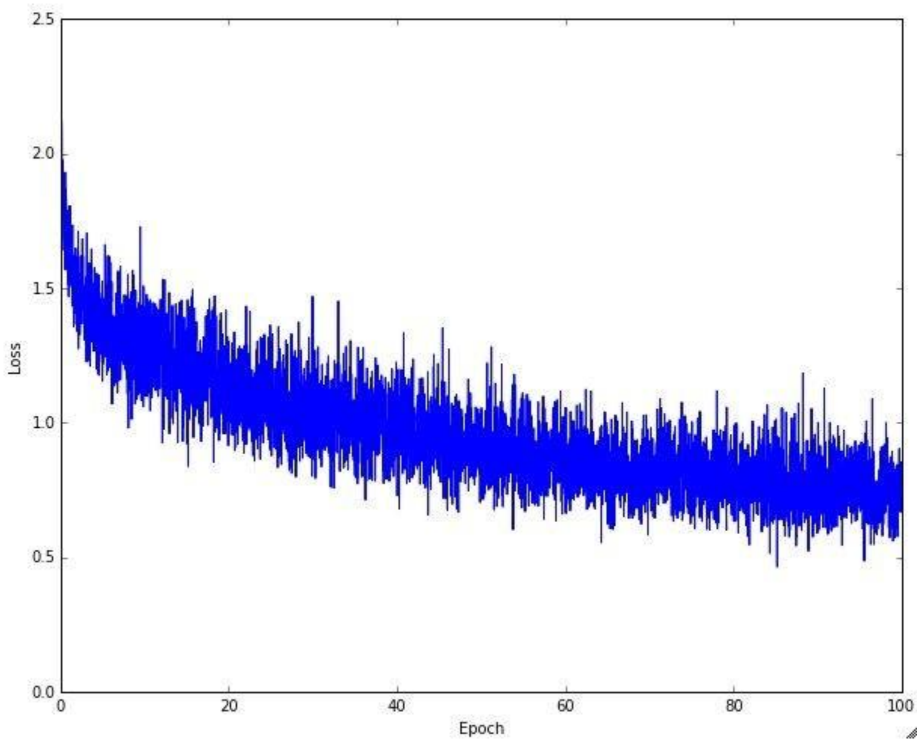
http://stanford.edu/~boyd/cvxbook/

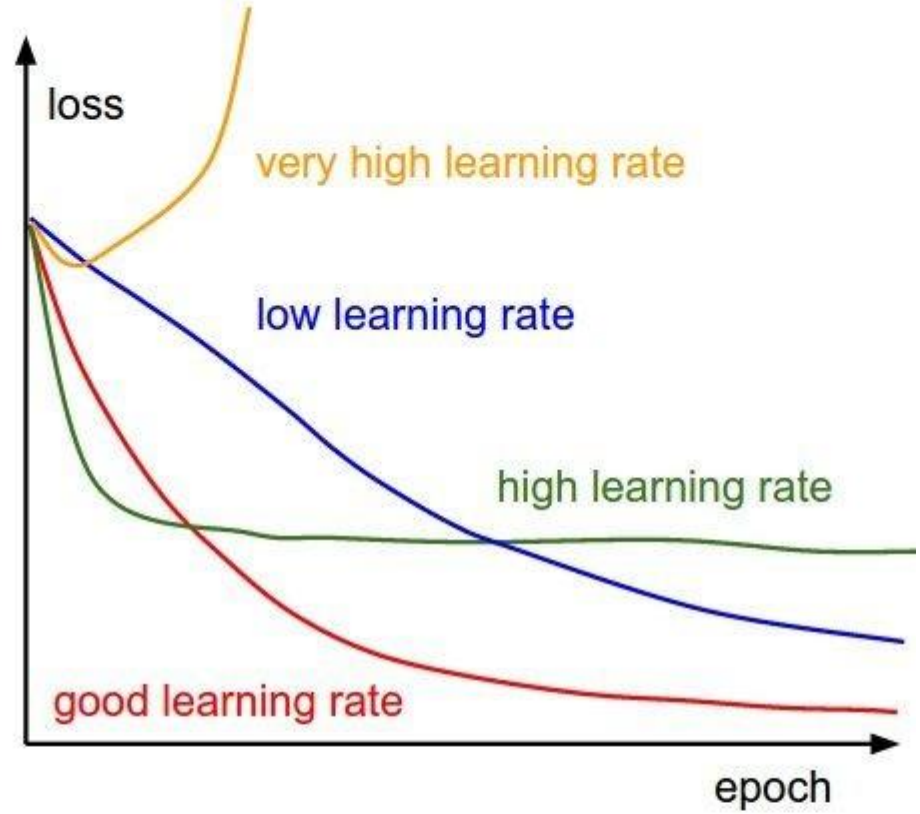


Slide credit: Subhransu Maji

Learning rate selection

The effects of step size (or "learning rate")





Andrej Karpathy

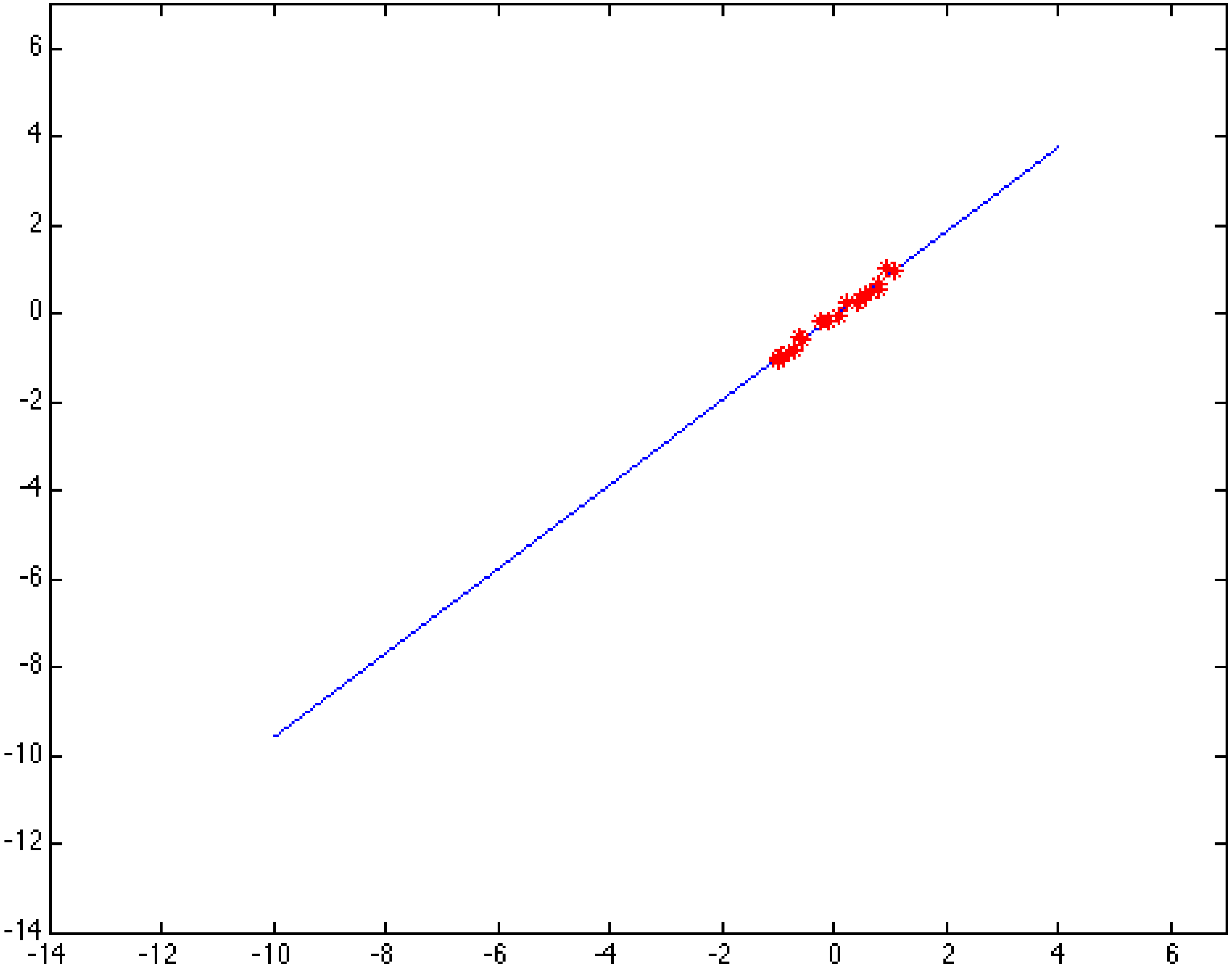
Summary: Closed form vs gradient descent

- Closed form:
  - Deterministic and elegant
  - Need to compute pseudoinverse of a large matrix from all data samples' features
- Gradient descent:
  - Update solution based on randomly chosen samples, iterate over all samples many times
  - Update model=weights with negative increment of the derivative of the loss function wrt weights
  - More efficient if lots of samples
  - Can be parallelized

Plan for Today

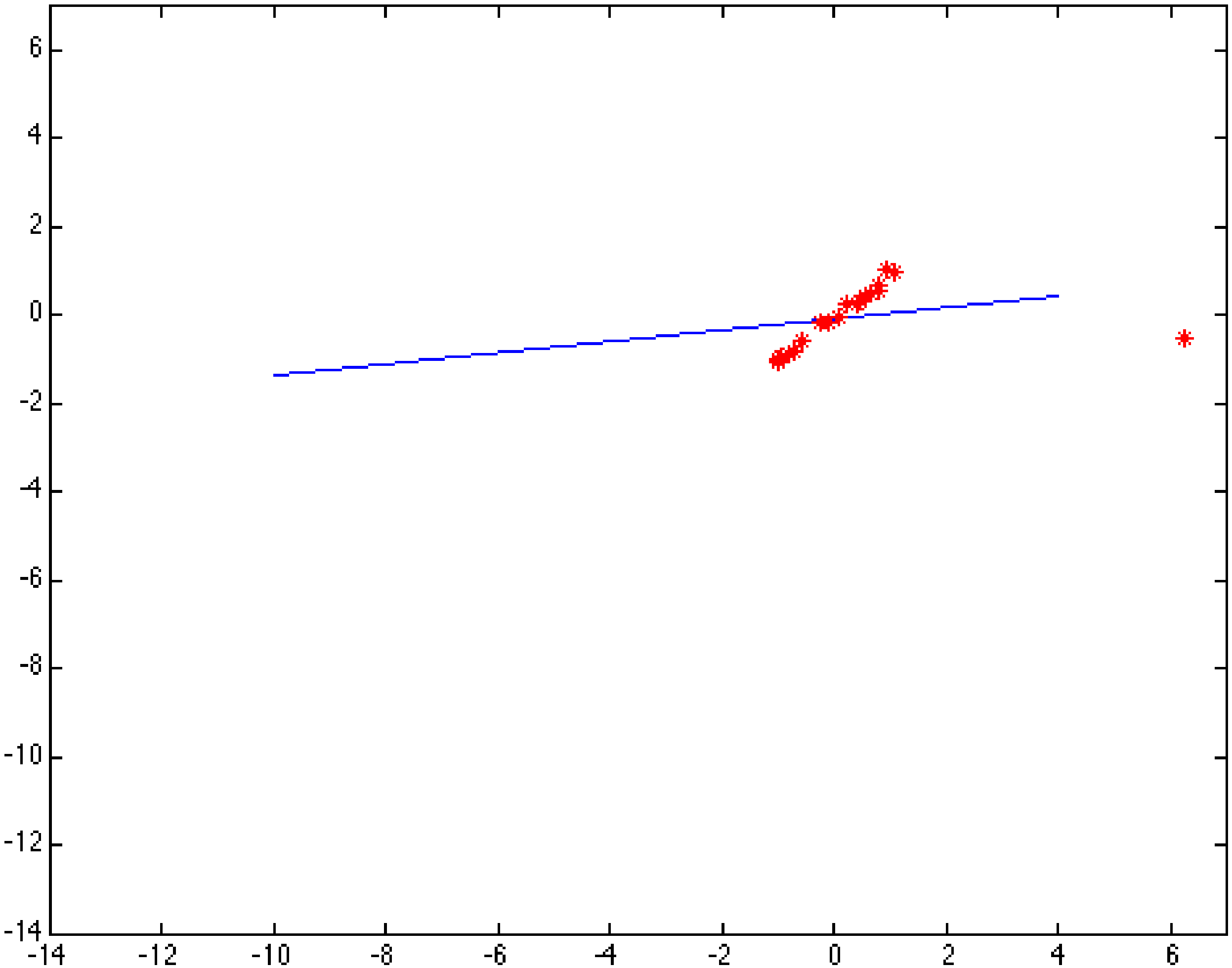
- Linear regression
- Closed-form solution via least squares
- Solution via gradient descent
- Dealing with outliers
- Generalization: bias and variance
- Regularization

Outliers affect least squares fit



Kristen Grauman

Outliers affect least squares fit



Kristen Grauman

Hypothesize and test

1. Try all possible parameter combinations
   - Repeatedly sample enough points to solve for parameters
   - Each point votes for all consistent parameters
   - E.g. each point votes for all possible lines on which it might lie

2. Score the given parameters: Number of consistent points

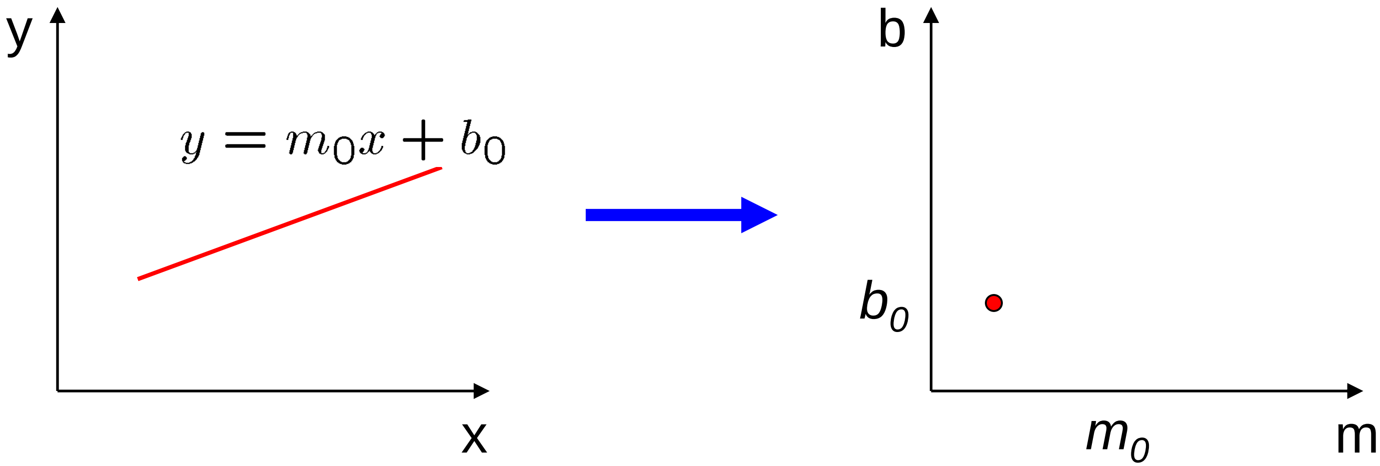
3. Choose the optimal parameters using the scores

- Noise & clutter features?
  - They will cast votes too, *but* typically their votes should be inconsistent with the majority of "good" features

- Two methods: Hough transform and RANSAC

Adapted from Derek Hoiem and Kristen Grauman

Hough transform for finding lines



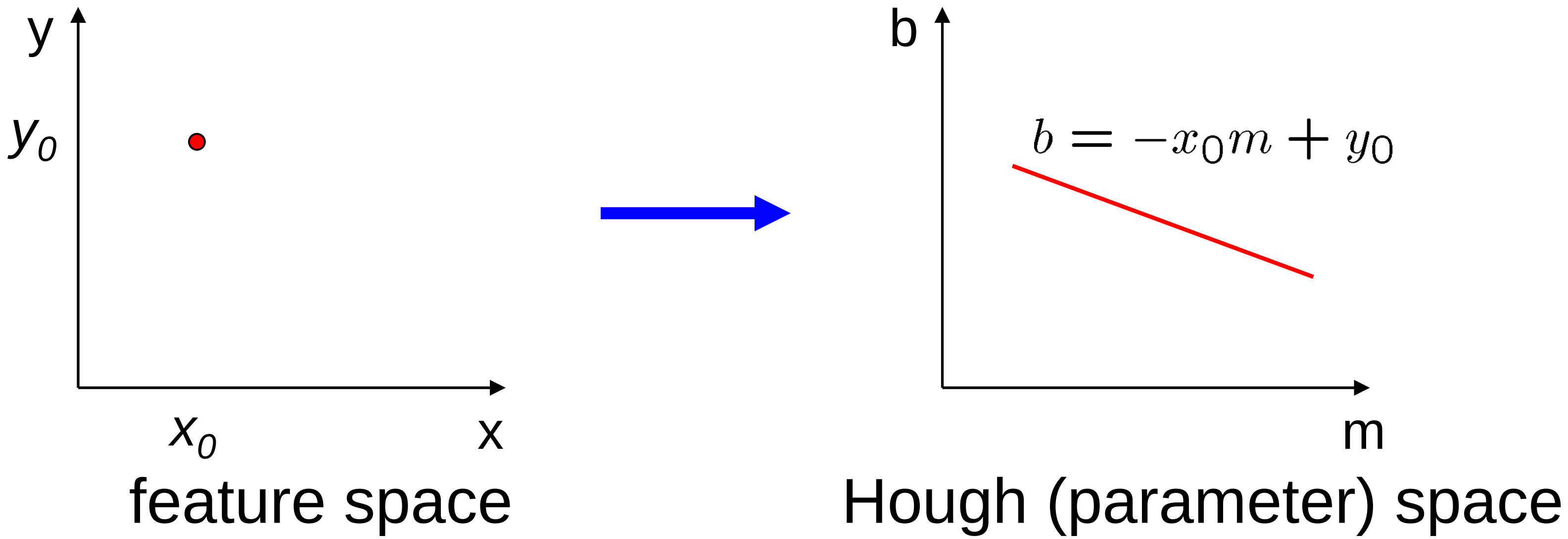
feature space

Hough (parameter) space

Connection between feature (x,y) and Hough (m,b) spaces

- A line in the feat space corresponds to a point in Hough space

Hough transform for finding lines

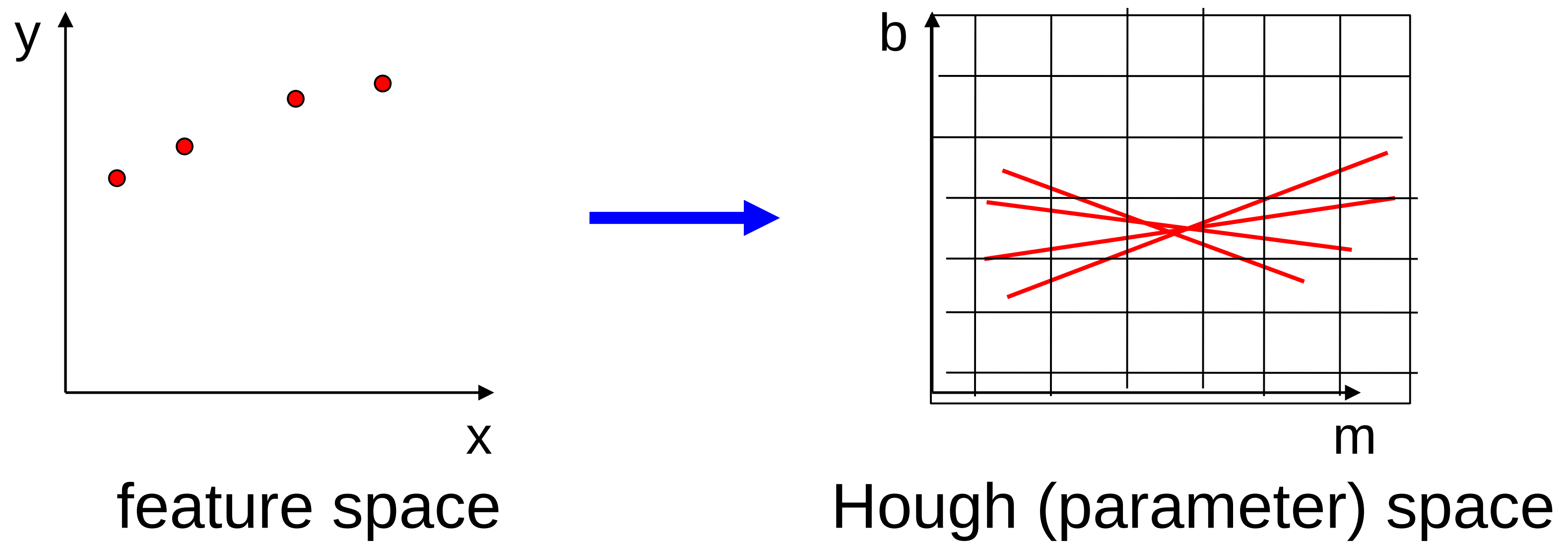


Connection between feature (x,y) and Hough (m,b) spaces

- A line in the feat space corresponds to a point in Hough space
- What does a point (x_0, y_0) in the feature space map to?
  - Answer: the solutions of $b = -x_0 m + y_0$
  - This is a line in Hough space

Steve Seitz

Hough transform for finding lines



How can we use this to find the most likely parameters (m,b) for the most prominent line in the feature space?

- Let each edge point in feature space *vote* for a set of possible parameters in Hough space
- Accumulate votes in discrete set of bins; parameters with the most votes indicate line in feature space

Steve Seitz

RANdom Sample Consensus (RANSAC)

- <u>RANSAC loop</u>:

1. Randomly select a *seed group* of ***s*** points on which to base model estimate
2. Fit model to these ***s*** points
3. Find *inliers* to this model (i.e., points whose distance from the line is less than ***t***)
4. If there are ***d*** or more inliers, re-compute estimate of model on all of the inliers
5. Repeat ***N*** times

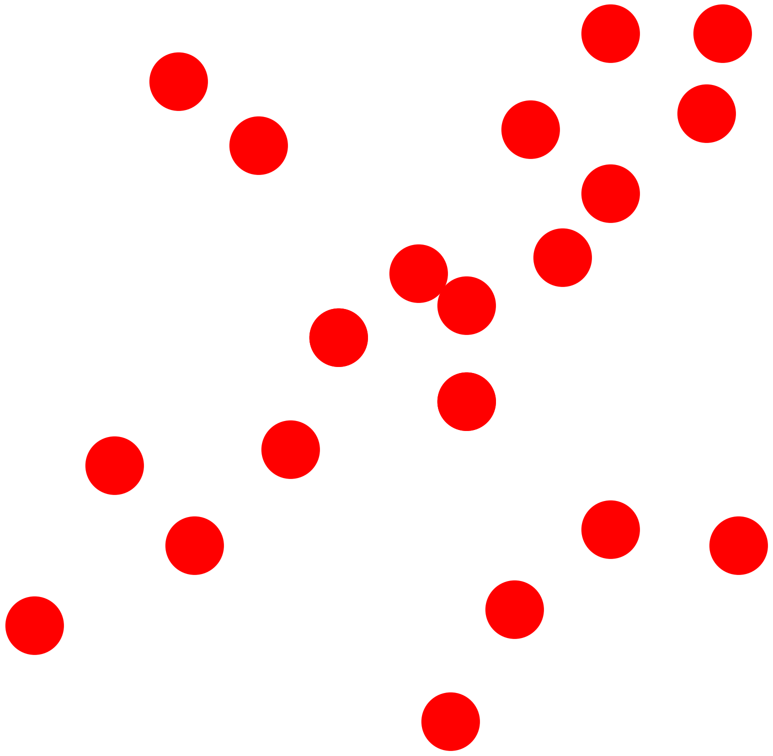
- Keep the model with the largest number of inliers

Modified from Kristen Grauman and Svetlana Lazebnik

RANSAC

(RANdom SAmple Consensus) :

Fischler & Bolles in '81.



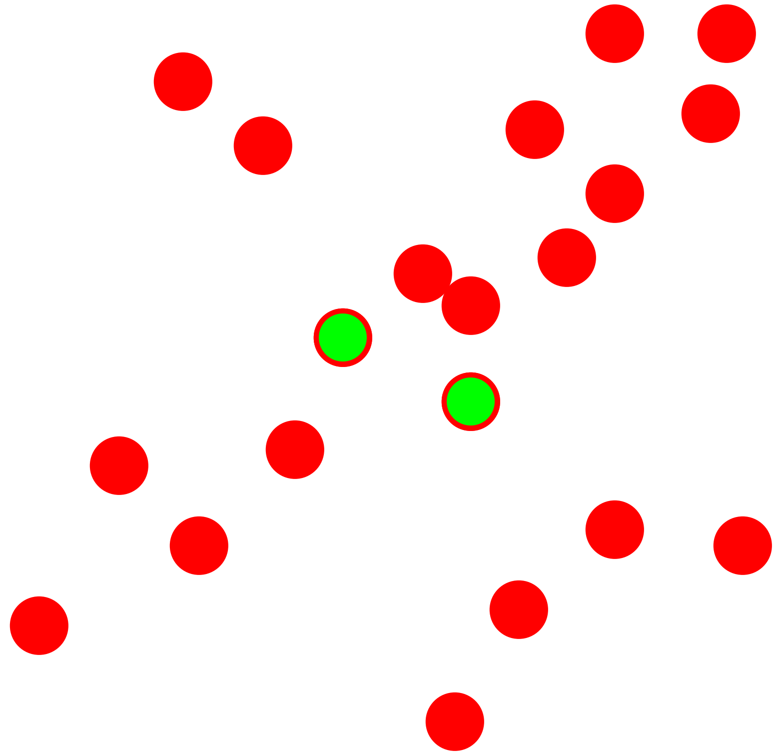
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC

Line fitting example



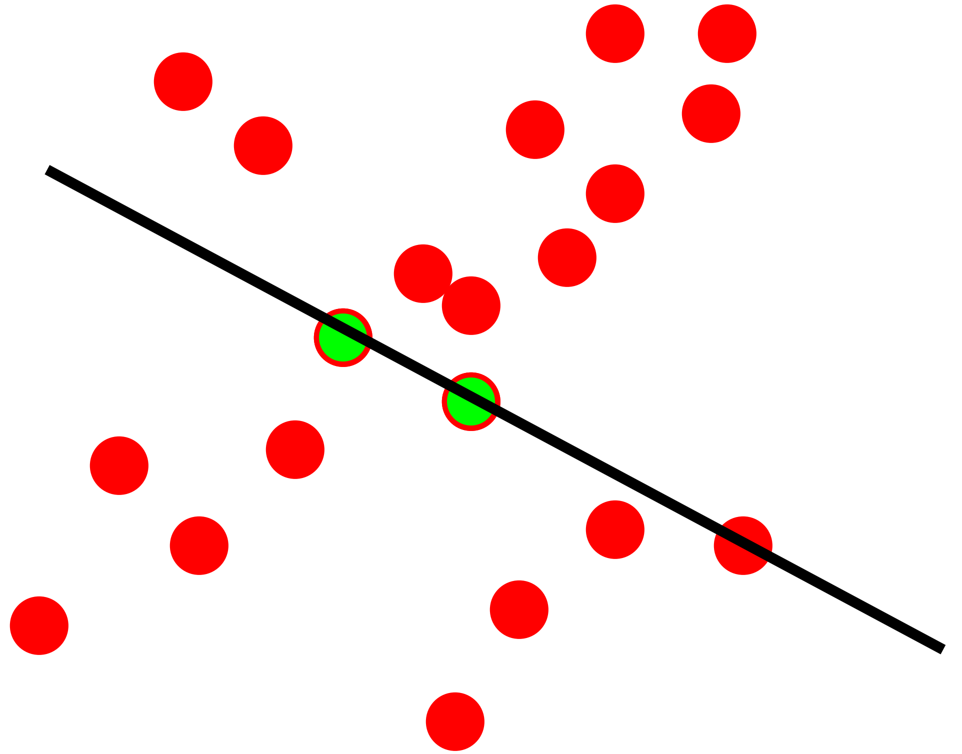
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC

Line fitting example



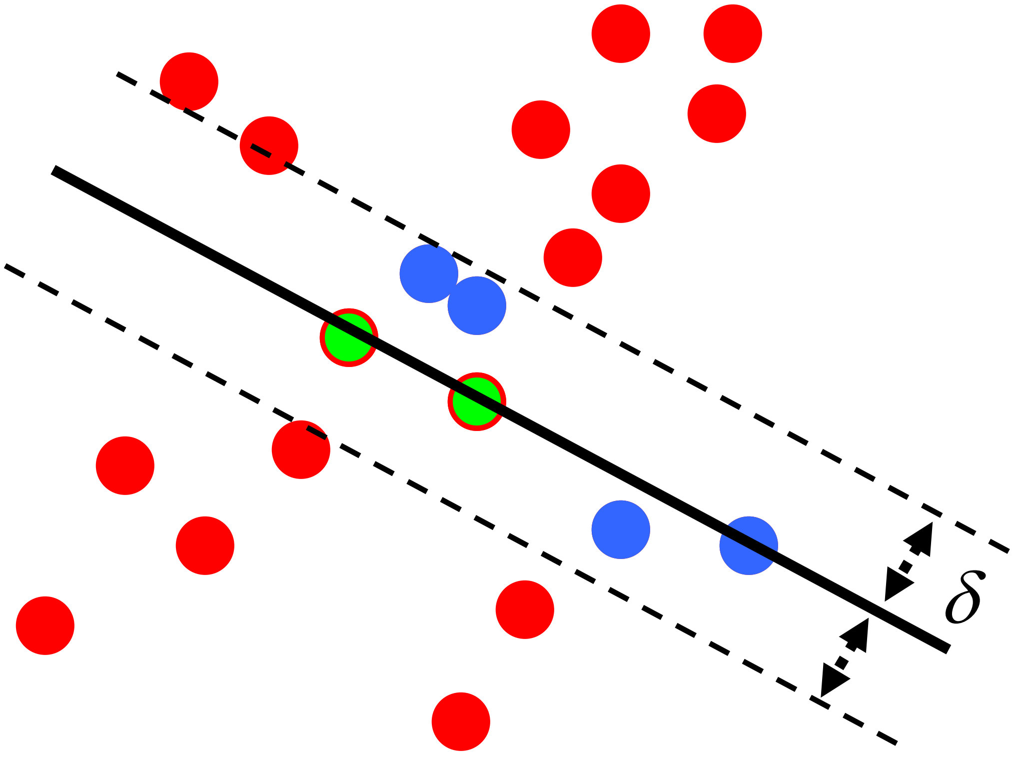
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC

Line fitting example



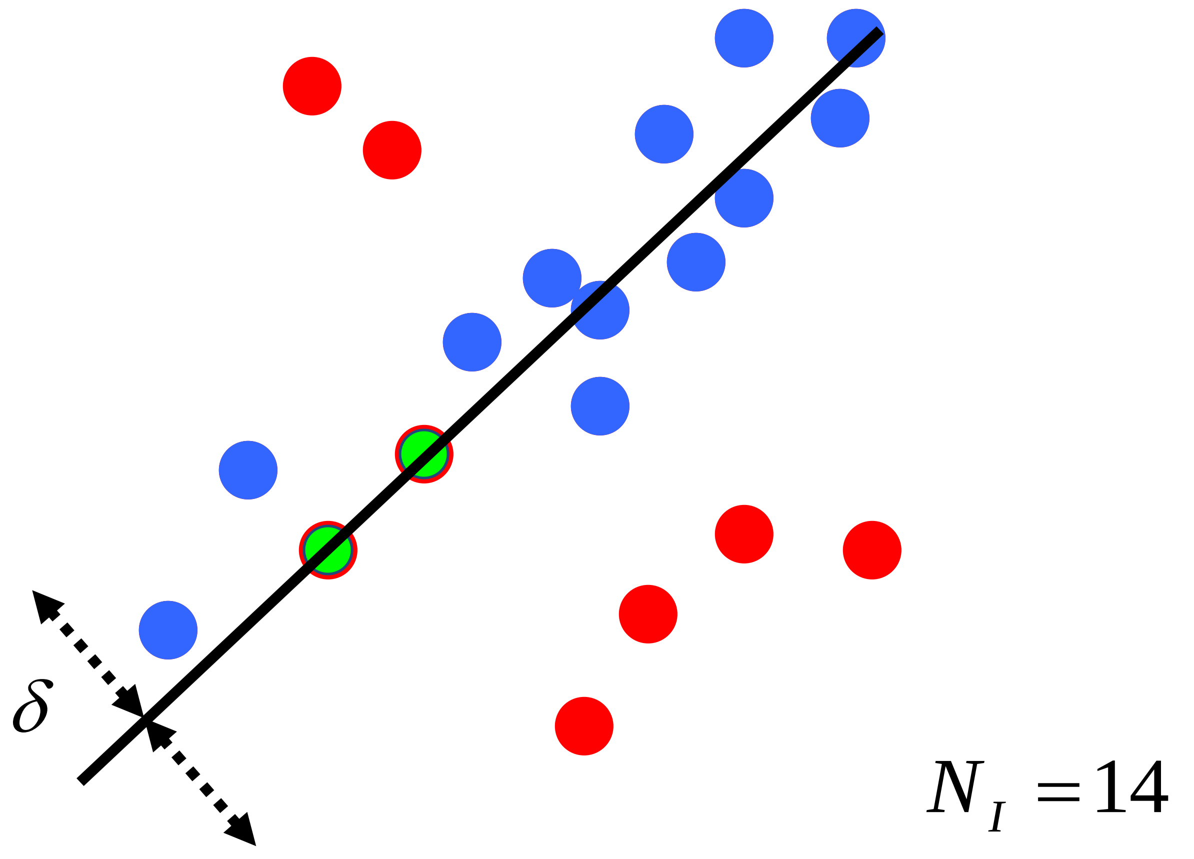
$$N_I = 6$$

Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC



Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Plan for Today

- Linear regression
- Closed-form solution via least squares
- Solution via gradient descent
- Dealing with outliers
- Generalization: bias and variance
- Regularization

Generalization

Training set (labels known)

Test set (labels unknown)

- How well does a learned model generalize from the data it was trained on to a new test set?

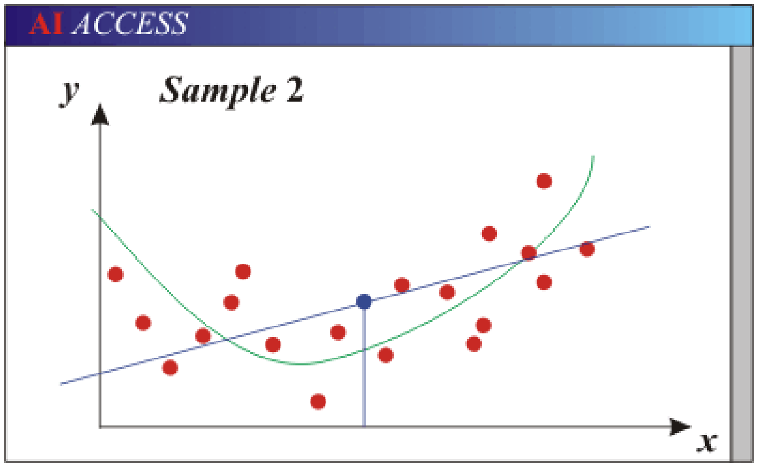
Slide credit: L. Lazebnik

Generalization

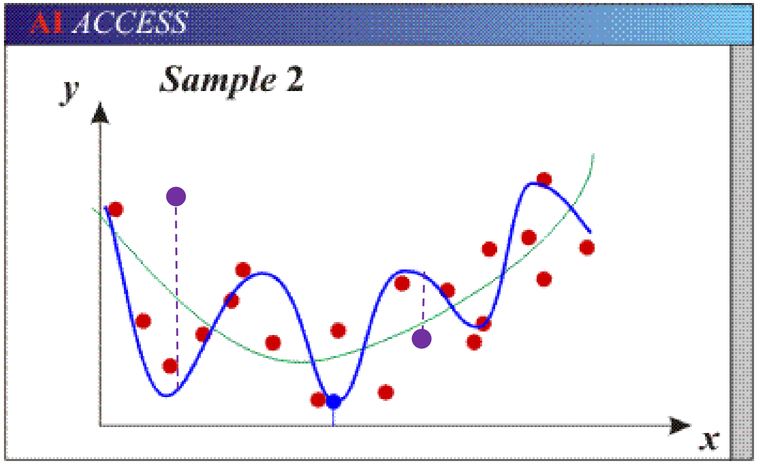
- Components of expected loss
  - **Noise** in our observations: unavoidable
  - **Bias:** how much avg model over all training sets differs from true model
    - Error due to inaccurate assumptions/simplifications by the model
  - **Variance:** how much models estimated from different training sets differ from each other
- **Underfitting:** model too “simple” to represent all relevant class characteristics
  - High bias and low variance
  - High training error and high test error
- **Overfitting:** model too “complex” and fits irrelevant characteristics (noise) in the data
  - Low bias and high variance
  - Low training error and high test error

Adapted from L. Lazebnik

Bias-Variance Trade-off



- Models with too few parameters are inaccurate because of a large bias (not enough flexibility).



- Models with too many parameters are inaccurate because of a large variance (too much sensitivity to the sample).

Red dots = training data (all that we see before we ship off our model!)

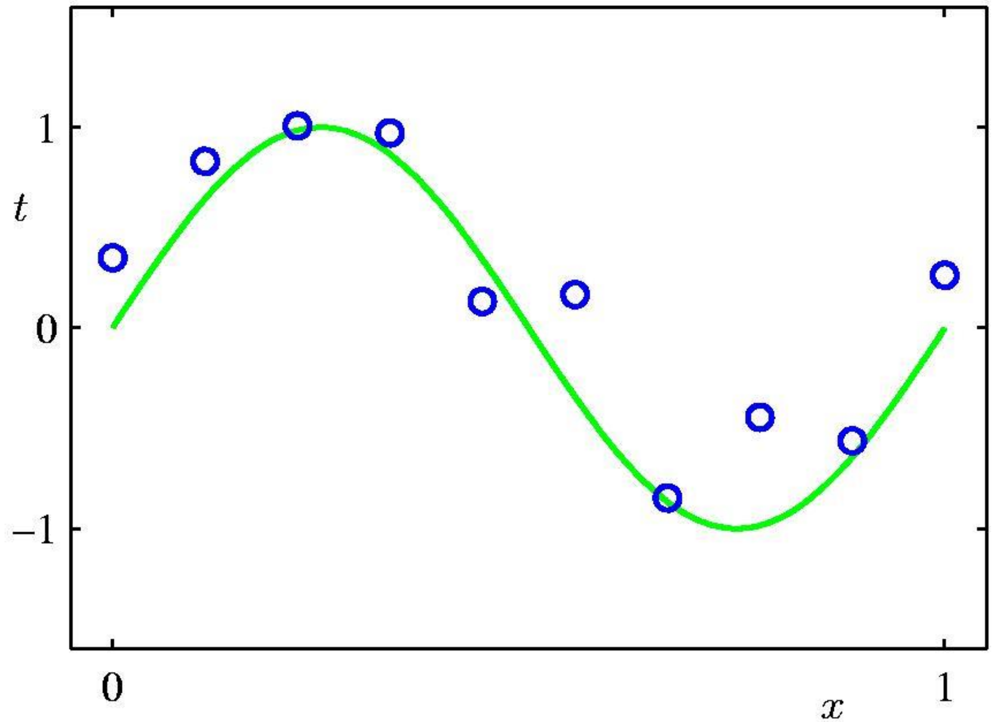
Purple dots = possible test points

Green curve = true underlying model

Blue curve = our predicted model/fit

Adapted from D. Hoiem

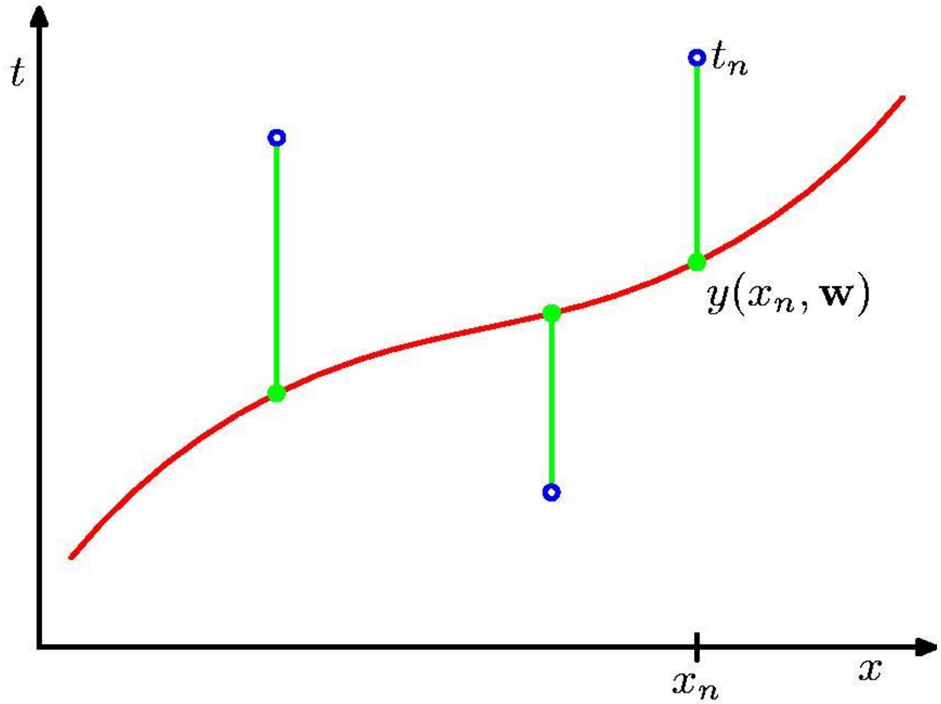
Polynomial Curve Fitting



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

Slide credit: Chris Bishop

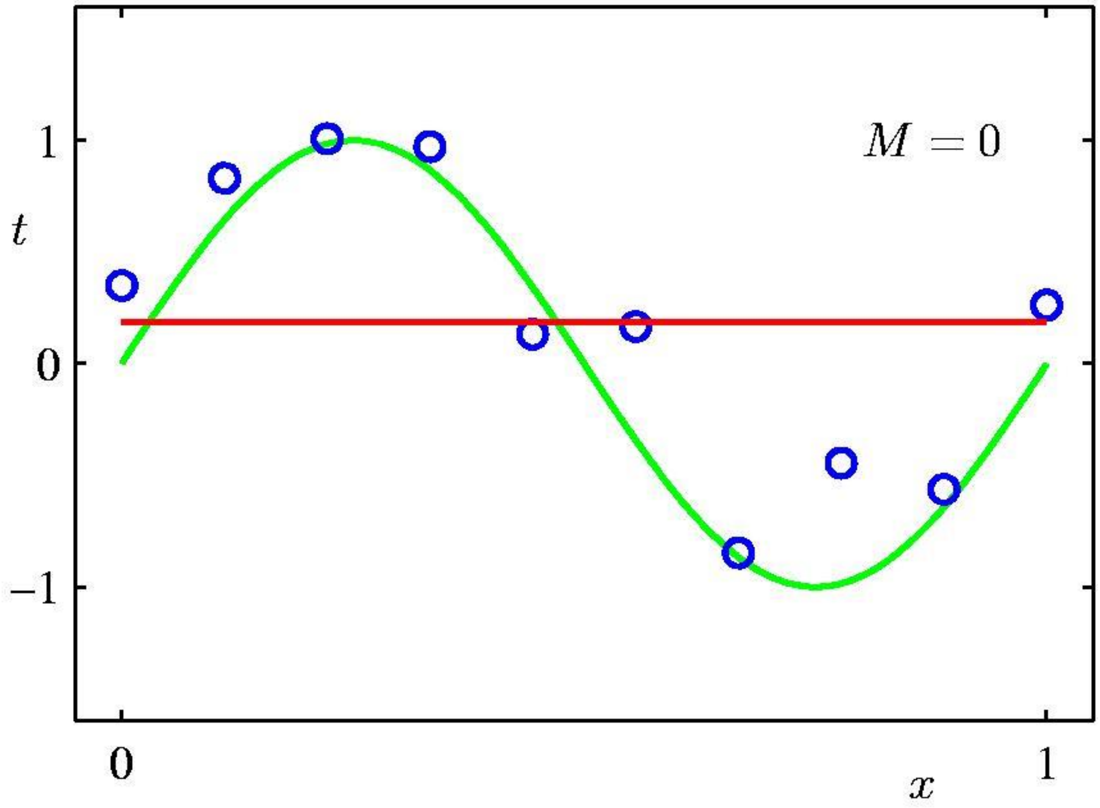
Sum-of-Squares Error Function



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

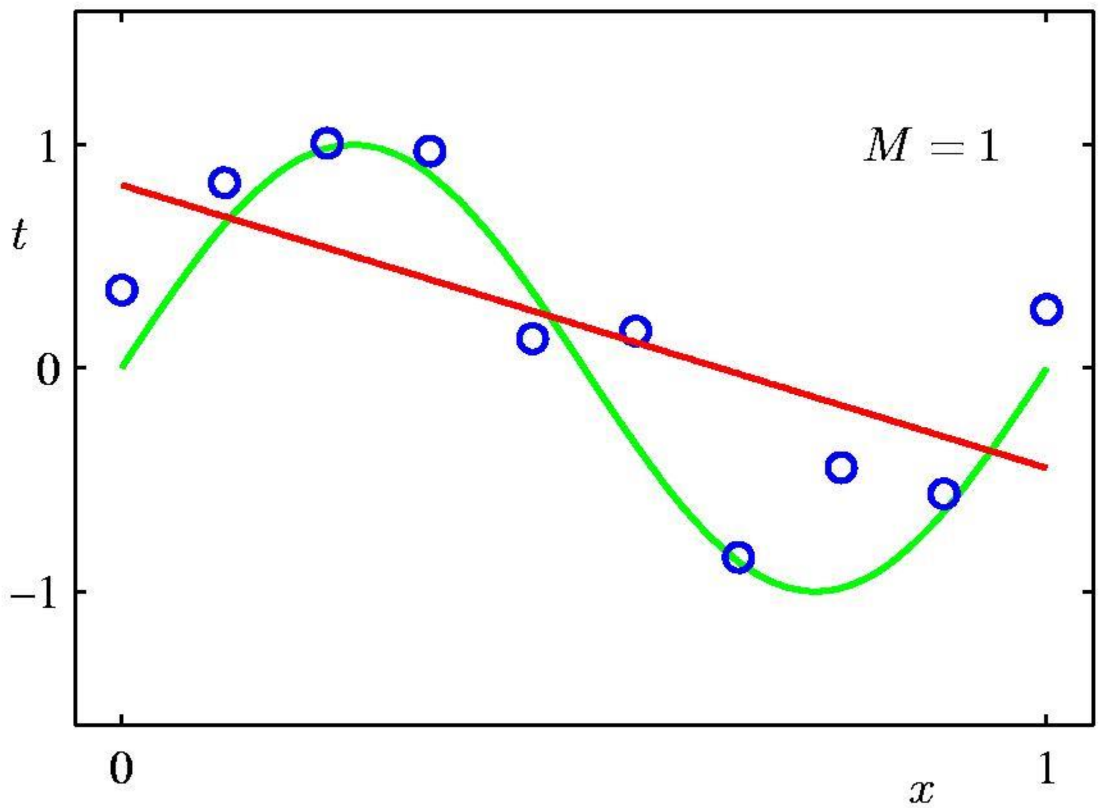
Slide credit: Chris Bishop

0^{th} Order Polynomial



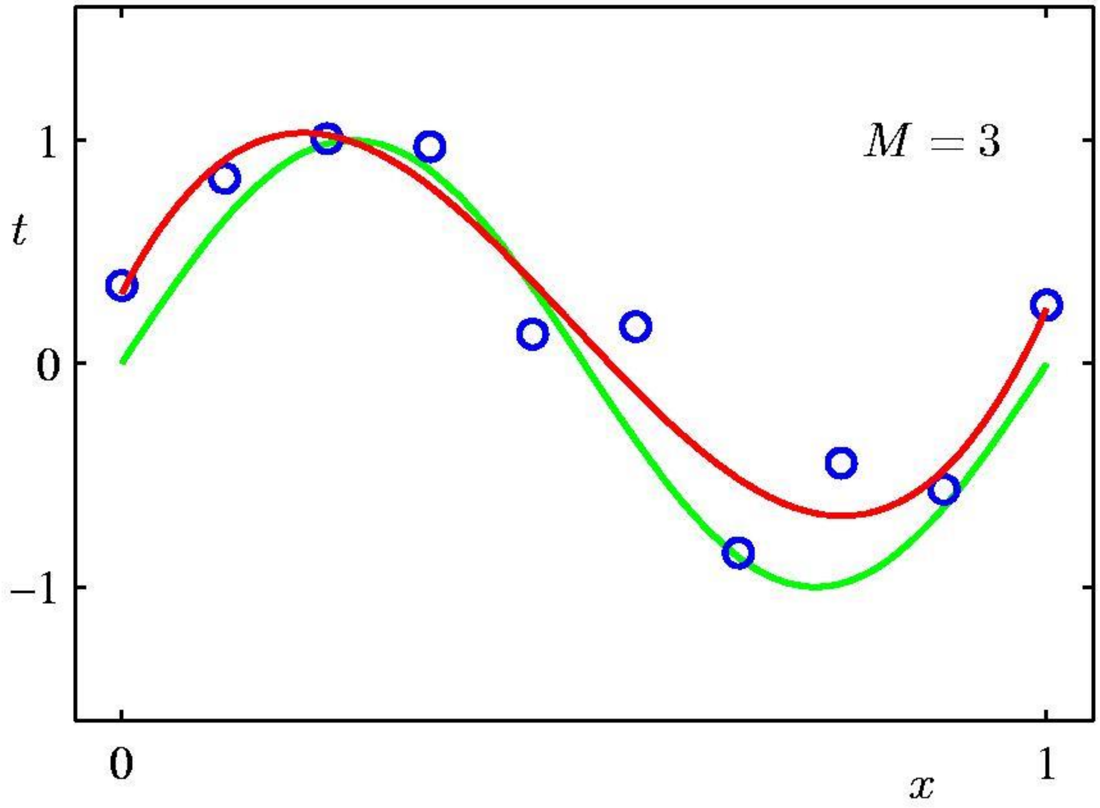
Slide credit: Chris Bishop

1^{st} Order Polynomial



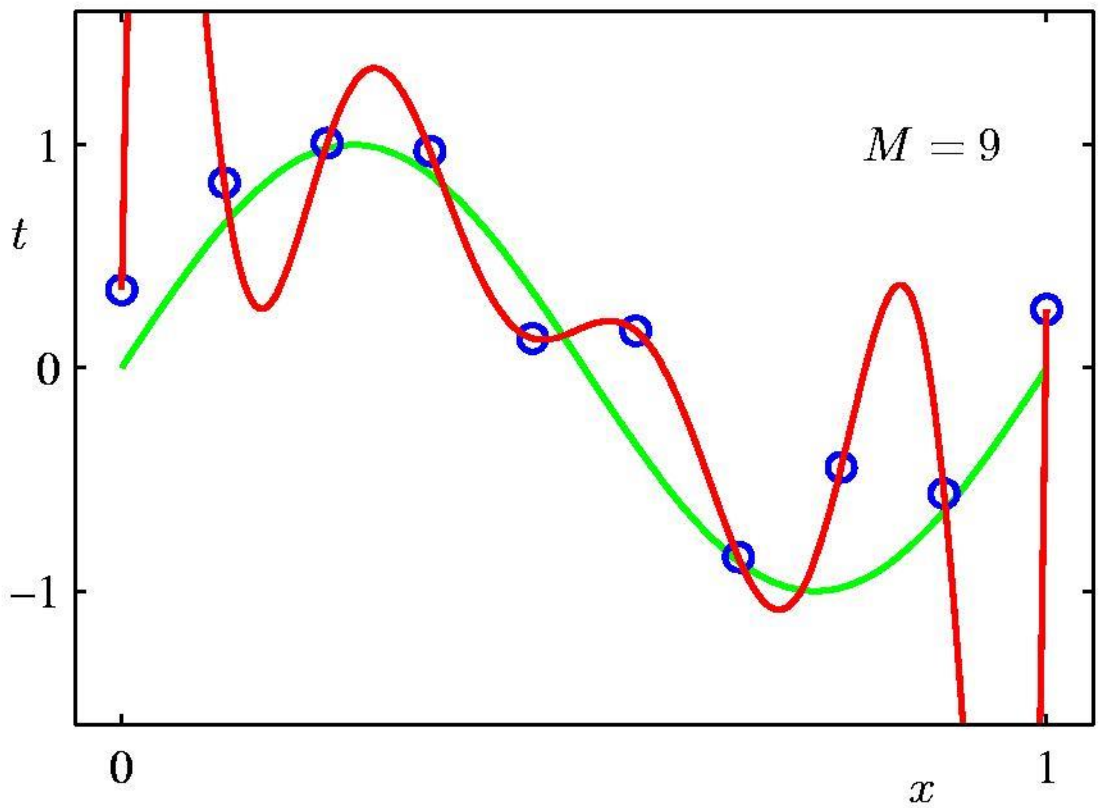
Slide credit: Chris Bishop

3rd Order Polynomial



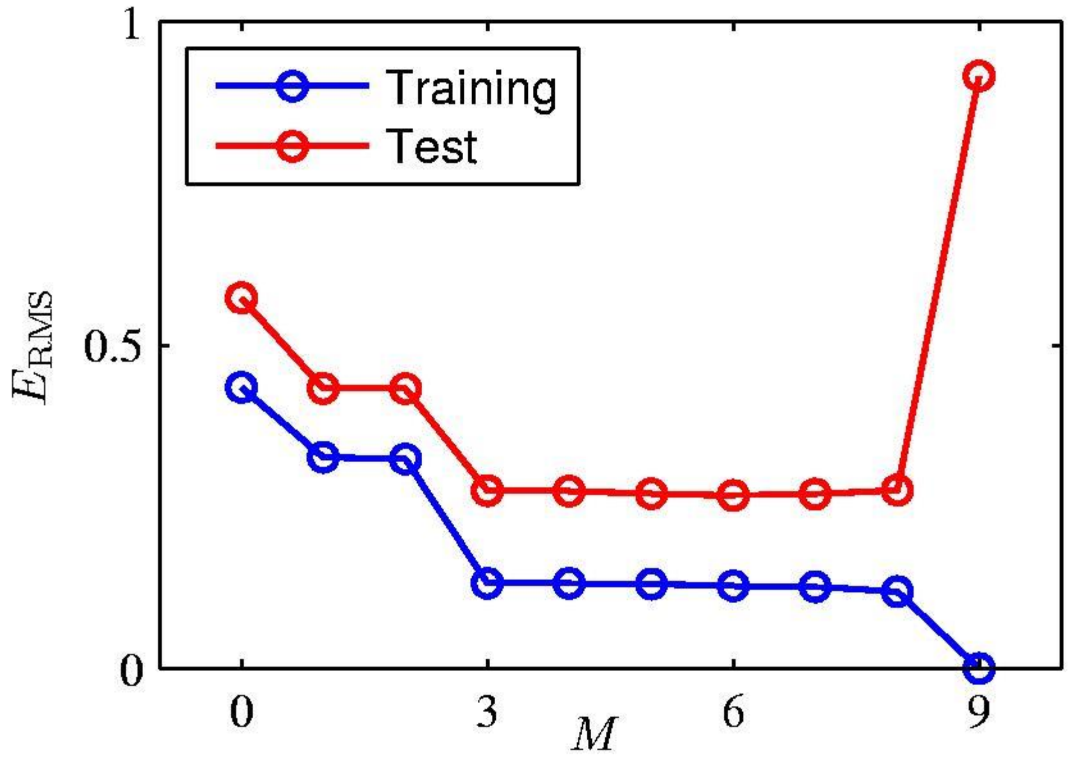
Slide credit: Chris Bishop

9th Order Polynomial



Slide credit: Chris Bishop

Over-fitting

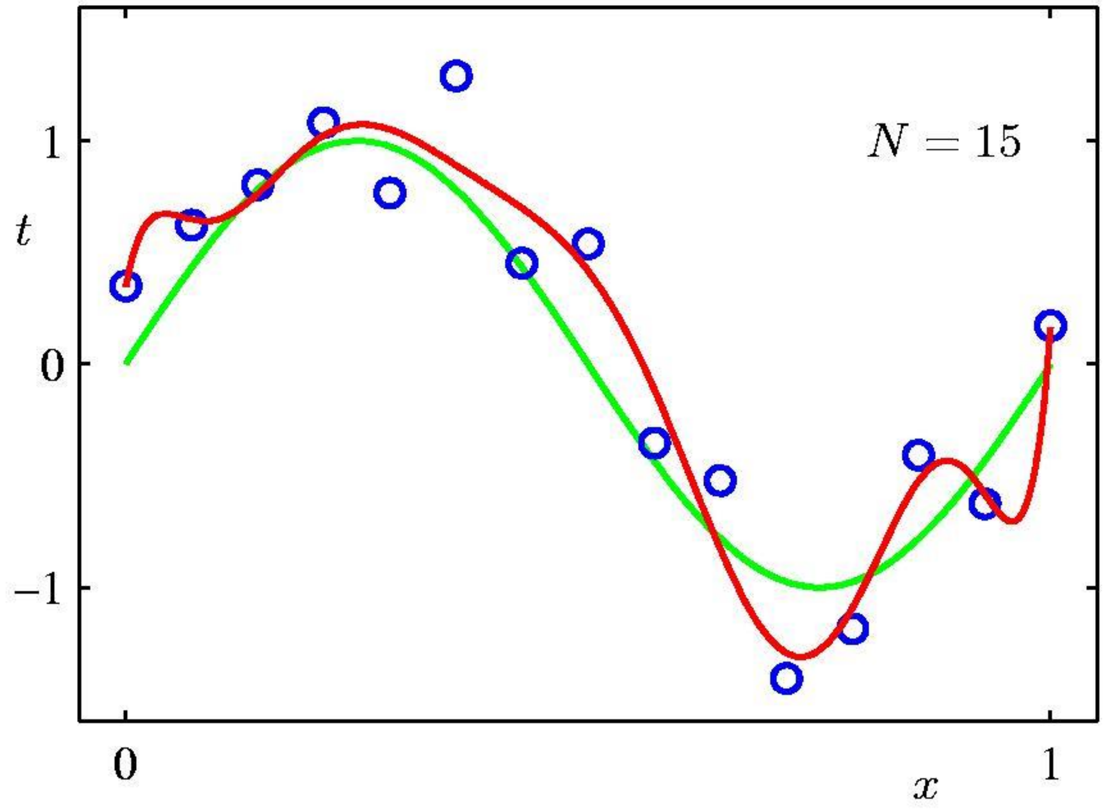


Root-Mean-Square (RMS) Error: $E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^\star)/N}$

Slide credit: Chris Bishop

Effect of Data Set Size

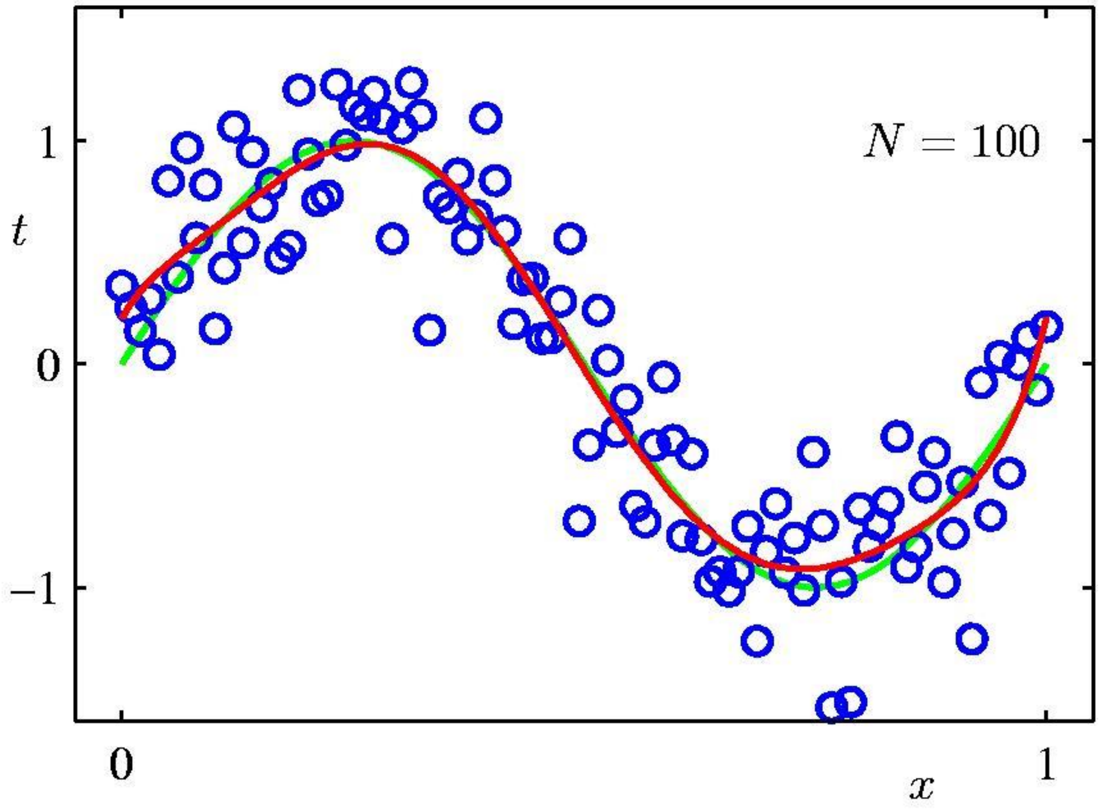
9th Order Polynomial



Slide credit: Chris Bishop

Effect of Data Set Size

9^{th} Order Polynomial



Slide credit: Chris Bishop

Regularization

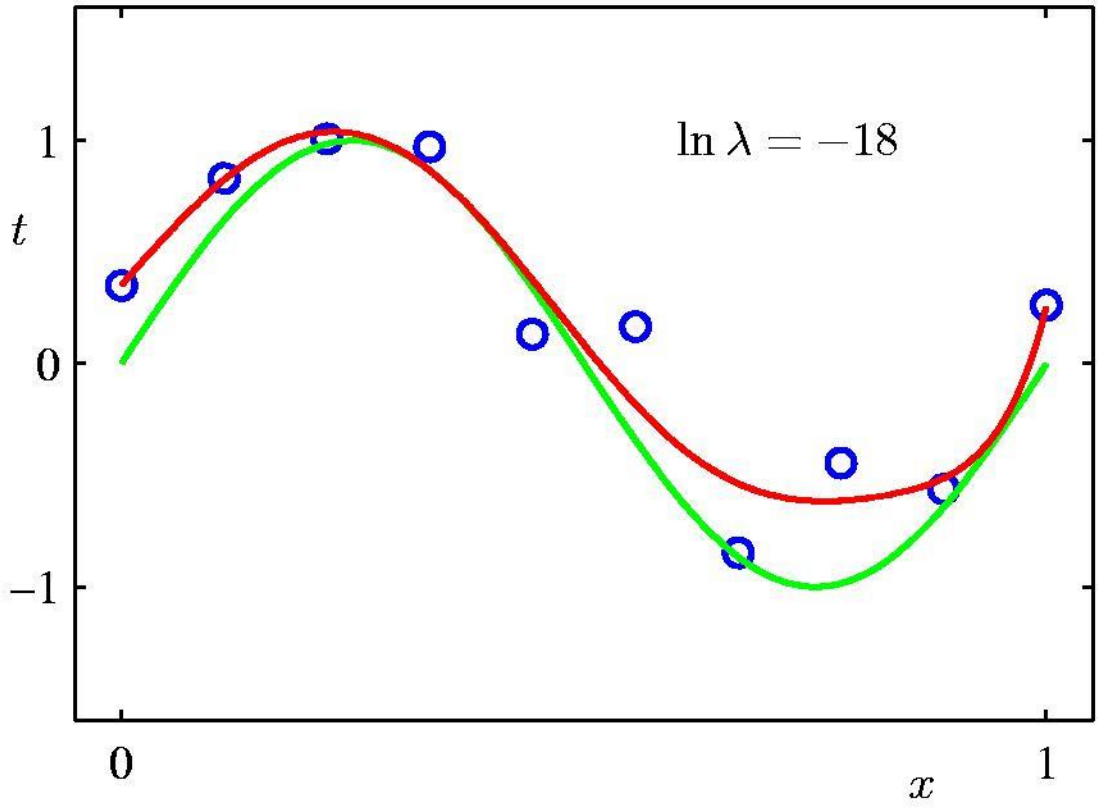
- Penalize large coefficient values

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

- (Remember: We want to minimize this expression.)

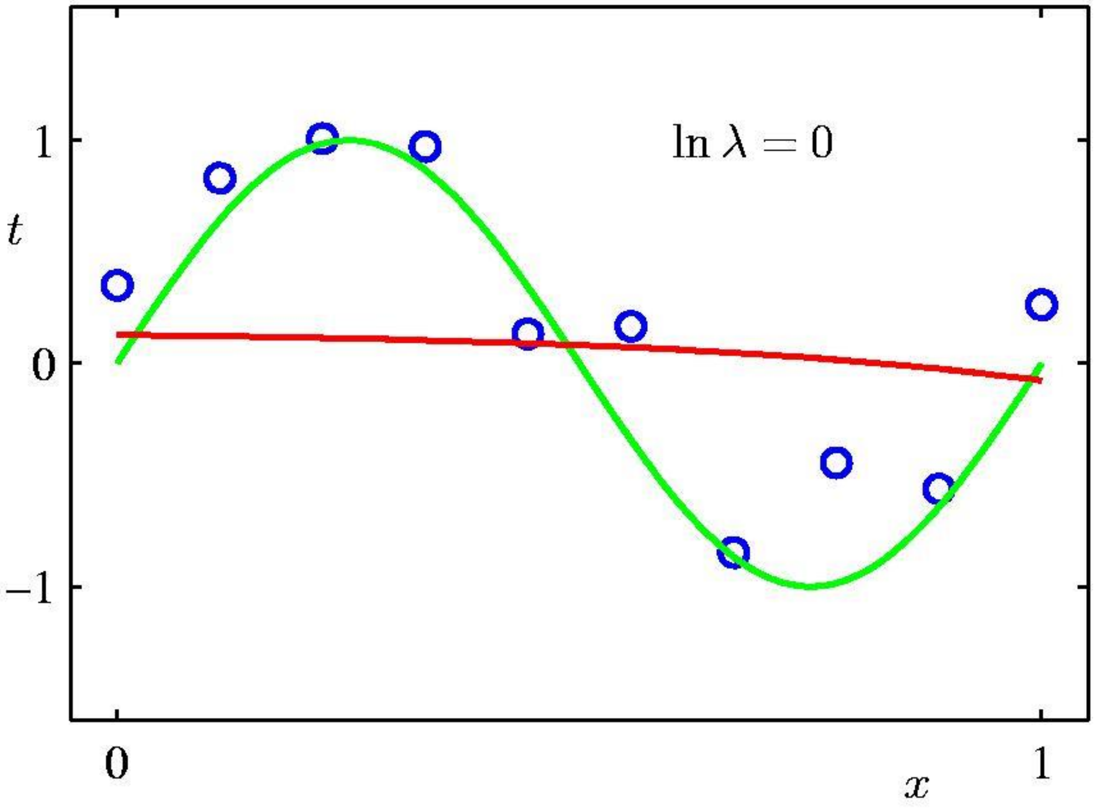
Adapted from Chris Bishop

Regularization



Slide credit: Chris Bishop

Regularization



Slide credit: Chris Bishop

Polynomial Coefficients

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

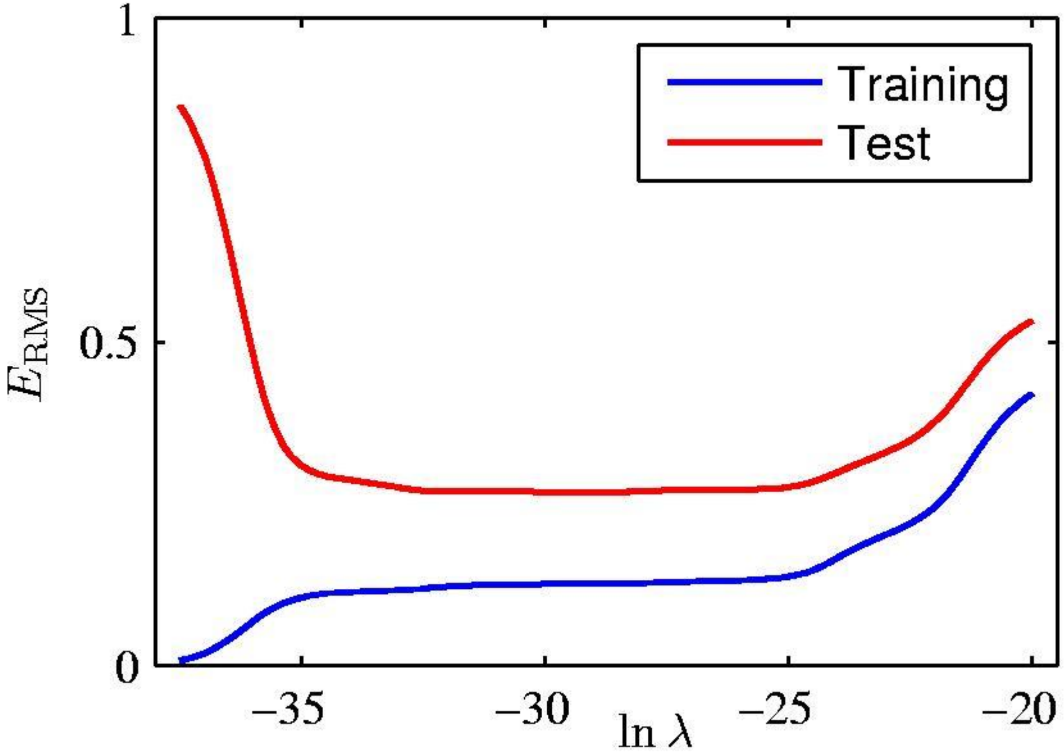
Slide credit: Chris Bishop

Polynomial Coefficients

	No regularization		Huge regularization
	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01

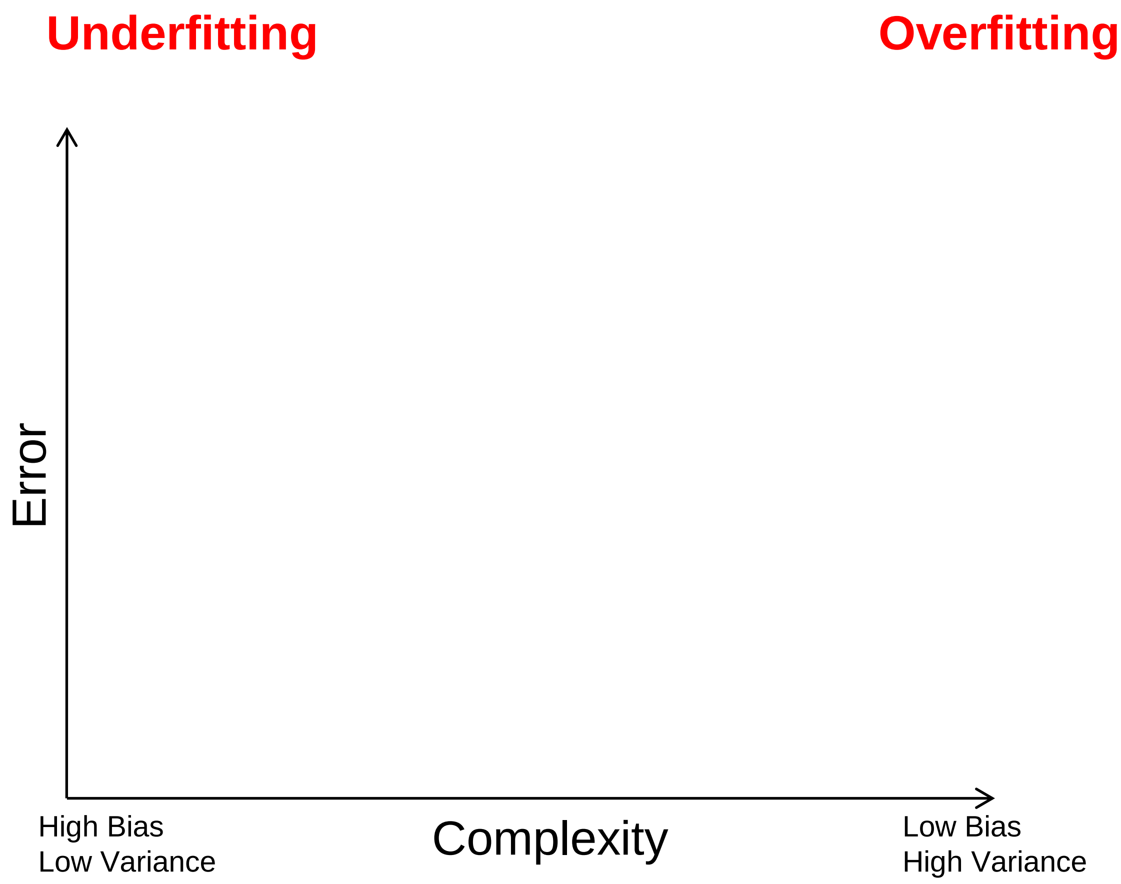
Adapted from Chris Bishop

Effect of Regularization



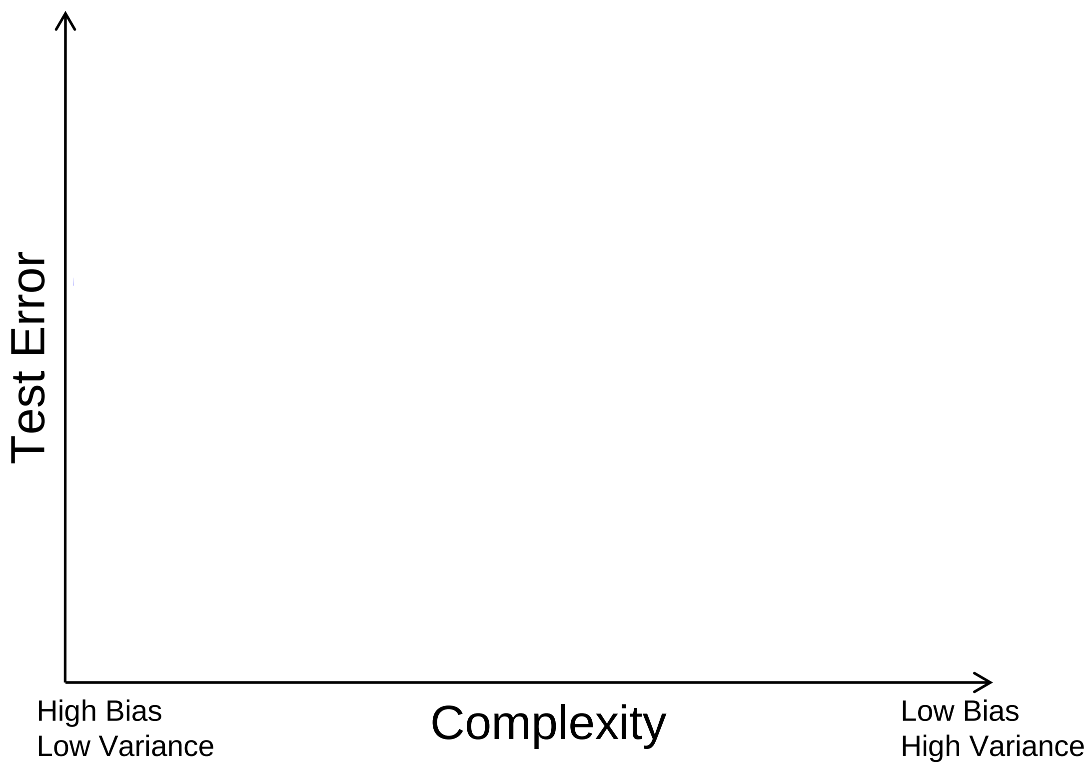
Slide credit: Chris Bishop

Training vs test error



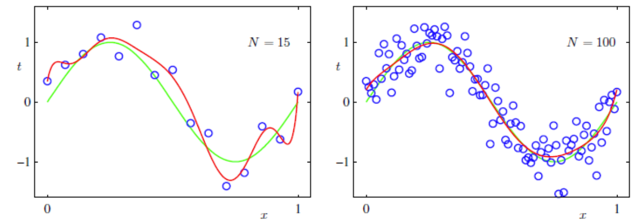
Slide credit: D. Hoiem

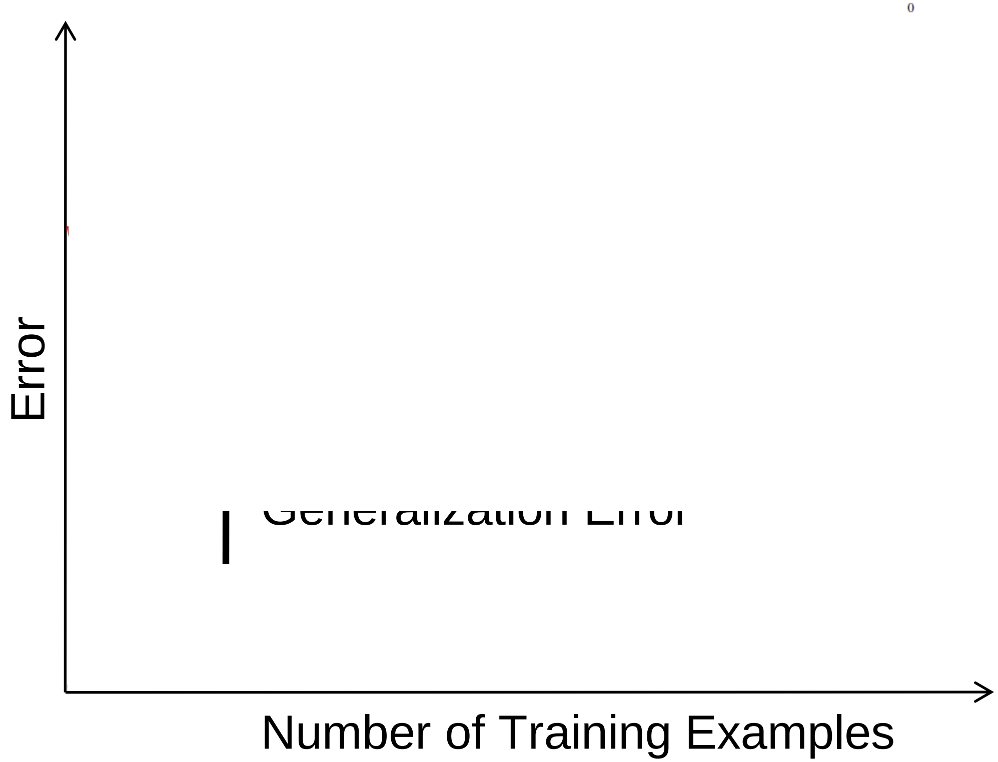
Effect of Training Set Size



Effect of Training Set Size

Fixed prediction model

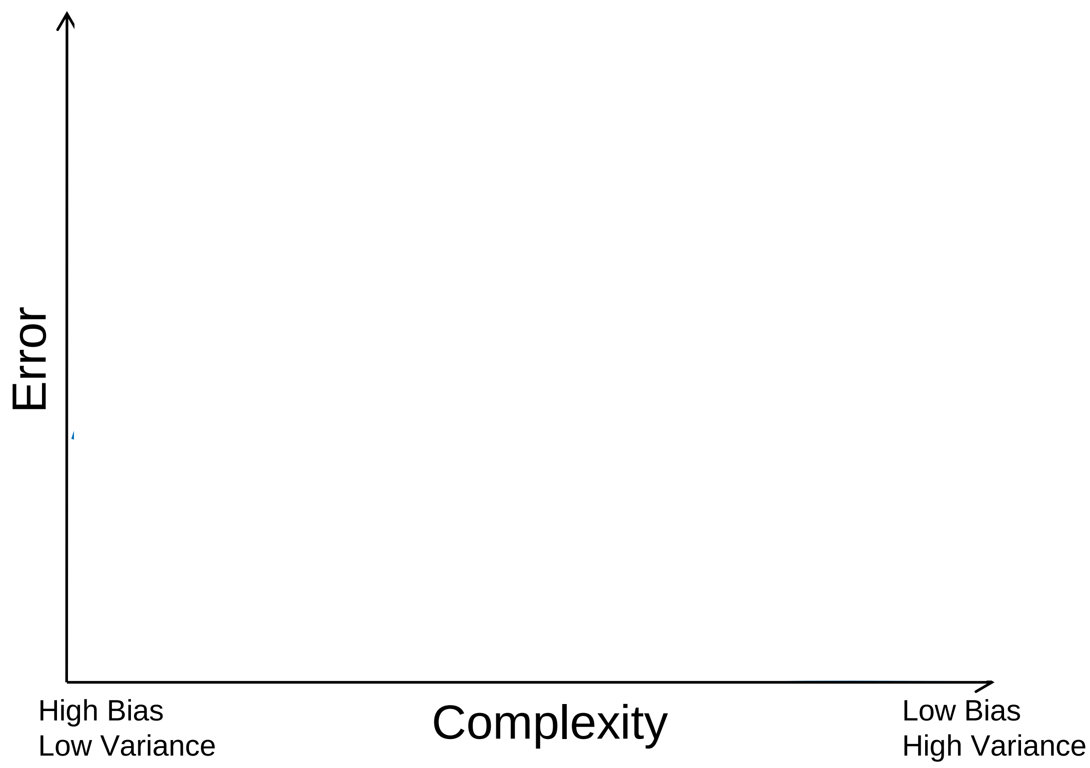




Adapted from D. Hoiem

Choosing the trade-off between bias and variance

- Need validation set (separate from the test set)



Slide credit: D. Hoiem

How to reduce variance?

- Choose a simpler classifier
- Regularize the parameters
- Get more training data

Slide credit: D. Hoiem

Overfitting take-away

- Three kinds of error
  - Inherent: unavoidable
  - Bias: due to over-simplifications
  - Variance: due to inability to perfectly estimate parameters from limited data
- Try simple models first, and use increasingly powerful models with more training data
- Important note: Overfitting applies to all tasks in machine learning, not just regression!

Adapted from D. Hoiem