

CSC 411: Lecture 02: Linear Regression

Class based on Raquel Urtasun & Rich Zemel's lectures

Sanja Fidler

University of Toronto

Jan 13, 2016

(Most plots in this lecture are from Bishop's book)

Problems for Today

- What should I watch this Friday?

Find Movies, TV shows, Celebrities and more... All

Movies, TV & Showtimes Celebs, Events & Photos News & Community Watchlist

The Martian (2015)
PG-13 | 144 min | Adventure, Comedy, Drama | 2 October 2015 (USA) 9
8.1 Your rating: ★★★★★★★★★★/10
Ratings: 8.1/10 from 271,829 users Metascore: 80/100
Reviews: 750 user | 499 critic | 46 from Metacritic.com

During a manned mission to Mars, Astronaut Mark Watney is presumed dead after a fierce storm and left behind by his crew. But Watney has survived and finds himself stranded and alone on the hostile planet. With only meager supplies, he must draw upon his ingenuity, wit and spirit to subsist and find a way to signal to Earth that he is alive.

Director: Ridley Scott
Writers: Drew Goddard (screenplay), Andy Weir (book)
Stars: Matt Damon, Jessica Chastain, Kristen Wiig | See full cast and crew »

+ Watchlist Watch Trailer Share...

Problems for Today

- What should I watch this Friday?



The image shows a screenshot of the IMDb website for the movie "Point Break" (2015). The page features the IMDb logo at the top left, followed by a search bar and navigation links for "Movies, TV & Showtimes", "Celebs, Events & Photos", "News & Community", and "Watchlist". The main content area displays the movie's poster, which features two surfers riding a large wave. The title "FIND YOUR BREAKING POINT" is overlaid on the poster, with "POINT BREAK" in orange at the bottom. To the right of the poster, the movie's title "Point Break (2015)" is shown in bold, along with its rating of PG-13, runtime of 114 min, genres of Action, Crime, Sport, and release date of 25 December 2015 (USA). A yellow star rating of 5.4 is displayed, along with the number of ratings (5.4/10 from 7,322 users), Metascore (34/100), and reviews from Metacritic.com (60 user | 84 critic | 19). Below the rating, a plot summary states: "A young FBI agent infiltrates an extraordinary team of extreme sports athletes he suspects of masterminding a string of unprecedented, sophisticated corporate heists. 'Point Break' is inspired by the classic 1991 hit." Director Ericson Core, Writers Kurt Wimmer and Rick King, and Stars Edgar Ramírez, Luke Bracey, and Ray Winstone are listed with links to their credits. At the bottom, there are buttons for "+ Watchlist", "Watch Trailer", and "Share...".

Problems for Today

- **Goal:** Predict movie rating automatically!

Find Movies, TV shows, Celebrities and more... All

Movies, TV & Showtimes Celebs, Events & Photos News & Community Watchlist

Point Break (2015) 15

PG-13 | 114 min | 25 December 2015

Predict this automatically!

Our rating: ★★★★★★★★★★★ 5.4 /10

Ratings: 5.4/10 from 7,322 users Metascore: 34/100

Reviews: 60 user | 84 critic | 19 from Metacritic.com

A young FBI agent infiltrates an extraordinary team of extreme sports athletes he suspects of masterminding a string of unprecedented, sophisticated corporate heists. "Point Break" is inspired by the classic 1991 hit.

Director: Ericson Core

Writers: Kurt Wimmer (screenplay), Rick King (story), 5 more credits »

Stars: Édgar Ramírez, Luke Bracey, Ray Winstone | See full cast and crew »

See More on IMDb Pro »

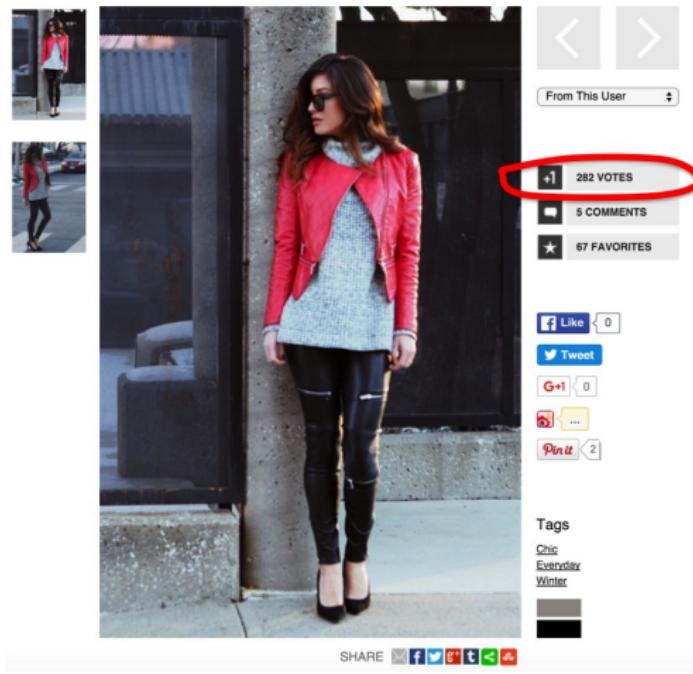
+ Watchlist Watch Trailer Share...

Problems for Today

- **Goal:** How many followers will I get?

Red Leather Jacket

Updated on Jan 09, 2016



Problems for Today

- Goal: Predict the price of the house

The screenshot shows the homepage of the Nationwide House Price Index. At the top, there is a navigation bar with links: Why choose Nationwide?, Have your say, Corporate information, Media, Policy & Legal, House Price Index (which is highlighted in blue), and Investor relations. Below the navigation bar is a large image of several houses. Overlaid on this image is a white rectangular box containing the text "Nationwide" in red and "House Price Index" in large blue letters. Below this box are five buttons: Headlines, House Price calculator, Report archive, Download data, and Methodology. The "House Price calculator" button is highlighted in red. Below these buttons, the text "House Price Calculator" is displayed in red. To the left of the calculator, the word "Instructions" is written in blue. To the right of the calculator, a note in blue text states: "Please note: The Nationwide House Price Calculator is intended to illustrate general movement in prices only. The calculator is based on the Nationwide House Price Index. Results are based on movements in prices in the regions of the UK rather than in specific towns and cities. The data is based on movements in the price of a typical property in the region, and cannot take account of differences in quality of fixtures and fittings". A list of instructions follows:

- Property Value: Enter the price paid for, or a more recent valuation of your property. Please ensure the value is entered without commas, for example 150000, rather than 150,000.
- Valuation Date 1: The date when your property was purchased, or revalued.
- Valuation Date 2: Date for which you would like a new estimate of your property's value.
- Region: Select region which the property is situated in. If you are not sure which region the property is in, click on the link below to find your region.

Regression

- What do all these problems have in common?

Regression

- What do all these problems have in common?
 - ▶ Continuous **outputs**, we'll call these t
(eg, a rating: a real number between 0-10, # of followers, house price)

Regression

- What do all these problems have in common?
 - ▶ Continuous **outputs**, we'll call these t
(eg, a rating: a real number between 0-10, # of followers, house price)
- What do I need in order to predict these outputs?
Predicting continuous outputs is called **regression**

Regression

- What do all these problems have in common?
 - ▶ Continuous **outputs**, we'll call these t
(eg, a rating: a real number between 0-10, # of followers, house price)
- What do I need in order to predict these outputs?
Predicting continuous outputs is called **regression**
 - ▶ **Features** (inputs), we'll call these x (or \mathbf{x} if vectors)

Regression

- What do all these problems have in common?
 - ▶ Continuous **outputs**, we'll call these t
(eg, a rating: a real number between 0-10, # of followers, house price)

- What do I need in order to predict these outputs?

Predicting continuous outputs is called **regression**

- ▶ **Features** (inputs), we'll call these x (or \mathbf{x} if vectors)
- ▶ **Training examples**, many $x^{(i)}$ for which $t^{(i)}$ is known (eg, many movies for which we know the rating)

Regression

- What do all these problems have in common?
 - ▶ Continuous **outputs**, we'll call these t
(eg, a rating: a real number between 0-10, # of followers, house price)
- What do I need in order to predict these outputs?

Predicting continuous outputs is called **regression**

 - ▶ **Features** (inputs), we'll call these x (or \mathbf{x} if vectors)
 - ▶ **Training examples**, many $x^{(i)}$ for which $t^{(i)}$ is known (eg, many movies for which we know the rating)
 - ▶ A **model**, a function that represents the relationship between x and t

Regression

- What do all these problems have in common?
 - ▶ Continuous **outputs**, we'll call these t
(eg, a rating: a real number between 0-10, # of followers, house price)
- What do I need in order to predict these outputs?

Predicting continuous outputs is called **regression**

 - ▶ **Features** (inputs), we'll call these x (or \mathbf{x} if vectors)
 - ▶ **Training examples**, many $x^{(i)}$ for which $t^{(i)}$ is known (eg, many movies for which we know the rating)
 - ▶ A **model**, a function that represents the relationship between x and t
 - ▶ A **loss** or a **cost** or an **objective** function, which tells us how well our model approximates the training examples

Regression

- What do all these problems have in common?
 - ▶ Continuous **outputs**, we'll call these t
(eg, a rating: a real number between 0-10, # of followers, house price)
- What do I need in order to predict these outputs?

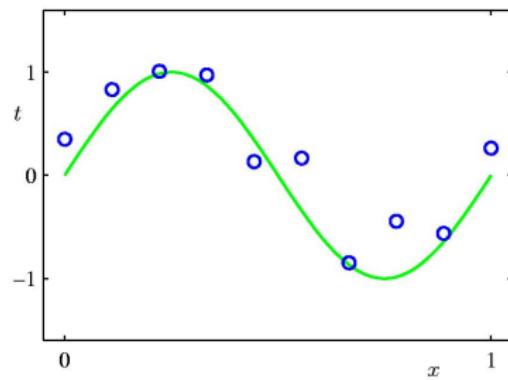
Predicting continuous outputs is called **regression**

 - ▶ **Features** (inputs), we'll call these x (or \mathbf{x} if vectors)
 - ▶ **Training examples**, many $x^{(i)}$ for which $t^{(i)}$ is known (eg, many movies for which we know the rating)
 - ▶ A **model**, a function that represents the relationship between x and t
 - ▶ A **loss** or a **cost** or an **objective** function, which tells us how well our model approximates the training examples
 - ▶ Optimization, a way of finding the parameters of our model that minimizes the loss function

Today: Linear Regression

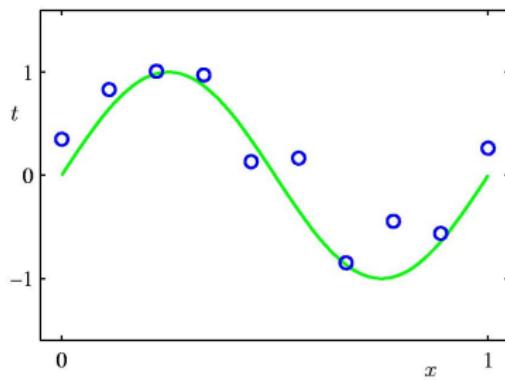
- Linear regression
 - ▶ continuous outputs
 - ▶ simple model (linear)
- Introduce key concepts:
 - ▶ loss functions
 - ▶ generalization
 - ▶ optimization
 - ▶ model complexity
 - ▶ regularization

Simple 1-D regression



- Circles are data points (i.e., training examples) that are given to us

Simple 1-D regression

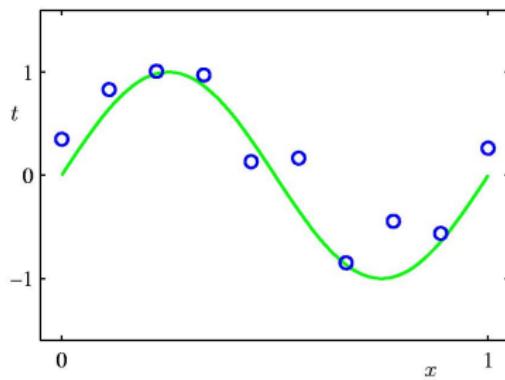


- Circles are data points (i.e., training examples) that are given to us
- The data points are uniform in x , but may be displaced in y

$$t(x) = f(x) + \epsilon$$

with ϵ some noise

Simple 1-D regression



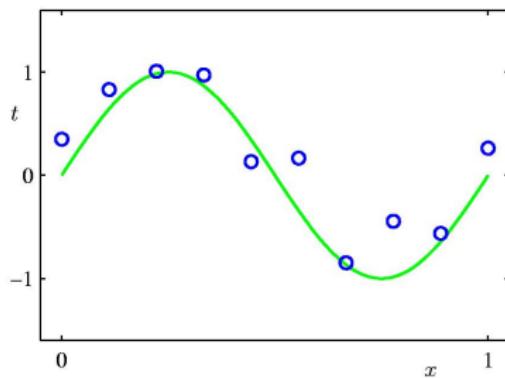
- Circles are data points (i.e., training examples) that are given to us
- The data points are uniform in x , but may be displaced in y

$$t(x) = f(x) + \epsilon$$

with ϵ some noise

- In green is the "true" curve that we don't know

Simple 1-D regression



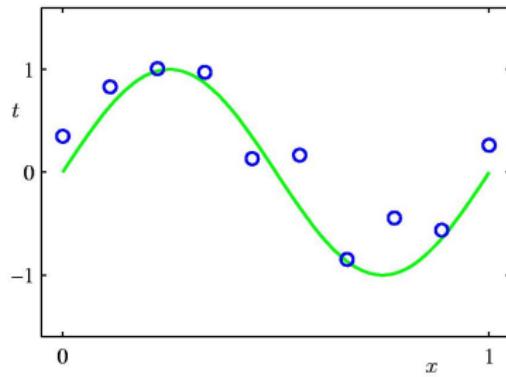
- Circles are data points (i.e., training examples) that are given to us
- The data points are uniform in x , but may be displaced in y

$$t(x) = f(x) + \epsilon$$

with ϵ some noise

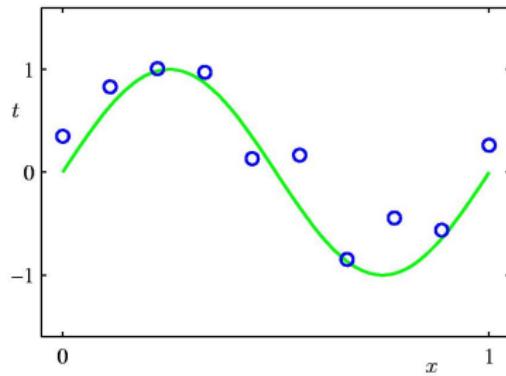
- In green is the "true" curve that we don't know
- Goal: We want to fit a curve to these points

Simple 1-D regression



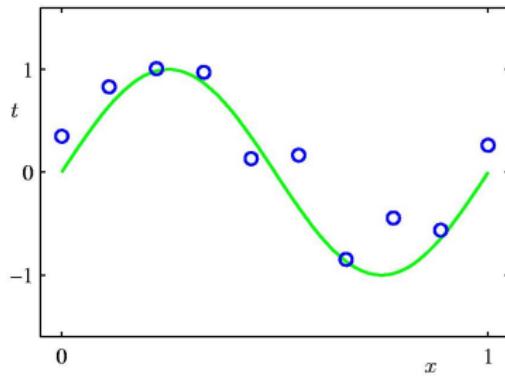
- Key Questions:

Simple 1-D regression



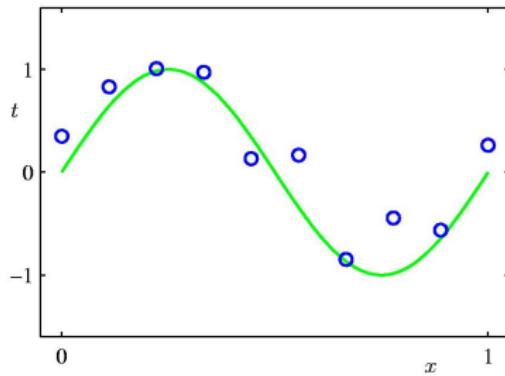
- Key Questions:
 - ▶ How do we parametrize the **model**?

Simple 1-D regression



- Key Questions:
 - ▶ How do we parametrize the model?
 - ▶ What loss (objective) function should we use to judge the fit?

Simple 1-D regression



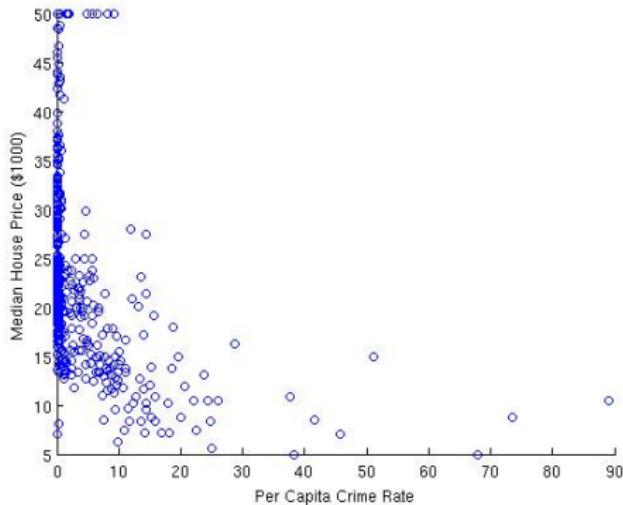
- Key Questions:
 - ▶ How do we parametrize the **model**?
 - ▶ What **loss (objective) function** should we use to judge the fit?
 - ▶ How do we optimize fit to unseen test data (**generalization**)?

Example: Boston Housing data

- Estimate median house price in a neighborhood based on neighborhood statistics

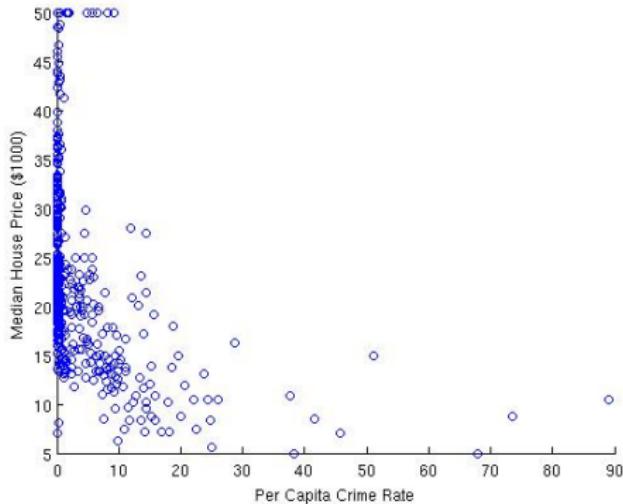
Example: Boston Housing data

- Estimate median house price in a neighborhood based on neighborhood statistics
- Look at first possible attribute (feature): per capita crime rate



Example: Boston Housing data

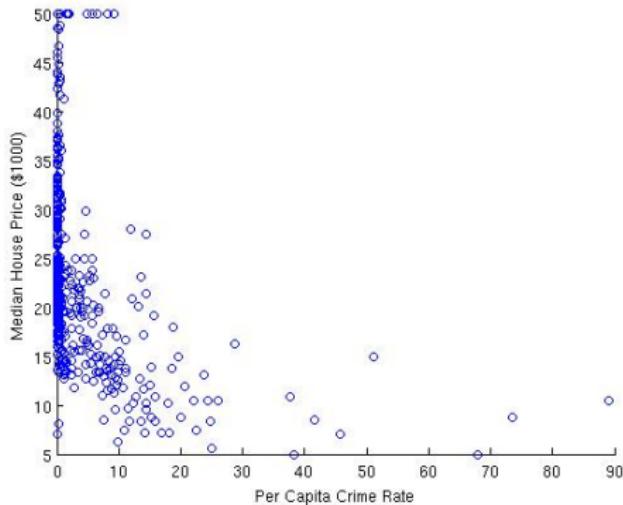
- Estimate median house price in a neighborhood based on neighborhood statistics
- Look at first possible attribute (feature): per capita crime rate



- Use this to predict house prices in other neighborhoods

Example: Boston Housing data

- Estimate median house price in a neighborhood based on neighborhood statistics
- Look at first possible attribute (feature): per capita crime rate



- Use this to predict house prices in other neighborhoods
- Is this a good input (attribute) to predict house prices?

Represent the Data

- Data is described as pairs $\mathcal{D} = \{(x^{(1)}, t^{(1)}), \dots, (x^{(N)}, t^{(N)})\}$
 - ▶ $x \in \mathbb{R}$ is the **input feature** (per capita crime rate)
 - ▶ $t \in \mathbb{R}$ is the **target output** (median house price)
 - ▶ (i) simply indicates the training examples (we have N in this case)

Represent the Data

- Data is described as pairs $\mathcal{D} = \{(x^{(1)}, t^{(1)}), \dots, (x^{(N)}, t^{(N)})\}$
 - ▶ $x \in \mathbb{R}$ is the **input feature** (per capita crime rate)
 - ▶ $t \in \mathbb{R}$ is the **target output** (median house price)
 - ▶ (i) simply indicates the training examples (we have N in this case)
- Here t is continuous, so this is a **regression problem**

Represent the Data

- Data is described as pairs $\mathcal{D} = \{(x^{(1)}, t^{(1)}), \dots, (x^{(N)}, t^{(N)})\}$
 - ▶ $x \in \mathbb{R}$ is the **input feature** (per capita crime rate)
 - ▶ $t \in \mathbb{R}$ is the **target output** (median house price)
 - ▶ (i) simply indicates the training examples (we have N in this case)
- Here t is continuous, so this is a **regression problem**
- Model outputs y , an estimate of t

$$y(x) = w_0 + w_1 x$$

Represent the Data

- Data is described as pairs $\mathcal{D} = \{(x^{(1)}, t^{(1)}), \dots, (x^{(N)}, t^{(N)})\}$
 - ▶ $x \in \mathbb{R}$ is the **input feature** (per capita crime rate)
 - ▶ $t \in \mathbb{R}$ is the **target output** (median house price)
 - ▶ (i) simply indicates the training examples (we have N in this case)
- Here t is continuous, so this is a **regression problem**
- Model outputs y , an estimate of t

$$y(x) = w_0 + w_1 x$$

- What type of **model** did we choose?

Represent the Data

- Data is described as pairs $\mathcal{D} = \{(x^{(1)}, t^{(1)}), \dots, (x^{(N)}, t^{(N)})\}$
 - ▶ $x \in \mathbb{R}$ is the **input feature** (per capita crime rate)
 - ▶ $t \in \mathbb{R}$ is the **target output** (median house price)
 - ▶ (i) simply indicates the training examples (we have N in this case)
- Here t is continuous, so this is a **regression problem**
- Model outputs y , an estimate of t

$$y(x) = w_0 + w_1 x$$

- What type of **model** did we choose?
- Divide the dataset into training and testing examples

Represent the Data

- Data is described as pairs $\mathcal{D} = \{(x^{(1)}, t^{(1)}), \dots, (x^{(N)}, t^{(N)})\}$
 - ▶ $x \in \mathbb{R}$ is the **input feature** (per capita crime rate)
 - ▶ $t \in \mathbb{R}$ is the **target output** (median house price)
 - ▶ (i) simply indicates the training examples (we have N in this case)
- Here t is continuous, so this is a **regression problem**
- Model outputs y , an estimate of t

$$y(x) = w_0 + w_1 x$$

- What type of **model** did we choose?
- Divide the dataset into training and testing examples
 - ▶ Use the training examples to construct hypothesis, or function approximator, that maps x to predicted y

Represent the Data

- Data is described as pairs $\mathcal{D} = \{(x^{(1)}, t^{(1)}), \dots, (x^{(N)}, t^{(N)})\}$
 - ▶ $x \in \mathbb{R}$ is the **input feature** (per capita crime rate)
 - ▶ $t \in \mathbb{R}$ is the **target output** (median house price)
 - ▶ (i) simply indicates the training examples (we have N in this case)
- Here t is continuous, so this is a **regression problem**
- Model outputs y , an estimate of t

$$y(x) = w_0 + w_1 x$$

- What type of **model** did we choose?
- Divide the dataset into training and testing examples
 - ▶ Use the training examples to construct hypothesis, or function approximator, that maps x to predicted y
 - ▶ Evaluate hypothesis on test set

Noise

- A simple model typically does not exactly fit the data – lack of fit can be considered noise

Noise

- A simple model typically does not exactly fit the data – lack of fit can be considered noise
- Sources of noise:

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, eg noise in per-capita crime)

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, eg noise in per-capita crime)
 - ▶ Errors in data targets (mis-labeling, eg noise in house prices)

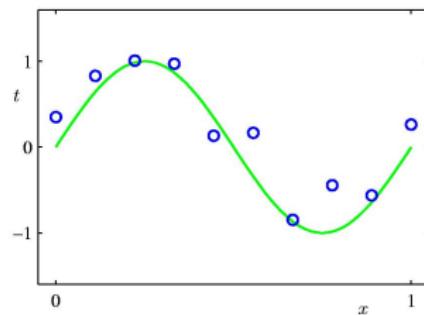
Noise

- 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, eg noise in per-capita crime)
 - ▶ Errors in data targets (mis-labeling, eg noise in house prices)
 - ▶ Additional attributes not taken into account by data attributes, affect target values (latent variables). In the example, what else could affect house prices?

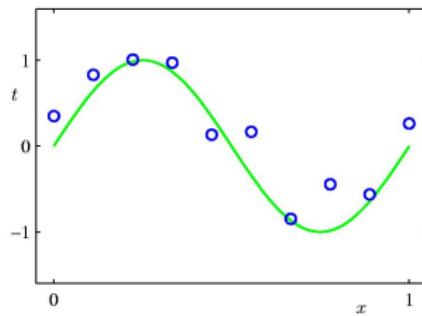
Noise

- 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, eg noise in per-capita crime)
 - ▶ Errors in data targets (mis-labeling, eg noise in house prices)
 - ▶ Additional attributes not taken into account by data attributes, affect target values (latent variables). In the example, what else could affect house prices?
 - ▶ Model may be too simple to account for data targets

Least-Squares Regression



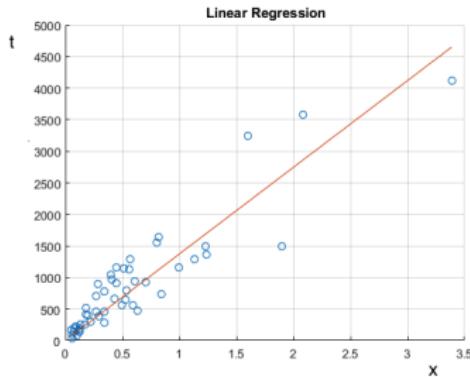
Least-Squares Regression



- Define a model

$$y(x) = \text{function}(x, \mathbf{w})$$

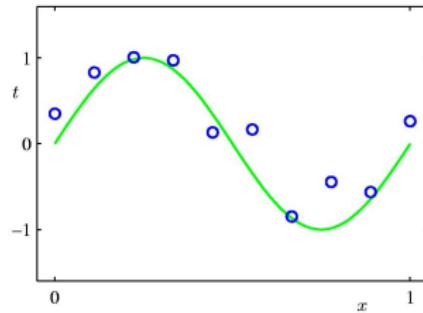
Least-Squares Regression



- Define a model

Linear: $y(x) = w_0 + w_1 x$

Least-Squares Regression



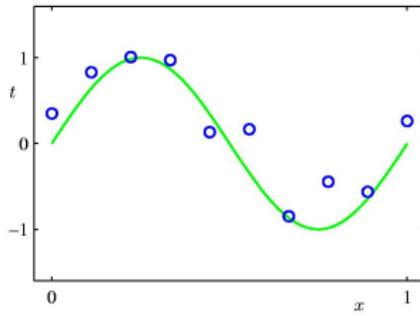
- Define a model

Linear: $y(x) = w_0 + w_1 x$

- Standard loss/cost/objective function measures the squared error between y and the true value t

$$\ell(\mathbf{w}) = \sum_{n=1}^N [t^{(n)} - y(x^{(n)})]^2$$

Least-Squares Regression



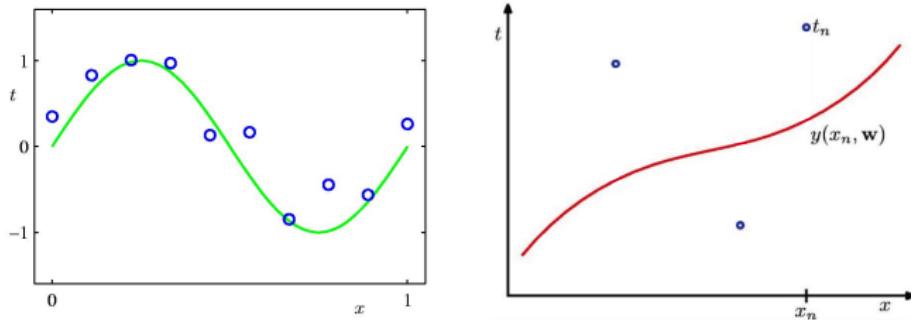
- Define a model

Linear: $y(x) = w_0 + w_1 x$

- Standard loss/cost/objective function measures the squared error between y and the true value t

Linear model: $\ell(\mathbf{w}) = \sum_{n=1}^N [t^{(n)} - (w_0 + w_1 x^{(n)})]^2$

Least-Squares Regression



- Define a model

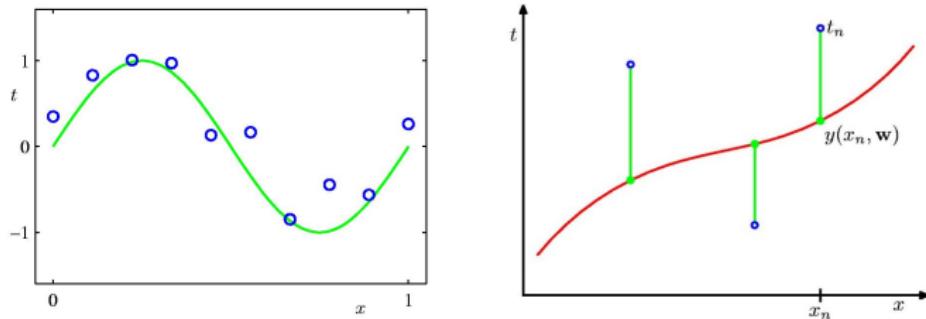
Linear: $y(x) = w_0 + w_1 x$

- Standard loss/cost/objective function measures the squared error between y and the true value t

Linear model: $\ell(\mathbf{w}) = \sum_{n=1}^N [t^{(n)} - (w_0 + w_1 x^{(n)})]^2$

- For a particular hypothesis ($y(x)$ defined by a choice of \mathbf{w} , drawn in red), what does the loss represent geometrically?

Least-Squares Regression



- Define a model

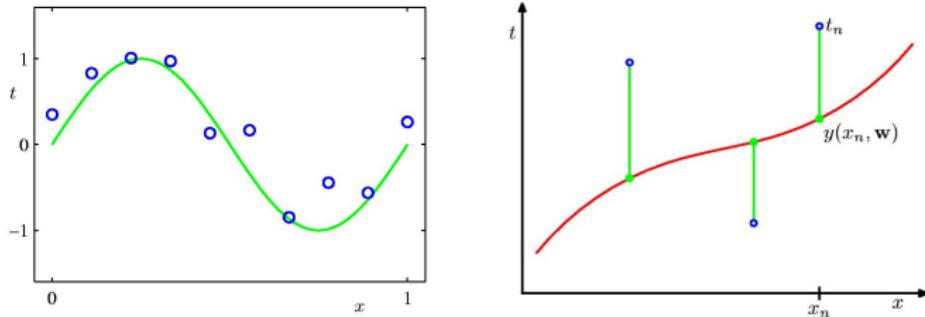
Linear: $y(x) = w_0 + w_1 x$

- Standard loss/cost/objective function measures the squared error between y and the true value t

Linear model: $\ell(\mathbf{w}) = \sum_{n=1}^N [t^{(n)} - (w_0 + w_1 x^{(n)})]^2$

- The loss for the red hypothesis is the **sum of the squared vertical errors** (squared lengths of green vertical lines)

Least-Squares Regression



- Define a model

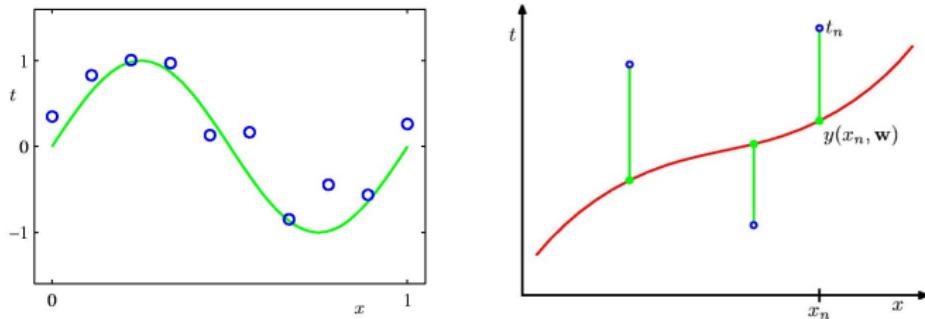
Linear: $y(x) = w_0 + w_1 x$

- Standard loss/cost/objective function measures the squared error between y and the true value t

Linear model: $\ell(\mathbf{w}) = \sum_{n=1}^N [t^{(n)} - (w_0 + w_1 x^{(n)})]^2$

- How do we obtain weights $\mathbf{w} = (w_0, w_1)$?

Least-Squares Regression



- Define a model

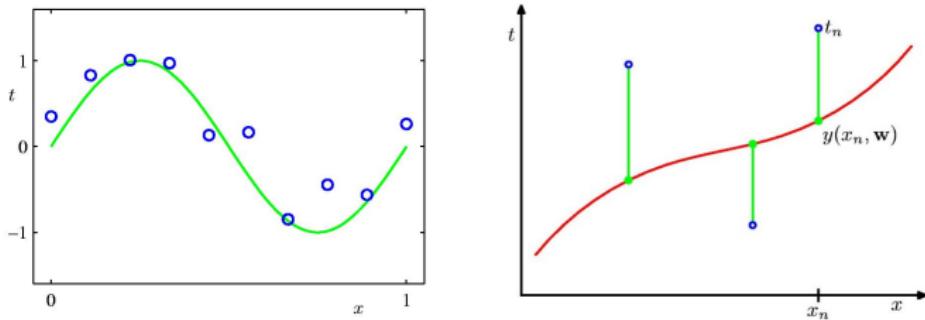
Linear: $y(x) = w_0 + w_1 x$

- Standard loss/cost/objective function measures the squared error between y and the true value t

Linear model: $\ell(\mathbf{w}) = \sum_{n=1}^N [t^{(n)} - (w_0 + w_1 x^{(n)})]^2$

- How do we obtain weights $\mathbf{w} = (w_0, w_1)$? Find \mathbf{w} that minimizes loss $\ell(\mathbf{w})$

Least-Squares Regression



- Define a model

Linear: $y(x) = w_0 + w_1 x$

- Standard loss/cost/objective function measures the squared error between y and the true value t

Linear model: $\ell(\mathbf{w}) = \sum_{n=1}^N [t^{(n)} - (w_0 + w_1 x^{(n)})]^2$

- How do we obtain weights $\mathbf{w} = (w_0, w_1)$?
- For the linear model, what kind of a function is $\ell(\mathbf{w})$?

Optimizing the Objective

- One straightforward method: [gradient descent](#)

Optimizing the Objective

- One straightforward method: [gradient descent](#)
 - ▶ initialize \mathbf{w} (e.g., randomly)

Optimizing the Objective

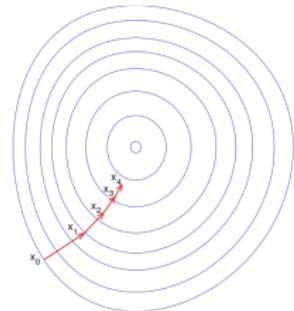
- One straightforward method: [gradient descent](#)
 - ▶ initialize \mathbf{w} (e.g., randomly)
 - ▶ repeatedly update \mathbf{w} based on the gradient

$$\mathbf{w} \leftarrow \mathbf{w} - \lambda \frac{\partial \ell}{\partial \mathbf{w}}$$

Optimizing the Objective

- One straightforward method: [gradient descent](#)
 - initialize \mathbf{w} (e.g., randomly)
 - repeatedly update \mathbf{w} based on the gradient

$$\mathbf{w} \leftarrow \mathbf{w} - \lambda \frac{\partial \ell}{\partial \mathbf{w}}$$

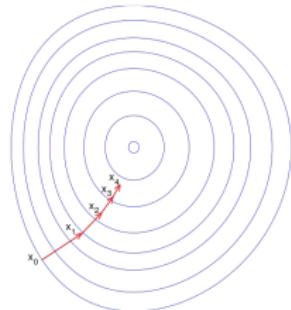


Optimizing the Objective

- One straightforward method: [gradient descent](#)
 - initialize \mathbf{w} (e.g., randomly)
 - repeatedly update \mathbf{w} based on the gradient

$$\mathbf{w} \leftarrow \mathbf{w} - \lambda \frac{\partial \ell}{\partial \mathbf{w}}$$

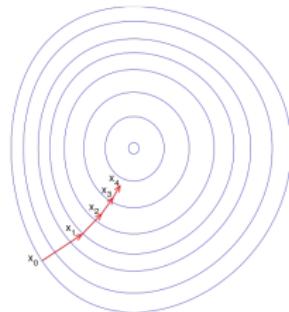
- λ is the [learning rate](#)



Optimizing the Objective

- One straightforward method: [gradient descent](#)
 - initialize \mathbf{w} (e.g., randomly)
 - repeatedly update \mathbf{w} based on the gradient

$$\mathbf{w} \leftarrow \mathbf{w} - \lambda \frac{\partial \ell}{\partial \mathbf{w}}$$



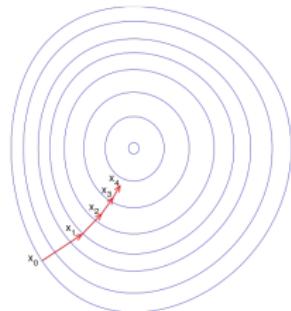
- λ is the [learning rate](#)
- For a single training case, this gives the [LMS update rule](#):

$$\mathbf{w} \leftarrow \mathbf{w} + 2\lambda(t^{(n)} - y(x^{(n)}))x^{(n)}$$

Optimizing the Objective

- One straightforward method: **gradient descent**
 - initialize \mathbf{w} (e.g., randomly)
 - repeatedly update \mathbf{w} based on the gradient

$$\mathbf{w} \leftarrow \mathbf{w} - \lambda \frac{\partial \ell}{\partial \mathbf{w}}$$



- λ is the **learning rate**
- For a single training case, this gives the **LMS update rule**:

$$\mathbf{w} \leftarrow \mathbf{w} + 2\lambda \underbrace{(t^{(n)} - y(x^{(n)}))}_{\text{error}} x^{(n)}$$

- Note: As error approaches zero, so does the update (\mathbf{w} stops changing)

Optimizing Across Training Set

- Two ways to generalize this for all examples in training set:

Optimizing Across Training Set

- Two ways to generalize this for all examples in training set:
 1. **Batch updates**: sum or average updates across every example n , then change the parameter values

$$\mathbf{w} \leftarrow \mathbf{w} + 2\lambda \sum_{n=1}^N (t^{(n)} - y(x^{(n)}))x^{(n)}$$

Optimizing Across Training Set

- Two ways to generalize this for all examples in training set:
 1. **Batch updates**: sum or average updates across every example n , then change the parameter values

$$\mathbf{w} \leftarrow \mathbf{w} + 2\lambda \sum_{n=1}^N (t^{(n)} - y(x^{(n)}))x^{(n)}$$

2. **Stochastic/online updates**: update the parameters for each training case in turn, according to its own gradients

Algorithm 1 Stochastic gradient descent

- 1: Randomly shuffle examples in the training set
- 2: **for** $i = 1$ to N **do**
- 3: Update:

$$\mathbf{w} \leftarrow \mathbf{w} + 2\lambda(t^{(i)} - y(x^{(i)}))x^{(i)} \quad (\text{update for a linear model})$$

- 4: **end for**

Optimizing Across Training Set

- Two ways to generalize this for all examples in training set:
 1. **Batch updates**: sum or average updates across every example n , then change the parameter values

$$\mathbf{w} \leftarrow \mathbf{w} + 2\lambda \sum_{n=1}^N (t^{(n)} - y(x^{(n)}))x^{(n)}$$

2. **Stochastic/online updates**: update the parameters for each training case in turn, according to its own gradients
 - ▶ Underlying assumption: sample is independent and identically distributed (i.i.d.)

Analytical Solution?

- For some objectives we can also find the optimal solution analytically
- This is the case for linear least-squares regression
- How?

Analytical Solution?

- For some objectives we can also find the optimal solution analytically
- This is the case for linear least-squares regression
- How?
- Compute the derivatives of the objective wrt \mathbf{w} and equate with 0

Analytical Solution?

- For some objectives we can also find the optimal solution analytically
- This is the case for linear least-squares regression
- How?
- Compute the derivatives of the objective wrt \mathbf{w} and equate with 0
- Define:

$$\mathbf{t} = [t^{(1)}, t^{(2)}, \dots, t^{(N)}]^T$$
$$\mathbf{x} = \begin{bmatrix} 1, x^{(1)} \\ 1, x^{(2)} \\ \vdots \\ 1, x^{(N)} \end{bmatrix}$$

- Then:

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

(work it out!)

Multi-dimensional Inputs

- One method of extending the model is to consider other input dimensions

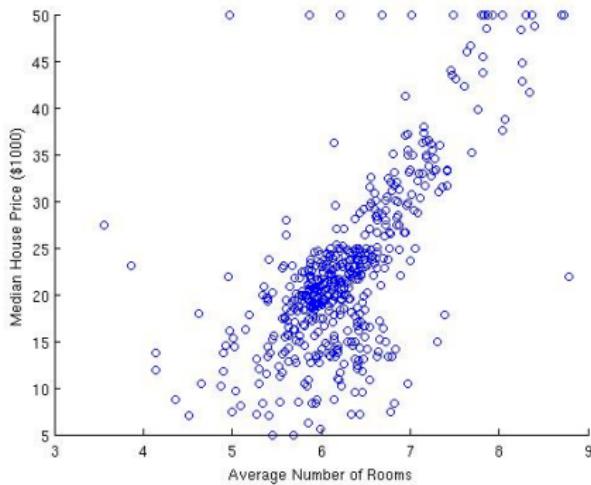
$$y(\mathbf{x}) = w_0 + w_1x_1 + w_2x_2$$

Multi-dimensional Inputs

- One method of extending the model is to consider other input dimensions

$$y(\mathbf{x}) = w_0 + w_1x_1 + w_2x_2$$

- In the Boston housing example, we can look at the number of rooms



Linear Regression with Multi-dimensional Inputs

- Imagine now we want to predict the median house price from these multi-dimensional observations

Linear Regression with Multi-dimensional Inputs

- Imagine now we want to predict the median house price from these multi-dimensional observations
- Each house is a data point n , with observations indexed by j :

$$\mathbf{x}^{(n)} = \left(x_1^{(n)}, \dots, x_j^{(n)}, \dots, x_d^{(n)} \right)$$

Linear Regression with Multi-dimensional Inputs

- Imagine now we want to predict the median house price from these multi-dimensional observations
- Each house is a data point n , with observations indexed by j :

$$\mathbf{x}^{(n)} = \left(x_1^{(n)}, \dots, x_j^{(n)}, \dots, x_d^{(n)} \right)$$

- We can incorporate the bias w_0 into \mathbf{w} , by using $x_0 = 1$, then

$$y(\mathbf{x}) = w_0 + \sum_{j=1}^d w_j x_j = \mathbf{w}^T \mathbf{x}$$

Linear Regression with Multi-dimensional Inputs

- Imagine now we want to predict the median house price from these multi-dimensional observations
- Each house is a data point n , with observations indexed by j :

$$\mathbf{x}^{(n)} = \left(x_1^{(n)}, \dots, x_j^{(n)}, \dots, x_d^{(n)} \right)$$

- We can incorporate the bias w_0 into \mathbf{w} , by using $x_0 = 1$, then

$$y(\mathbf{x}) = w_0 + \sum_{j=1}^d w_j x_j = \mathbf{w}^T \mathbf{x}$$

- We can then solve for $\mathbf{w} = (w_0, w_1, \dots, w_d)$. How?

Linear Regression with Multi-dimensional Inputs

- Imagine now we want to predict the median house price from these multi-dimensional observations
- Each house is a data point n , with observations indexed by j :

$$\mathbf{x}^{(n)} = \left(x_1^{(n)}, \dots, x_j^{(n)}, \dots, x_d^{(n)} \right)$$

- We can incorporate the bias w_0 into \mathbf{w} , by using $x_0 = 1$, then

$$y(\mathbf{x}) = w_0 + \sum_{j=1}^d w_j x_j = \mathbf{w}^T \mathbf{x}$$

- We can then solve for $\mathbf{w} = (w_0, w_1, \dots, w_d)$. How?
- We can use gradient descent to solve for each coefficient, or compute \mathbf{w} analytically (how does the solution change?)

More Powerful Models?

- What if our linear model is not good? How can we create a more complicated model?

Fitting a Polynomial

- What if our linear model is not good? How can we create a more complicated model?
- We can create a more complicated model by defining input variables that are combinations of components of x

Fitting a Polynomial

- What if our linear model is not good? How can we create a more complicated model?
- We can create a more complicated model by defining input variables that are combinations of components of \mathbf{x}
- Example: an M -th order polynomial function of one dimensional feature x :

$$y(x, \mathbf{w}) = w_0 + \sum_{j=1}^M w_j x^j$$

where x^j is the j -th power of x

Fitting a Polynomial

- What if our linear model is not good? How can we create a more complicated model?
- We can create a more complicated model by defining input variables that are combinations of components of \mathbf{x}
- Example: an M -th order polynomial function of one dimensional feature x :

$$y(x, \mathbf{w}) = w_0 + \sum_{j=1}^M w_j x^j$$

where x^j is the j -th power of x

- We can use the same approach to optimize for the weights \mathbf{w}

Fitting a Polynomial

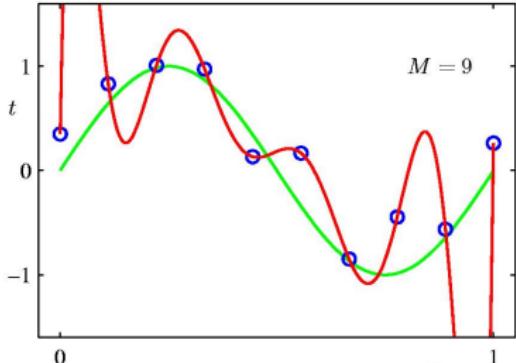
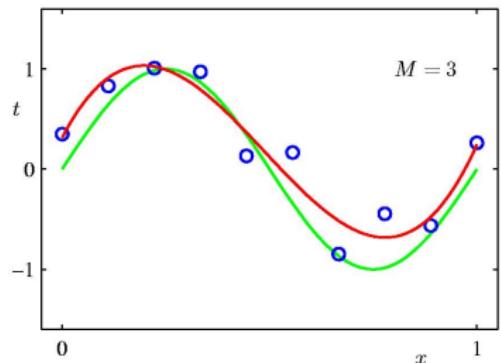
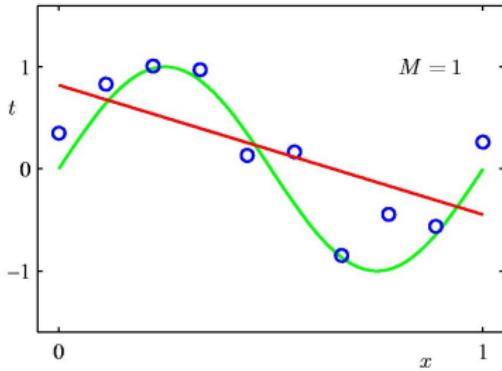
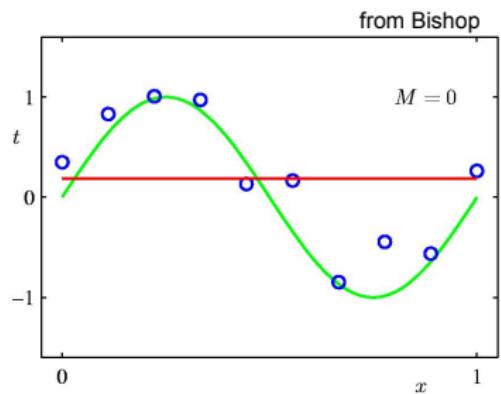
- What if our linear model is not good? How can we create a more complicated model?
- We can create a more complicated model by defining input variables that are combinations of components of \mathbf{x}
- Example: an M -th order polynomial function of one dimensional feature x :

$$y(x, \mathbf{w}) = w_0 + \sum_{j=1}^M w_j x^j$$

where x^j is the j -th power of x

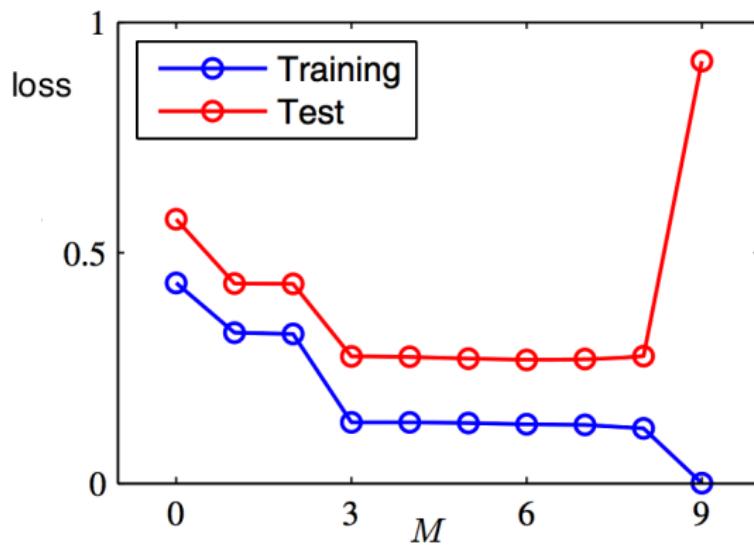
- We can use the same approach to optimize for the weights \mathbf{w}
- How do we do that?

Which Fit is Best?



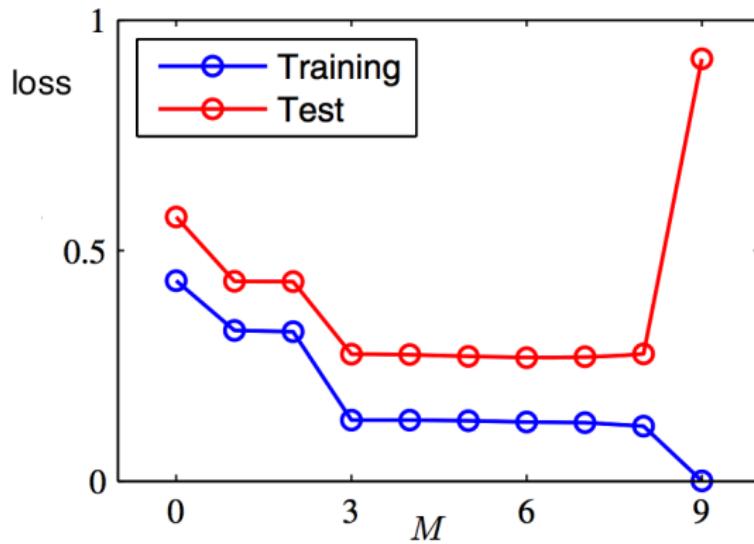
Generalization

- Generalization = model's ability to predict the held out data
- What is happening?



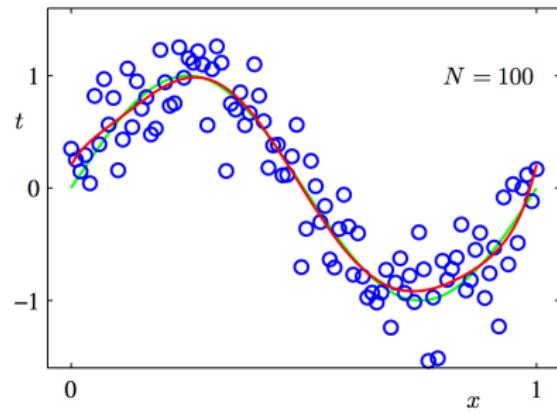
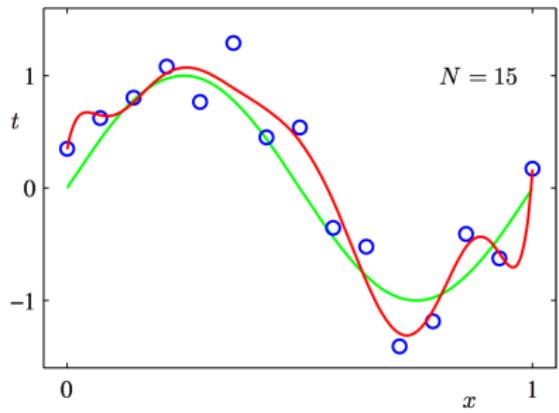
Generalization

- Generalization = model's ability to predict the held out data
- What is happening?
- Our model with $M = 9$ overfits the data (it models also noise)



Generalization

- Generalization = model's ability to predict the held out data
- What is happening?
- Our model with $M = 9$ overfits the data (it models also noise)
- Not a problem if we have lots of training examples



Generalization

- Generalization = model's ability to predict the held out data
- What is happening?
- Our model with $M = 9$ overfits the data (it models also noise)
- Let's look at the estimated weights for various M in the case of fewer examples

	$M = 0$	$M = 1$	$M = 6$	$M = 9$
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

Generalization

- Generalization = model's ability to predict the held out data
- What is happening?
- Our model with $M = 9$ overfits the data (it models also noise)
- Let's look at the estimated weights for various M in the case of fewer examples
- The weights are becoming huge to compensate for the noise

Generalization

- Generalization = model's ability to predict the held out data
- What is happening?
- Our model with $M = 9$ overfits the data (it models also noise)
- Let's look at the estimated weights for various M in the case of fewer examples
- The weights are becoming huge to compensate for the noise
- One way of dealing with this is to encourage the weights to be small (this way no input dimension will have too much influence on prediction). This is called regularization.

Regularized Least Squares

- Increasing the input features this way can complicate the model considerably

Regularized Least Squares

- Increasing the input features this way can complicate the model considerably
- **Goal:** select the appropriate model complexity automatically

Regularized Least Squares

- Increasing the input features this way can complicate the model considerably
- Goal: select the appropriate model complexity automatically
- Standard approach: regularization

$$\tilde{\ell}(\mathbf{w}) = \sum_{n=1}^N [t^{(n)} - (w_0 + w_1 x^{(n)})]^2 + \alpha \mathbf{w}^T \mathbf{w}$$

Regularized Least Squares

- Increasing the input features this way can complicate the model considerably
- Goal: select the appropriate model complexity automatically
- Standard approach: regularization

$$\tilde{\ell}(\mathbf{w}) = \sum_{n=1}^N [t^{(n)} - (w_0 + w_1 x^{(n)})]^2 + \alpha \mathbf{w}^T \mathbf{w}$$

- Intuition: Since we are minimizing the loss, the second term will encourage smaller values in \mathbf{w}

Regularized Least Squares

- Increasing the input features this way can complicate the model considerably
- Goal: select the appropriate model complexity automatically
- Standard approach: regularization

$$\tilde{\ell}(\mathbf{w}) = \sum_{n=1}^N [t^{(n)} - (w_0 + w_1 x^{(n)})]^2 + \alpha \mathbf{w}^T \mathbf{w}$$

- Intuition: Since we are minimizing the loss, the second term will encourage smaller values in \mathbf{w}
- The penalty on the squared weights is known as ridge regression in statistics

Regularized Least Squares

- Increasing the input features this way can complicate the model considerably
- Goal: select the appropriate model complexity automatically
- Standard approach: regularization

$$\tilde{\ell}(\mathbf{w}) = \sum_{n=1}^N [t^{(n)} - (w_0 + w_1 x^{(n)})]^2 + \alpha \mathbf{w}^T \mathbf{w}$$

- Intuition: Since we are minimizing the loss, the second term will encourage smaller values in \mathbf{w}
- The penalty on the squared weights is known as ridge regression in statistics
- Leads to a modified update rule for gradient descent:

$$\mathbf{w} \leftarrow \mathbf{w} + 2\lambda \left[\sum_{n=1}^N (t^{(n)} - y(x^{(n)}))x^{(n)} - \alpha \mathbf{w} \right]$$

Regularized Least Squares

- Increasing the input features this way can complicate the model considerably
- Goal: select the appropriate model complexity automatically
- Standard approach: regularization

$$\tilde{\ell}(\mathbf{w}) = \sum_{n=1}^N [t^{(n)} - (w_0 + w_1 x^{(n)})]^2 + \alpha \mathbf{w}^T \mathbf{w}$$

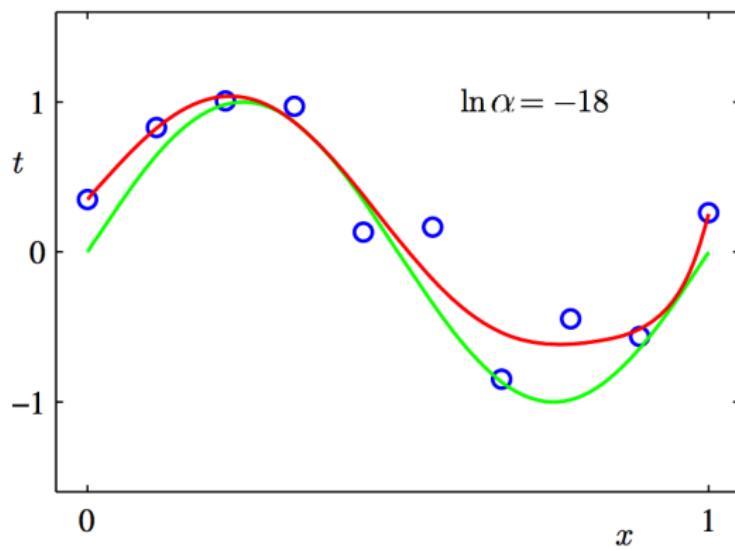
- Intuition: Since we are minimizing the loss, the second term will encourage smaller values in \mathbf{w}
- The penalty on the squared weights is known as ridge regression in statistics
- Leads to a modified update rule for gradient descent:

$$\mathbf{w} \leftarrow \mathbf{w} + 2\lambda \left[\sum_{n=1}^N (t^{(n)} - y(x^{(n)}))x^{(n)} - \alpha \mathbf{w} \right]$$

- Also has an analytical solution: $\mathbf{w} = (\mathbf{X}^T \mathbf{X} + \alpha \mathbf{I})^{-1} \mathbf{X}^T \mathbf{t}$ (verify!)

Regularized least squares

- Better generalization
- Choose α carefully



1-D regression illustrates key concepts

- Data fits – is linear model best ([model selection](#))?

1-D regression illustrates key concepts

- Data fits – is linear model best ([model selection](#))?
 - ▶ Simple models may not capture all the important variations ([signal](#)) in the data: [underfit](#)

1-D regression illustrates key concepts

- Data fits – is linear model best ([model selection](#))?
 - ▶ Simple models may not capture all the important variations ([signal](#)) in the data: [underfit](#)
 - ▶ More complex models may [overfit](#) the training data (fit not only the signal but also the [noise](#) in the data), especially if not enough data to constrain model

1-D regression illustrates key concepts

- Data fits – is linear model best ([model selection](#))?
 - ▶ Simple models may not capture all the important variations ([signal](#)) in the data: [underfit](#)
 - ▶ More complex models may [overfit](#) the training data (fit not only the signal but also the [noise](#) in the data), especially if not enough data to constrain model
- One method of assessing fit: test [generalization](#) = model's ability to predict the held out data

1-D regression illustrates key concepts

- Data fits – is linear model best ([model selection](#))?
 - ▶ Simple models may not capture all the important variations ([signal](#)) in the data: [underfit](#)
 - ▶ More complex models may [overfit](#) the training data (fit not only the signal but also the [noise](#) in the data), especially if not enough data to constrain model
- One method of assessing fit: test [generalization](#) = model's ability to predict the held out data
- Optimization is essential: stochastic and batch iterative approaches; analytic when available

So...

- Which movie will you watch?



Now Playing

REFINE YOUR SEARCH



Alvin And The Chipmunks: The Road Chip

1h 30m | Comedy, Family

[View Ratings and Warnings](#)

[BUY TICKETS](#)

[TRAILER](#)



Anomalis

1h 31m | Comedy, Animation, Fantasy

[View Ratings and Warnings](#)

[BUY TICKETS](#)

[TRAILER](#)



Baigao Mastani (Hindi w/e.t.)

2h 30m | Foreign Language, Drama, Romance, History

[View Ratings and Warnings](#)

[BUY TICKETS](#)



Prince Caspian (Disney)

1h 55m | Action, Foreign Language, Comedy

[View Ratings and Warnings](#)

[BUY TICKETS](#)



Brooklyn

1h 52m | Drama

[View Ratings and Warnings](#)

[BUY TICKETS](#)

[TRAILER](#)