

# CSC 411: Lecture 02: Linear Regression


Richard Zemel, Raquel Urtasun and Sanja Fidler

University of Toronto


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

# Problems for Today

- What should I watch this Friday?

All

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



**The Martian** (2015)

PG-13 | 144 min | [Adventure, Comedy, Drama](#) | [2 October 2015 \(USA\)](#)

**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...](#)

[See More on IMDb Pro »](#)

# Problems for Today

- What should I watch this Friday?



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

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

## Point Break (2015)

PG-13 | 114 min | Action, Crime, Sport | 25 December 2015 (USA)

**Your rating:** ★★★★★★ -/10  
Ratings: 5.4/10 from 7,322 users Metascore: 34/100  
Reviews: 60 user | 84 critic | 19 from Metacritic.com

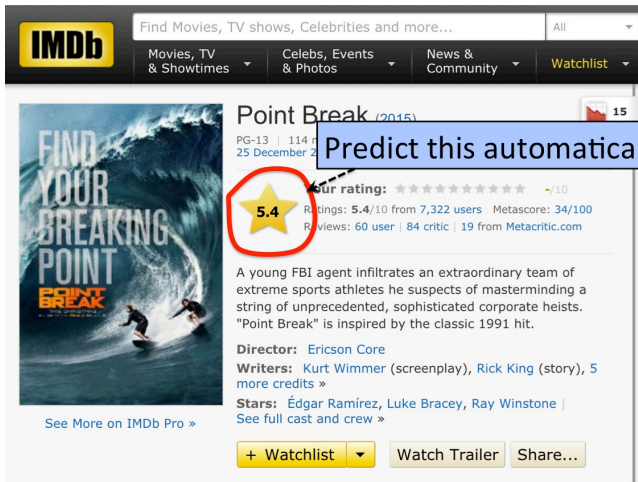
A young FBI agent infiltrates an extraordinary team of extreme sports athletes he suspects of masterminding a string of unprecedented, sophisticated corporate heists. "Point Break" is inspired by the classic 1991 hit.

**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](#) »

[+ Watchlist](#) [Watch Trailer](#) [Share...](#)

# Problems for Today

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



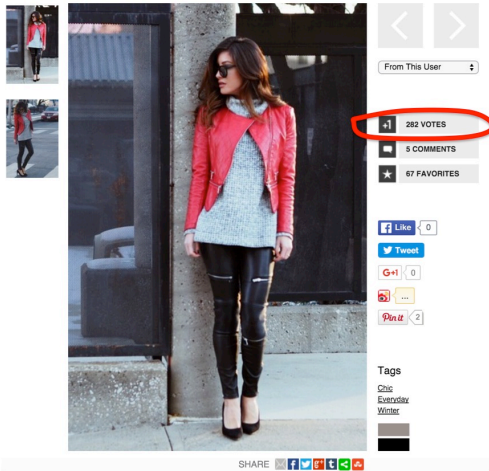
The image shows a screenshot of the IMDb website for the movie "Point Break (2015)". The IMDb logo is in the top left. A search bar contains the text "Find Movies, TV shows, Celebrities and more...". Below the search bar are navigation tabs: "Movies, TV & Showtimes", "Celebs, Events & Photos", "News & Community", and "Watchlist". The movie poster for "Point Break" is on the left, with the text "FIND YOUR BREAKING POINT" and "CRIME BREAK". The movie title "Point Break (2015)" is displayed, along with its rating "PG-13" and release date "25 December 2015". A callout box with a blue background and white text says "Predict this automatically!" with a dashed arrow pointing to a yellow star containing the number "5.4". Below the star, the text reads "Our rating: ★★★★★ -/10". Further down, it says "Ratings: 5.4/10 from 7,322 users" and "Metascore: 34/100". Below that, it says "Reviews: 60 user | 84 critic | 19 from Metacritic.com". A short synopsis follows: "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." Credits for Director (Ericson Core), Writers (Kurt Wimmer, Rick King), and Stars (Édgar Ramírez, Luke Bracey, Ray Winstone) are listed. At the bottom, there are buttons for "+ Watchlist", "Watch Trailer", and "Share...".

# Problems for Today

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

Red Leather Jacket

Updated on Jan 09, 2016



The image shows a fashion blog post. The main image is a woman standing outdoors, wearing a red leather jacket, a grey textured sweater, and black leggings. To the left of the main image are two smaller thumbnail images of the same woman in the same outfit. To the right of the main image is a sidebar with social media sharing options and engagement metrics. The engagement metrics are: +1 (circled in red), 282 VOTES, 5 COMMENTS, and 67 FAVORITES. Below these are social media sharing buttons for Facebook (Like 0), Twitter (Tweet), Google+ (0), and Pinterest (2). At the bottom of the sidebar are the tags: Chic, Everyday, and Winter. At the bottom of the main image area is a 'SHARE' button with icons for Facebook, Twitter, Google+, and LinkedIn.

# Problems for Today

- **Goal:** Predict the price of the house

Why choose Nationwide? | Have your say | Corporate information | Media, Policy & Legal | **House Price Index** | Investor relations

## Nationwide House Price Index

Headlines | **House Price calculator** | Report archive | Download data | Methodology

### House Price Calculator

#### Instructions

- 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.

**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 fittings.

# Regression

- What do all these problems have in common?

# Regression

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

# Regression

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

# Regression

- What do all these problems have in common?
  - ▶ Continuous **outputs**, we'll call these  $t$   
(e.g., a rating: a real number between 0-10, # of followers, house price)
- Predicting continuous outputs is called **regression**
- What do I need in order to **predict** these outputs?
  - ▶ **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  $t$   
(e.g., a rating: a real number between 0-10, # of followers, house price)
- Predicting continuous outputs is called **regression**
- What do I need in order to **predict** these outputs?
  - ▶ **Features** (inputs), we'll call these  $x$  (or  $\mathbf{x}$  if vectors)
  - ▶ **Training examples**, many  $x^{(i)}$  for which  $t^{(i)}$  is known (e.g., many movies for which we know the rating)

# Regression

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

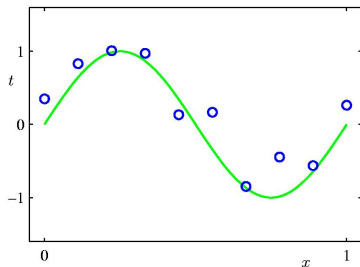
- What do all these problems have in common?
  - ▶ Continuous **outputs**, we'll call these  $t$   
(e.g., a rating: a real number between 0-10, # of followers, house price)
- Predicting continuous outputs is called **regression**
- What do I need in order to **predict** these outputs?
  - ▶ **Features**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

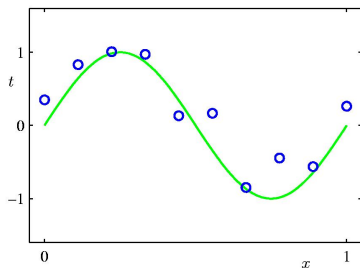


# Simple 1-D regression



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

# Simple 1-D regression

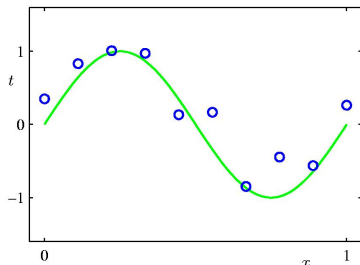


- Circles are data points (i.e., training examples) that are given to us
- The data points are uniform in  $x$ , but may be displaced in  $y$

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

with  $\epsilon$  some noise

# Simple 1-D regression



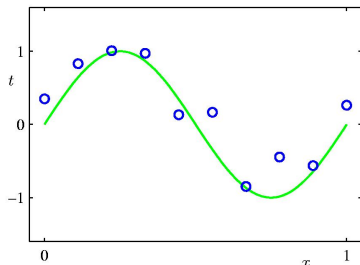
- Circles are data points (i.e., training examples) that are given to us
- The data points are uniform in  $x$ , but may be displaced in  $y$

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

with  $\epsilon$  some noise

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

# Simple 1-D regression



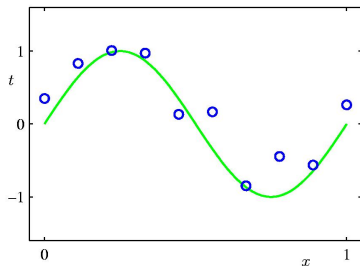
- Circles are data points (i.e., training examples) that are given to us
- The data points are uniform in  $x$ , but may be displaced in  $y$

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

with  $\epsilon$  some noise

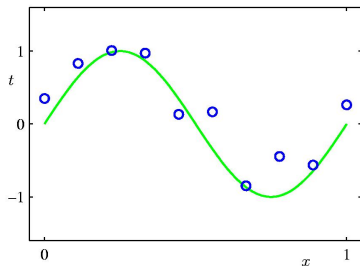
- In **green** is the "true" curve that we don't know
- **Goal**: We want to fit a curve to these points

# Simple 1-D regression



- Key Questions:

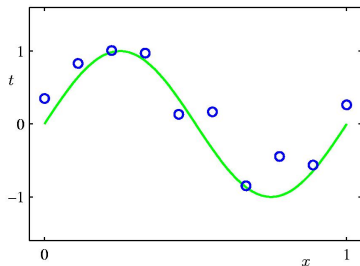
# Simple 1-D regression



- Key Questions:

- ▶ How do we parametrize the [model](#)?

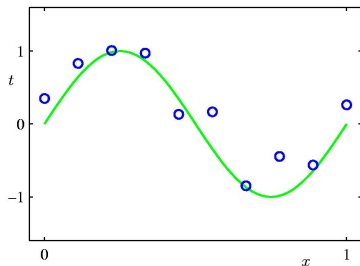
# Simple 1-D regression



- 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**)?

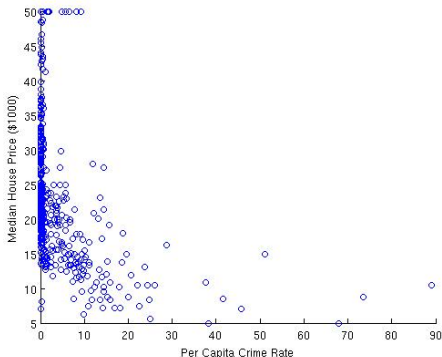


## Example: Boston Housing data

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

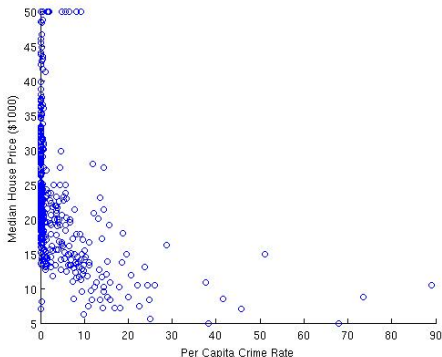
# Example: Boston Housing data

- 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

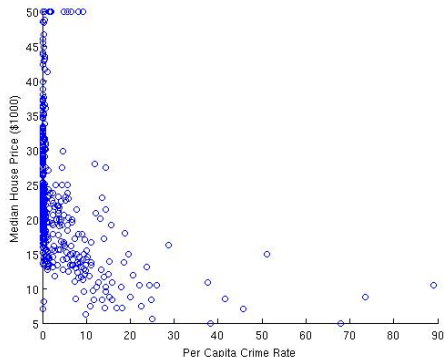
- Estimate median house price in a neighborhood based on neighborhood statistics
- Look at first possible attribute (feature): per capita crime rate



- 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  $t$

$$y(x) = w_0 + w_1x$$

# 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  $t$

$$y(x) = w_0 + w_1x$$

- 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  $t$

$$y(x) = w_0 + w_1x$$

- 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  $t$

$$y(x) = w_0 + w_1x$$

- 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  $t$

$$y(x) = w_0 + w_1x$$

- 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

- 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**, e.g., noise in per-capita crime)

# Noise

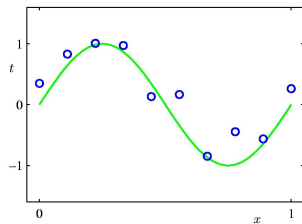
- A simple model typically does not exactly fit the data
  - ▶ lack of fit can be considered **noise**
- Sources of noise:
  - ▶ Imprecision in data attributes (**input noise**, e.g., noise in per-capita crime)
  - ▶ Errors in data targets (**mis-labeling**, e.g., noise in house prices)

- 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**, e.g., noise in per-capita crime)
  - ▶ Errors in data targets (**mis-labeling**, e.g., 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?

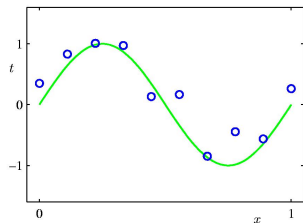


- A simple model typically does not exactly fit the data
  - ▶ lack of fit can be considered **noise**
- Sources of noise:
  - ▶ Imprecision in data attributes (**input noise**, e.g., noise in per-capita crime)
  - ▶ Errors in data targets (**mis-labeling**, e.g., noise in house prices)
  - ▶ **Additional 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



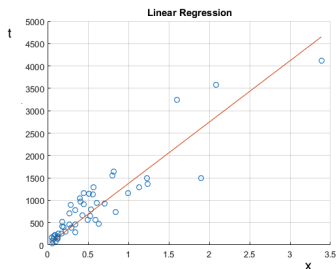
# Least-Squares Regression



- Define a model

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

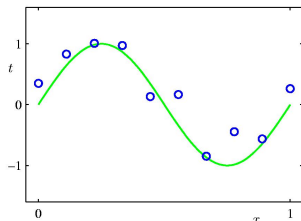
# Least-Squares Regression



- Define a model

Linear:  $y(x) = w_0 + w_1x$

# Least-Squares Regression



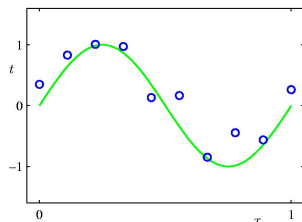
- Define a model

Linear:  $y(x) = w_0 + w_1x$

- 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



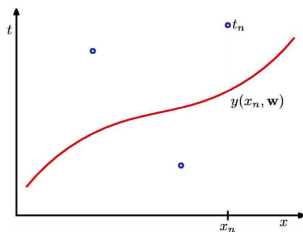
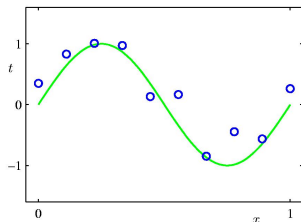
- Define a model

Linear:  $y(x) = w_0 + w_1x$

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

Linear model: 
$$\ell(\mathbf{w}) = \sum_{n=1}^N [t^{(n)} - (w_0 + w_1x^{(n)})]^2$$

# Least-Squares Regression



- Define a model

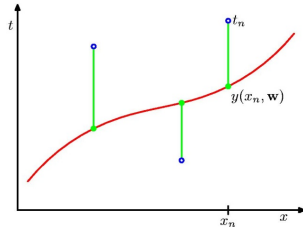
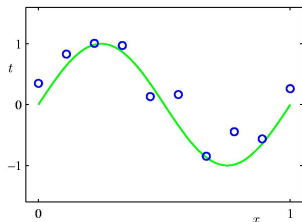
Linear:  $y(x) = w_0 + w_1x$

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

Linear model: 
$$\ell(\mathbf{w}) = \sum_{n=1}^N [t^{(n)} - (w_0 + w_1x^{(n)})]^2$$

- 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:  $y(x) = w_0 + w_1x$

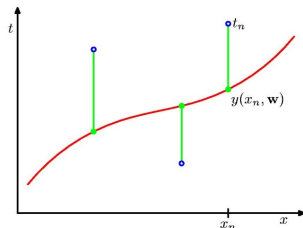
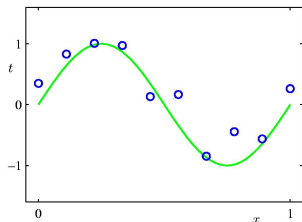
- Standard **loss/cost/objective function** measures the **squared error** between  $y$  and the true value  $t$

Linear model: 
$$\ell(\mathbf{w}) = \sum_{n=1}^N [t^{(n)} - (w_0 + w_1x^{(n)})]^2$$

- 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

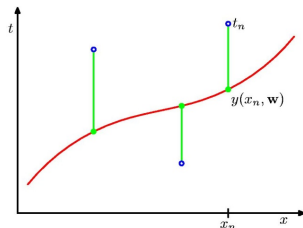
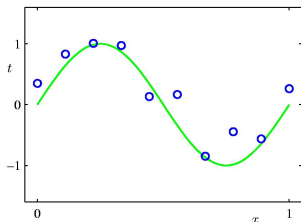
Linear:  $y(x) = w_0 + w_1x$

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

Linear model: 
$$\ell(\mathbf{w}) = \sum_{n=1}^N [t^{(n)} - (w_0 + w_1x^{(n)})]^2$$

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

# Least-Squares Regression



- Define a model

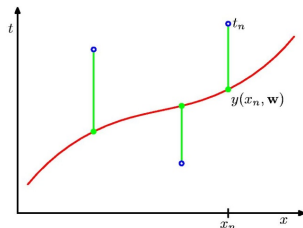
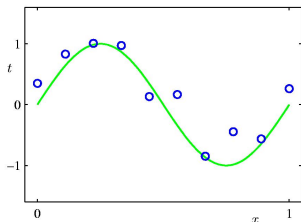
Linear:  $y(x) = w_0 + w_1x$

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

Linear model: 
$$\ell(\mathbf{w}) = \sum_{n=1}^N [t^{(n)} - (w_0 + w_1x^{(n)})]^2$$

- 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

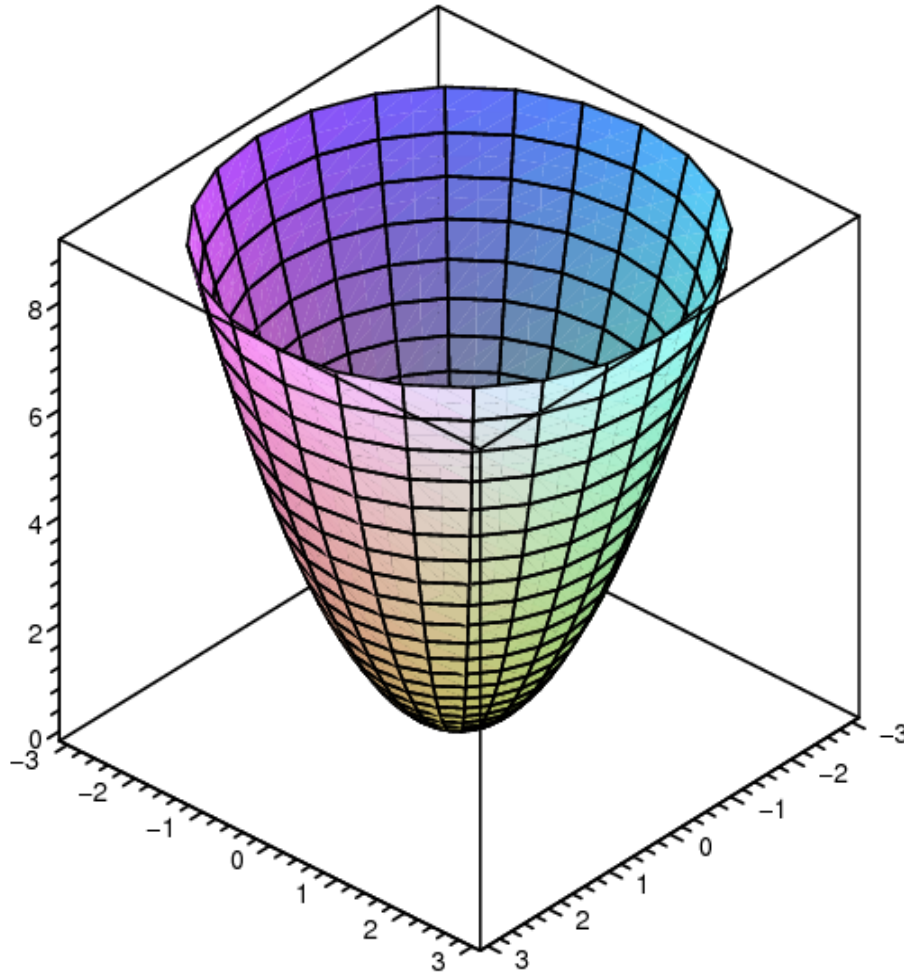
Linear:  $y(x) = w_0 + w_1x$

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

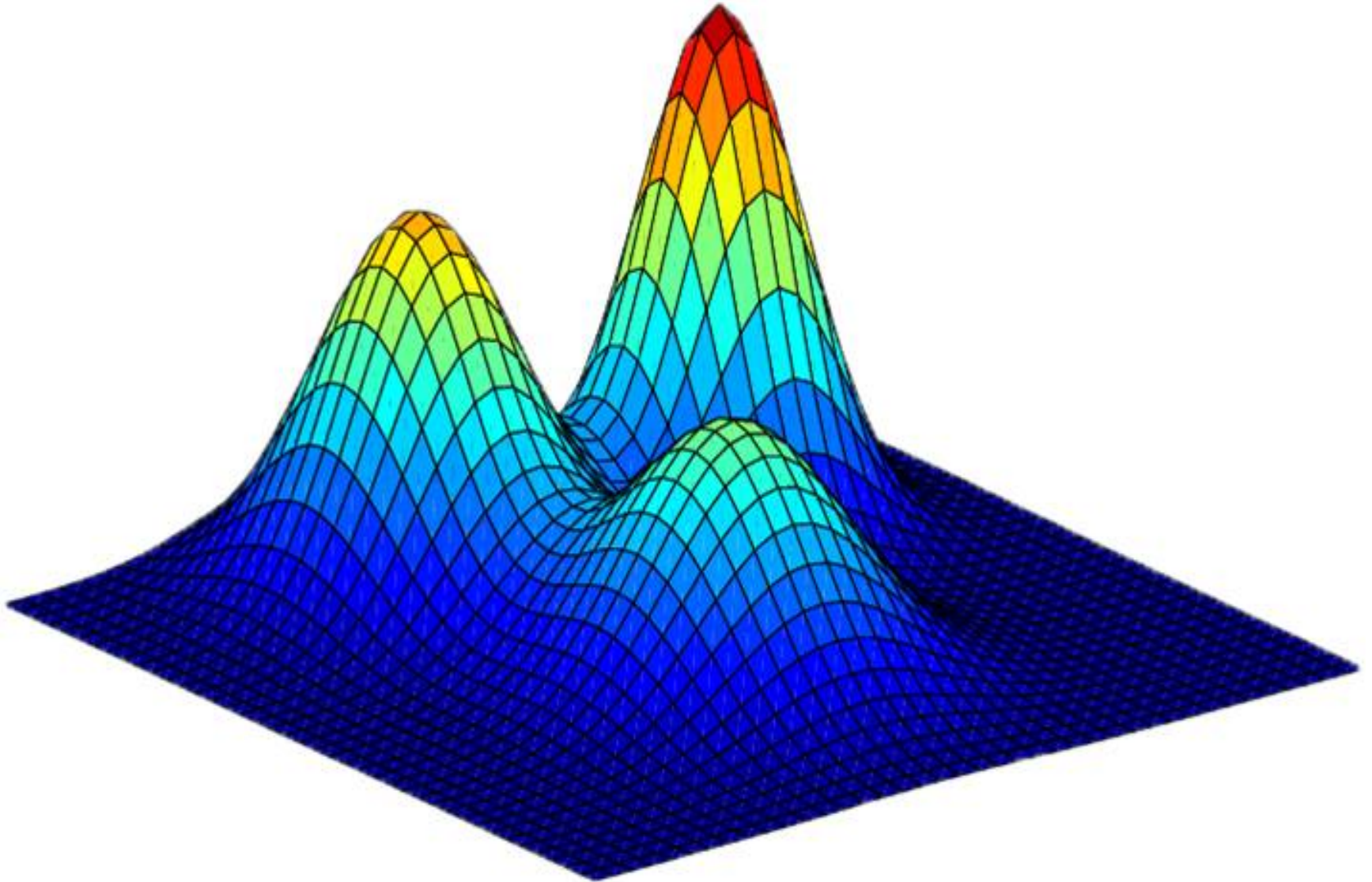
Linear model: 
$$\ell(\mathbf{w}) = \sum_{n=1}^N [t^{(n)} - (w_0 + w_1x^{(n)})]^2$$

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

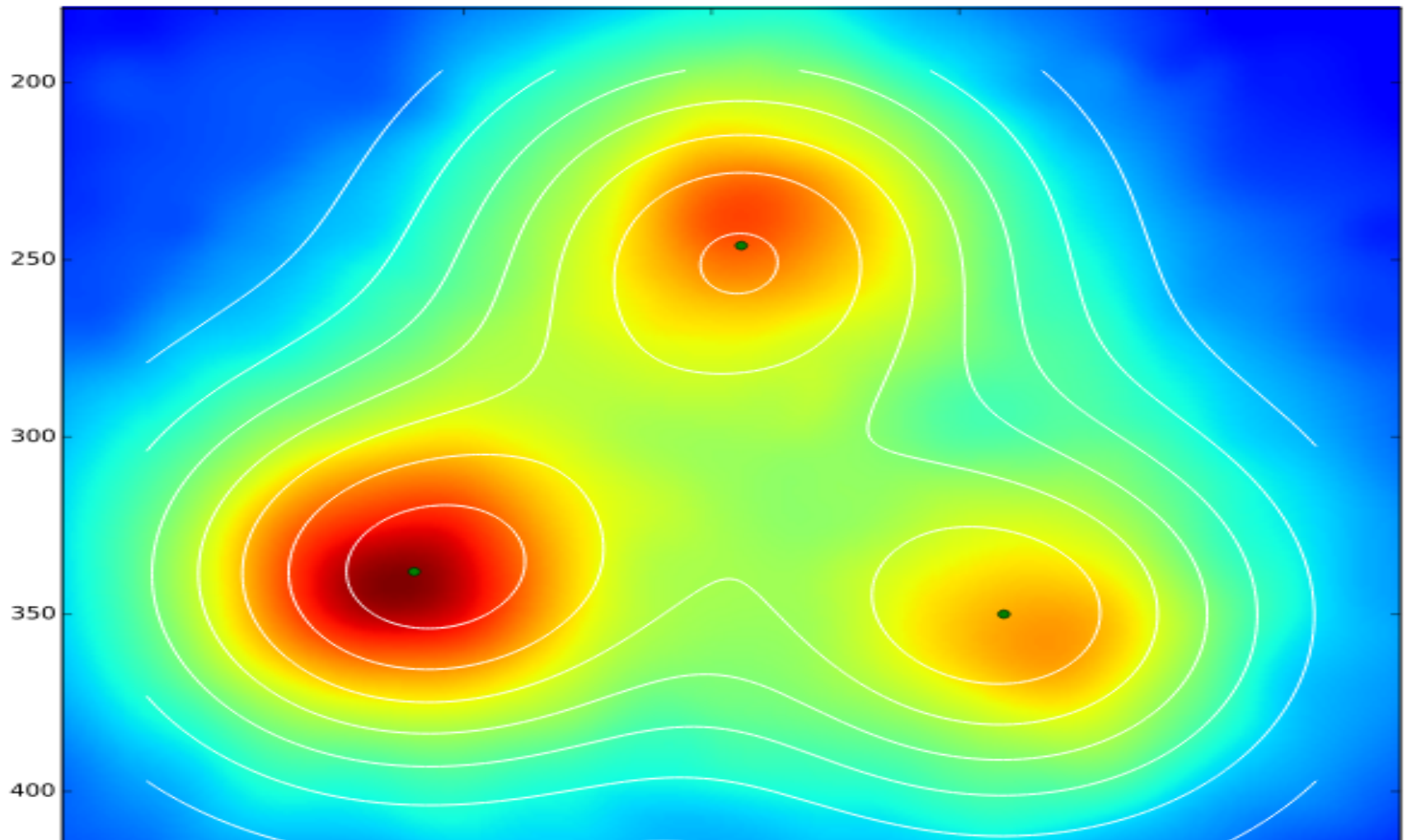
$$\ell(\mathbf{w}) = w_0^2 + w_1^2$$



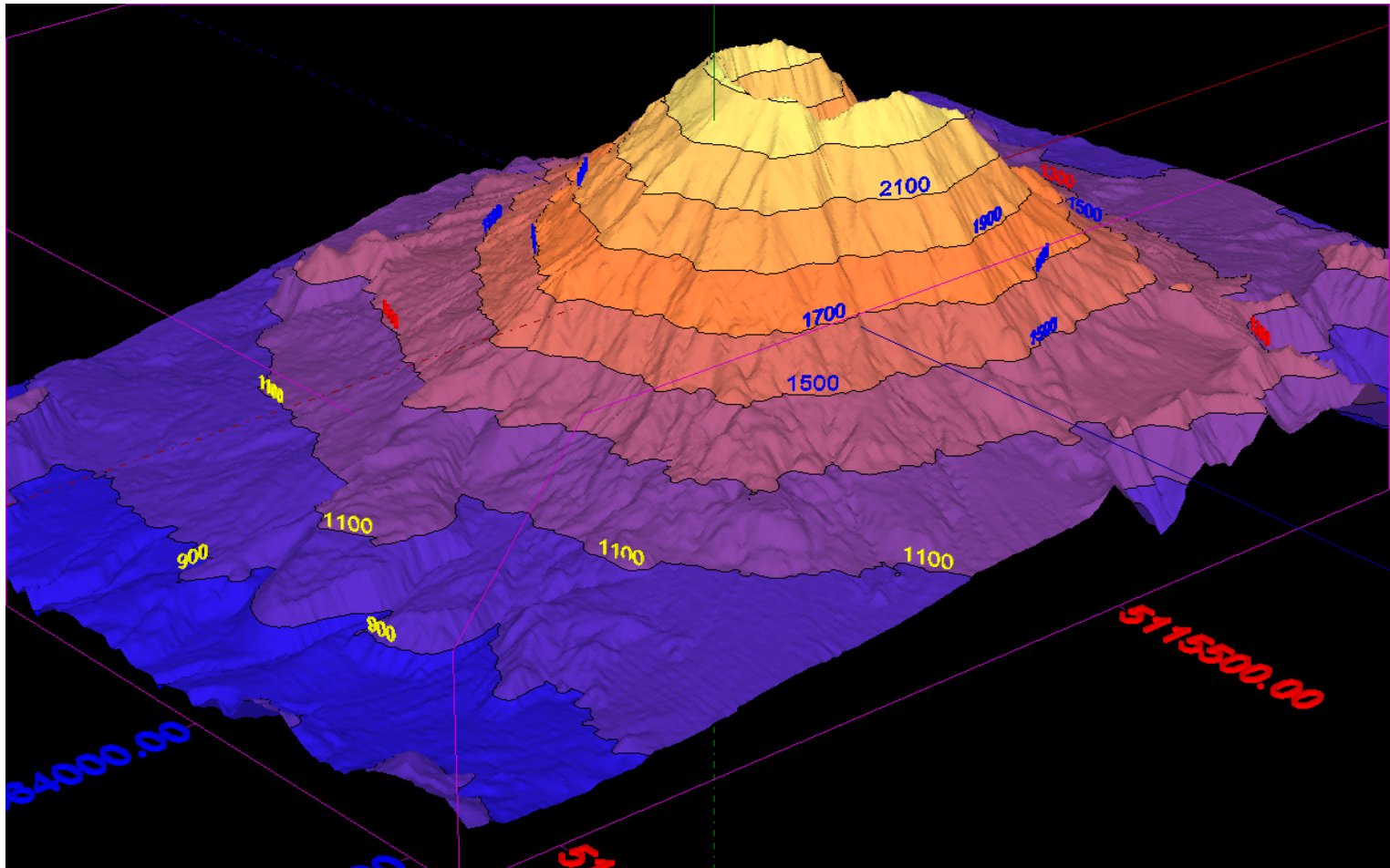
# Mixture of three Gaussians



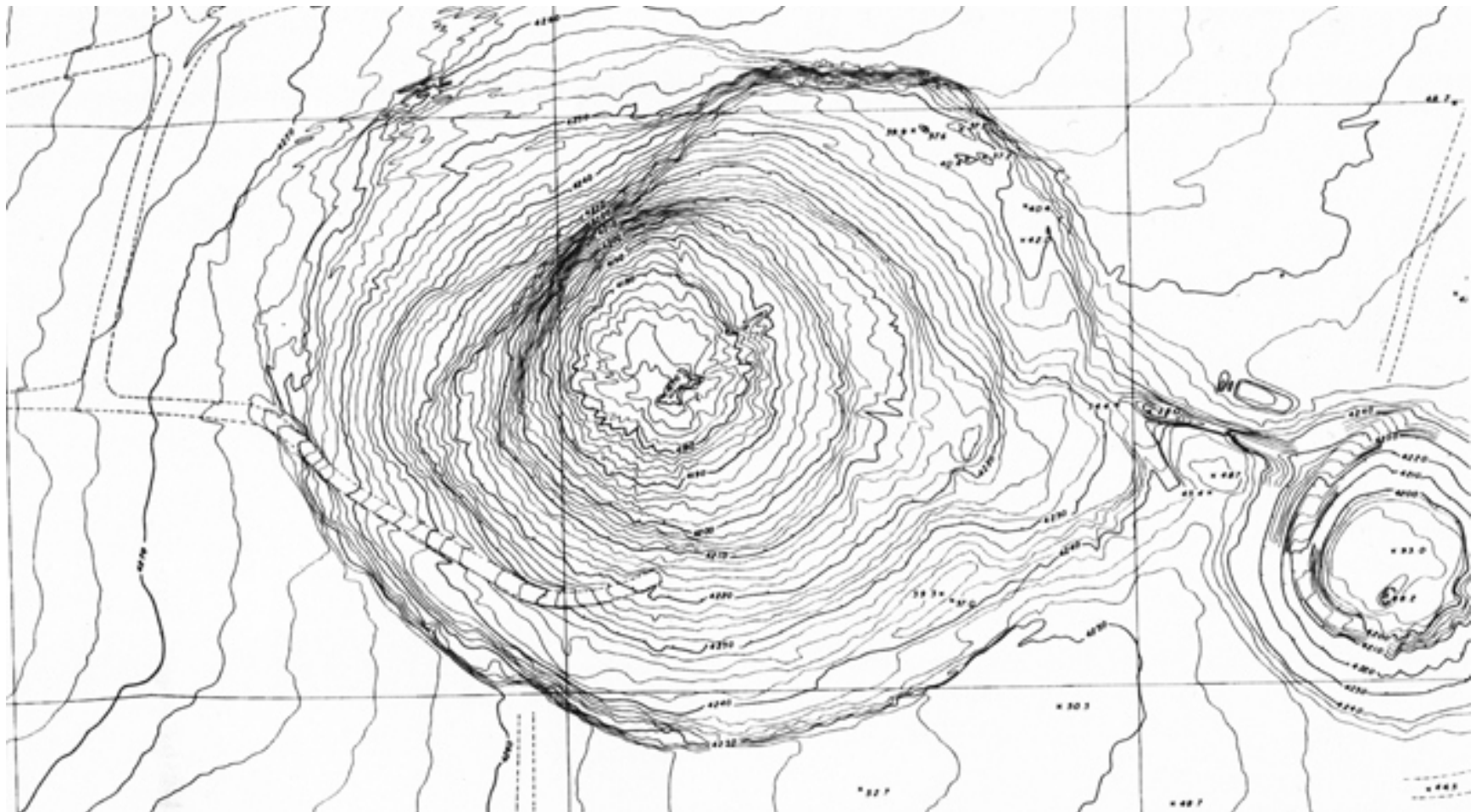
# MOG contours



# Contour Maps



# Contour Map





# 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

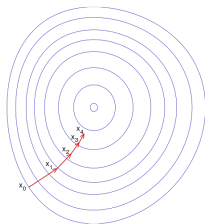
- One straightforward method: **gradient descent**
  - ▶ initialize  $\mathbf{w}$  (e.g., randomly)
  - ▶ repeatedly update  $\mathbf{w}$  based on the gradient

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

# Optimizing the Objective

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

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

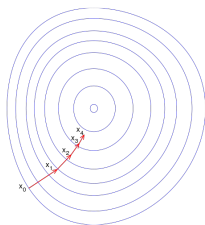


# Optimizing the Objective

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

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

- $\lambda$  is the **learning rate**



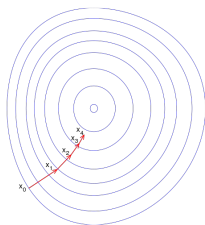
# Optimizing the Objective

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

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

- $\lambda$  is the **learning rate**
- For a **single training case**, this gives the **LMS update rule** (Least Mean Squares):

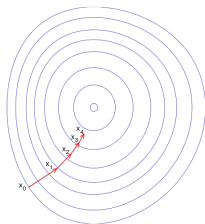
$$\mathbf{w} \leftarrow \mathbf{w} + 2\lambda(t^{(n)} - y(x^{(n)}))x^{(n)}$$



# Optimizing the Objective

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

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



- $\lambda$  is the **learning rate**
- For a **single training case**, this gives the **LMS update rule** (Least Mean Squares):

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

- 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:

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

# 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:

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

- 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

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

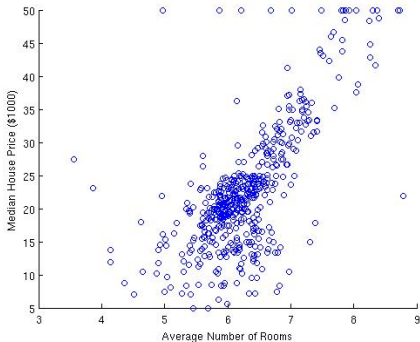


# Multi-dimensional Inputs

- 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  $\mathbf{x}$



# 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

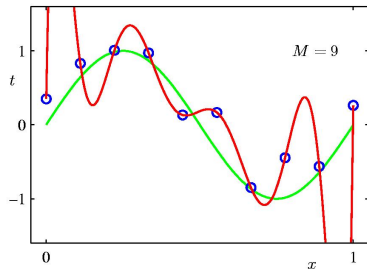
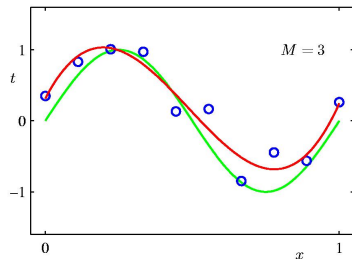
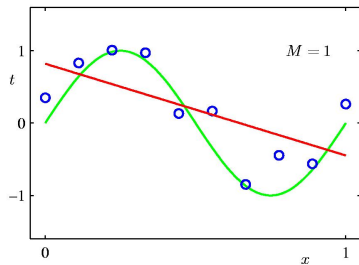
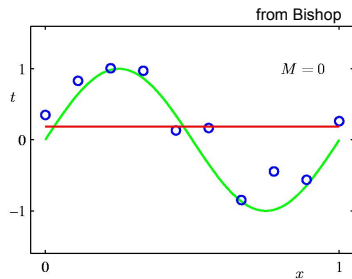
- What if our linear model is not good? How can we create a **more complicated model**?
- We can create a more complicated model by defining input variables that are combinations of components of  $\mathbf{x}$
- Example: an  $M$ -th order polynomial function of one dimensional feature  $x$ :

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

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

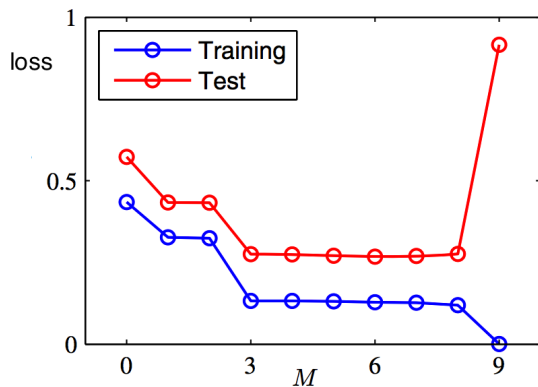
- We can use the same approach to optimize for the weights  $\mathbf{w}$
- How do we do that?

# Which Fit is Best?



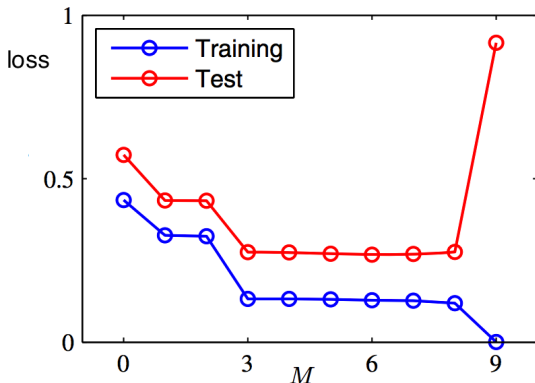
# Generalization

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



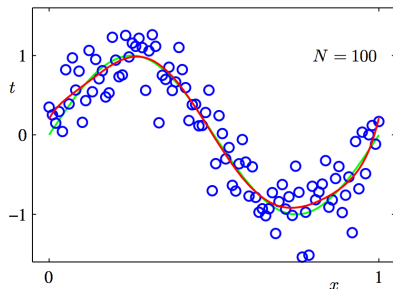
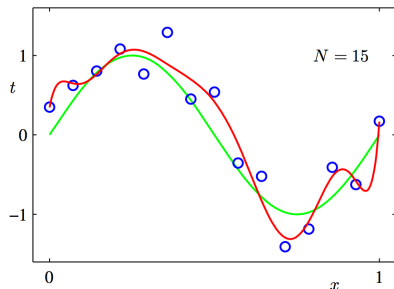
# Generalization

- **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  $\mathbf{w}$
- When we use the penalty on the squared weights we have **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  $\mathbf{w}$
- When we use the penalty on the squared weights we have **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**

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

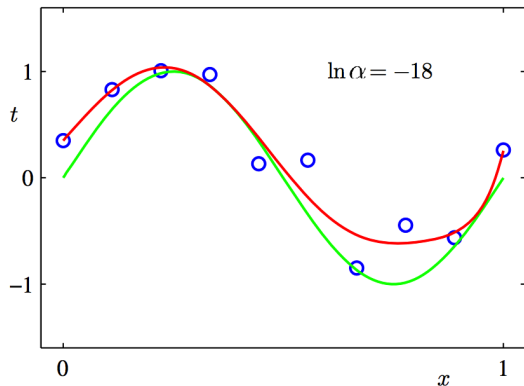
- Intuition: Since we are minimizing the loss, the second term will encourage smaller values in  $\mathbf{w}$
- When we use the penalty on the squared weights we have **ridge regression** in statistics
- Leads to a **modified update rule** for gradient descent:

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

- 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  $\alpha$  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](#)



**Anomalisa**

1h 31m | Comedy, Animation, Fantasy  
[View Ratings and Warnings](#)

[BUY TICKETS](#) [TRAILER](#)



**Django Masters (Behind the Scenes)**

2h 30m | Foreign Language, Drama, Romance, History  
[View Ratings and Warnings](#)

[BUY TICKETS](#)



**Beauty And The Beast (Filipino version)**

1h 59m | Action, Foreign Language, Comedy  
[View Ratings and Warnings](#)

[BUY TICKETS](#)



**Brooklyn**

1h 52m | Drama  
[View Ratings and Warnings](#)

[BUY TICKETS](#) [TRAILER](#)