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?

The Martian (2015)
PG-13 | 144 min | Adventure, Comedy, Drama | 2 October 2015 (USA)

Your rating: 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 More on IMDb Pro »
Problems for Today

- What should I watch this Friday?

**Point Break (2015)**

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

Your rating: 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 »
**Goal**: Predict movie rating automatically!

Predict this automatically!
Problems for Today

- **Goal:** How many followers will I get?
Problems for Today

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

---

**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 in 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?
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 $x$ 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 \( x \) 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 $x$ 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
Circles are data points (i.e., training examples) that are given to us.
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.
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.
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:

- 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)?
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)?
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

![Scatter plot showing the relationship between median house price and per capita crime rate.](scatter_plot.png)
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?
Data is described as pairs \( \mathcal{D} = \{(x^{(1)}, t^{(1)}), \ldots, (x^{(N)}, t^{(N)})\} \)

- \( x \in \mathbb{R} \) 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 $\mathcal{D} = \{(x^{(1)}, t^{(1)}), \ldots, (x^{(N)}, t^{(N)})\}$
  - $x \in \mathbb{R}$ 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**

$\text{Model outputs } y, \text{ an estimate of } y(x) = w_0 + w_1 x$

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
Represent the Data

- Data is described as pairs \( D = \{(x^{(1)}, t^{(1)}), \cdots, (x^{(N)}, t^{(N)})\} \)
  - \( x \in \mathbb{R} \) 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 $\mathcal{D} = \{(x^{(1)}, t^{(1)}), \ldots, (x^{(N)}, t^{(N)})\}$
  - $x \in \mathbb{R}$ 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?

Zemel, Urtasun, Fidler (UofT)  
CSC 411: 02-Regression
Represent the Data

- Data is described as pairs $\mathcal{D} = \{(x^{(1)}, t^{(1)}), \ldots, (x^{(N)}, t^{(N)})\}$
  - $x \in \mathbb{R}$ 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 \( \mathcal{D} = \{(x^{(1)}, t^{(1)}), \ldots, (x^{(N)}, t^{(N)})\} \)
  - \( x \in \mathbb{R} \) 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 $\mathcal{D} = \{(x^{(1)}, t^{(1)}), \ldots , (x^{(N)}, t^{(N)})\}$
  - $x \in \mathbb{R}$ 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
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)
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?
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)
  - 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

[Graph showing a plot with labeled axes and data points.]

Define a model

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

How do we obtain weights $w = (w_0, w_1)$?

For the linear model, what kind of a function is $\ell(w)$?
Define a model

\[ y(x) = \text{function}(x, w) \]
Least-Squares Regression

- Define a model

Linear: \[ y(x) = w_0 + w_1 x \]
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 \( t \)

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

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

For a particular hypothesis (\( y(x) \) defined by a choice of \( w \), drawn in red), what does the loss represent geometrically?
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:
\[
\ell(w) = \sum_{n=1}^{N} [t^{(n)} - (w_0 + w_1 x^{(n)})]^2
\]

The loss for the red hypothesis is the sum of the squared vertical errors (squared lengths of green vertical lines)
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: \( \ell(w) = \sum_{n=1}^{N} [t^{(n)} - (w_0 + w_1 x^{(n)})]^2 \)

How do we obtain weights \( w = (w_0, w_1) \)?
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:  
  \[
  \ell(w) = \sum_{n=1}^{N} [t^{(n)} - (w_0 + w_1 x^{(n)})]^2
  \]

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

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

How do we obtain weights \( w = (w_0, w_1) \)?

For the linear model, what kind of a function is \( \ell(w) \)?
Optimizing the Objective

- One straightforward method: gradient descent

- Initialize \( w \) (e.g., randomly)
- Repeatedly update \( w \) based on the gradient

\[
\dot{w} = w - \lambda \frac{\partial \ell}{\partial w}
\]

\( \lambda \) is the learning rate

For a single training case, this gives the LMS update rule (Least Mean Squares):

Note: As error approaches zero, so does the update (\( w \) stops changing)
Optimizing the Objective

- One straightforward method: gradient descent
  - initialize $\mathbf{w}$ (e.g., randomly)
Optimizing the Objective

- One straightforward method: gradient descent
  - initialize $w$ (e.g., randomly)
  - repeatedly update $w$ based on the gradient

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

Note: As error approaches zero, so does the update ($w$ stops changing)
Optimizing the Objective

- One straightforward method: gradient descent
  - initialize $w$ (e.g., randomly)
  - repeatedly update $w$ based on the gradient

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

Note: As error approaches zero, so does the update ($w$ stops changing)
Optimizing the Objective

- One straightforward method: gradient descent
  - initialize $w$ (e.g., randomly)
  - repeatedly update $w$ based on the gradient
    \[
    w \leftarrow w - \lambda \frac{\partial \ell}{\partial w}
    \]

- $\lambda$ 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 $w$ (e.g., randomly)
  - repeatedly update $w$ based on the gradient
    \[
    w \leftarrow w - \lambda \frac{\partial \ell}{\partial w} \]

- $\lambda$ is the learning rate

- For a single training case, this gives the LMS update rule (Least Mean Squares):
  \[
  w \leftarrow w + 2\lambda \left( t^{(n)} - y(x^{(n)}) \right) x^{(n)} \]

- Note: As error approaches zero, so does the update ($w$ stops changing)
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
     
     \[
     w \leftarrow w + 2\lambda N \sum_{n=1}^{N} \left( t(n) - y(x(n)) \right) x(n)
     \]

  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.)
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

   

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

   

   Underlying assumption: sample is independent and identically distributed (i.i.d.)
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

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

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:

   $$w \leftarrow w + 2\lambda(t^{(i)} - y(x^{(i)}))x^{(i)}$$ (update for a linear model)
4. end for
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.)
For some objectives we can also find the optimal solution analytically.

This is the case for linear least-squares regression.

How?
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 \( w \) and equate with 0.
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:

\[
\mathbf{t} = [t^{(1)}, t^{(2)}, \ldots, t^{(N)}]^T
\]

\[
\mathbf{X} = \begin{bmatrix}
1, x^{(1)} \\
1, x^{(2)} \\
\vdots \\
1, x^{(N)}
\end{bmatrix}
\]
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 $w$ and equate with 0. Define:

$$t = [t^{(1)}, t^{(2)}, \ldots, t^{(N)}]^T$$

$$X = \begin{bmatrix} 1, x^{(1)} \\ 1, x^{(2)} \\ \vdots \\ 1, x^{(N)} \end{bmatrix}$$

Then:

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

(work it out!)
Multi-dimensional Inputs

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

\[ y(x) = w_0 + w_1 x_1 + w_2 x_2 \]
Multi-dimensional Inputs

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

\[ y(x) = w_0 + w_1 x_1 + w_2 x_2 \]

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

![Graph showing the relationship between median house price and average number of rooms.](image-url)
Imagine now we want to predict the median house price from these multi-dimensional observations:

$$x(n) = (x(n)_1, \cdots, x(n)_j, \cdots, x(n)_d)$$

We can incorporate the bias $w_0$ into $w$, by using $x_0 = 1$, then

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

We can then solve for $w = (w_0, w_1, \cdots, w_d)$. How?

We can use gradient descent to solve for each coefficient, or compute $w$ analytically (how does the solution change?)
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 $j$:

\[
x^{(n)} = \left( x_1^{(n)}, \ldots, x_j^{(n)}, \ldots, x_d^{(n)} \right)
\]
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 $j$:

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

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

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

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

We can incorporate the bias $w_0$ into $w$, by using $x_0 = 1$, then

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

We can then solve for $w = (w_0, w_1, \cdots, w_d)$. How?
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 $j$:

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

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 $\mathbf{w} = (w_0, w_1, \cdots, w_d)$. How?

We can use gradient descent to solve for each coefficient, or compute $\mathbf{w}$ analytically (how does the solution change?)
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 $x$.

Example: an $M$-th order polynomial function of one dimensional feature $x$:

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

where $x_j$ is the $j$-th power of $x$.
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 $x$

Example: an $M$-th order polynomial function of one dimensional feature $x$:

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

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 $x$

- Example: an $M$-th order polynomial function of one dimensional feature $x$:

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

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 $x$

- Example: an $M$-th order polynomial function of one dimensional feature $x$:

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

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

- We can use the same approach to optimize for the weights $w$
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 $x$

Example: an $M$-th order polynomial function of one dimensional feature $x$:

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

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

We can use the same approach to optimize for the weights $w$

How do we do that?
Which Fit is Best?

from Bishop

CSC 411: 02-Regression

Zemel, Urtasun, Fidler (UofT)
Generalization

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

![Graph showing loss against M with Training and Test lines.]
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

![Graph showing generalization with different numbers of training examples](image)
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

<table>
<thead>
<tr>
<th></th>
<th>$M = 0$</th>
<th>$M = 1$</th>
<th>$M = 6$</th>
<th>$M = 9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_0^*$</td>
<td>0.19</td>
<td>0.82</td>
<td>0.31</td>
<td>0.35</td>
</tr>
<tr>
<td>$w_1^*$</td>
<td></td>
<td>-1.27</td>
<td>7.99</td>
<td>232.37</td>
</tr>
<tr>
<td>$w_2^*$</td>
<td></td>
<td></td>
<td>-25.43</td>
<td>-5321.83</td>
</tr>
<tr>
<td>$w_3^*$</td>
<td></td>
<td></td>
<td>17.37</td>
<td>48568.31</td>
</tr>
<tr>
<td>$w_4^*$</td>
<td></td>
<td></td>
<td></td>
<td>-231639.30</td>
</tr>
<tr>
<td>$w_5^*$</td>
<td></td>
<td></td>
<td></td>
<td>640042.26</td>
</tr>
<tr>
<td>$w_6^*$</td>
<td></td>
<td></td>
<td></td>
<td>-1061800.52</td>
</tr>
<tr>
<td>$w_7^*$</td>
<td></td>
<td></td>
<td></td>
<td>1042400.18</td>
</tr>
<tr>
<td>$w_8^*$</td>
<td></td>
<td></td>
<td></td>
<td>-557682.99</td>
</tr>
<tr>
<td>$w_9^*$</td>
<td></td>
<td></td>
<td></td>
<td>125201.43</td>
</tr>
</tbody>
</table>
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

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

Intuition: Since we are minimizing the loss, the second term will encourage smaller values in \( w \)

When we use the penalty on the squared weights we have ridge regression in statistics

Leads to a modified update rule for gradient descent:

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

Also has an analytical solution:

\[ w = (X^T X + \alpha I)^{-1} X^T t \] (verify!)
Regularized Least Squares

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

\[
\tilde{\ell}(w) = \sum_{n=1}^{N} [t^{(n)} - (w_0 + w_1 x^{(n)})]^2 + \alpha w^T w
\]
Increasing the input features this way can complicate the model considerably

**Goal:** select the appropriate model complexity automatically

**Standard approach:** regularization

\[
\ell(w) = \sum_{n=1}^{N} [t^{(n)} - (w_0 + w_1x^{(n)})]^2 + \alpha w^T w
\]

**Intuition:** Since we are minimizing the loss, the second term will encourage smaller values in \( w \)
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

\[ \tilde{\ell}(\mathbf{w}) = \sum_{n=1}^{N} [t^{(n)} - (w_0 + w_1x^{(n)})]^2 + \alpha \mathbf{w}^T \mathbf{w} \]

- 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*

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

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

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

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

- Better generalization
- Choose $\alpha$ carefully

\[
\ln \alpha = -18
\]
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
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 \((\text{model selection})\)?
  - Simple models may not capture all the important variations \((\text{signal})\) in the data: \text{underfit}
  - More complex models may \text{overfit} the training data (fit not only the signal but also the \text{noise} in the data), especially if not enough data to constrain model

- One method of assessing fit: test \text{generalization} = model’s ability to predict the held out data

- \text{Optimization} is essential: stochastic and batch iterative approaches; analytic when available
So...

- Which movie will you watch?