Data 401
Machine Learning and Linear Regression

Dennis Sun

September 28, 2016

1. Machine Learning
2. Linear Regression
3. Categorical Predictors

1. Machine Learning
2. Linear Regression
3. Categorical Predictors

Traditional Artificial Intelligence

Traditional Artificial Intelligence

- Discover and hard-code the rules that humans use to make decisions.

Traditional Artificial Intelligence

- Discover and hard-code the rules that humans use to make decisions.
- For example, if you were trying to build a system that recognizes people names in texts, you might have several rules:
    - If the first letter is not capitalized, then it is not a name.
    - The first letter of a sentence is always capitalized.
    - ...

Traditional Artificial Intelligence

- Discover and hard-code the rules that humans use to make decisions.
- For example, if you were trying to build a system that recognizes people names in texts, you might have several rules:
  - If the first letter is not capitalized, then it is not a name.
  - The first letter of a sentence is always capitalized.
  - ...
- The system can deduce new rules from existing rules, e.g.,
  - If the first letter of a sentence is capitalized, then that word may or may not be a name.

Traditional Artificial Intelligence

- Discover and hard-code the rules that humans use to make decisions.
- For example, if you were trying to build a system that recognizes people names in texts, you might have several rules:
  - If the first letter is not capitalized, then it is not a name.
  - The first letter of a sentence is always capitalized.
  - ...
- The system can deduce new rules from existing rules, e.g.,
  - If the first letter of a sentence is capitalized, then that word may or may not be a name.
- **Pros:** Models were super interpretable!

Traditional Artificial Intelligence

- Discover and hard-code the rules that humans use to make decisions.
- For example, if you were trying to build a system that recognizes people names in texts, you might have several rules:
  - If the first letter is not capitalized, then it is not a name.
  - The first letter of a sentence is always capitalized.
  - ...
- The system can deduce new rules from existing rules, e.g.,
  - If the first letter of a sentence is capitalized, then that word may or may not be a name.
- **Pros:** Models were super interpretable!
- **Cons:** Performance plateaued quickly. There are too many rules, and we don't know what they are!

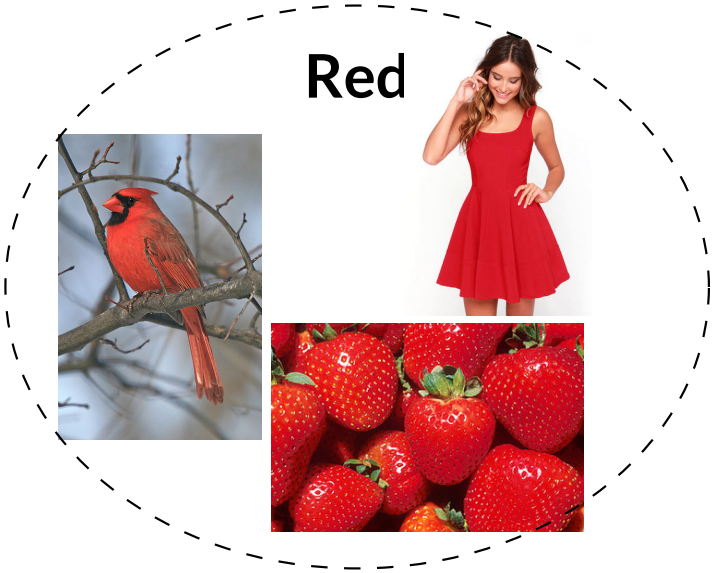
What is Machine Learning?

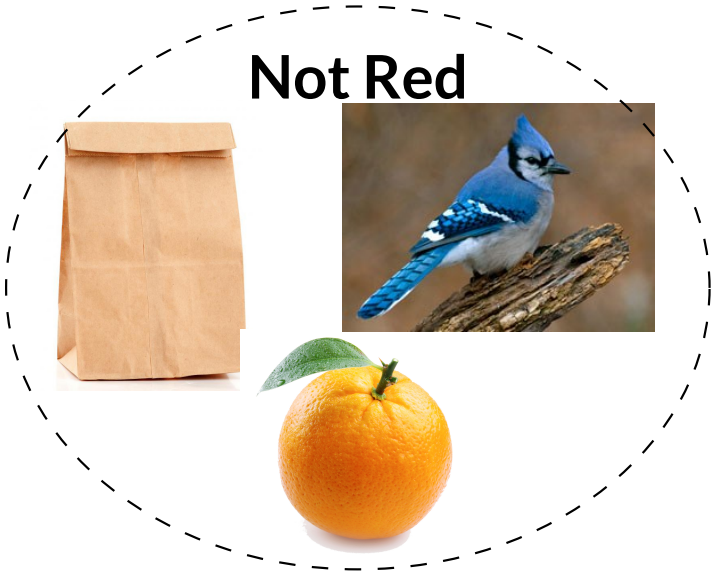
Pretend your neighbor does not know what the color red is. Try to explain to him or her in words what it is.

What is Machine Learning?

Pretend your neighbor does not know what the color red is. Try to explain to him or her in words what it is.

You've seen thousands of **training examples** in your life.

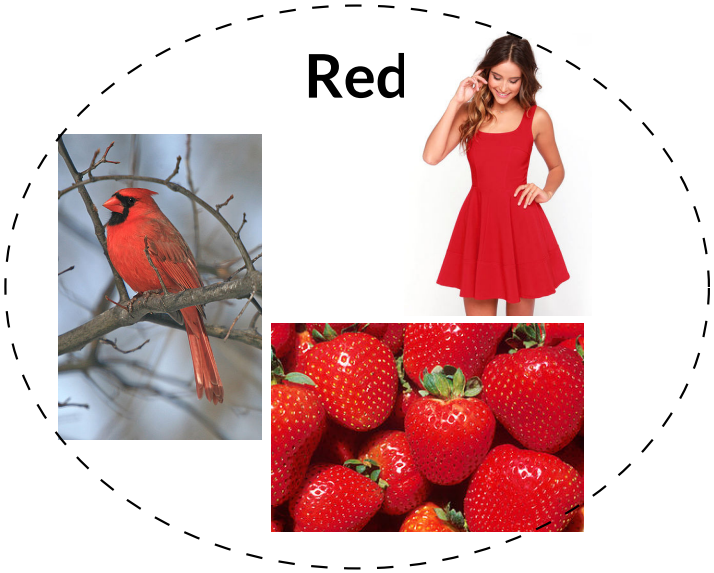


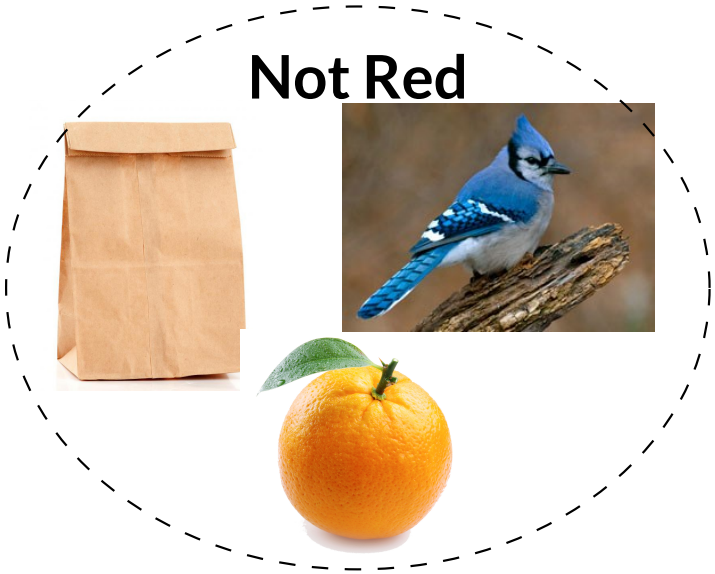


What is Machine Learning?

Pretend your neighbor does not know what the color red is. Try to explain to him or her in words what it is.

You've seen thousands of **training examples** in your life.



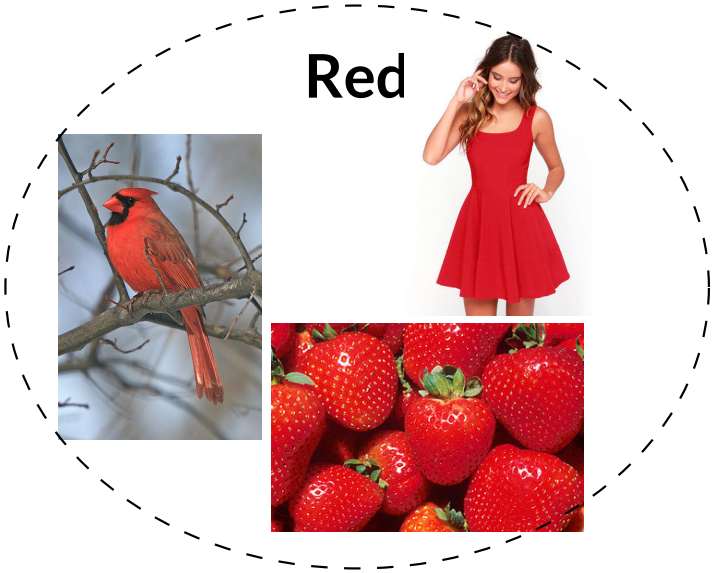


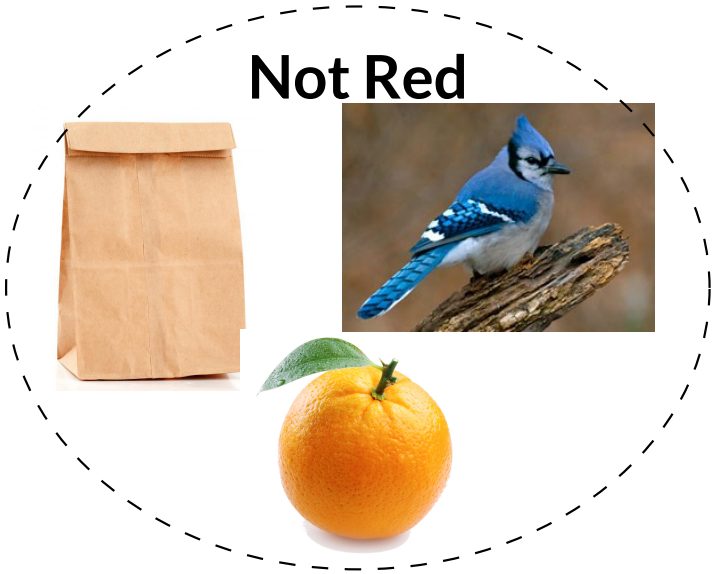
You're able to demonstrate your learning on **test examples**.

What is Machine Learning?

Pretend your neighbor does not know what the color red is. Try to explain to him or her in words what it is.

You've seen thousands of **training examples** in your life.





You're able to demonstrate your learning on **test examples**.

$\longrightarrow$ red

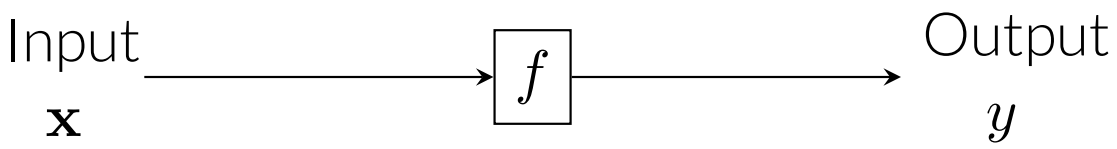
$\longrightarrow$ not red

What is Machine Learning?

Learning is the ability to generalize from training examples to make accurate predictions on test examples.

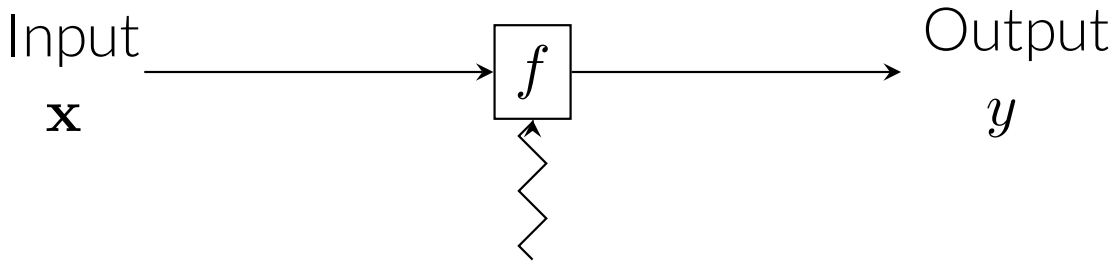
What is Machine Learning?

Learning is the ability to generalize from training examples to make accurate predictions on test examples.



What is Machine Learning?

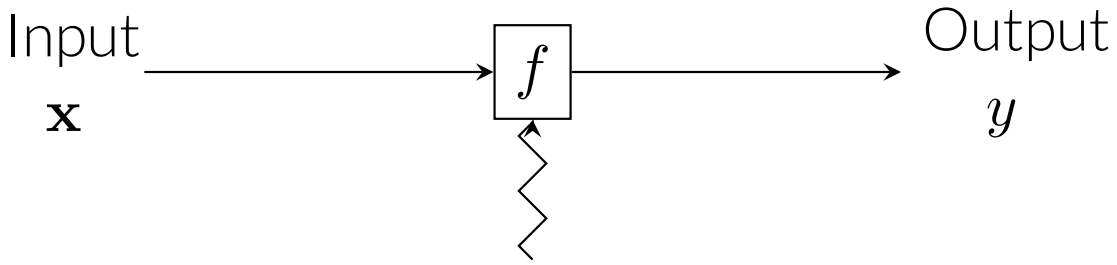
Learning is the ability to generalize from training examples to make accurate predictions on test examples.



Goal of Machine Learning: Estimate f using training observations $(\mathbf{x}_i, y_i)$

What is Machine Learning?

Learning is the ability to generalize from training examples to make accurate predictions on test examples.

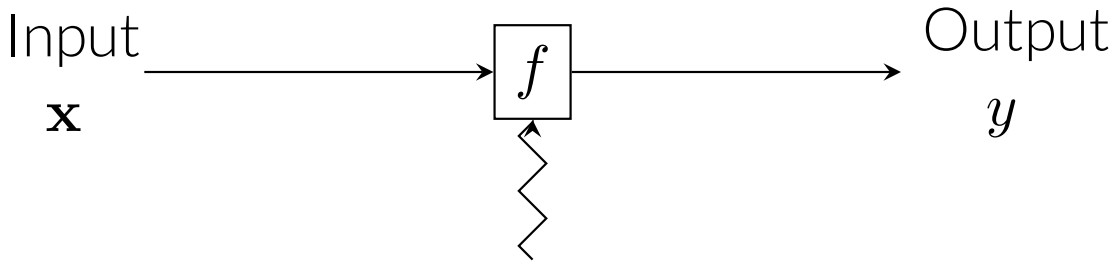


Goal of Machine Learning: Estimate f using training observations $(\mathbf{x}_i, y_i)$

Then, if we have a test input $\mathbf{x}^*$, we can predict its output as $f(\mathbf{x}^*)$.

What is Machine Learning?

Learning is the ability to generalize from training examples to make accurate predictions on test examples.



Goal of Machine Learning: Estimate f using training observations $(\mathbf{x}_i, y_i)$

Then, if we have a test input $\mathbf{x}^*$, we can predict its output as $f(\mathbf{x}^*)$.

Linear regression, k-nearest neighbors, random forests, SVMs, and deep learning are all just different ways of estimating f.

1 Machine Learning

2 Linear Regression

3 Categorical Predictors

(Multiple) Linear Regression

In linear regression, we assume that f is linear:

$$y = f(\mathbf{x}) + \epsilon = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p + \epsilon,$$

where ϵ represents noise.

(Multiple) Linear Regression

In linear regression, we assume that f is linear:

$$y = f(\mathbf{x}) + \epsilon = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p + \epsilon,$$

where ϵ represents noise.

How do we use training data $(\mathbf{x}_i, y_i)$ to estimate f (or equivalently, $\beta_0, \beta_1, ..., \beta_p$)?

(Multiple) Linear Regression

In linear regression, we assume that f is linear:

$$y = f(\mathbf{x}) + \epsilon = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p + \epsilon,$$

where ϵ represents noise.

How do we use training data $(\mathbf{x}_i, y_i)$ to estimate f (or equivalently, $\beta_0, \beta_1, ..., \beta_p$)?

Choose $\beta_0, \beta_1, ..., \beta_p$ to make f fit the training data as best as possible:

$$\sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip}))^2.$$

(Multiple) Linear Regression

Choose $\beta_0, \beta_1, ..., \beta_p$ to make $f(\mathbf{x})$ fit the training data as best as possible:

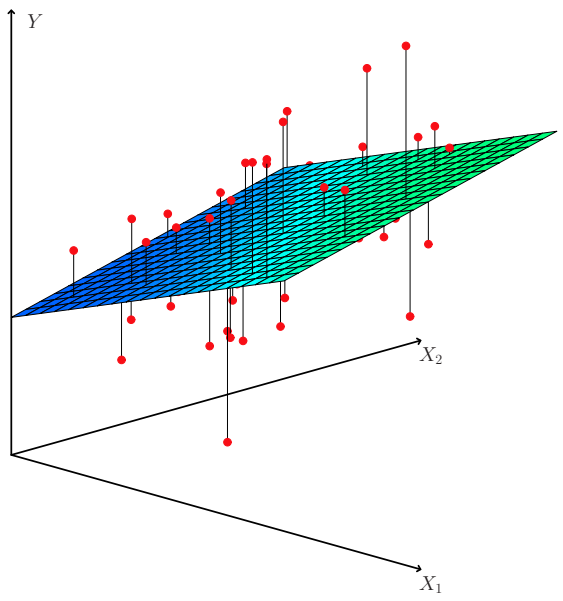
$$\sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip}))^2.$$

(Multiple) Linear Regression

Choose $\beta_0, \beta_1, ..., \beta_p$ to make $f(\mathbf{x})$ fit the training data as best as possible:

$$\sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip}))^2.$$

When $p = 2$, we are fitting a plane to the data.

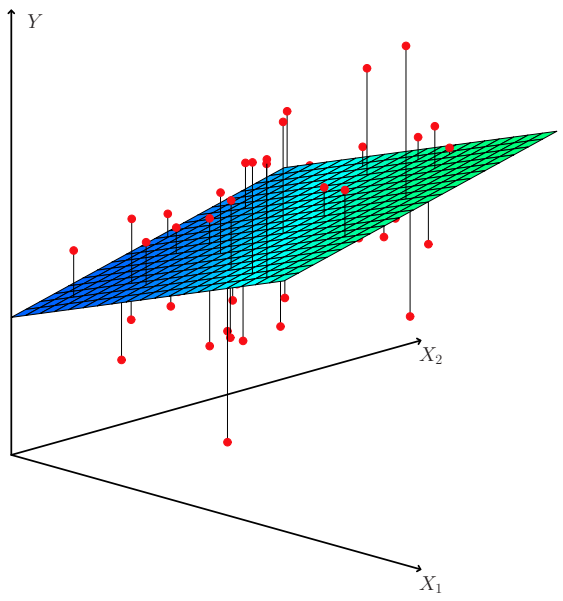


(Multiple) Linear Regression

Choose $\beta_0, \beta_1, ..., \beta_p$ to make $f(\mathbf{x})$ fit the training data as best as possible:

$$\sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip}))^2.$$

When $p = 2$, we are fitting a plane to the data.



When $p > 2$, we can't visualize it, but we can still do the math.

Linear Algebra Simplifies Life

Linear Algebra Simplifies Life

We can stack our training data into matrices and vectors:

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \qquad X = \begin{pmatrix} 1 & x_{11} & \dots & x_{1p} \\ 1 & x_{21} & \dots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \dots & x_{np} \end{pmatrix} \qquad \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix}.$$

Linear Algebra Simplifies Life

We can stack our training data into matrices and vectors:

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \qquad X = \begin{pmatrix} 1 & x_{11} & \dots & x_{1p} \\ 1 & x_{21} & \dots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \dots & x_{np} \end{pmatrix} \qquad \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix}.$$

Notice that $X\boldsymbol{\beta}$ is the vector of **predictions** or **fitted values** for the observations in our training set:

$$X\boldsymbol{\beta} = \begin{pmatrix} \beta_0 + \beta_1 x_{11} + \dots + \beta_p x_{1p} \\ \vdots \\ \beta_0 + \beta_1 x_{n1} + \dots + \beta_p x_{np} \end{pmatrix}$$

Linear Algebra Simplifies Life

We can stack our training data into matrices and vectors:

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \qquad X = \begin{pmatrix} 1 & x_{11} & \dots & x_{1p} \\ 1 & x_{21} & \dots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \dots & x_{np} \end{pmatrix} \qquad \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix}.$$

Notice that $X\boldsymbol{\beta}$ is the vector of **predictions** or **fitted values** for the observations in our training set:

$$X\boldsymbol{\beta} = \begin{pmatrix} \beta_0 + \beta_1 x_{11} + \ldots + \beta_p x_{1p} \\ \vdots \\ \beta_0 + \beta_1 x_{n1} + \ldots + \beta_p x_{np} \end{pmatrix}$$

and $\mathbf{y} - X\boldsymbol{\beta}$ is the vector of **residuals**:

$$\mathbf{y} - X\boldsymbol{\beta} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} - \begin{pmatrix} \beta_0 + \beta_1 x_{11} + \ldots + \beta_p x_{1p} \\ \vdots \\ \beta_0 + \beta_1 x_{n1} + \ldots + \beta_p x_{np} \end{pmatrix}$$

Why do we need to know the math?

Why do we have to know how linear regression works? Can't we just plug this into R / Scikit-learn?

Why do we need to know the math?

Why do we have to know how linear regression works? Can't we just plug this into R / Scikit-learn?

No! Remember, one of the challenges of data science is **big data**. It'll be hard to *load* a 10 GB file into R, much less run `lm` on it.

Why do we need to know the math?

Why do we have to know how linear regression works? Can't we just plug this into R / Scikit-learn?

No! Remember, one of the challenges of data science is **big data**. It'll be hard to *load* a 10 GB file into R, much less run `lm` on it.

As a data scientist, you will need to understand the nuts and bolts because you may need to implement these methods from scratch.

Solving for $\boldsymbol{\beta}$

How do we find $\boldsymbol{\beta}$ that minimizes

$$L(\boldsymbol{\beta}) = \sum_{i=1}^{n} (y_i - (\beta_0 x_{i0} + \beta_1 x_{i1} + ... + \beta_p x_{ip}))^2,$$

where $x_{i0} = 1$?

Solving for $\boldsymbol{\beta}$

How do we find $\boldsymbol{\beta}$ that minimizes

$$L(\boldsymbol{\beta}) = \sum_{i=1}^{n} (y_i - (\beta_0 x_{i0} + \beta_1 x_{i1} + ... + \beta_p x_{ip}))^2,$$

where $x_{i0} = 1$?

Take the derivative, set it equal to 0. Solve for $\boldsymbol{\beta}$.

Solving for $\boldsymbol{\beta}$

How do we find $\boldsymbol{\beta}$ that minimizes

$$L(\boldsymbol{\beta}) = \sum_{i=1}^{n} (y_i - (\beta_0 x_{i0} + \beta_1 x_{i1} + ... + \beta_p x_{ip}))^2,$$

where $x_{i0} = 1$?

Take the derivative, set it equal to 0. Solve for $\boldsymbol{\beta}$.

$$\frac{\partial L}{\partial \beta_j} = 2 \sum_{i=1}^{n} x_{ij} (y_i - (\beta_0 x_{i0} + \beta_1 x_{i1} + ... + \beta_p x_{ip})) = 0.$$

Solving for $\boldsymbol{\beta}$

How do we find $\boldsymbol{\beta}$ that minimizes

$$L(\boldsymbol{\beta}) = \sum_{i=1}^{n} (y_i - (\beta_0 x_{i0} + \beta_1 x_{i1} + ... + \beta_p x_{ip}))^2,$$

where $x_{i0} = 1$?

Take the derivative, set it equal to 0. Solve for $\boldsymbol{\beta}$.

$$\frac{\partial L}{\partial \beta_j} = 2 \sum_{i=1}^{n} x_{ij} (y_i - (\beta_0 x_{i0} + \beta_1 x_{i1} + ... + \beta_p x_{ip})) = 0.$$

This is a linear system of $p + 1$ equations with $p + 1$ unknowns.

The Miracle of Linear Algebra

We need to solve

$$
\begin{aligned}
\textstyle\sum_{i=1}^{n} x_{i0}(y_i - (\beta_0 x_{i0} + \beta_1 x_{i1} + ... + \beta_p x_{ip})) &= 0 \\
\textstyle\sum_{i=1}^{n} x_{i1}(y_i - (\beta_0 x_{i0} + \beta_1 x_{i1} + ... + \beta_p x_{ip})) &= 0 \\
\vdots \qquad\qquad\qquad\qquad\qquad\qquad &= \vdots \\
\textstyle\sum_{i=1}^{n} x_{ip}(y_i - (\beta_0 x_{i0} + \beta_1 x_{i1} + ... + \beta_p x_{ip})) &= 0.
\end{aligned}
$$

The Miracle of Linear Algebra

We need to solve

$$\begin{aligned}
\textstyle\sum_{i=1}^n x_{i0}(y_i - (\beta_0 x_{i0} + \beta_1 x_{i1} + ... + \beta_p x_{ip})) &= 0 \\
\textstyle\sum_{i=1}^n x_{i1}(y_i - (\beta_0 x_{i0} + \beta_1 x_{i1} + ... + \beta_p x_{ip})) &= 0 \\
\vdots \qquad\qquad\qquad\qquad\qquad\qquad &= \vdots \\
\textstyle\sum_{i=1}^n x_{ip}(y_i - (\beta_0 x_{i0} + \beta_1 x_{i1} + ... + \beta_p x_{ip})) &= 0.
\end{aligned}$$

Linear algebra comes to the rescue! We can rewrite this as

$$X^T(\mathbf{y} - X\boldsymbol{\beta}) = \mathbf{0}$$

The Miracle of Linear Algebra

We need to solve

$$\begin{aligned}
\textstyle\sum_{i=1}^n x_{i0}(y_i - (\beta_0 x_{i0} + \beta_1 x_{i1} + ... + \beta_p x_{ip})) &= 0 \\
\textstyle\sum_{i=1}^n x_{i1}(y_i - (\beta_0 x_{i0} + \beta_1 x_{i1} + ... + \beta_p x_{ip})) &= 0 \\
\vdots \qquad\qquad\qquad\qquad\qquad\qquad &= \vdots \\
\textstyle\sum_{i=1}^n x_{ip}(y_i - (\beta_0 x_{i0} + \beta_1 x_{i1} + ... + \beta_p x_{ip})) &= 0.
\end{aligned}$$

Linear algebra comes to the rescue! We can rewrite this as

$$X^T(\mathbf{y} - X\boldsymbol{\beta}) = \mathbf{0}$$

Now if we solve for $\boldsymbol{\beta}$, we obtain

$$\hat{\boldsymbol{\beta}} = (X^T X)^{-1} X^T \mathbf{y}$$

Multiple Regression in Code

In-Class Exercise

Open the notebook `Implementing Linear Regression.ipynb` *and implement the* `lm` *function, assuming that all predictors are quantitative. Check that your function returns the same coefficients as scikit-learn.*

Hints:

- `np.dot` *can be used to multiply matrices.*
- `X.T` *returns the transpose of the matrix* `X`.
- `np.linalg.inv` *can be used to invert matrices.*

Categorical Predictors

Some of the variables in the `autos` dataset are categorical, like the make of the car (e.g., alfa-romero, audi, bmw, etc.). How do we incorporate variables like this into linear regression?

Categorical Predictors

Some of the variables in the `autos` dataset are categorical, like the make of the car (e.g., alfa-romero, audi, bmw, etc.). How do we incorporate variables like this into linear regression?

We can expand a categorical variable with k levels into $k - 1$ binary variables, with one level as a baseline.

Categorical Predictors

Some of the variables in the `autos` dataset are categorical, like the make of the car (e.g., alfa-romero, audi, bmw, etc.). How do we incorporate variables like this into linear regression?

We can expand a categorical variable with k levels into $k-1$ binary variables, with one level as a baseline.

For example, there are 22 makes of cars. Choose alfa-romero as the baseline. Then, we include 21 binary variables in our regression:

Categorical Predictors

Some of the variables in the `autos` dataset are categorical, like the make of the car (e.g., alfa-romero, audi, bmw, etc.). How do we incorporate variables like this into linear regression?

We can expand a categorical variable with k levels into $k-1$ binary variables, with one level as a baseline.

For example, there are 22 makes of cars. Choose alfa-romero as the baseline. Then, we include 21 binary variables in our regression:

$$y = \beta_0 + \beta_1 X_{audi} + \beta_2 X_{bmw} + \beta_3 X_{honda} + ... + \beta_{21} X_{volvo} + ... + \epsilon$$

Categorical Predictors

$$y = \beta_0 + \beta_1 X_{audi} + \beta_2 X_{bmw} + \beta_3 X_{honda} + ... + \beta_{21} X_{volvo} + \epsilon$$

Let's assume that these are the only variables in our regression.

Categorical Predictors

$$y = \beta_0 + \beta_1 X_{audi} + \beta_2 X_{bmw} + \beta_3 X_{honda} + ... + \beta_{21} X_{volvo} + \epsilon$$

Let's assume that these are the only variables in our regression.

- How would you interpret β_0?

Categorical Predictors

$$y = \beta_0 + \beta_1 X_{audi} + \beta_2 X_{bmw} + \beta_3 X_{honda} + ... + \beta_{21} X_{volvo} + \epsilon$$

Let's assume that these are the only variables in our regression.

- How would you interpret β_0?
- How about β_1?

Categorical Predictors

$$y = \beta_0 + \beta_1 X_{audi} + \beta_2 X_{bmw} + \beta_3 X_{honda} + ... + \beta_{21} X_{volvo} + \epsilon$$

Let's assume that these are the only variables in our regression.

- How would you interpret β_0?
- How about β_1?
- We fit linear regression to this data, so we form:

$$\begin{pmatrix} \text{bmw} \\ \text{audi} \\ \text{bmw} \\ \text{alfa-romero} \\ \vdots \end{pmatrix} \longrightarrow X = \begin{pmatrix} 1 & 0 & 1 & 0 & \cdots & 0 \\ 1 & 1 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 1 & 0 & \cdots & 0 \\ 1 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \end{pmatrix}.$$

Categorical Predictors

$$y = \beta_0 + \beta_1 X_{audi} + \beta_2 X_{bmw} + \beta_3 X_{honda} + ... + \beta_{21} X_{volvo} + \epsilon$$

Let's assume that these are the only variables in our regression.

- How would you interpret β_0?
- How about β_1?
- We fit linear regression to this data, so we form:

$$\begin{pmatrix} \text{bmw} \\ \text{audi} \\ \text{bmw} \\ \text{alfa-romero} \\ \vdots \end{pmatrix} \longrightarrow X = \begin{pmatrix} 1 & 0 & 1 & 0 & \cdots & 0 \\ 1 & 1 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 1 & 0 & \cdots & 0 \\ 1 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \end{pmatrix}.$$

We also have a vector $\mathbf{y}$ with the prices of the cars. We'll calculate $\hat{\boldsymbol{\beta}} = (X^T X)^{-1} X^T \mathbf{y}$.

Categorical Predictors

$$y = \beta_0 + \beta_1 X_{audi} + \beta_2 X_{bmw} + \beta_3 X_{honda} + ... + \beta_{21} X_{volvo} + \epsilon$$

Let's assume that these are the only variables in our regression.

- How would you interpret β_0?
- How about β_1?
- We fit linear regression to this data, so we form:

$$\begin{pmatrix} \text{bmw} \\ \text{audi} \\ \text{bmw} \\ \text{alfa-romero} \\ \vdots \end{pmatrix} \longrightarrow X = \begin{pmatrix} 1 & 0 & 1 & 0 & \cdots & 0 \\ 1 & 1 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 1 & 0 & \cdots & 0 \\ 1 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \end{pmatrix}.$$

We also have a vector $\mathbf{y}$ with the prices of the cars. We'll calculate $\hat{\boldsymbol{\beta}} = (X^T X)^{-1} X^T \mathbf{y}$.

What do you think the estimates $\hat{\beta}_0$ and $\hat{\beta}_1$ will be?

Ordinal Predictors

An ordinal variable is like a categorical variable, except that the levels have an ordering.

Ordinal Predictors

An ordinal variable is like a categorical variable, except that the levels have an ordering.

For example, the number of cylinders is (arguably) an ordinal variable, with levels two, three, four, five, six, eight, twelve.

Ordinal Predictors

An ordinal variable is like a categorical variable, except that the levels have an ordering.

For example, the number of cylinders is (arguably) an ordinal variable, with levels two, three, four, five, six, eight, twelve.

For ordinal variables, we also expand the variable into $k - 1$ binary variables. But we often code it slightly differently:

Ordinal Predictors

An ordinal variable is like a categorical variable, except that the levels have an ordering.

For example, the number of cylinders is (arguably) an ordinal variable, with levels two, three, four, five, six, eight, twelve.

For ordinal variables, we also expand the variable into $k-1$ binary variables. But we often code it slightly differently:

$$\begin{pmatrix} \text{two} \\ \text{three} \\ \text{four} \\ \text{five} \\ \vdots \end{pmatrix} \longrightarrow X = \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 1 & 0 & \cdots & 0 \\ 1 & 1 & 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \end{pmatrix}.$$

Ordinal Predictors

An ordinal variable is like a categorical variable, except that the levels have an ordering.

For example, the number of cylinders is (arguably) an ordinal variable, with levels two, three, four, five, six, eight, twelve.

For ordinal variables, we also expand the variable into $k-1$ binary variables. But we often code it slightly differently:

$$\begin{pmatrix} \text{two} \\ \text{three} \\ \text{four} \\ \text{five} \\ \vdots \end{pmatrix} \longrightarrow X = \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 1 & 0 & \cdots & 0 \\ 1 & 1 & 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \end{pmatrix}.$$

So the predicted price of a car with three cylinders would be $\beta_0 + \beta_1$, while for one with four cylinders it would be $\beta_0 + \beta_1 + \beta_2$.

Ordinal Predictors

An ordinal variable is like a categorical variable, except that the levels have an ordering.

For example, the number of cylinders is (arguably) an ordinal variable, with levels two, three, four, five, six, eight, twelve.

For ordinal variables, we also expand the variable into $k-1$ binary variables. But we often code it slightly differently:

$$\begin{pmatrix} \text{two} \\ \text{three} \\ \text{four} \\ \text{five} \\ \vdots \end{pmatrix} \longrightarrow X = \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 1 & 0 & \cdots & 0 \\ 1 & 1 & 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \end{pmatrix}.$$

So the predicted price of a car with three cylinders would be $\beta_0 + \beta_1$, while for one with four cylinders it would be $\beta_0 + \beta_1 + \beta_2$.

Each coefficient β_j represents the change in price from level $j-1$ to level j (rather than from a common baseline to level j).

Multiple Regression with Categorical Predictors

In-Class Exercise

Now, modify your `lm` *function so that it automatically handles categorical variables.*

Hints:

- *Read the documentation on Pandas' "categorical" data type. It is analogous to* `R`*'s factors.*