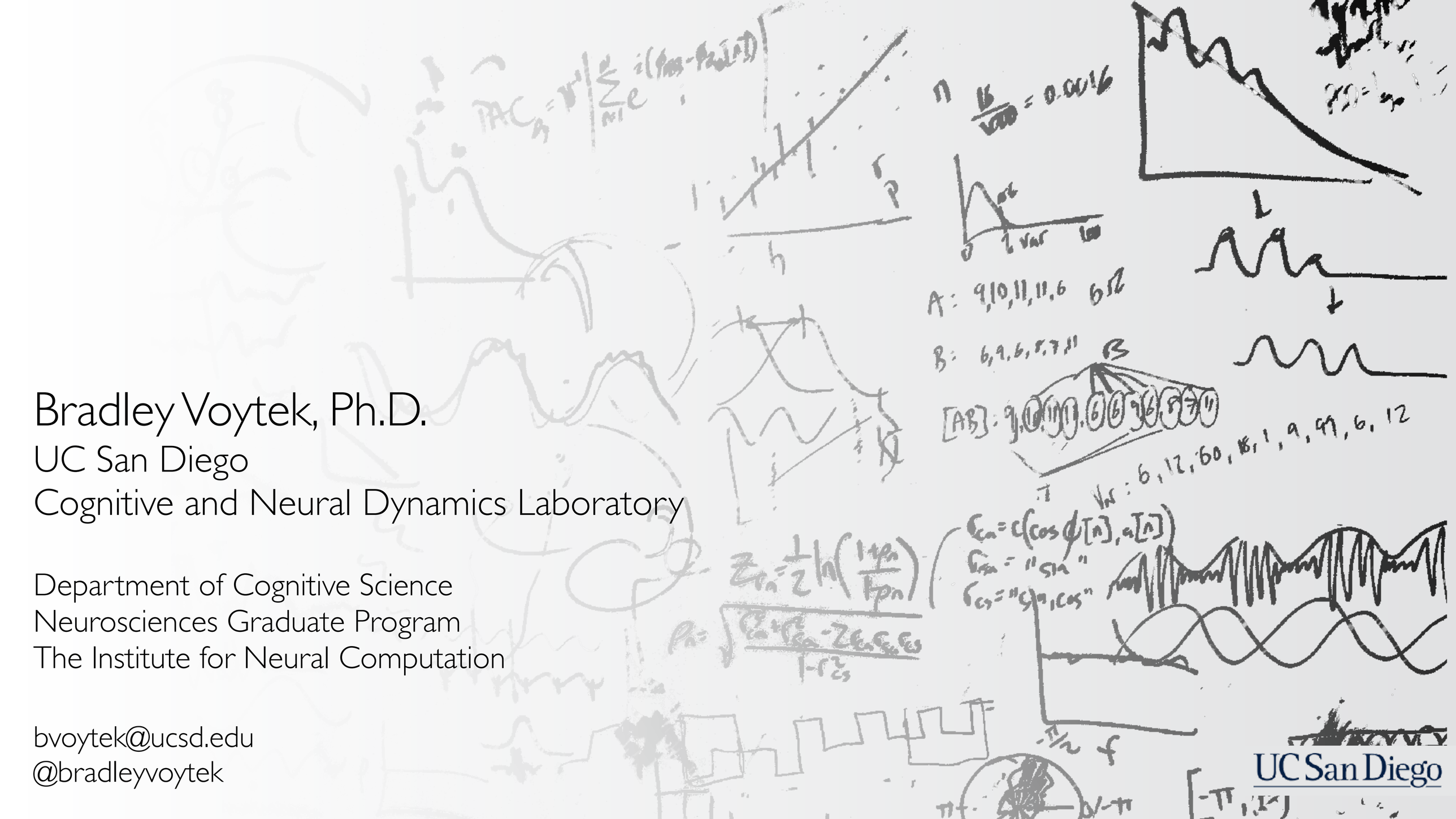


Bradley Voytek, Ph.D.

UC San Diego
Cognitive and Neural Dynamics Laboratory

Department of Cognitive Science
Neurosciences Graduate Program
The Institute for Neural Computation

bvoytek@ucsd.edu
@bradleyvoytek

UC San Diego

Administrative stuff

Halicioglu Data Science Institute *tomorrow*, Friday, March 2.

The student sessions will be from 2:00p to 5:00p in the lobby of the Qualcomm Institute down in Warren.

Research posters will be in the lobby beginning at 2p

COGS 108 student project presentations and judging will begin at 3:00p.

Free cookies and light refreshments!

Administrative stuff

WiDS San Diego: Global Women in Data Science Conference

Rady Data Analytics Club
Monday, March 5, 2018 from 4:30 PM to 7:00 PM (PST)
San Diego, CA

Ticket Information

TYPE	END		QUANTITY
UCSD Students, Faculty and Staff	Mar 5, 2018	Free	0
Data Science Professionals and Others	Mar 5, 2018	Free	0

Register

Source: https://www.eventbrite.com/e/wids-san-diego-global-women-in-data-science-conference-tickets-42659327196

COGS 108
Data Science in Practice

Modeling and Model Validation

UC San Diego

Simple models in practice

Import libraries

We will be using the library statsmodels to run our linear regression

```
import numpy as np
import statsmodels.formula.api as smf
import pandas as pd
import scipy as sp

%matplotlib notebook
%config InlineBackend.figure_format = 'retina'
%matplotlib inline
import matplotlib.pyplot as plt
```

Simple models in practice

1a. Fitting a line

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

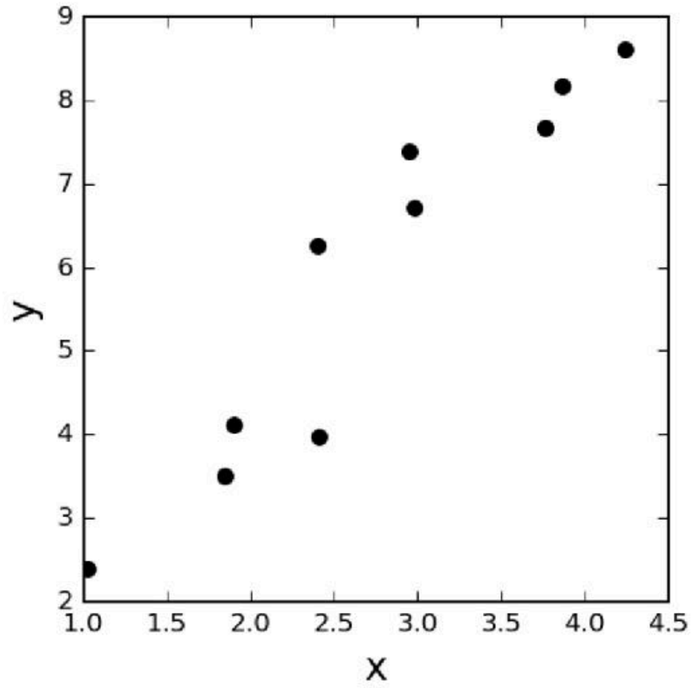
```
# Define true statistics relating x and y
N_points = 10
true_beta0 = 0
true_beta1 = 2
noise_stdev = 1

# Set random seed
np.random.seed(0)

# Generate correlated data
x = np.random.randn(N_points) + 2
y = true_beta0 + true_beta1*x + np.random.randn(N_points)*noise_stdev
print('x=', x)
print('y=', y)
```

Simple models in practice

```
# Plot x and y
plt.figure(figsize=(4,4))
plt.plot(x, y, 'k.', ms=12)
plt.xlabel('x',size=15)
plt.ylabel('y',size=15)
plt.show()
```



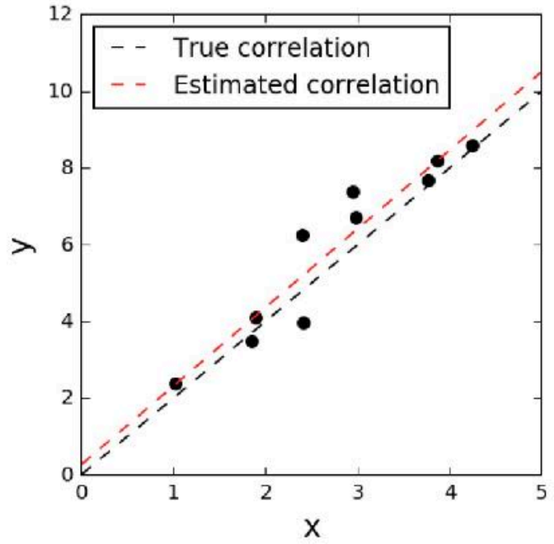
Simple models in practice

```
# Fit line to data
A = np.vstack([x, np.ones(len(x))]).T
m, b = np.linalg.lstsq(A, y)[0]
print('True statistics: y =', true_beta1, '*x +', true_beta0)
print('Estimated stats: y =', m, '*x +', b)
print('R squared (fraction of variance explained) =',
      np.round(sp.stats.pearsonr(x,y)[0],2))
```

```
True statistics: y = 2 *x + 0
Estimated stats: y = 2.04994401661 *x + 0.26389814044
R squared (fraction of variance explained) = 0.95
```

Simple models in practice

```
# Plot fitted line
plt.figure(figsize=(4,4))
plt.plot(x, y, 'k.', ms=12)
plt.plot([0,5], [true_beta1*x+true_beta0 for x in [0,5]], 'k--',label='True correlation')
plt.plot([0,5], [m*x+b for x in [0,5]], 'r--',label='Estimated correlation')
plt.xlabel('x',size=15)
plt.ylabel('y',size=15)
plt.xlim((0,5))
plt.legend(loc='best')
plt.show()
```



Simple models in practice

1b. Outliers and normality of errors

Minimizing the squared error of the line fit works well if our data is normally distributed.

In this section, we simulate some data in which the error is not normally distributed, and see the fitted line is not as accurate.

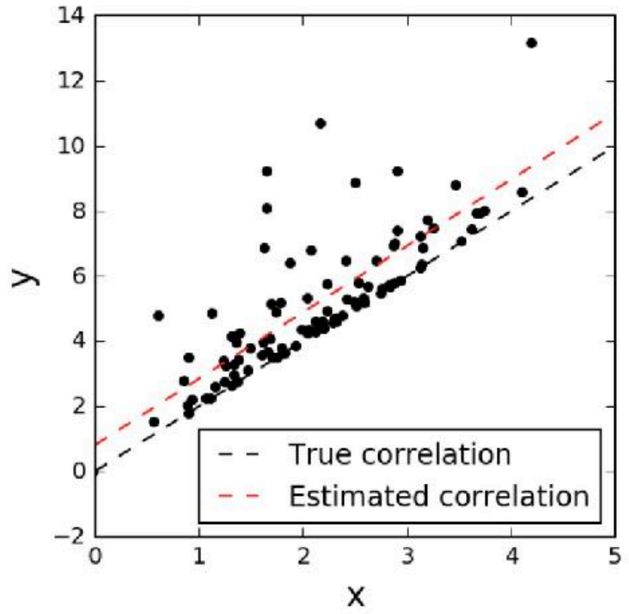
```
# Simulate data with non-normal distribution of error
np.random.seed(1)
N_points = 100
x = np.random.randn(N_points) + 2
y = true_beta0 + true_beta1*x + np.random.randn(N_points)**2
```

```
# Fit line to data
A = np.vstack([x, np.ones(len(x))]).T
m, b = np.linalg.lstsq(A, y)[0]
print('True statistics: y =', true_beta1, '*x +', true_beta0)
print('Estimated stats: y =', m, '*x +', b)
print('R squared (fraction of variance explained) =',np.round(sp.stats.pearsonr(x,y)[0],2))
```

```
True statistics: y = 2 *x + 0
Estimated stats: y = 2.04202383003 *x + 0.805367000302
R squared (fraction of variance explained) = 0.82
```

Simple models in practice

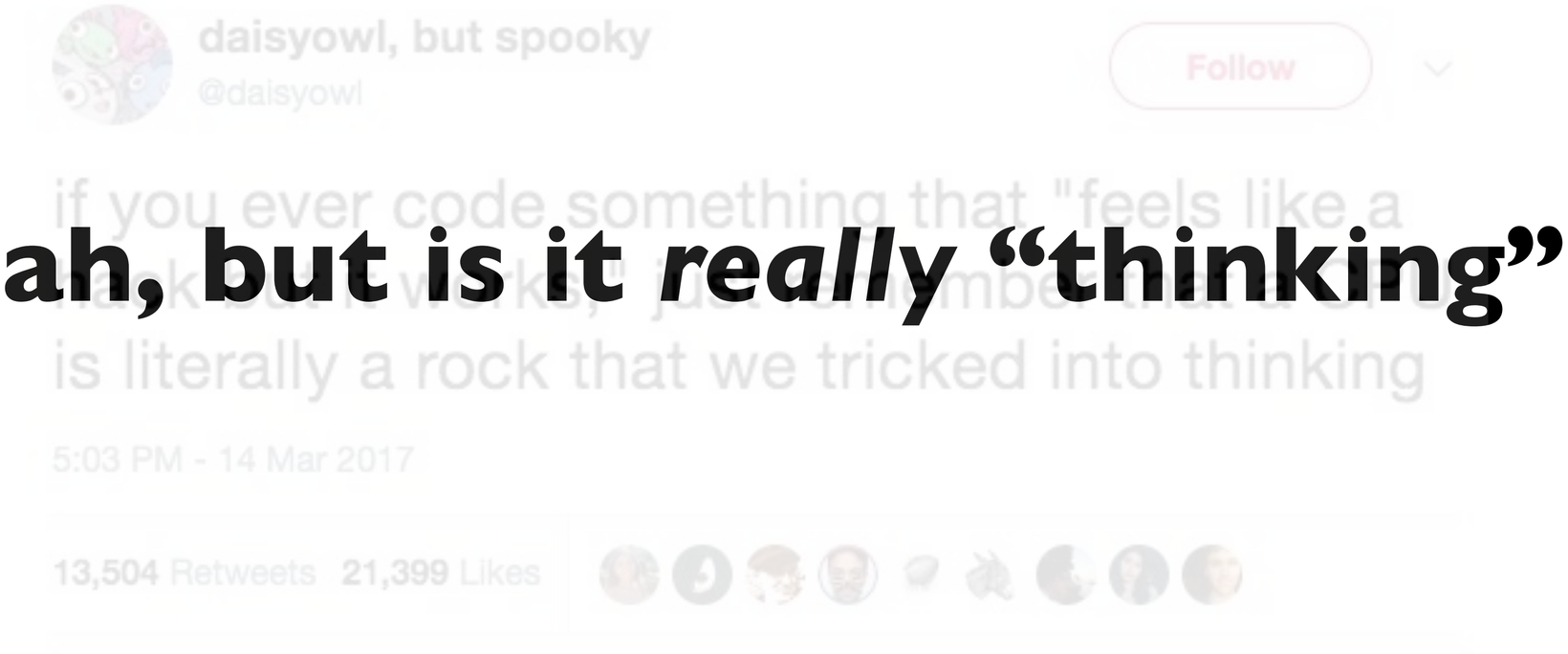
```
# Plot fitted line
plt.figure(figsize=(4,4))
plt.plot(x, y, 'k.', ms=8)
plt.plot([0,5], [true_beta1*x+true_beta0 for x in [0,5]], 'k--',label='True correlation')
plt.plot([0,5], [m*x+b for x in [0,5]], 'r--', label='Estimated correlation')
plt.xlabel('x',size=15)
plt.ylabel('y',size=15)
plt.xlim((0,5))
plt.legend(loc='best')
plt.show()
```



Machine Learning basics

Cognitive Science

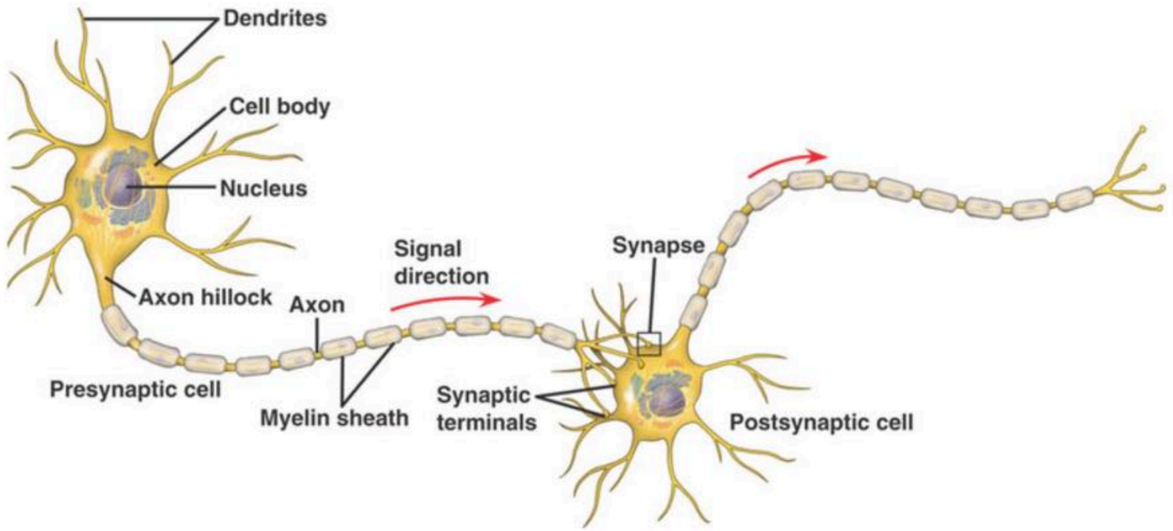
Cognitive Science



Yeah, but is it *really* “thinking”?

Cogsci: Convergence

WHAT IS A NEURON?

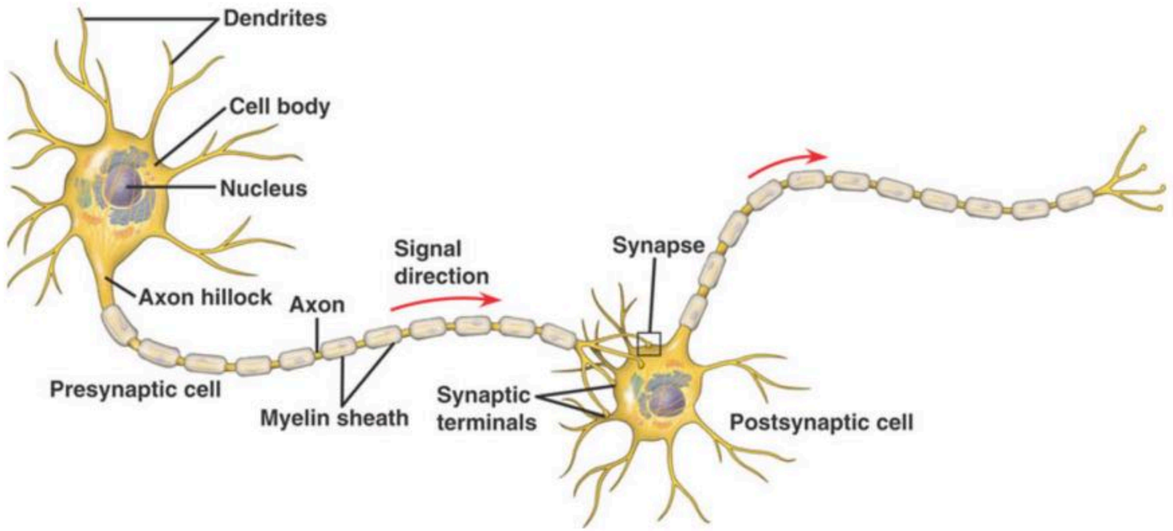


- Receives signal on synapse
- When trigger sends signal on axon

Source: A Jalis, Senior Technical Trainer, Amazon Web Services

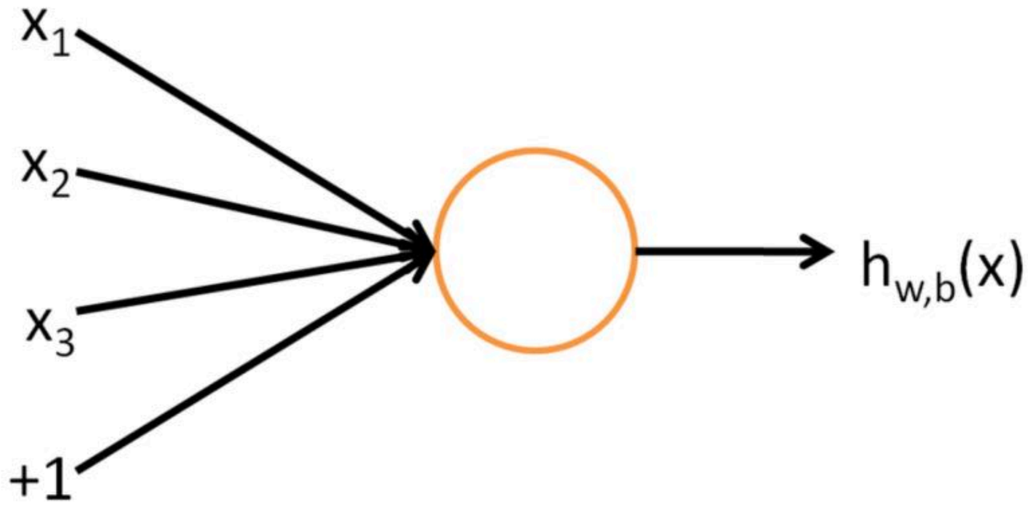
Cogsci: Convergence

WHAT IS A NEURON?



- Receives signal on synapse
- When trigger sends signal on axon

MATHEMATICAL NEURON



- Mathematical abstraction, inspired by biological neuron
- Either on or off based on sum of input

Source: A Jalis, Senior Technical Trainer, Amazon Web Services

Cogsci: Divergence

The learning procedure, in its current form, is not a plausible model of learning in brains. However, applying the procedure to various tasks shows that interesting internal representations can be constructed by gradient descent in weight-space, and this suggests that it is worth looking for more biologically plausible ways of doing gradient descent in neural networks.

UC San Diego

Source: Rumelhart, Hinton, Williams, *Nature* 1986

One hell of a divergence

Google AI defeats world's Go champion in gripping 'man vs. machine' film

UC San Diego

Source: Every tech article from 2010-2017

Data and Deep Learning

Google AI defeats world's Go champion in gripping 'man vs. machine' film

The AI Revolution: Why You Need to Learn About Deep Learning

UC San Diego

Source: Every tech article from 2010-2017

Data and Deep Learning

Google AI defeats world's Go champion in gripping 'man vs. machine' film

The AI Revolution: Why You Need to Learn About

Google's artificial intelligence can actually help the environment

UC San Diego

Source: Every tech article from 2010-2017

Data and Deep Learning

Google AI defeats world's Go champion in gripping 'man vs. machine' film

The AI Revolution: Why You Need to Learn About Deep Learning

Google's artificial intelligence can actually help the environment

A computer's newfound 'intuition' beats world poker champs

UC San Diego

Source: Every tech article from 2010-2017

Data and Deep Learning

Google AI defeats world's Go champion in gripping 'man vs. machine' film

The AI Revolution: Why You Need to Learn About Deep Learning

Google's artificial intelligence can actually help the environment

A computer's newfound 'intuition' beats world poker champs

'AI can solve world's biggest problems' - Google Brain engineer

UC San Diego

Source: Every tech article from 2010-2017

What is machine learning?

"Machine Learning (ML) is a fascinating field of artificial intelligence (AI) research and practice where we investigate how computer agents can improve their perception, cognition, and action with experience. Machine Learning is about machines improving from data, knowledge, experience, and interaction." - Manuela Veloso, Head of ML at Carnegie Mellon

What is machine learning?

"Machine learning is the science of getting computers to act without being explicitly programmed" - Andrew Ng, Stanford, ex-Google, formerly chief scientist at Baidu, founder Coursera

What is machine learning?

"Machine learning is a method of data analysis that automates analytical model building." - SAS, the largest installed user base of analytics software, revenue of $3.2 billion/year

What does ML look like?

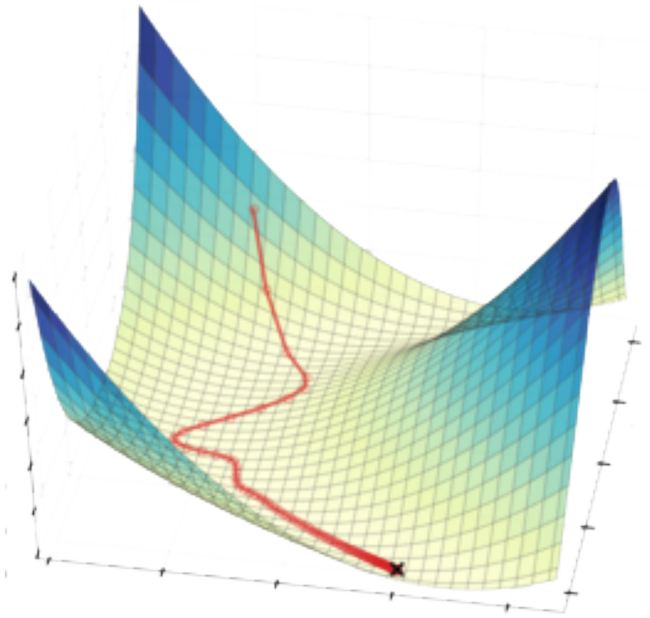
- Algorithms that can improve in performance with training or experience
- Typically by fitting large numbers of parameters
- Usually used in situations where it is difficult to define rules by hand
  - Face detection
  - Stock market prediction
  - Spam filters

What is ML really?

What is ML really?

What is ML really?

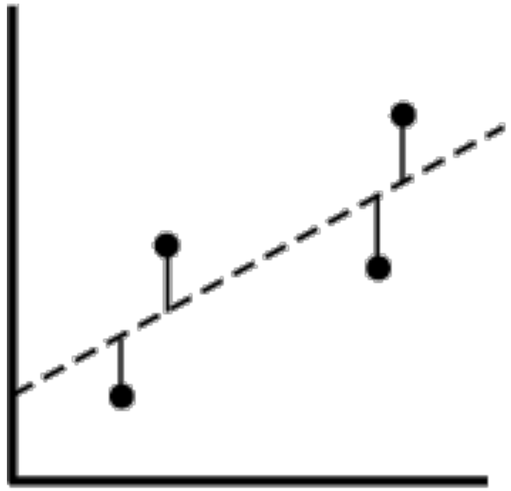
Pick a set of parameter values that minimize some objective function



What is ML really?

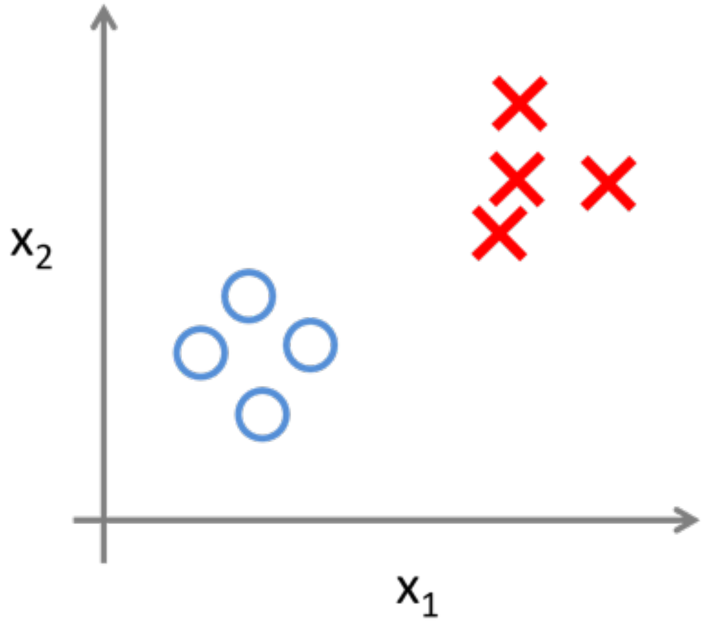
Pick a set of parameter values that minimize some objective function

$$\text{argmin}_{a,b} \sum_{i=1:n} (y_i - (ax_i + b))^2$$

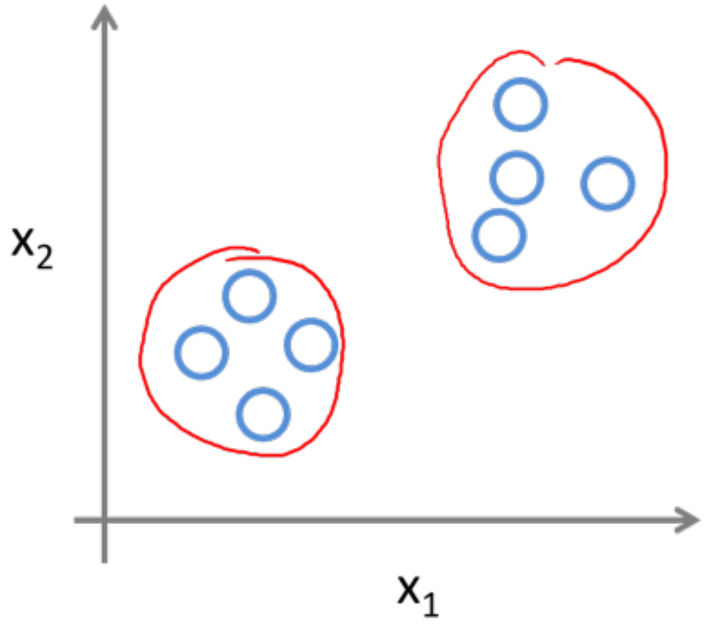


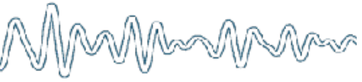
Two major modes of ML

Supervised Learning



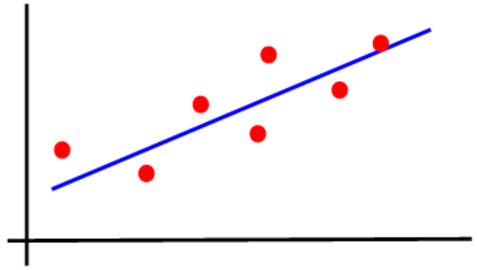
Unsupervised Learning





Three canonical problems

1. Regression - supervised
   - estimate parameters, e.g. of weight vs height

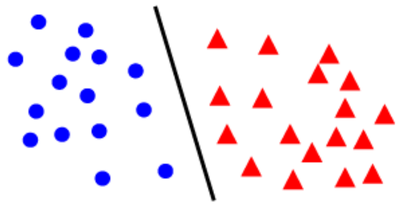


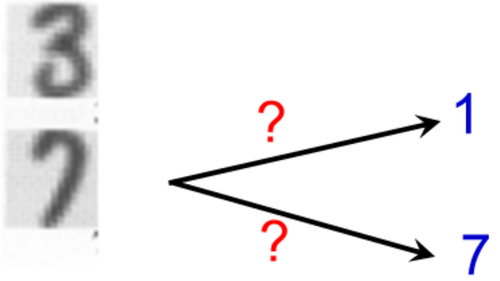
Source: Andrew Zisserman

Three canonical problems

1. Regression - supervised
   - estimate parameters, e.g. of weight vs height

2. Classification - supervised
   - estimate class, e.g. handwritten digit classification





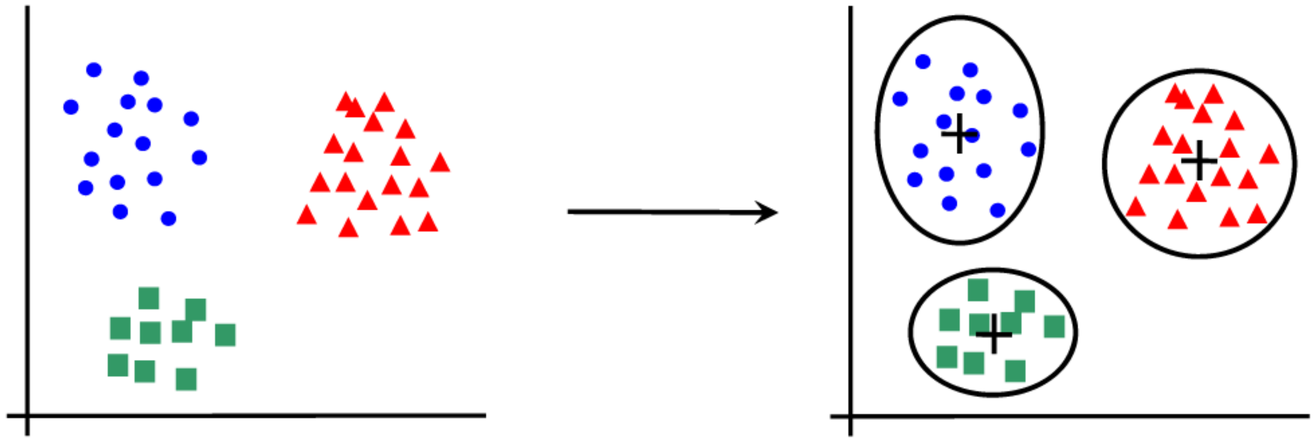
Source: Andrew Zisserman

Three canonical problems

1. Regression - supervised
   - estimate parameters, e.g. of weight vs height

2. Classification - supervised
   - estimate class, e.g. handwritten digit classification

3. Unsupervised learning – model the data
   - clustering



Source: Andrew Zisserman

Decision trees

- A sequence of tests.
- Representation very natural for humans.
- Style of many "How to" manuals and trouble-shooting procedures.

Decision trees

- http://www.r2d3.us/visual-intro-to-machine-learning-part-1/

Overfitting

- "With four parameters I can fit an elephant and with five I can make him wiggle his trunk."
    - John von Neumann

Overfitting

- "With four parameters I can fit an elephant and with five I can make him wiggle his trunk."
    - John von Neumann

Drawing an elephant with four complex parameters

Jürgen Mayer
Max Planck Institute of Molecular Cell Biology and Genetics, Pfotenhauerstr. 108, 01307 Dresden, Germany

Khaled Khairy
European Molecular Biology Laboratory, Meyerhofstraße. 1, 69117 Heidelberg, Germany

Jonathon Howard
Max Planck Institute of Molecular Cell Biology and Genetics, Pfotenhauerstr. 108, 01307 Dresden, Germany

(Received 20 August 2008; accepted 5 October 2009)

We define four complex numbers representing the parameters needed to specify an elephantine shape. The real and imaginary parts of these complex numbers are the coefficients of a Fourier coordinate expansion, a powerful tool for reducing the data required to define shapes. © *2010 American Association of Physics Teachers.*
[DOI: 10.1119/1.3254017]

Overfitting

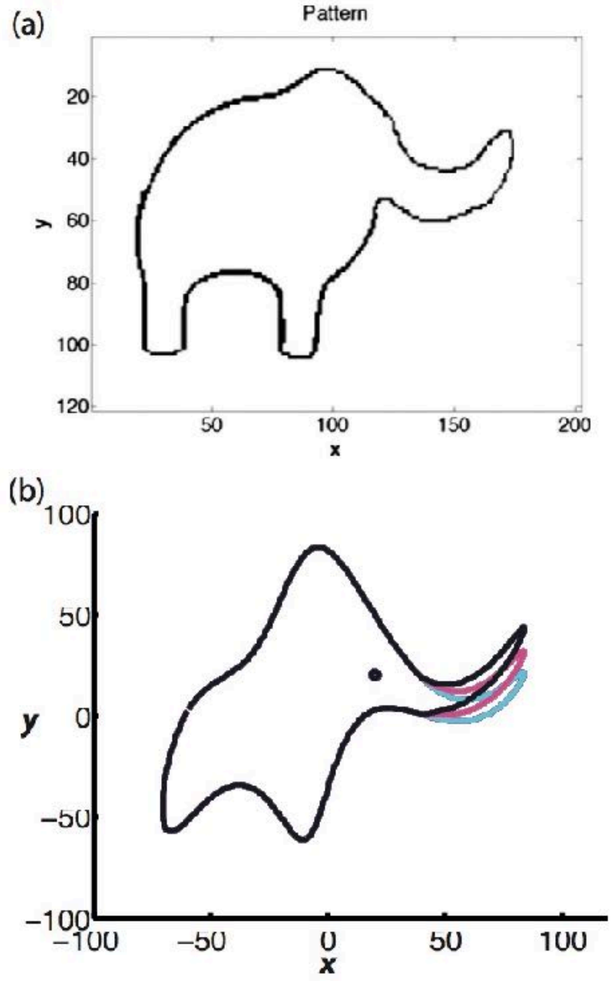
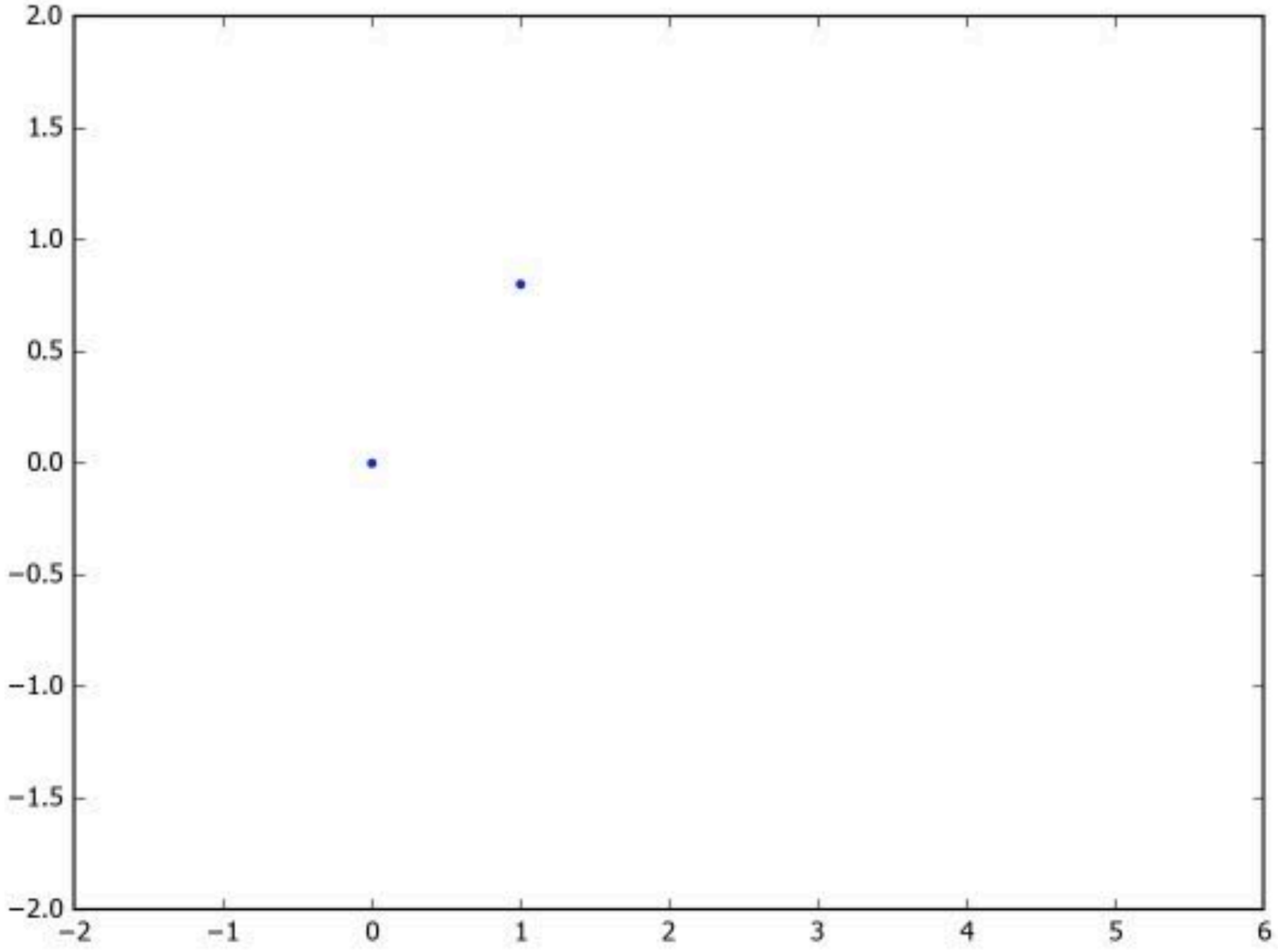
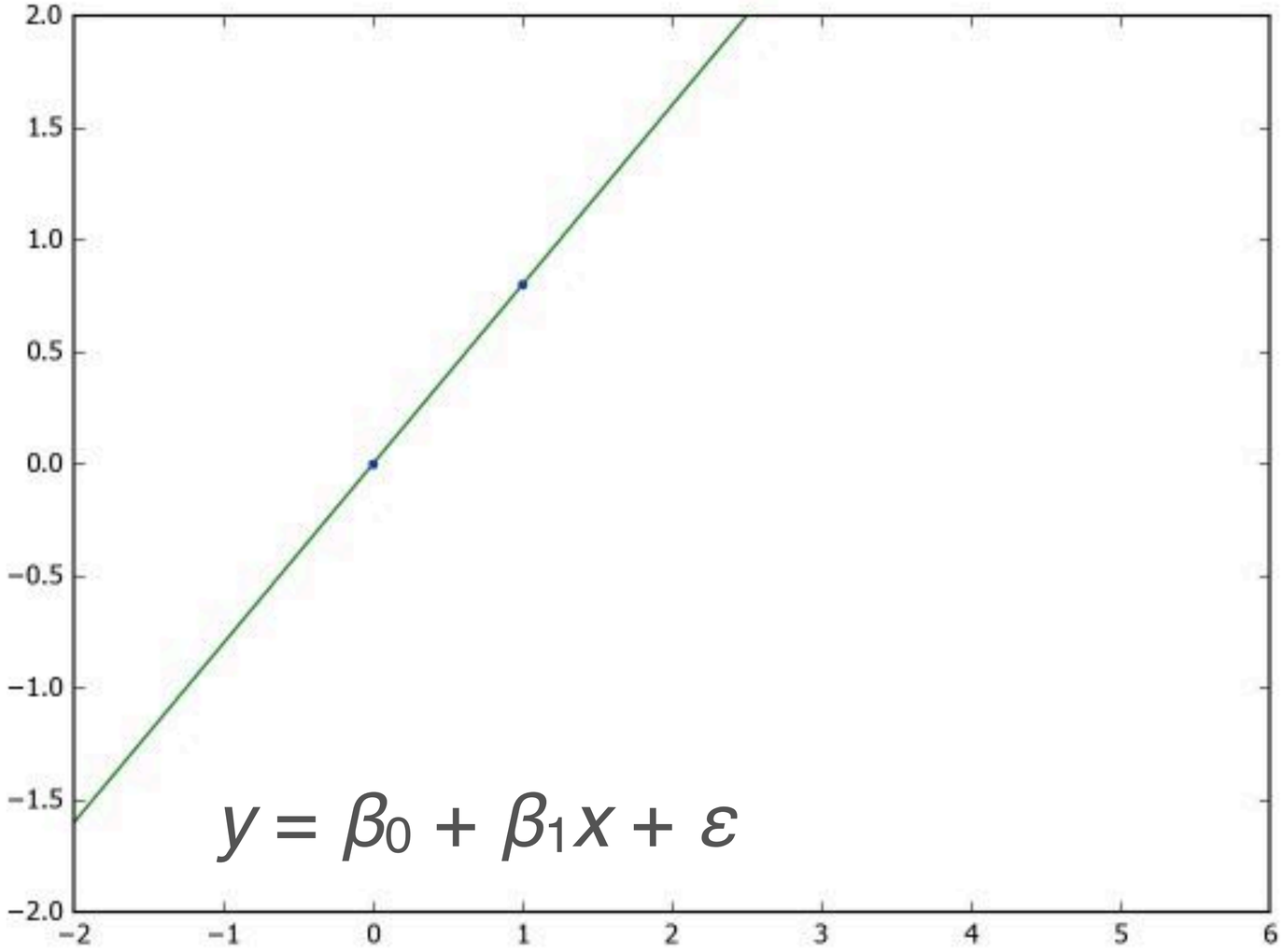


Fig. 1. (a) Outline of an elephant. (b) Three snapshots of the wiggling trunk.

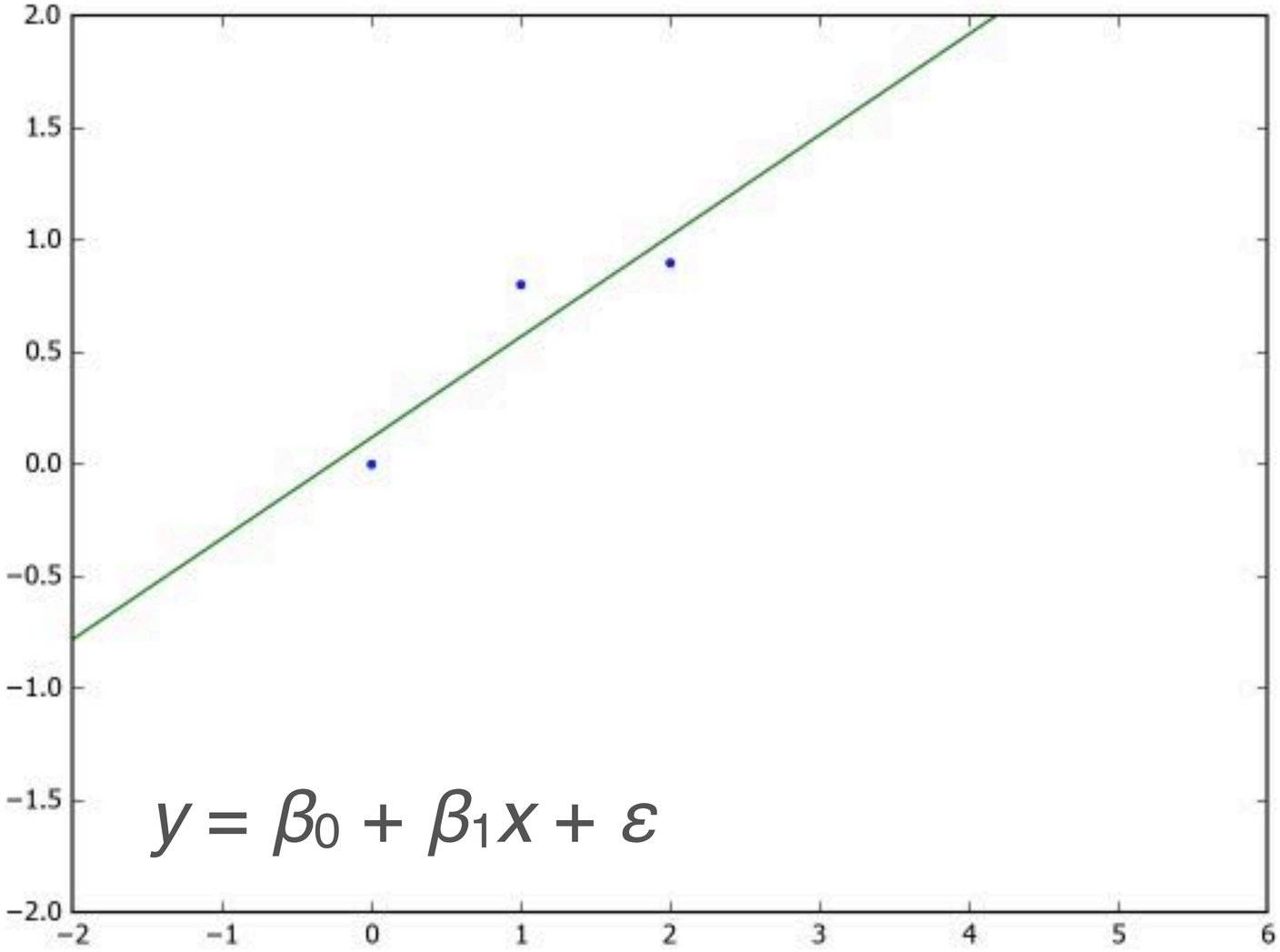
Overfitting



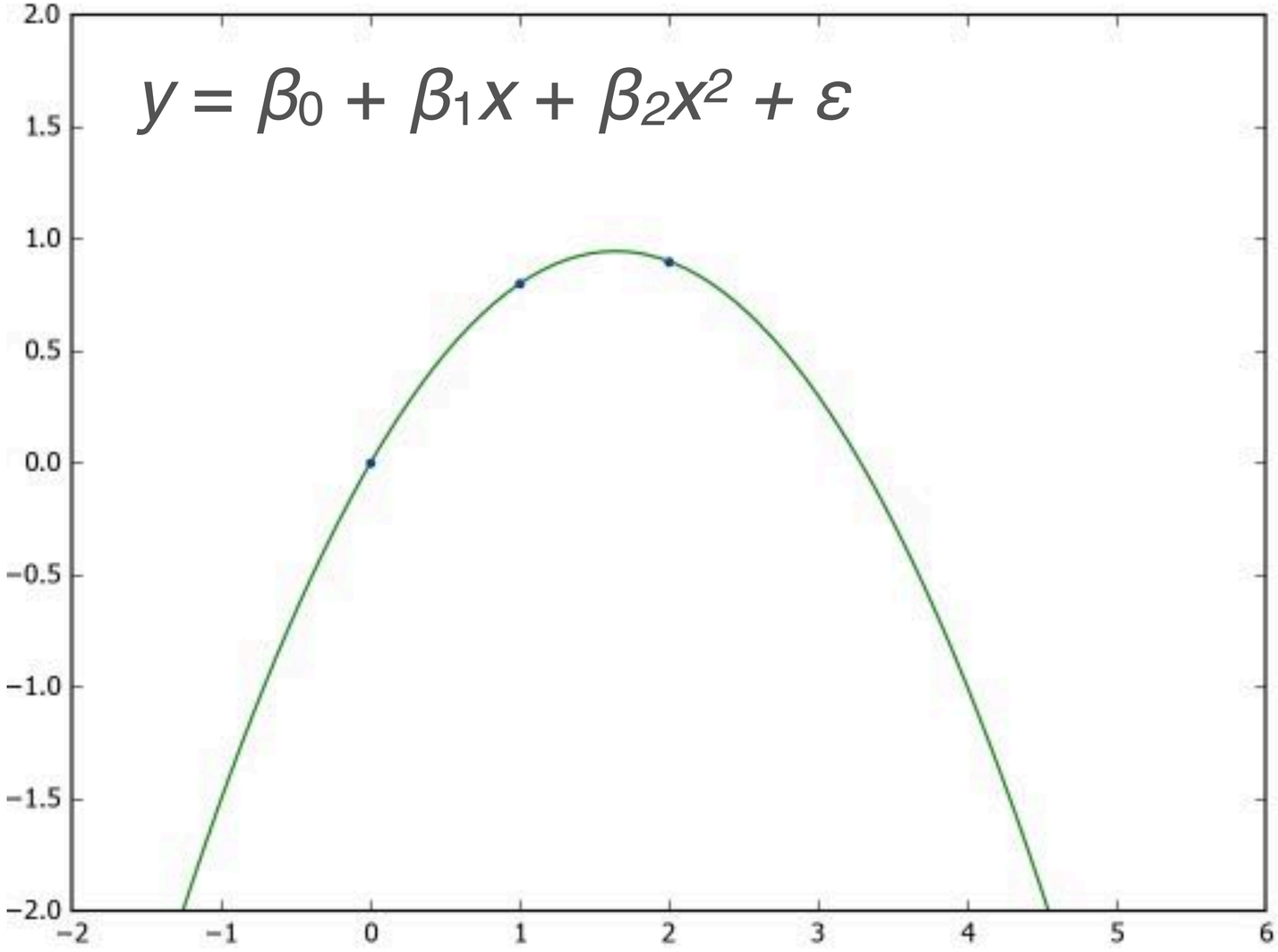
Overfitting



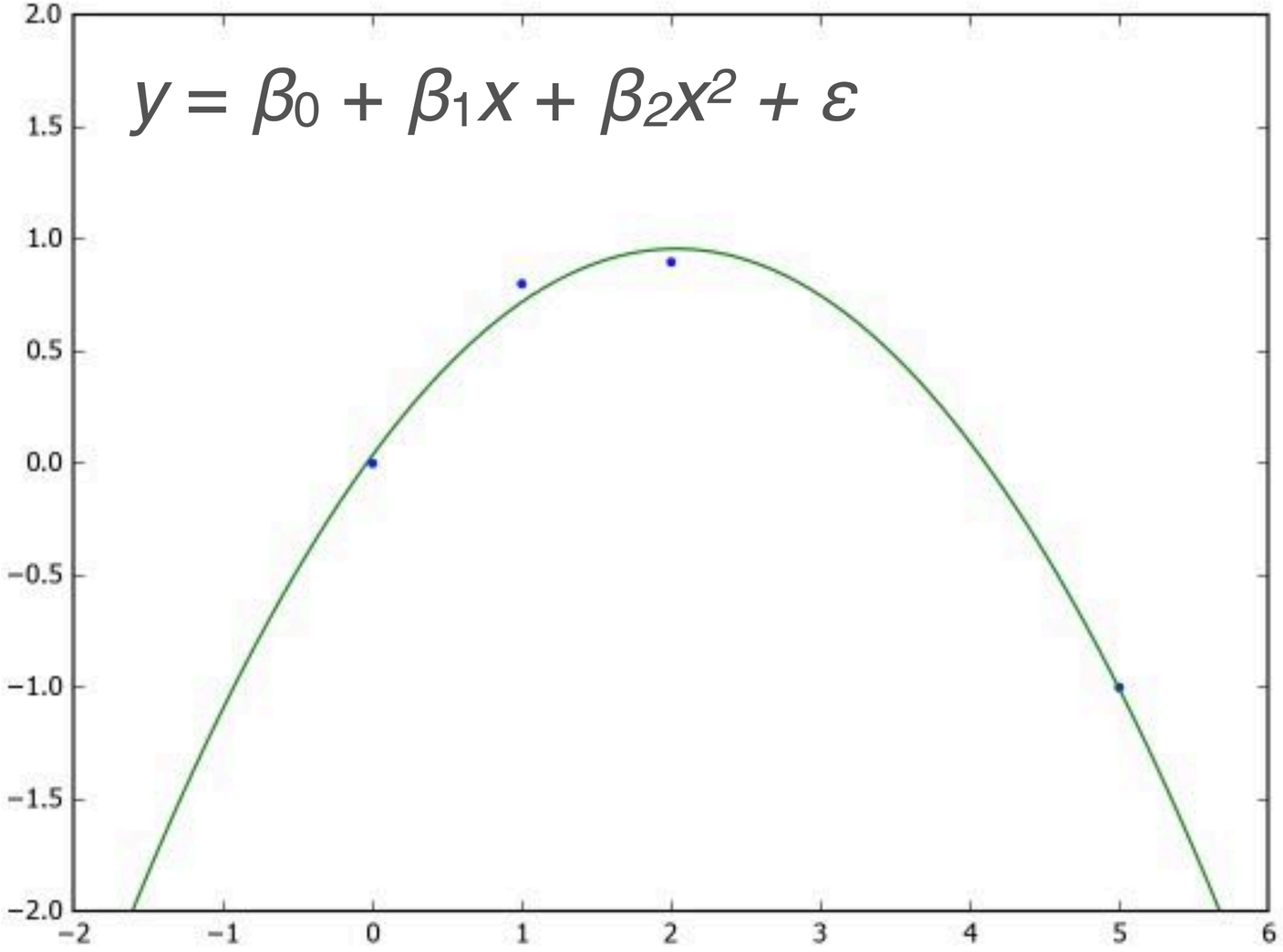
Overfitting



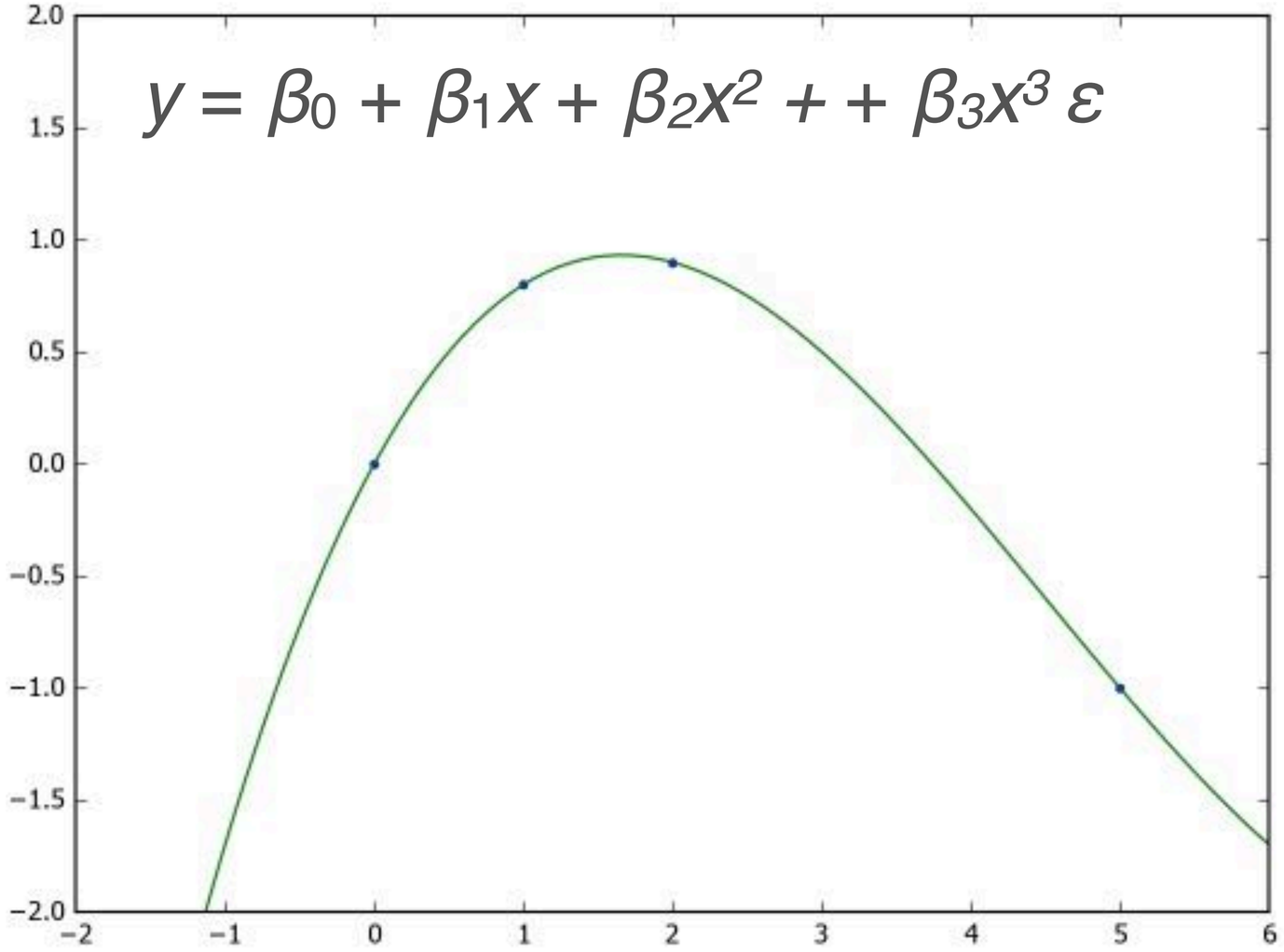
Overfitting



Overfitting



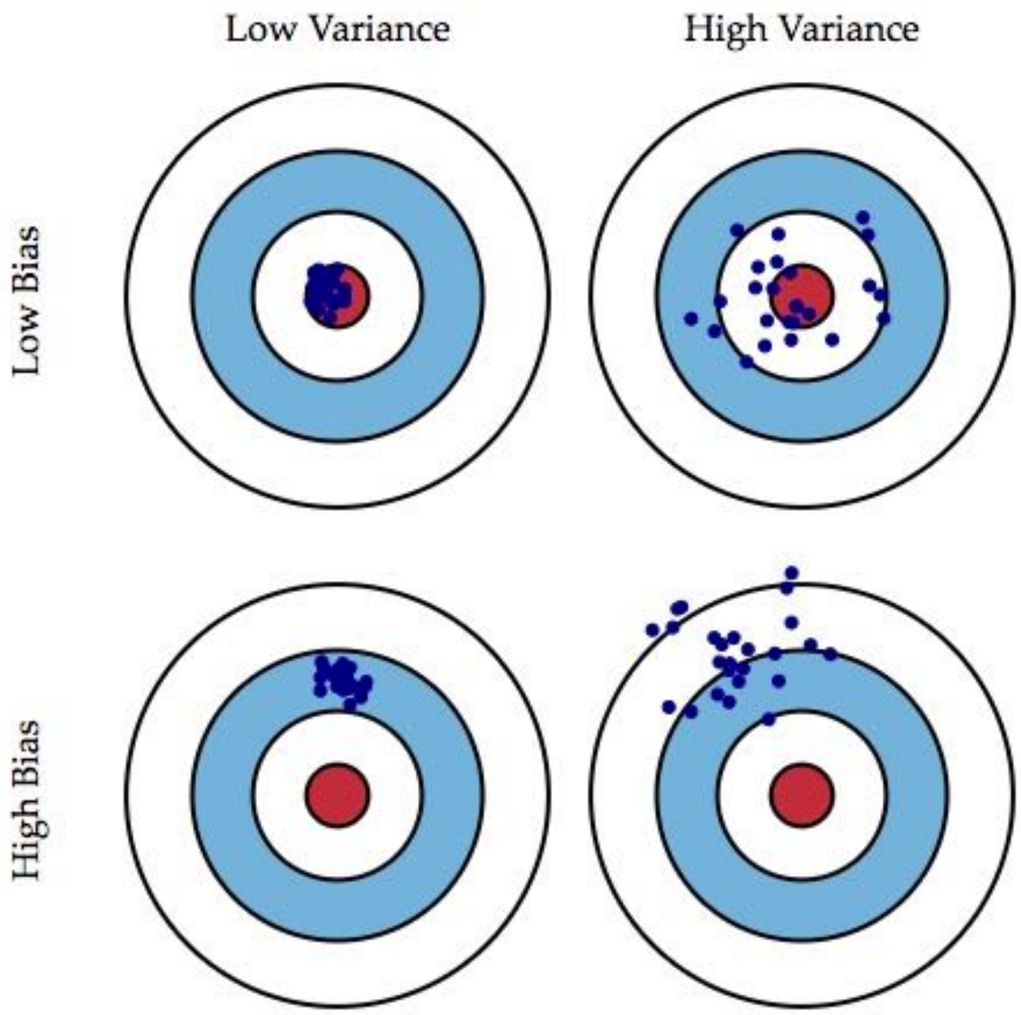
Overfitting



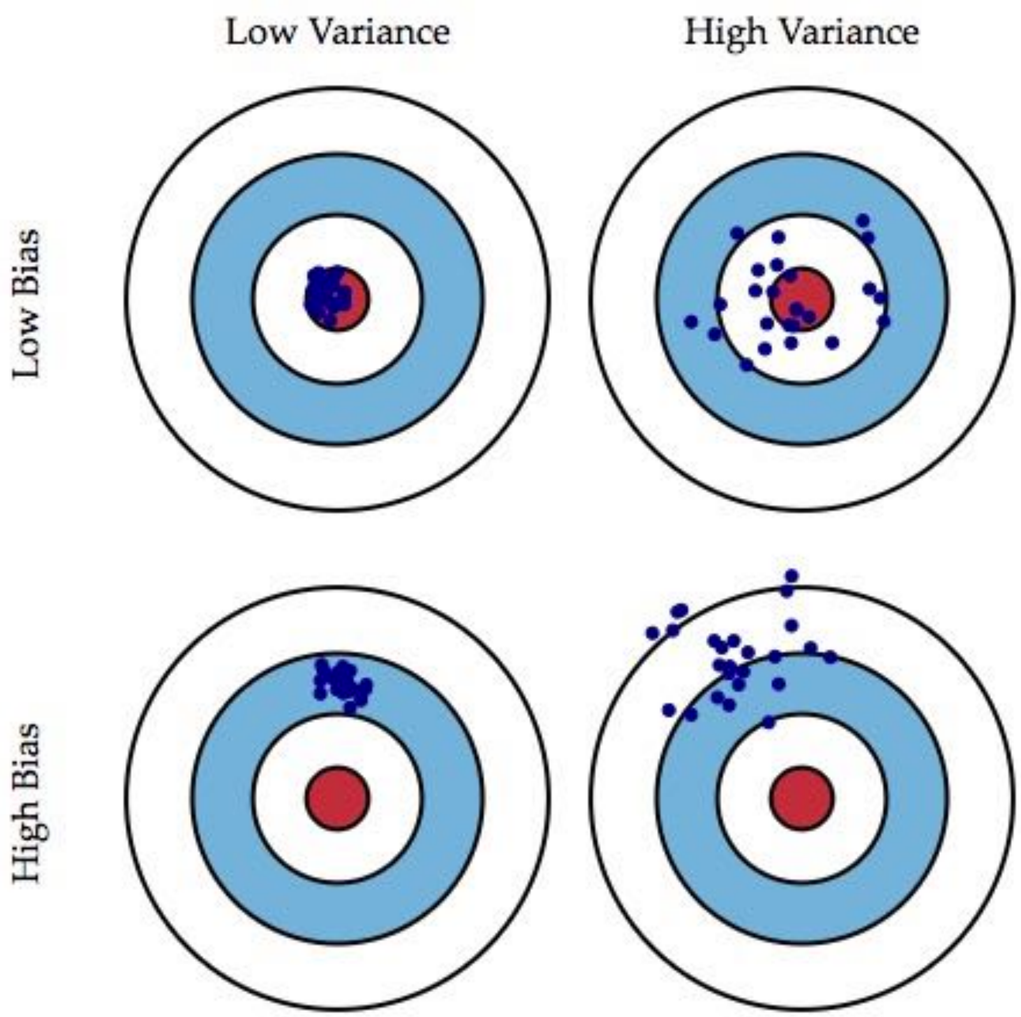
Model selection

- "Don't be too quick to turn on the computer. Bypassing the brain to compute by reflex is a sure recipe for disaster."
    - Good and Hardin, *Common Errors in Statistics (and How to Avoid Them)*

Bias/variance tradeoff



Bias/variance tradeoff



"Essentially, bias is how removed a model's predictions are from correctness, while variance is the degree to which these predictions vary between model iterations."

Source: kdnuggets

Rigor and intuition

In today's pattern recognition class my professor talked about PCA, eigenvectors & eigenvalues.

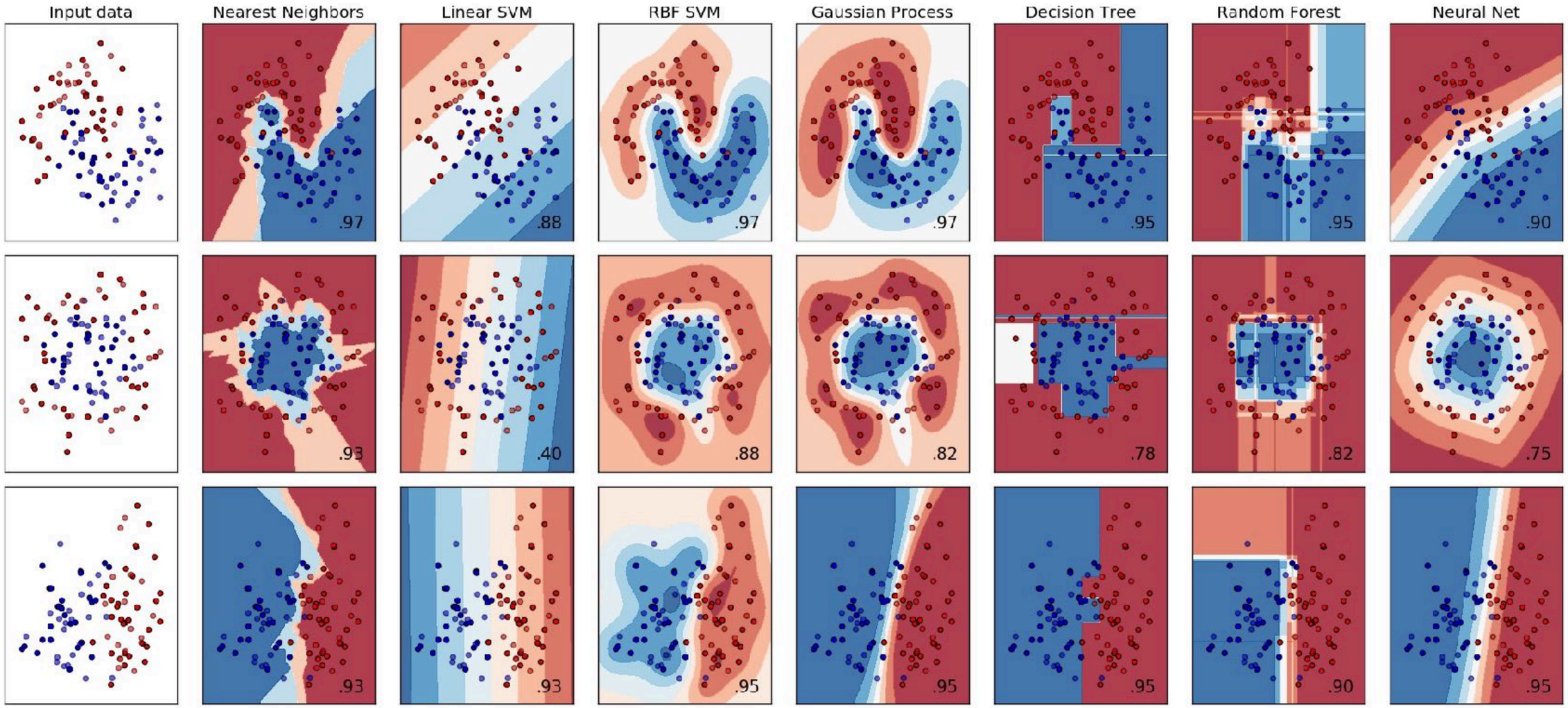
I got the mathematics of it. If I'm asked to find eigenvalues etc. I'll do it correctly like a machine. But I didn't **understand** it. I didn't get the purpose of it. I didn't get the feel of it. I strongly believe in

> you do not really understand something unless you can explain it to your grandmother -- Albert Einstein

Well, I can't explain these concepts to a layman or grandma.

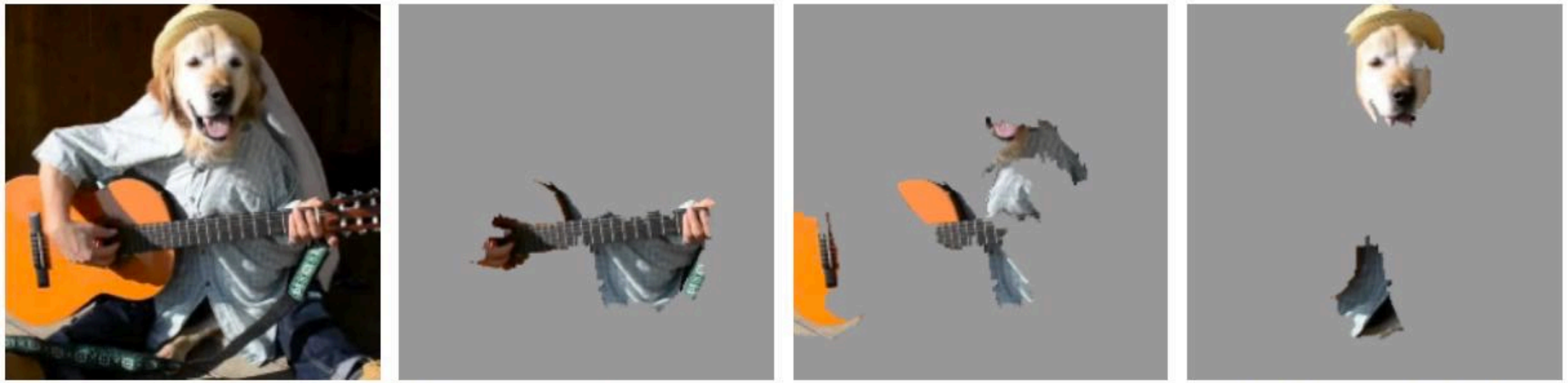
1. Why PCA, eigenvectors & eigenvalues? What was the *need* for these concepts?
2. How would you explain these to a layman?

Interpretability



Source: scikit-learn

Interpretability

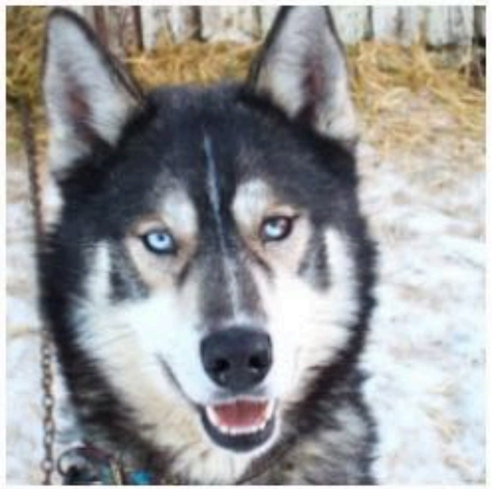


(a) Original Image (b) Explaining *Electric guitar* (c) Explaining *Acoustic guitar* (d) Explaining *Labrador*

Figure 4: Explaining an image classification prediction made by Google's Inception neural network. The top 3 classes predicted are "Electric Guitar" ($p = 0.32$), "Acoustic guitar" ($p = 0.24$) and "Labrador" ($p = 0.21$)

Source: Ribeiro, Singh, Guestrin, *KDD* 2016

Interpretability



(a) Husky classified as wolf

(b) Explanation

Figure 11: Raw data and explanation of a bad model's prediction in the "Husky vs Wolf" task.

Source: Ribeiro, Singh, Guestrin, *KDD* 2016

Interpretability

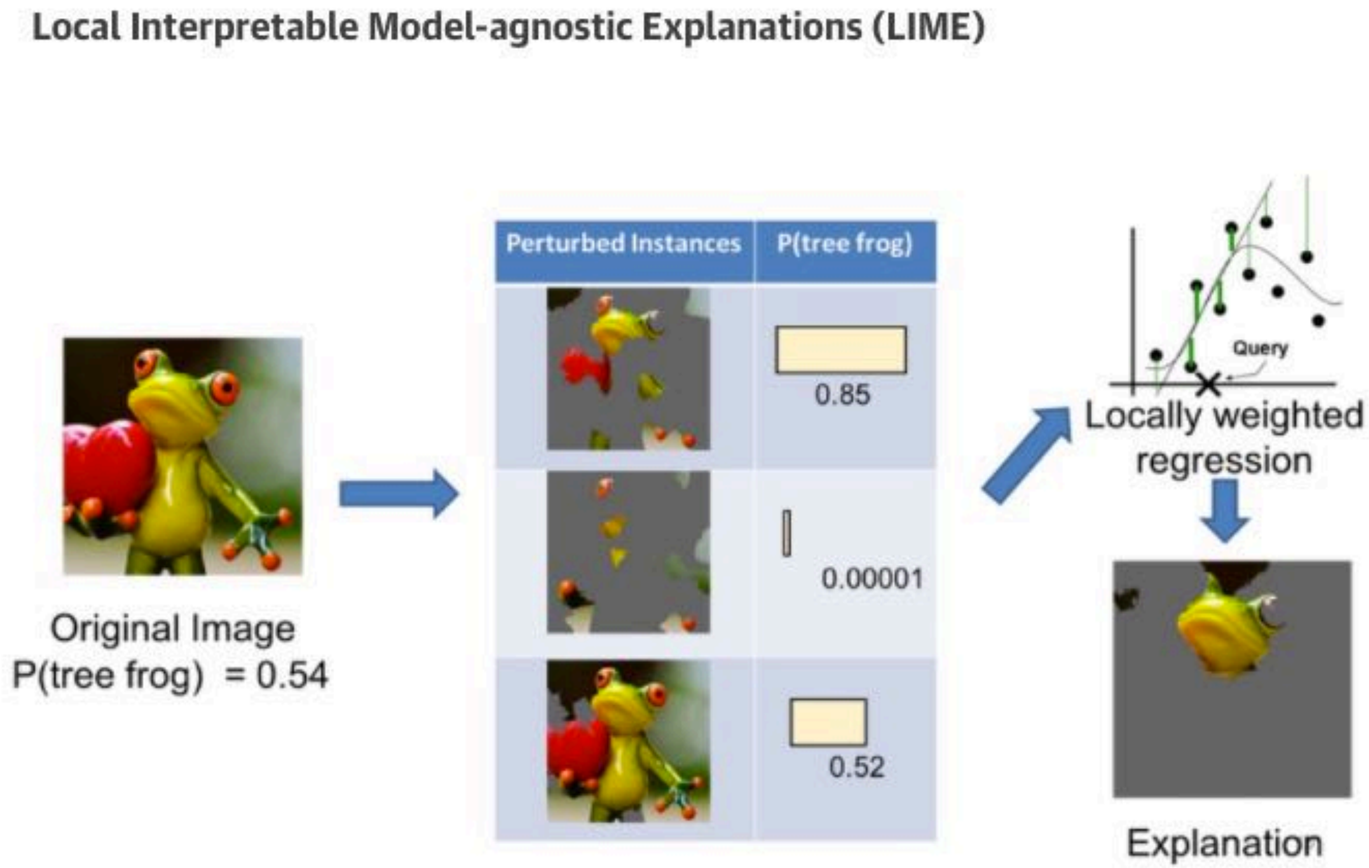


Figure 15. An illustration of the LIME process in which a weighted linear model is used to explain a single prediction from a complex neural network. Figure courtesy of Marco Tulio Ribeiro; image used with permission.

Source: Hall, Phan, Ambati, *O'Reilly Media* 2017

Feature selection

- In many cases, your features are given

Feature selection

- In many cases, your features are given
- But often you have to define your features

Feature selection

- Imagine creating a spam filter, when all you have is email text

Feature selection

- Imagine creating a spam filter, when all you have is email text
- How do you define your features in this case?

Feature selection

- Imagine creating a spam filter, when all you have is email text
- How do you define your features in this case?
  - How many words per email?

Feature selection

- Imagine creating a spam filter, when all you have is email text
- How do you define your features in this case?
  - How many words per email?
  - Frequency of misspellings?

Feature selection

- Imagine creating a spam filter, when all you have is email text
- How do you define your features in this case?
  - How many words per email?
  - Frequency of misspellings?
  - Domain of sender?

Feature selection

- Imagine creating a spam filter, when all you have is email text
- How do you define your features in this case?
  - How many words per email?
  - Frequency of misspellings?
  - Domain of sender?
  - Binary "contains_word_viagra"?

Feature selection

- Imagine creating a spam filter, when all you have is email text
- How do you define your features in this case?
  - How many words per email?
  - Frequency of misspellings?
  - Domain of sender?
  - Binary "contains_word_viagra"?

- How do you choose *meaningful* or *useful* features?

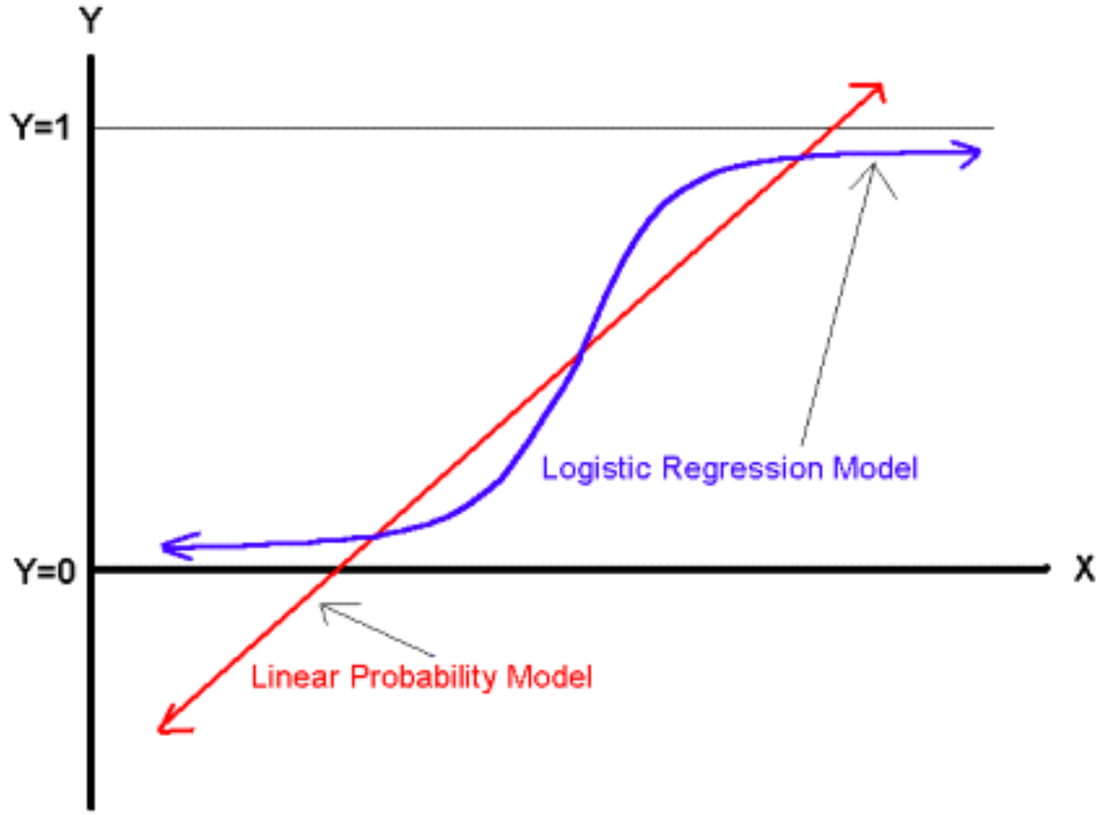
Binarization (dummy variables)

- Imagine creating a spam filter, when all you have is email text
- How do you define your features in this case?
  - How many words per email?
  - Frequency of misspellings?
  - Domain of sender?
  - **Binary “contains_word_viagra”?**

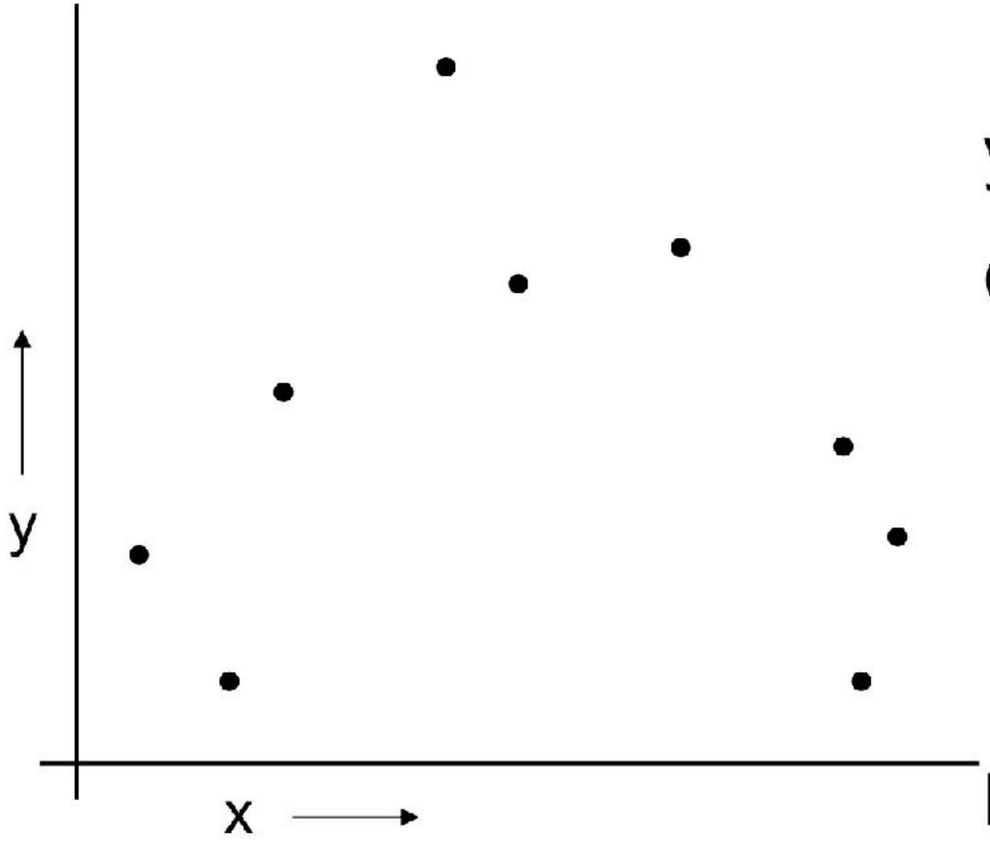
- How do you choose *meaningful* or *useful* features?

Binarization (dummy variables)

- Binarize ALL THE THINGS



Logic of cross-validation

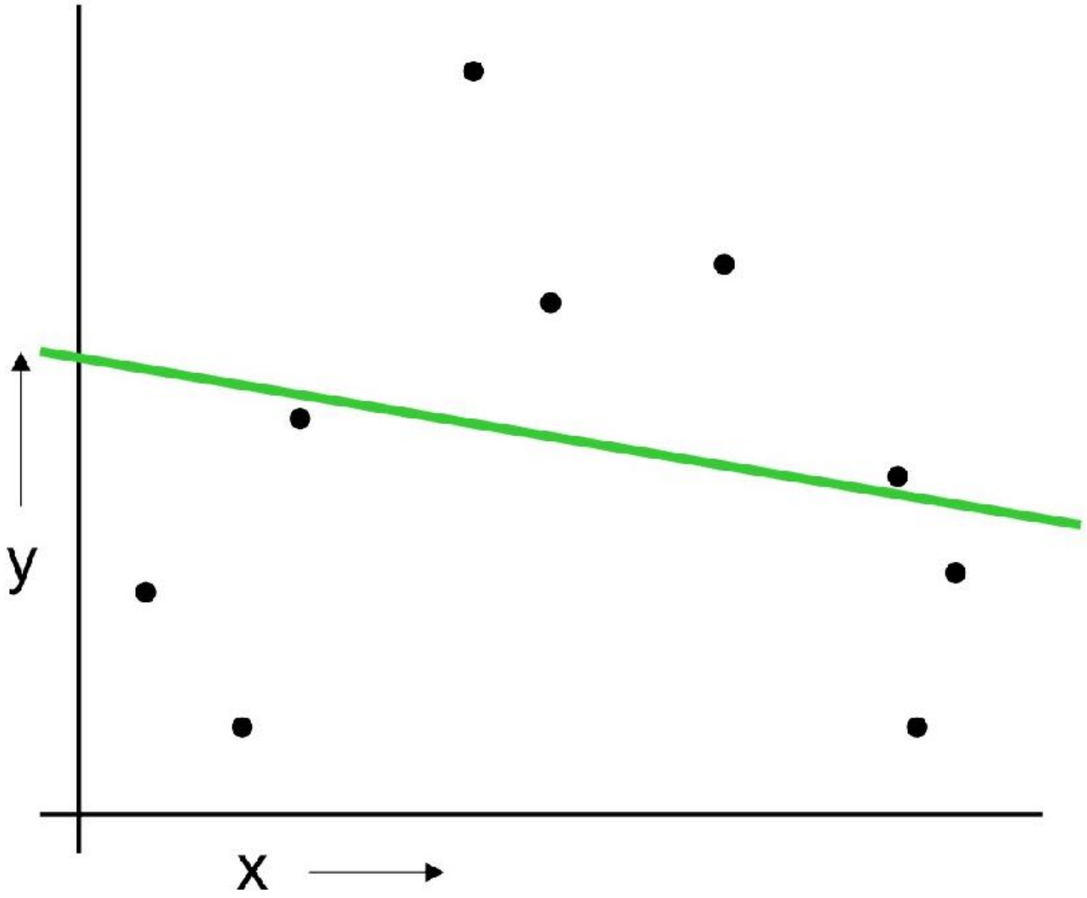


y = f(x) + noise

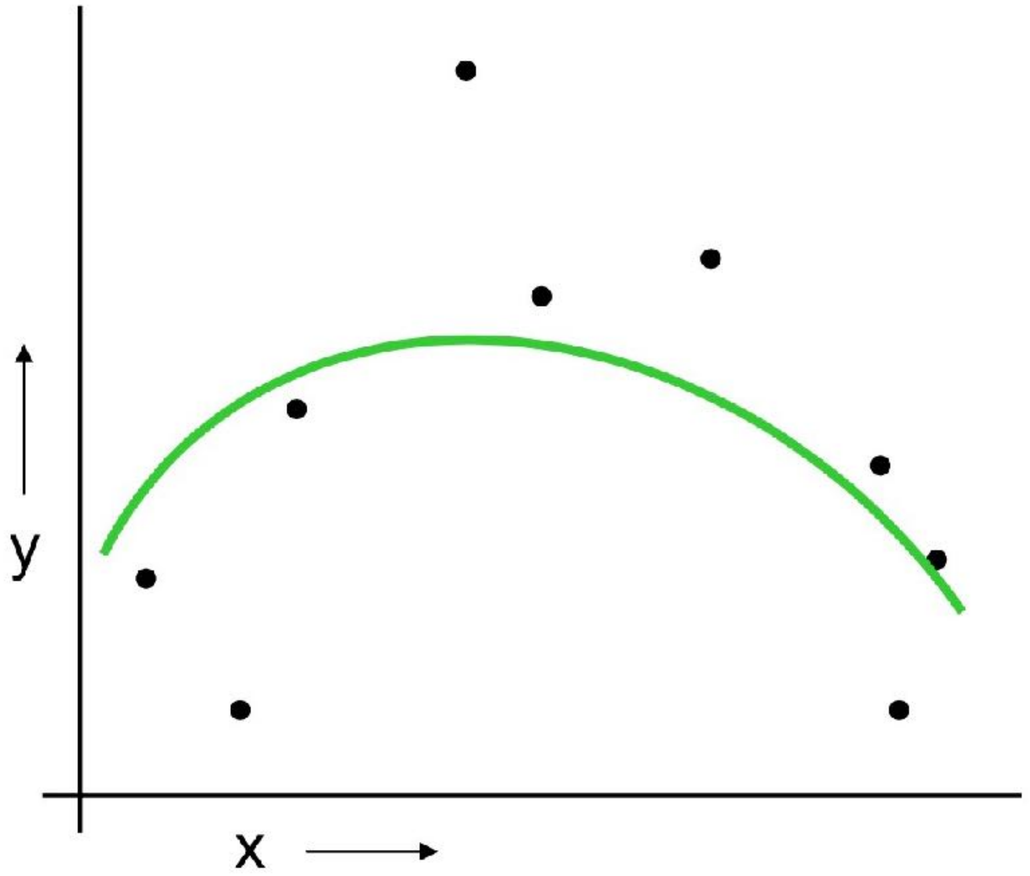
Can we learn f from this data?

Let's consider three methods

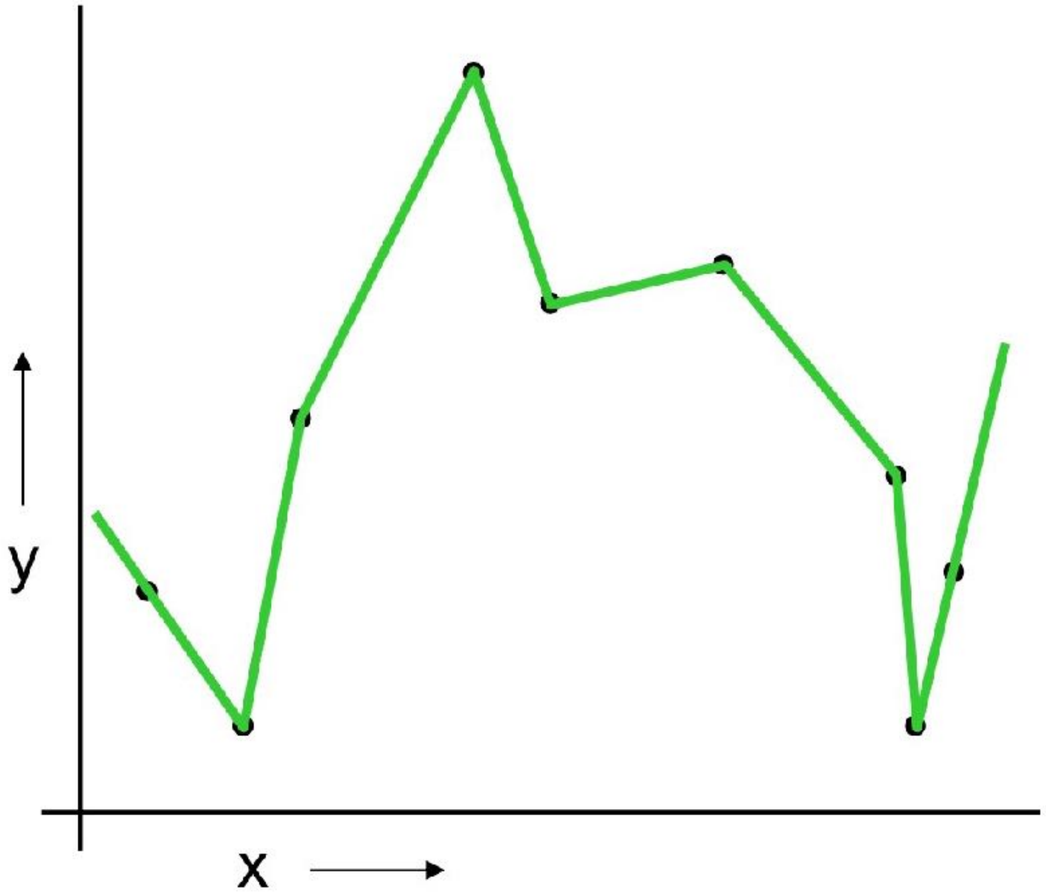
Linear regression



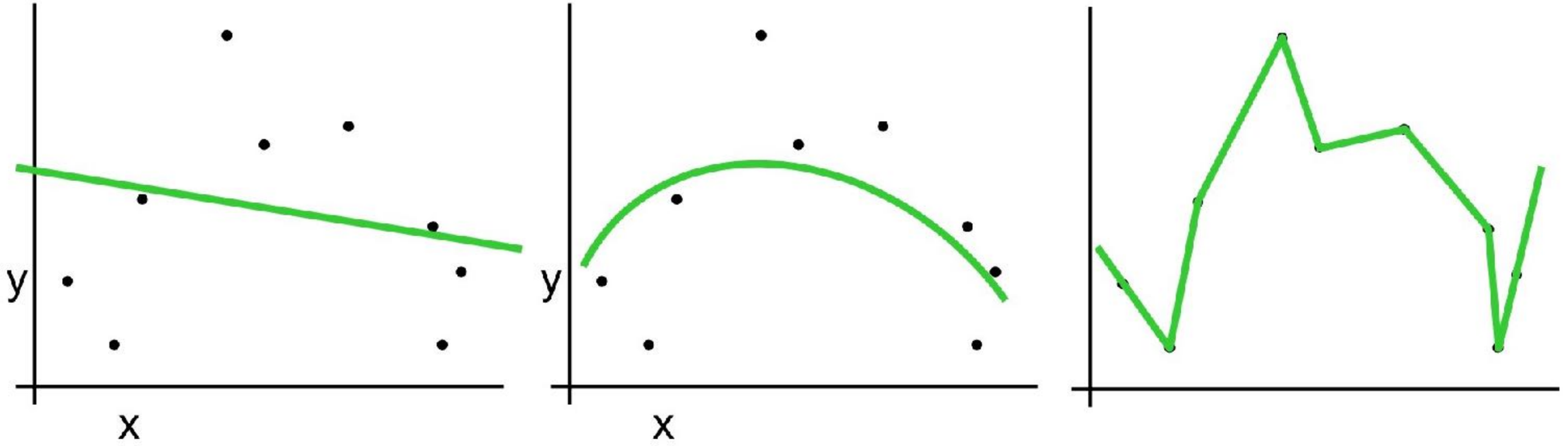
Quadratic regression



Piecewise linear nonparametric regression



Which is best?

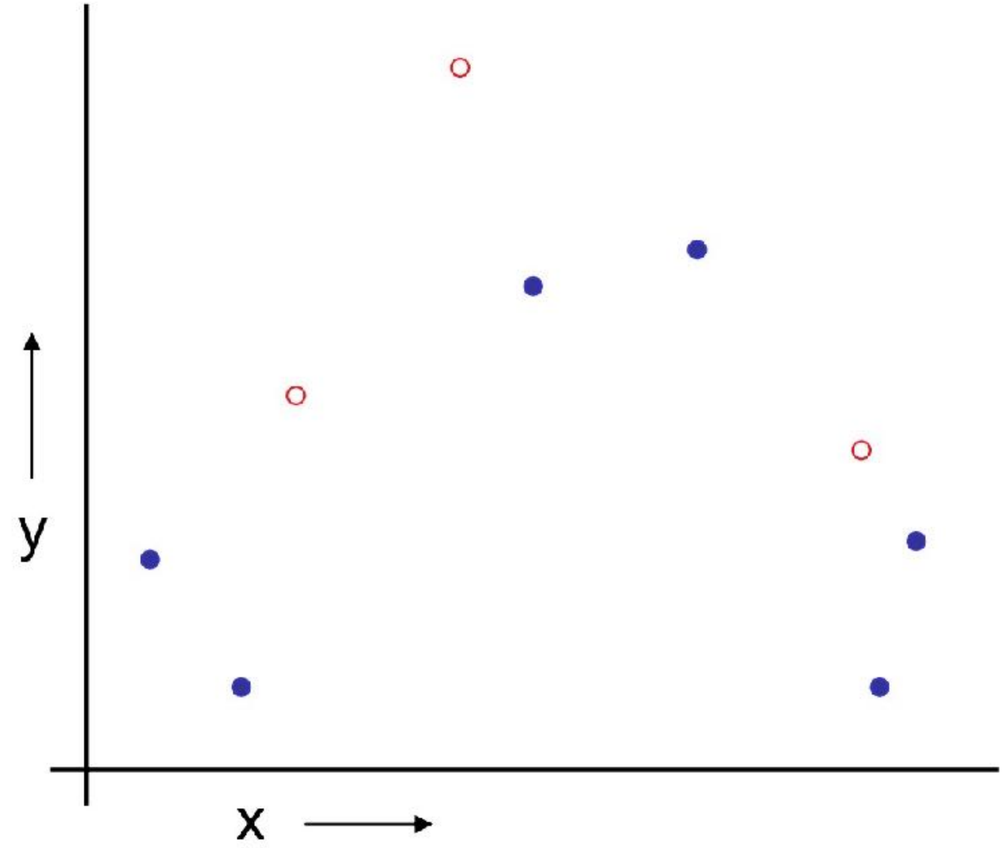


Why not choose the method with the best fit to the data?

Which is best?

"How well can we predict future data
drawn from the same distribution?"

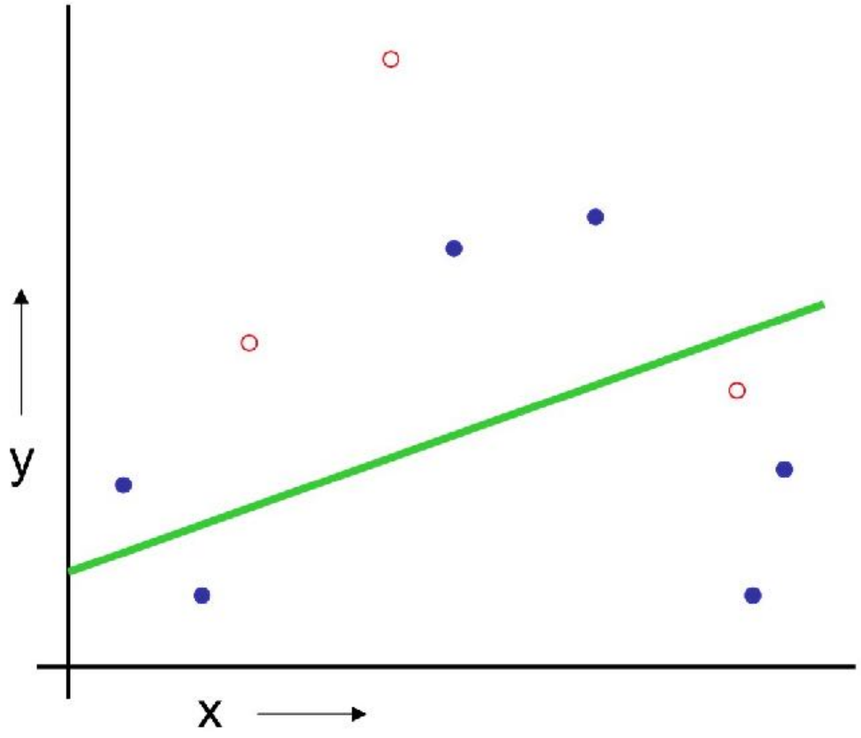
Test-set method



1. Randomly choose 30% of the data to be in a test set

2. The remainder is a training set

Test-set method



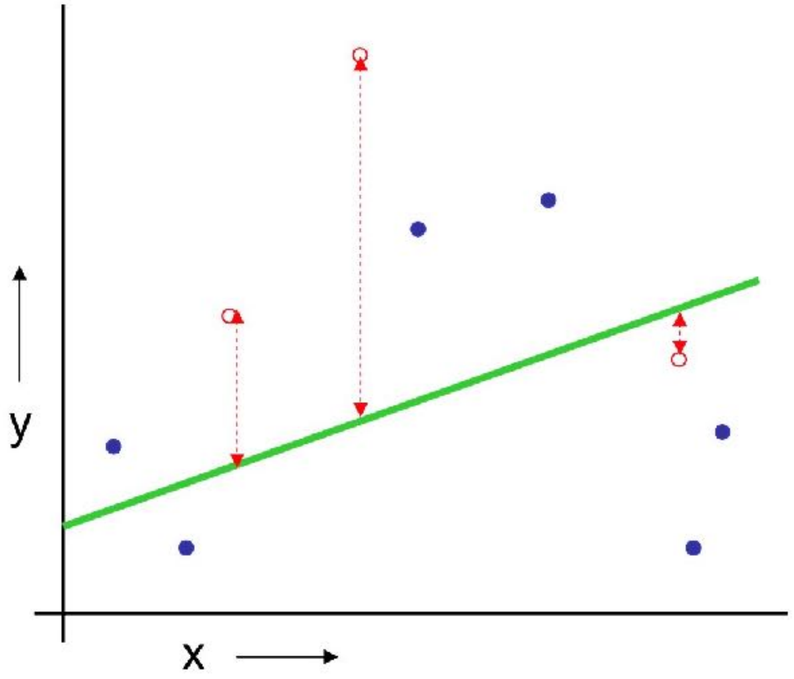
(Linear regression example)

1. Randomly choose 30% of the data to be in a test set

2. The remainder is a training set

3. Perform your regression on the training set

Test-set method



(Linear regression example)

Mean Squared Error = 2.4

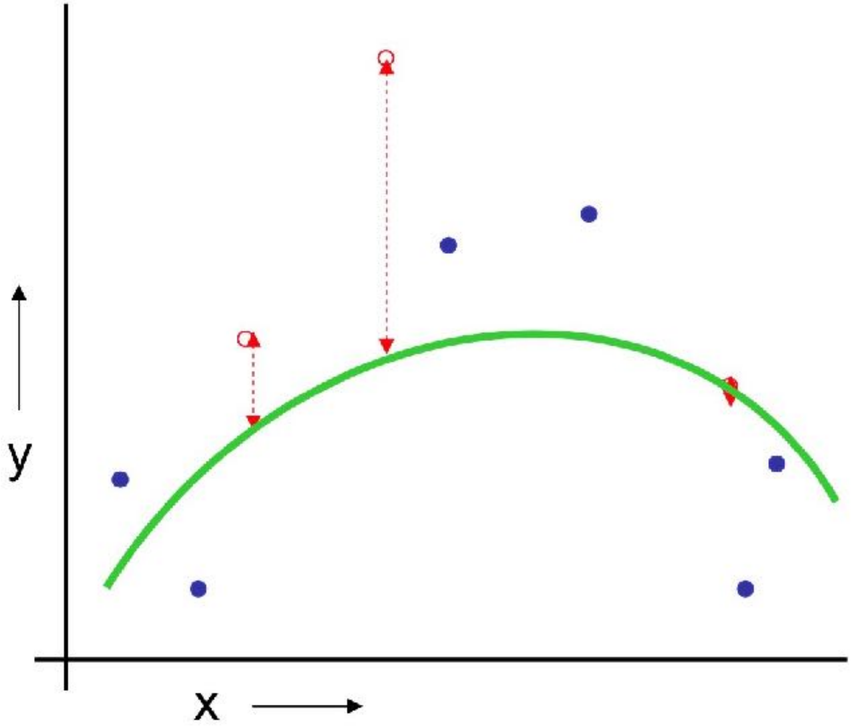
1. Randomly choose 30% of the data to be in a test set

2. The remainder is a training set

3. Perform your regression on the training set

4. Estimate your future performance with the test set

Test-set method



(Quadratic regression example)

Mean Squared Error = 0.9

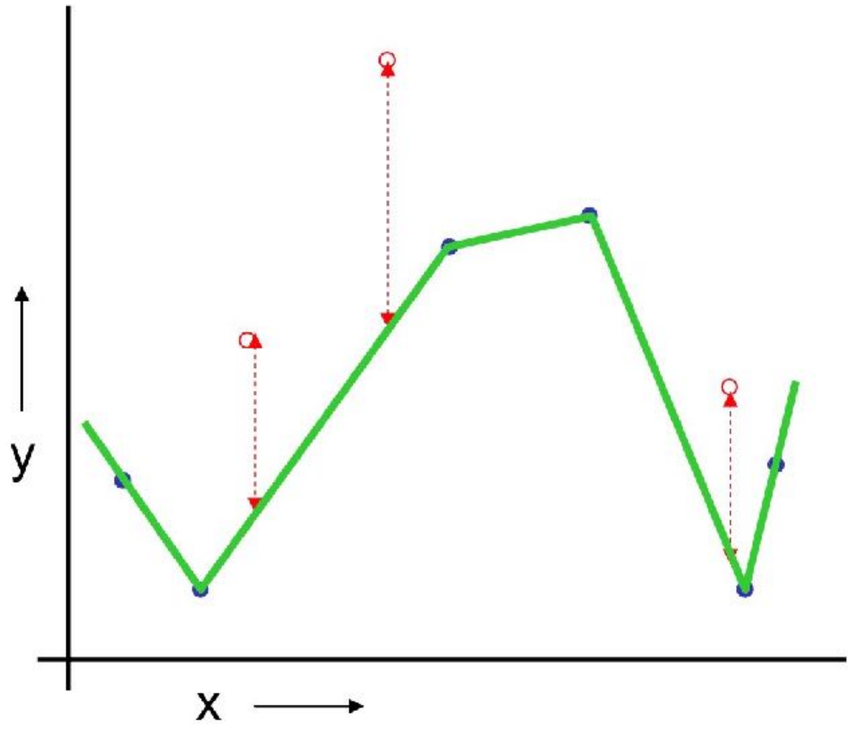
1. Randomly choose 30% of the data to be in a test set

2. The remainder is a training set

3. Perform your regression on the training set

4. Estimate your future performance with the test set

Test-set method



(Join the dots example)

Mean Squared Error = 2.2

1. Randomly choose 30% of the data to be in a test set

2. The remainder is a training set

3. Perform your regression on the training set

4. Estimate your future performance with the test set

Test-set method

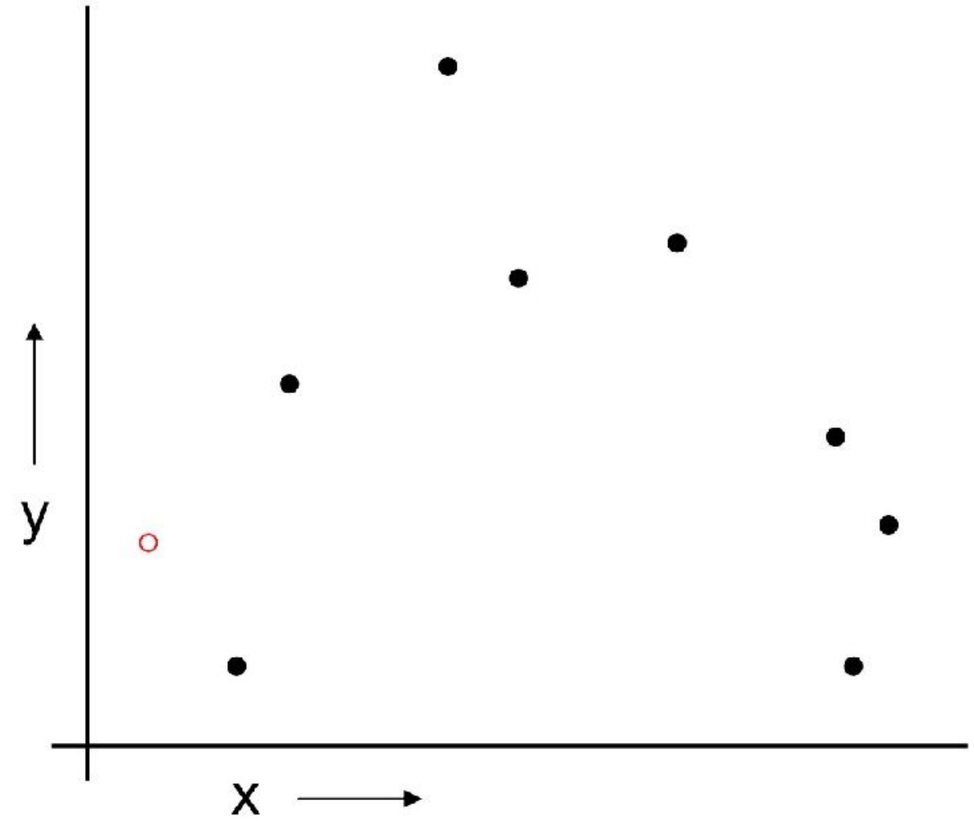
Good news:

- Very very simple
- Can then simply choose the method with the best test-set score

Bad news:

- Wastes data: we get an estimate of the best method to apply to 30% less data
- If we don't have much data, our test-set might just be lucky or unlucky

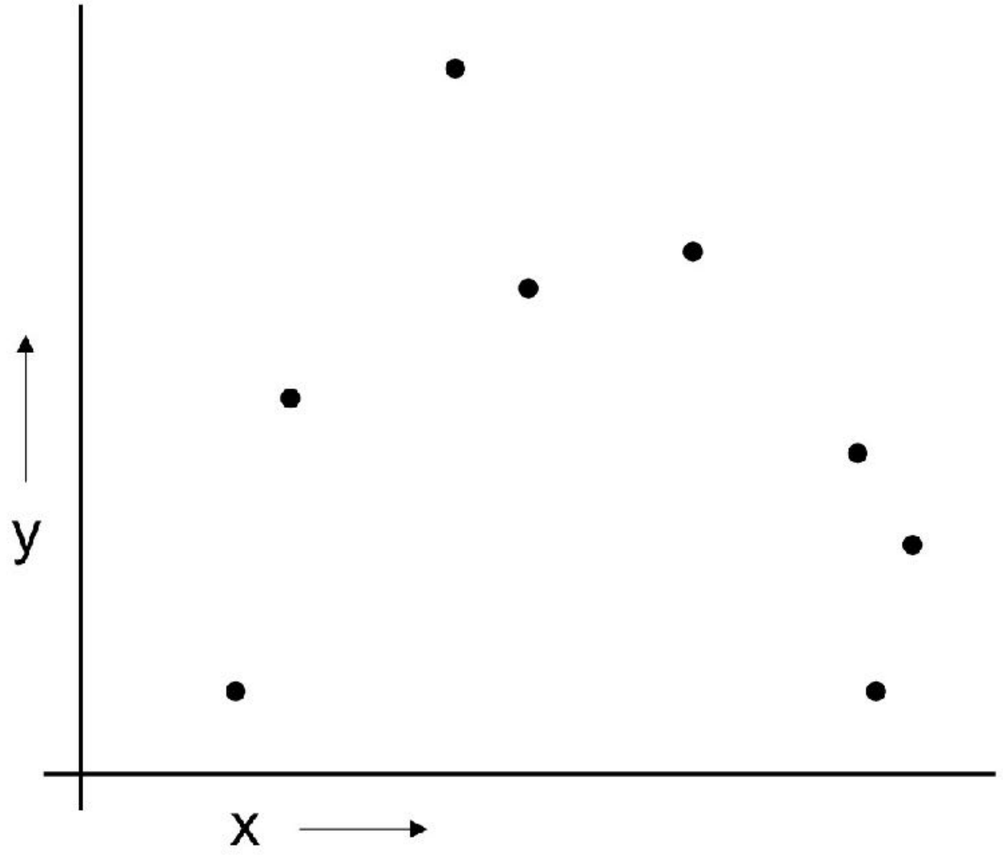
LOOCV



For k=1 to R

1. Let (x_k, y_k) be the k^{th} record

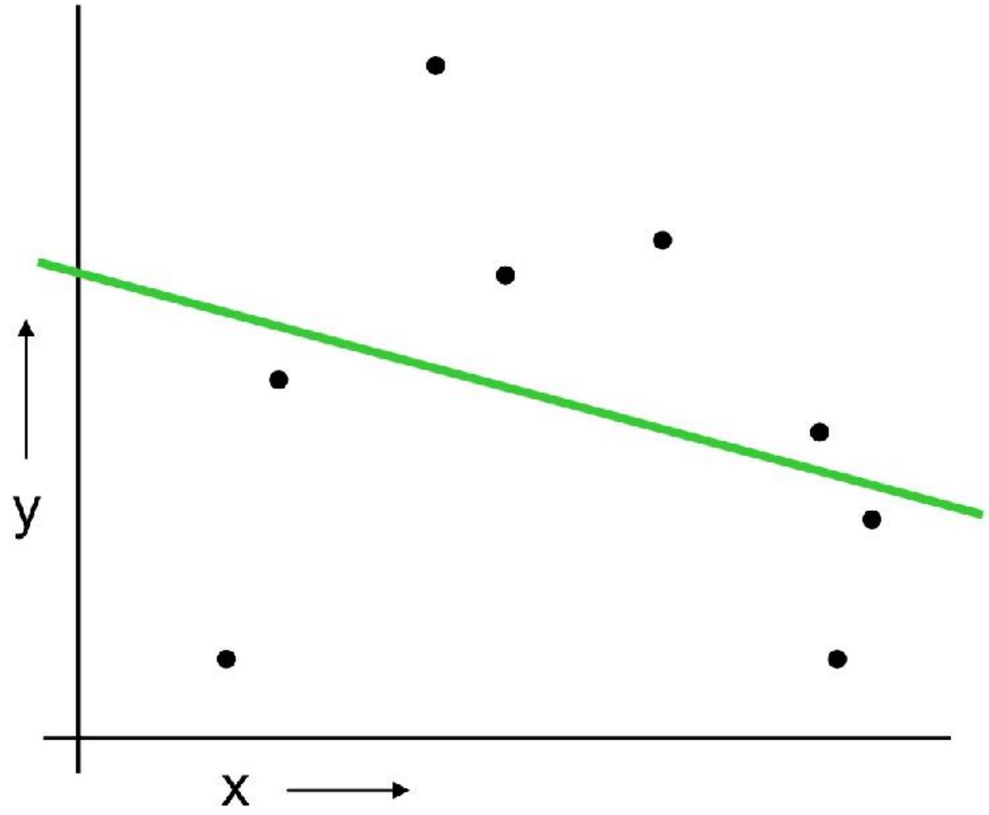
LOOCV



For k=1 to R

1. Let (x_k, y_k) be the k^{th} record
2. Temporarily remove (x_k, y_k) from the dataset

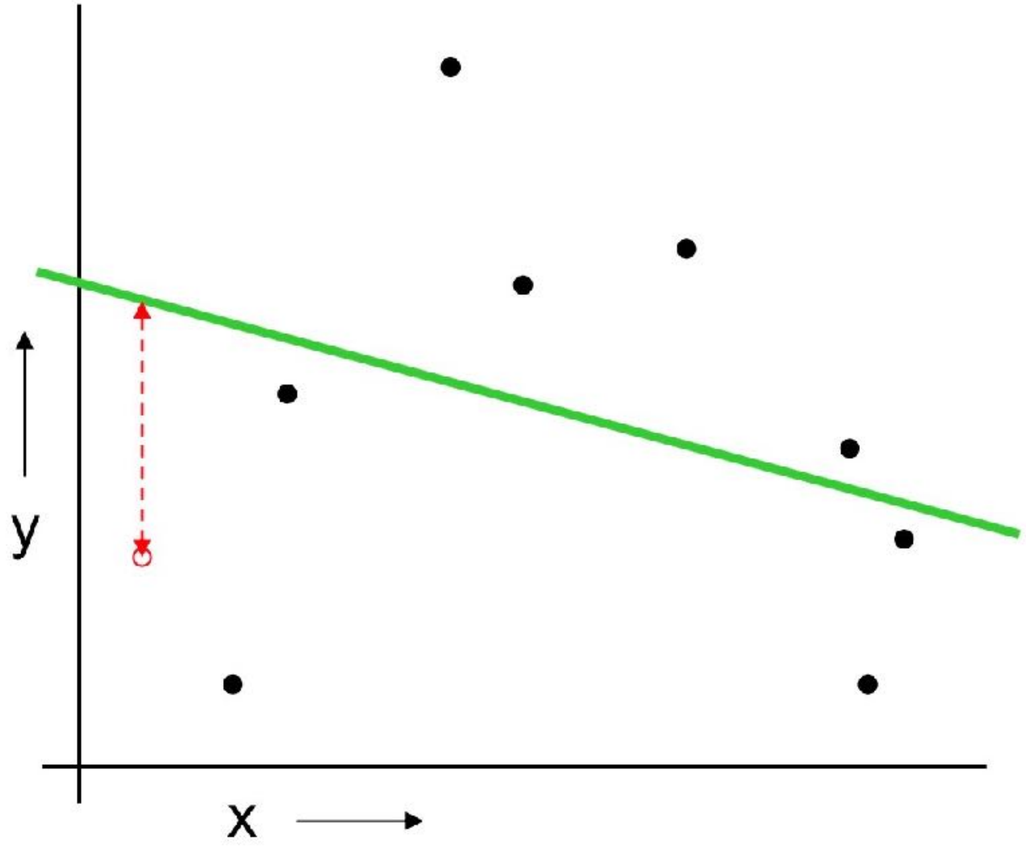
LOOCV



For k=1 to R

1. Let (x_k, y_k) be the k^{th} record
2. Temporarily remove (x_k, y_k) from the dataset
3. Train on the remaining R-1 datapoints

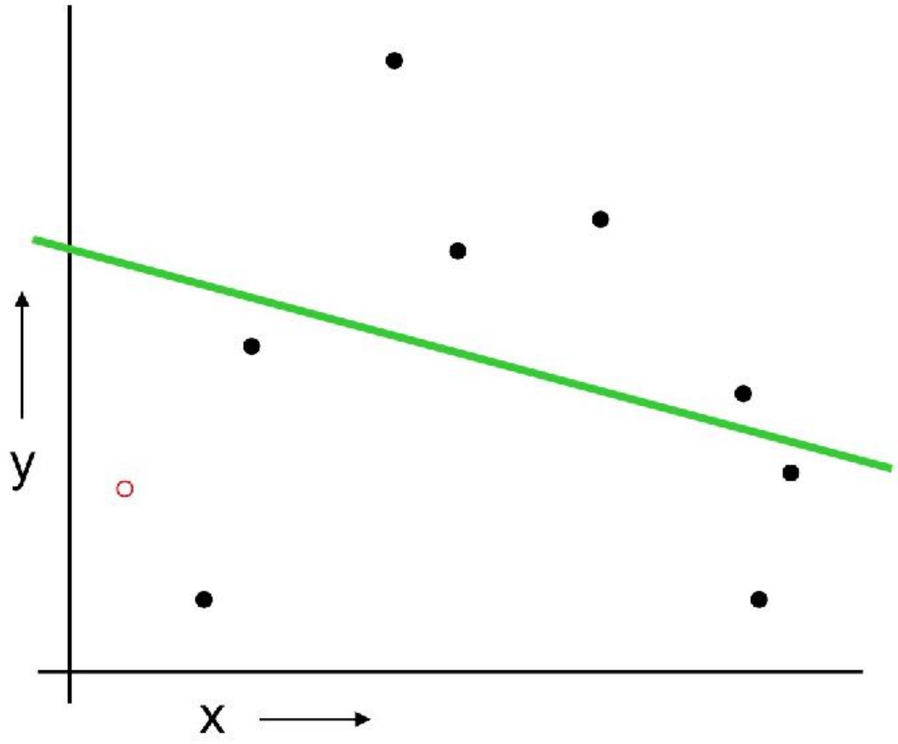
LOOCV



For k=1 to R

1. Let (x_k, y_k) be the k^{th} record
2. Temporarily remove (x_k, y_k) from the dataset
3. Train on the remaining R-1 datapoints
4. Note your error (x_k, y_k)

LOOCV

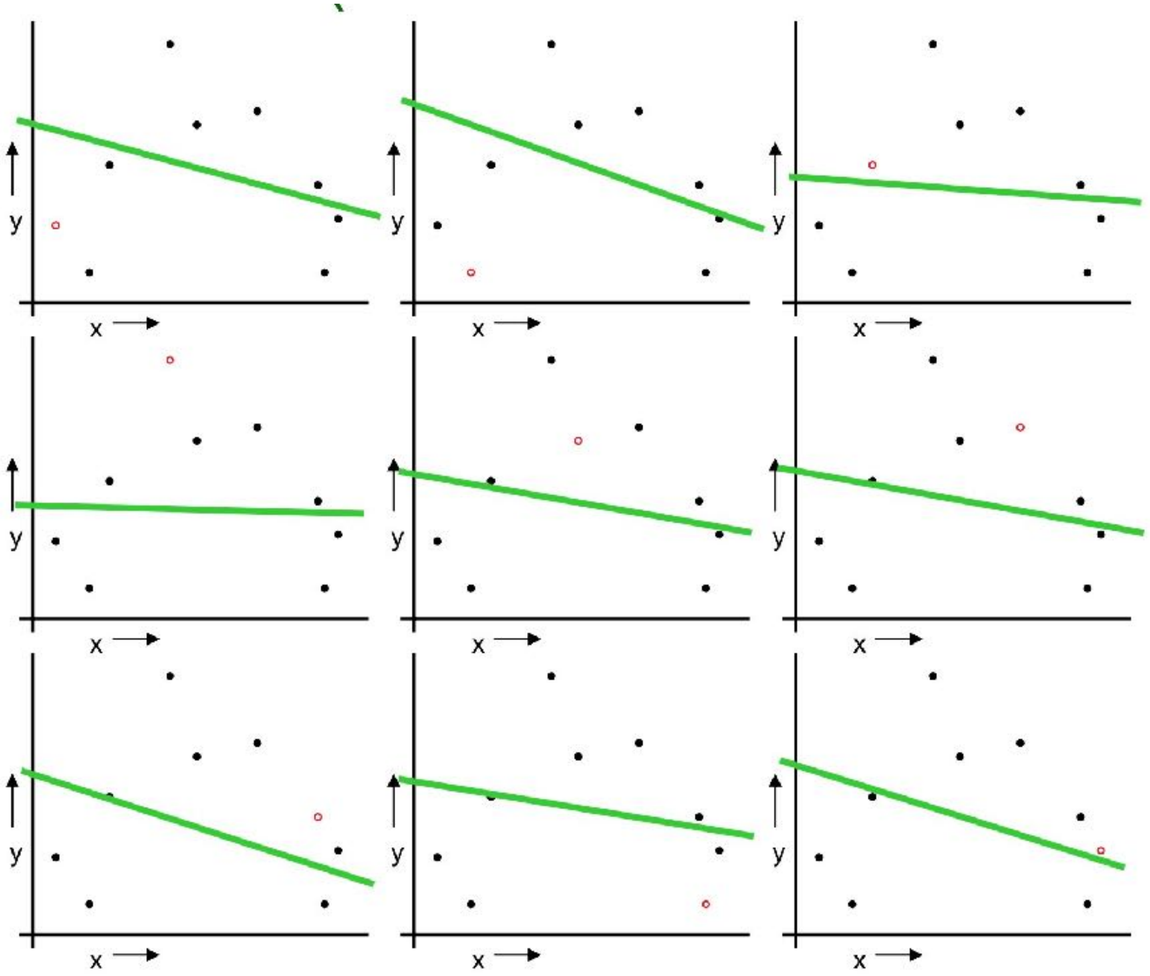


For k=1 to R

1. Let (x_k, y_k) be the k^{th} record
2. Temporarily remove (x_k, y_k) from the dataset
3. Train on the remaining R-1 datapoints
4. Note your error (x_k, y_k)

When you've done all points, report the mean error.

LOOCV



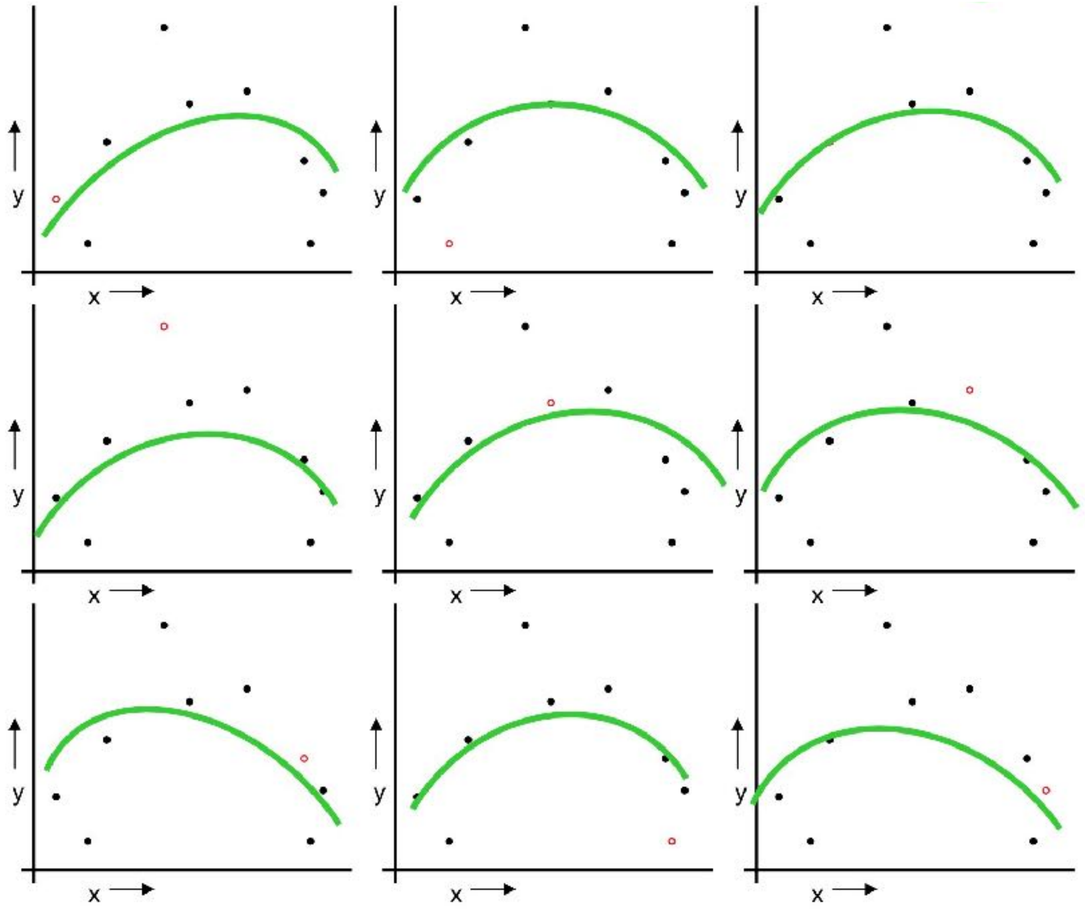
For k=1 to R

1. Let (x_k, y_k) be the k^{th} record
2. Temporarily remove (x_k, y_k) from the dataset
3. Train on the remaining R-1 datapoints
4. Note your error (x_k, y_k)

When you've done all points, report the mean error.

$$MSE_{LOOCV} = 2.12$$

LOOCV



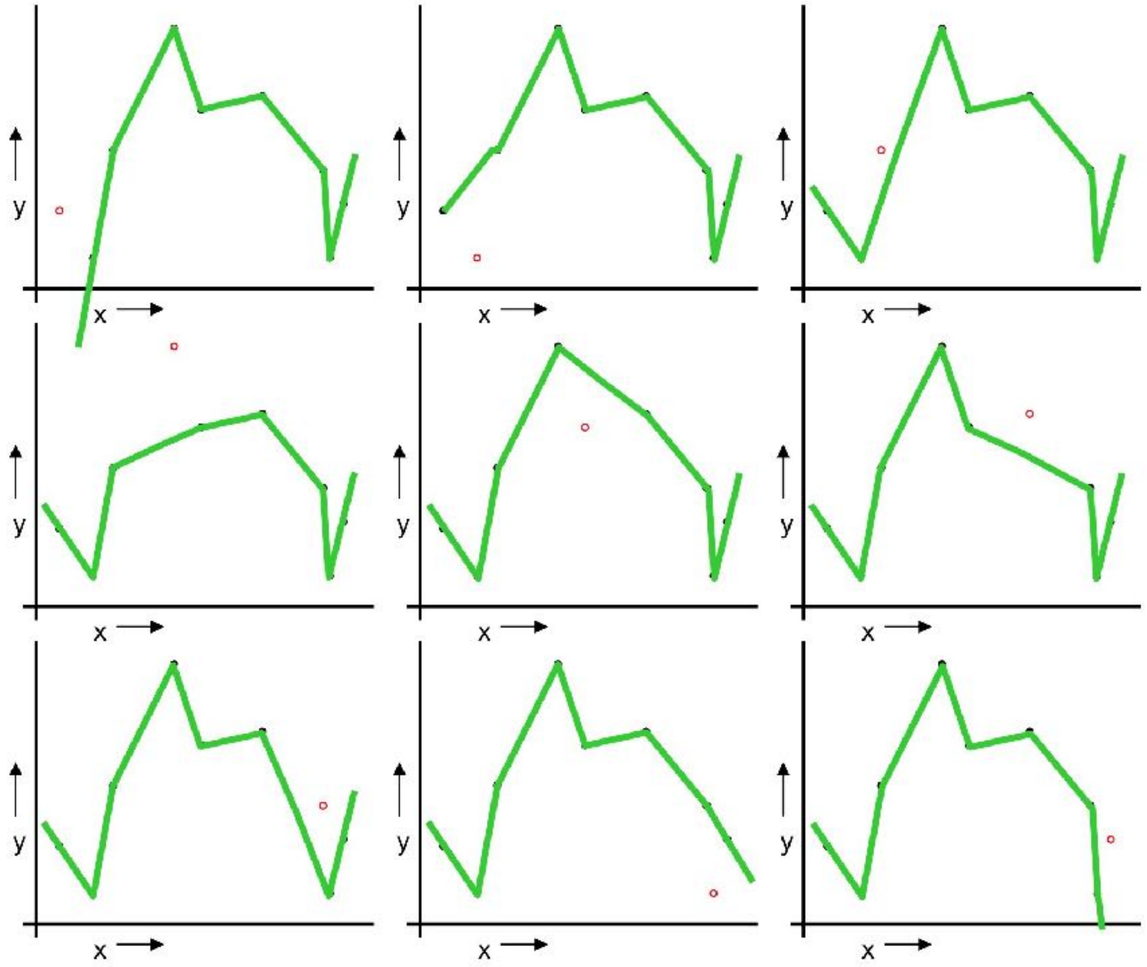
For k=1 to R

1. Let (x_k,y_k) be the k^{th} record
2. Temporarily remove (x_k,y_k) from the dataset
3. Train on the remaining R-1 datapoints
4. Note your error (x_k,y_k)

When you've done all points, report the mean error.

$MSE_{LOOCV} = 0.962$

LOOCV



For k=1 to R

1. Let (x_k, y_k) be the k^{th} record
2. Temporarily remove (x_k, y_k) from the dataset
3. Train on the remaining R-1 datapoints
4. Note your error (x_k, y_k)

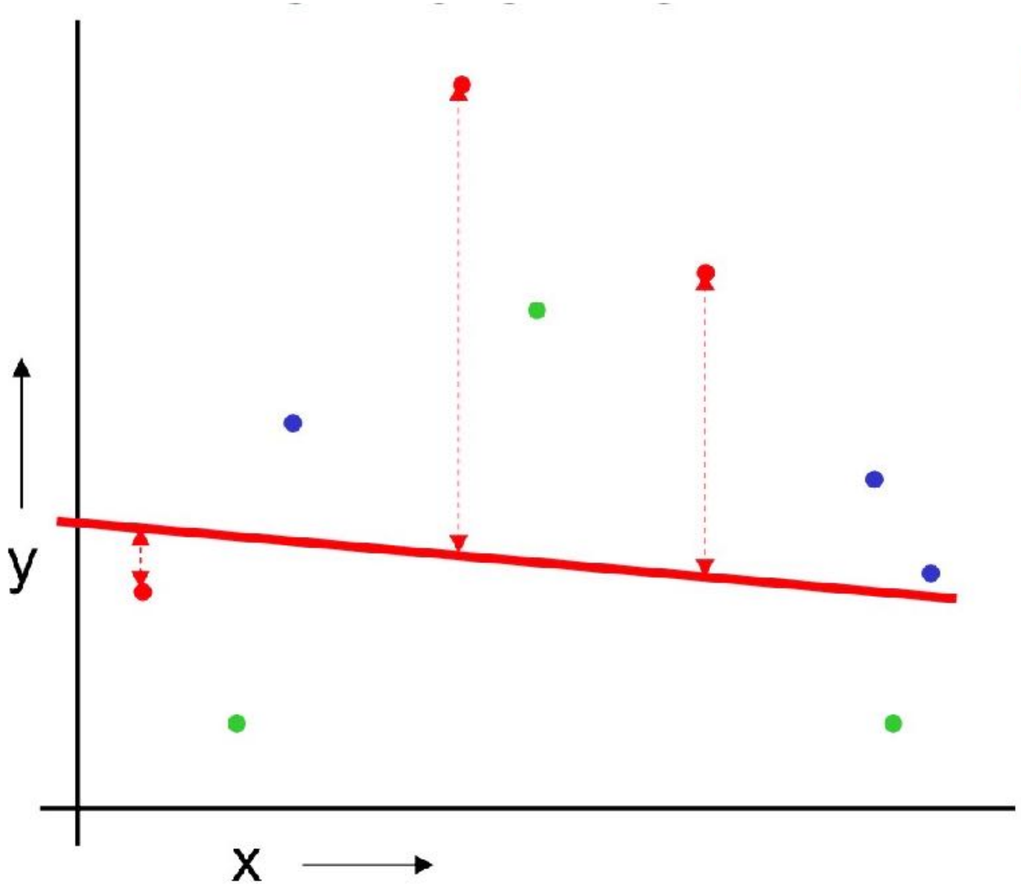
When you've done all points, report the mean error.

$MSE_{LOOCV} = 3.33$

Comparing cross-validations

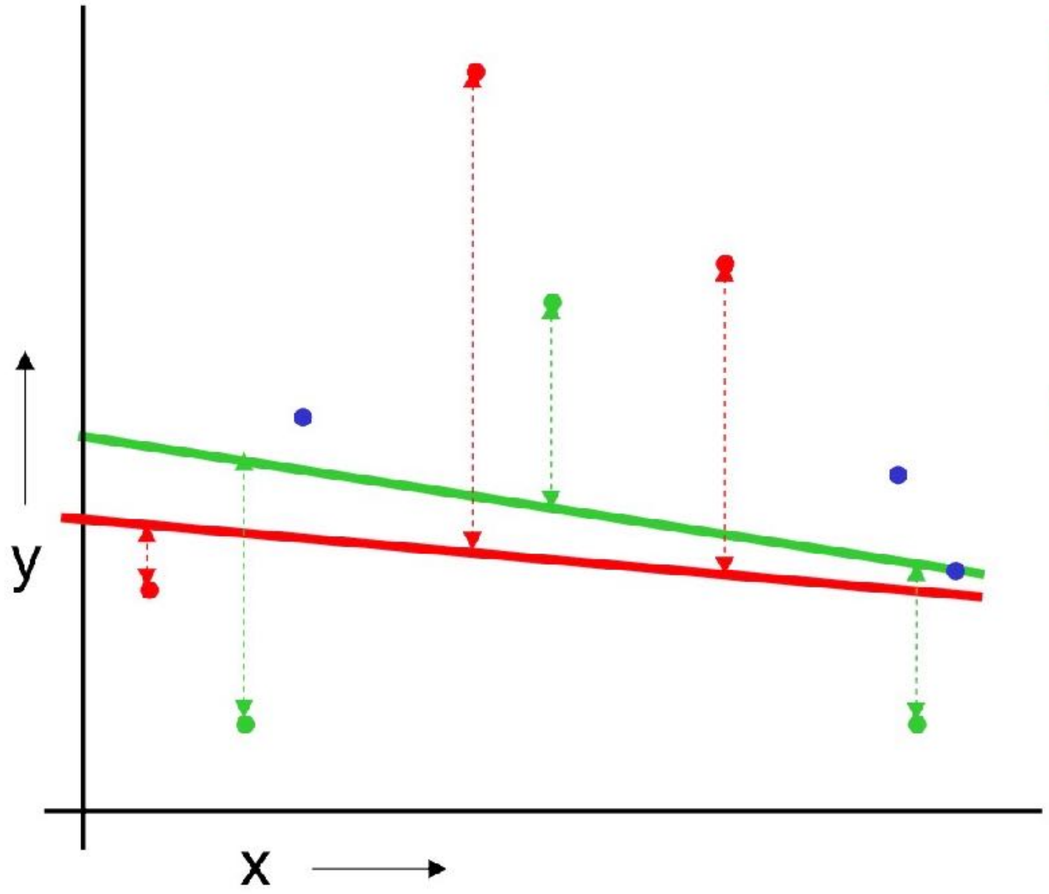
	Downside	Upside
Test-set	Variance: unreliable estimate of future performance	Cheap
Leave-one-out	Expensive. Has some weird behavior	Doesn't waste data

k-fold cross validation



For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

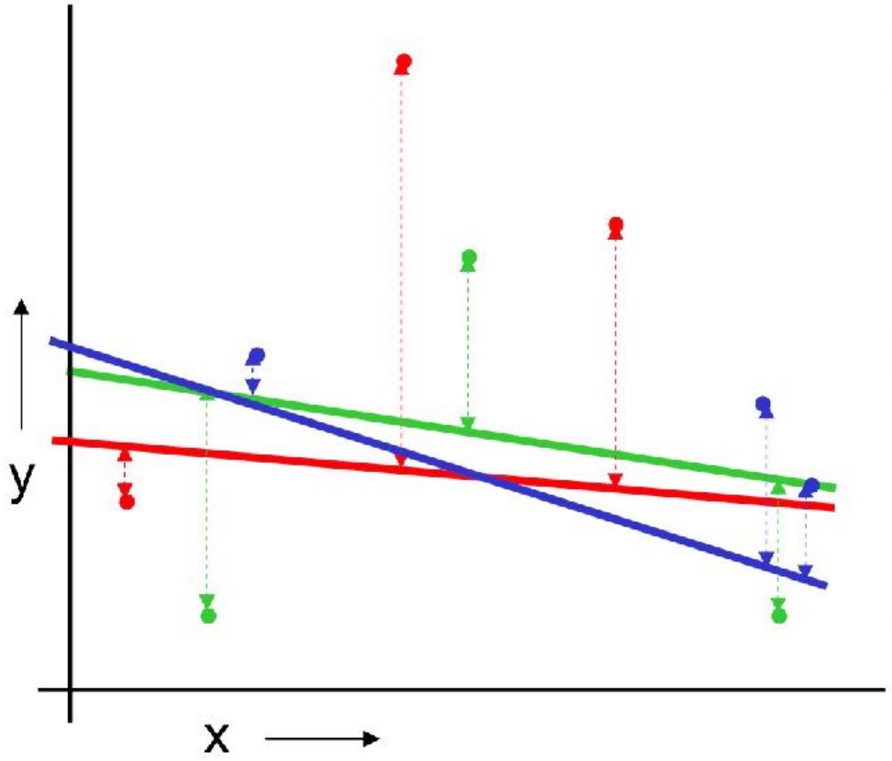
k-fold cross validation



For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

For the green partition: Train on all the points not in the green partition. Find the test-set sum of errors on the green points.

k-fold cross validation

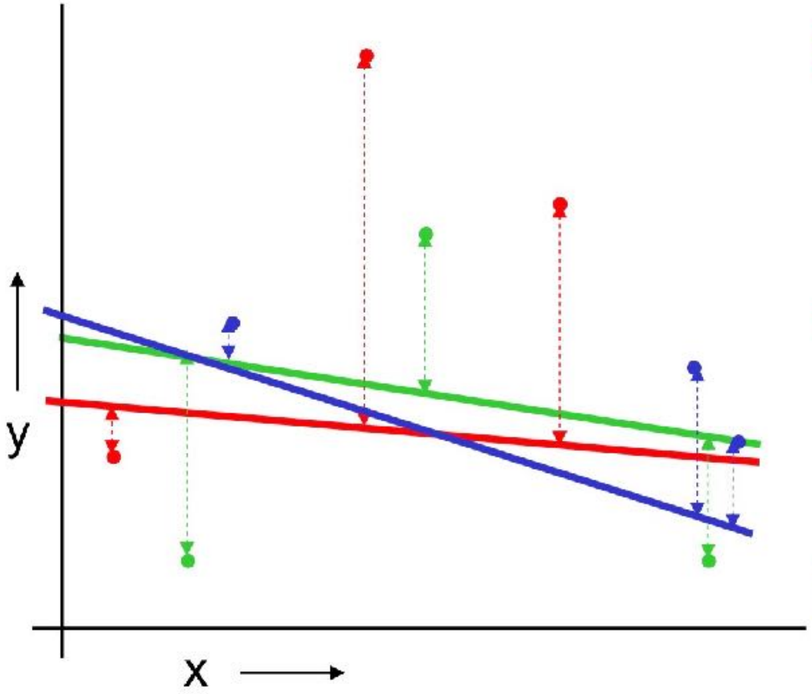


For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

For the green partition: Train on all the points not in the green partition. Find the test-set sum of errors on the green points.

For the blue partition: Train on all the points not in the blue partition. Find the test-set sum of errors on the blue points.

k-fold cross validation



Linear Regression
$MSE_{3FOLD}=2.05$

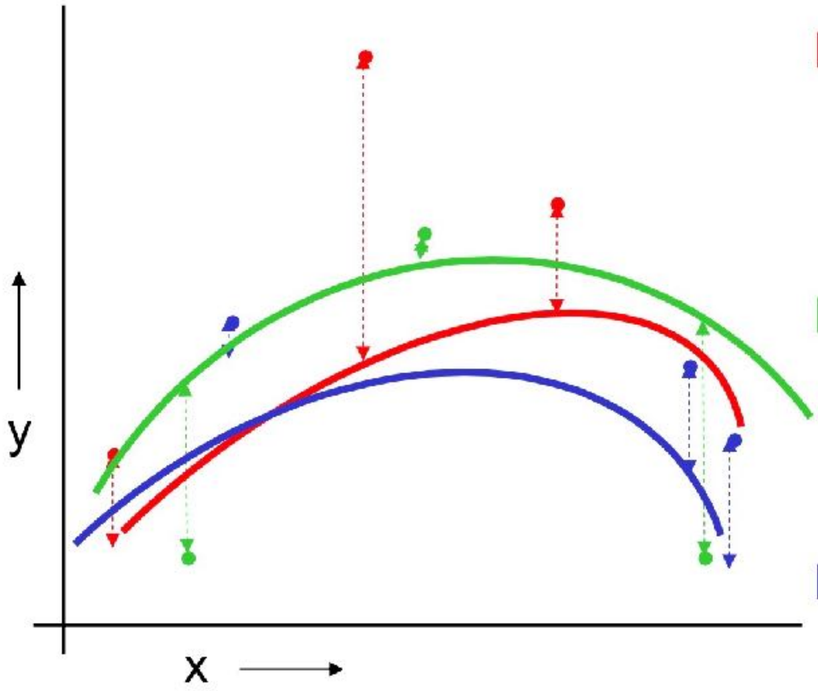
For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

For the green partition: Train on all the points not in the green partition. Find the test-set sum of errors on the green points.

For the blue partition: Train on all the points not in the blue partition. Find the test-set sum of errors on the blue points.

Then report the mean error

k-fold cross validation



Quadratic Regression

$MSE_{3FOLD}=1.11$

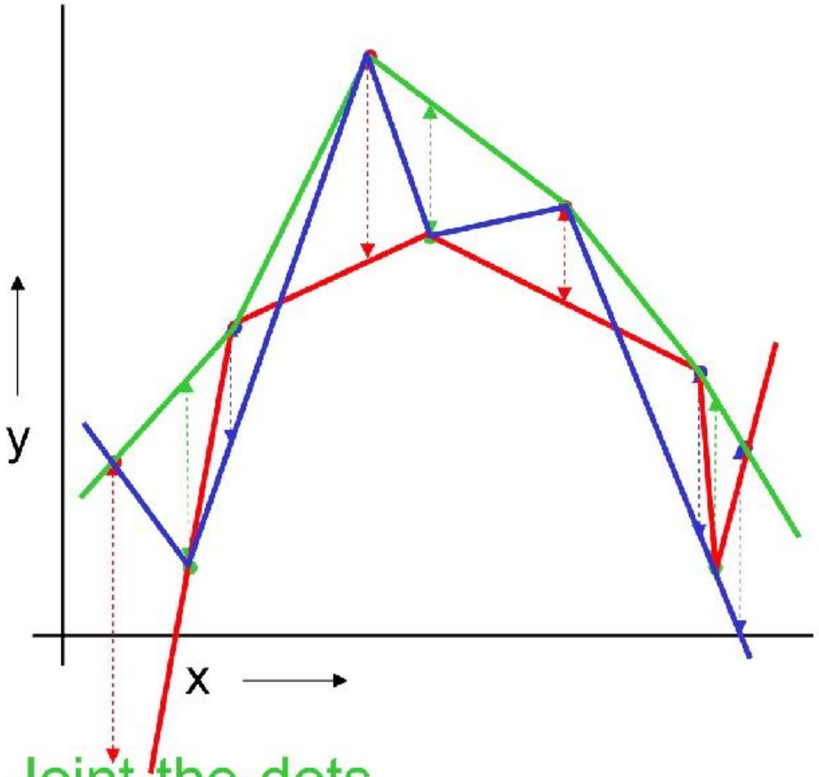
For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

For the green partition: Train on all the points not in the green partition. Find the test-set sum of errors on the green points.

For the blue partition: Train on all the points not in the blue partition. Find the test-set sum of errors on the blue points.

Then report the mean error

k-fold cross validation



Joint-the-dots

$MSE_{3FOLD}=2.93$

For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

For the green partition: Train on all the points not in the green partition. Find the test-set sum of errors on the green points.

For the blue partition: Train on all the points not in the blue partition. Find the test-set sum of errors on the blue points.

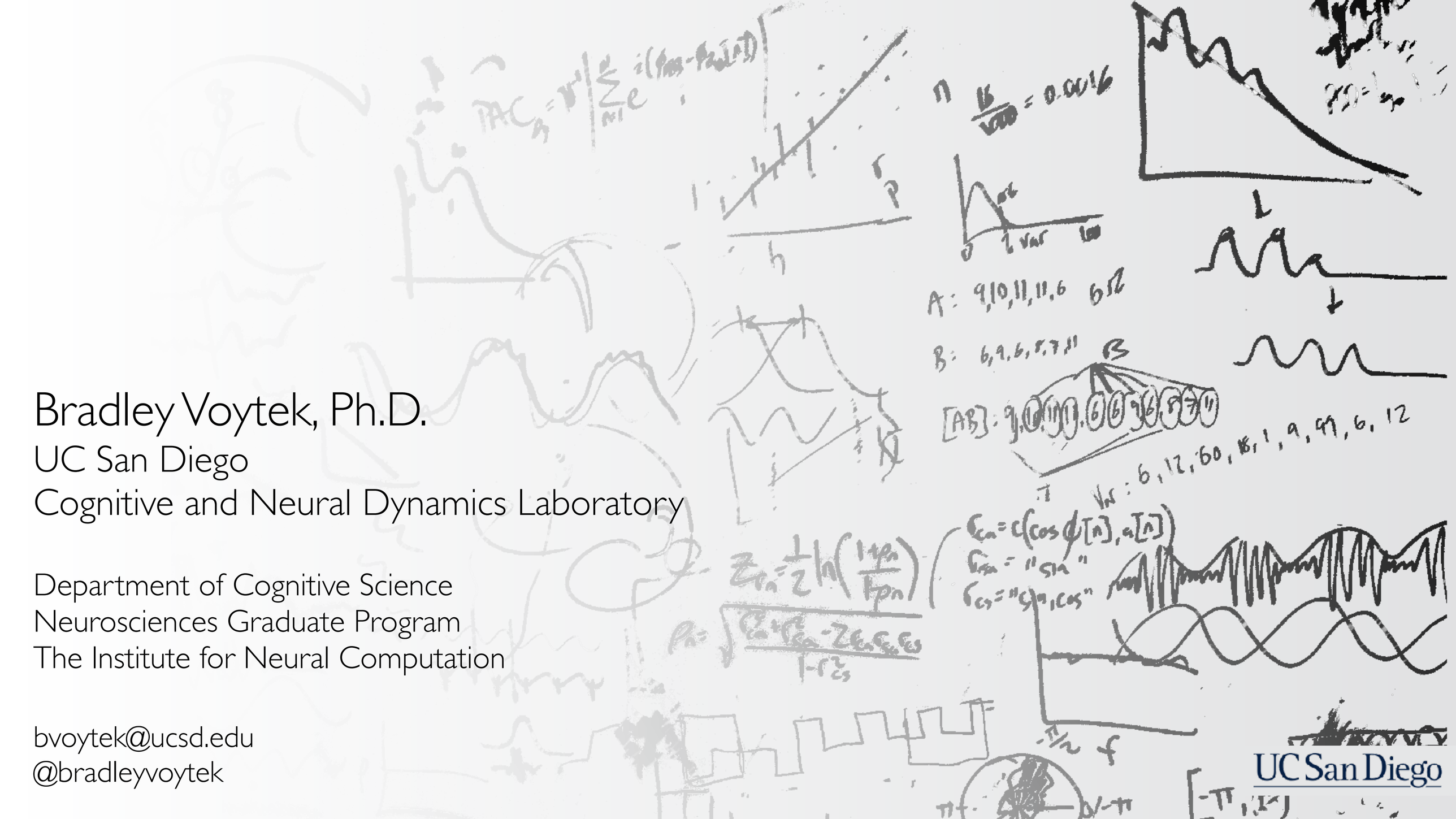
Then report the mean error

Trading off goodness of fit against model complexity

- If the model has as many degrees of freedom as the data, it can fit the training data perfectly

- But the objective in ML is generalization

- Can expect a model to generalize well if it explains the training data surprisingly well given the complexity of the model.



Bradley Voytek, Ph.D.

UC San Diego
Cognitive and Neural Dynamics Laboratory

Department of Cognitive Science
Neurosciences Graduate Program
The Institute for Neural Computation

bvoytek@ucsd.edu
@bradleyvoytek

UC San Diego