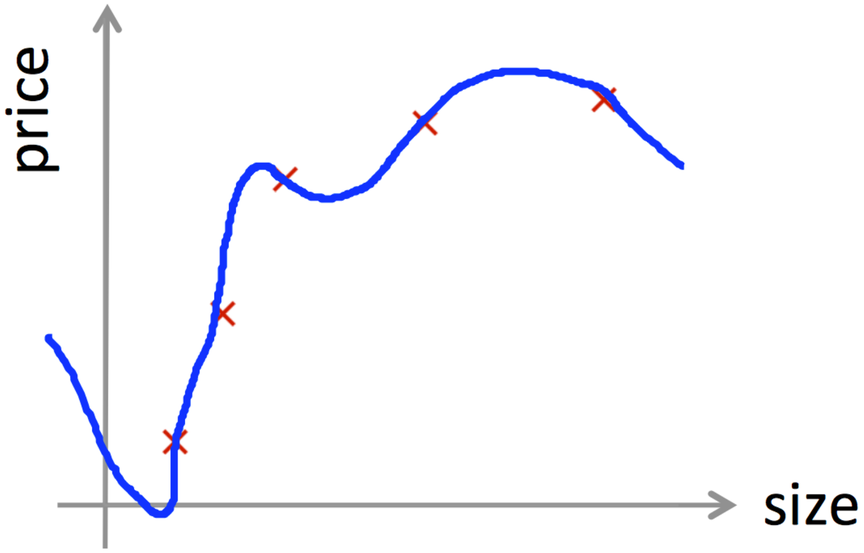
Evaluating a Learning Algorithm

Evaluating your hypothesis



$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

Fails to generalize to new examples not in training set.

$x_1 =$ size of house
$x_2 =$ no. of bedrooms
$x_3 =$ no. of floors
$x_4 =$ age of house
$x_5 =$ average income in neighborhood
$x_6 =$ kitchen size
$\vdots$
x_{100}

Evaluating a Learning Algorithm

Evaluating your hypothesis

Dataset:

Size	Price
2104	400
1600	330
2400	369
1416	232
3000	540
1985	300
1534	315
1427	199
1380	212
1494	243

70%

Training set

30%

Test set

$(x^{(1)}, y^{(1)})$
$(x^{(2)}, y^{(2)})$
$\vdots$
$(x^{(m)}, y^{(m)})$

$(x_{test}^{(1)}, y_{test}^{(1)})$
$(x_{test}^{(2)}, y_{test}^{(2)})$
$\vdots$
$(x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$

m_{test} = no. of test example

$(x_{test}^{(1)}, y_{test}^{(1)})$

Evaluating hypotheses

Evaluating a Learning Algorithm

Training/testing procedure for linear regression

- Learn parameter θ from training data (minimizing training error $J(\theta)$)

70 %

- Compute test set error:

$$J_{test}(\theta) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} \left(h_\theta(x_{test}^{(i)}) - y_{test}^{(i)} \right)^2$$

Training/testing procedure for logistic regression

- Learn parameter θ from training data
- Compute test set error:

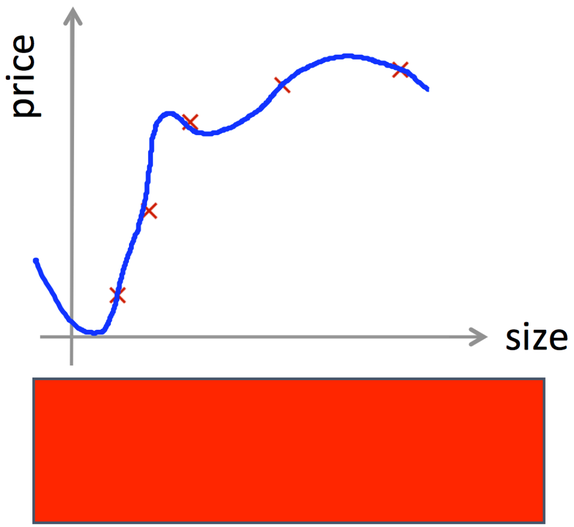
$$J_{test}(\theta) = -\frac{1}{m_{test}} \sum_{i=1}^{m_{test}} y_{test}^{(i)} \log h_\theta(x_{test}^{(i)}) + (1 - y_{test}^{(i)}) \log h_\theta(x_{test}^{(i)})$$

- Misclassification error (0/1 misclassification error):

Evaluating a Learning Algorithm

Model selection and train/validation/test sets

Overfitting example



Model selection

d = degree of polynomial

1. $h_\theta(x) = \theta_0 + \theta_1 x$ → $\theta^{(1)}$ → $J_{test}(\theta^{(1)})$
2. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$ → $\theta^{(2)}$ → $J_{test}(\theta^{(2)})$
3. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_3 x^3$ → $\theta^{(3)}$ → $J_{test}(\theta^{(3)})$

⋮ ⋮

10. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_{10} x^{10}$ → $\theta^{(10)}$ → $J_{test}(\theta^{(10)})$

Choose $\theta_0 + \ldots \theta_5 x^5$

How well does the model generalize? Report test set error $J_{test}(\theta^{(5)})$.

$\theta_0, \theta_1 \ldots$

Problem: $J_{test}(\theta^{(5)})$ is likely to be an optimistic estimate of generalization error. I.e. our extra parameter (d = degree of polynomial) is fit to test set.

Evaluating a Learning Algorithm

Model selection and train/validation/test sets

Evaluating your hypothesis

Dataset:

Size	Price	
2104	400	Training set
1600	330	
2400	369	
1416	232	
3000	540	
1985	300	
1534	315	Cross validation set (CV)
1427	199	
1380	212	test set
1494	243	

60%

20%

20%

$(x^{(1)}, y^{(1)})$
$(x^{(2)}, y^{(2)})$
$\vdots$
$(x^{(m)}, y^{(m)})$

$(x_{cv}^{(1)}, y_{cv}^{(1)})$
$(x_{cv}^{(2)}, y_{cv}^{(2)})$
$\vdots$
$(x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})})$

m_{cv} = no. of cv examples

$(x_{cv}^{(i)}, y_{cv}^{(i)})$

$(x_{test}^{(1)}, y_{test}^{(1)})$
$(x_{test}^{(2)}, y_{test}^{(2)})$
$\vdots$
$(x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$

m_{test}

Train/validation/test error

Training error:

$$J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Cross Validation error:

$$J_{cv}(\theta) = \frac{1}{2m_{cv}} \sum_{i=1}^{m_{cv}} (h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$$

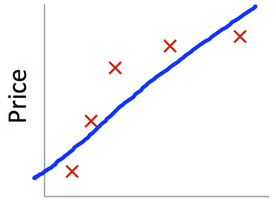
Test error:

$$J_{test}(\theta) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} (h_\theta(x_{test}^{(i)}) - y_{test}^{(i)})^2$$

Dianosing Bias vs. Variance

Bias vs. Variance

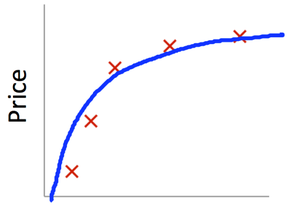
Bias/variance



$\theta_0 + \theta_1 x$

High bias (underfit)

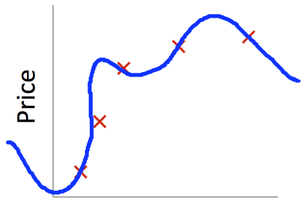
$d = 1$



$\theta_0 + \theta_1 x + \theta_2 x^2$

"Just right"

$d = 2$



$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

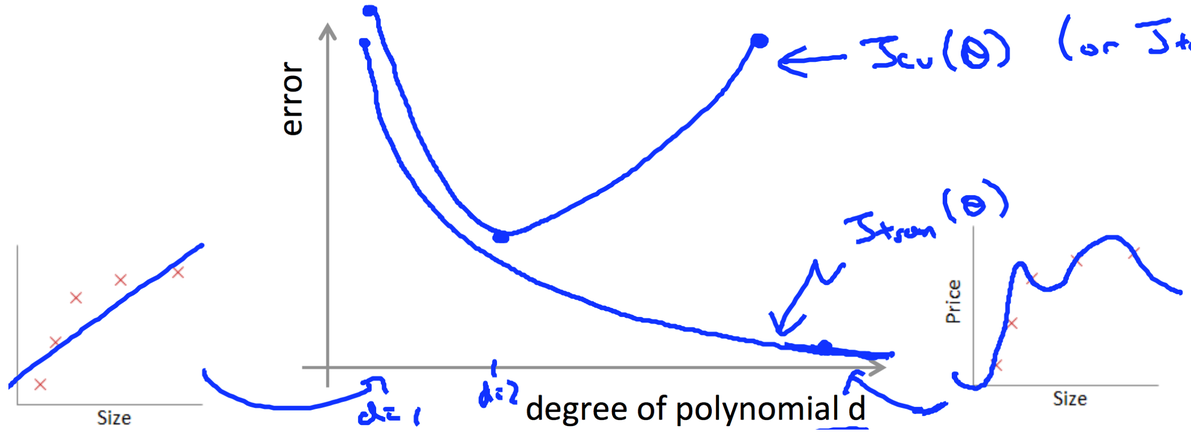
High variance (overfit)

$d = 4$

Bias/variance

Training error: $J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$

Cross validation error: $J_{cv}(\theta) = \frac{1}{2m_{cv}} \sum_{i=1}^{m_{cv}} (h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$ (or $J_{test}(\theta)$)

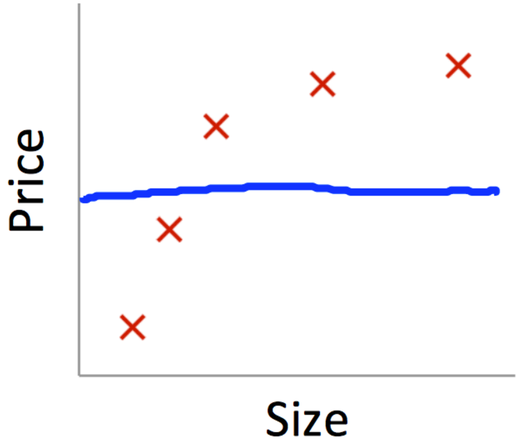


Bias vs. Variance

Linear regression with regularization

Model: $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m}\sum_{j=1}^{m}\theta_j^2$$

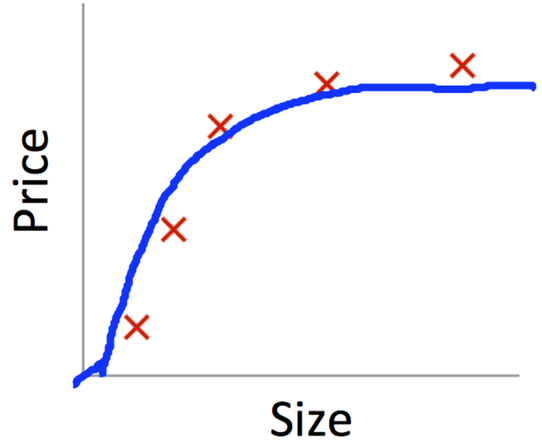


Large λ

High bias (underfit)

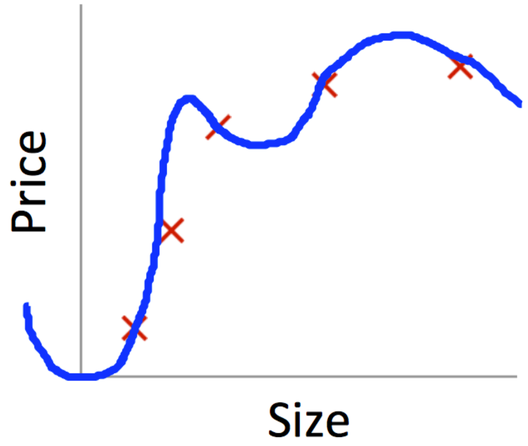
$\lambda = 10000.\ \theta_1 \approx 0, \theta_2 \approx 0, \ldots$

$h_\theta(x) \approx \theta_0$



Intermediate λ

"Just right"



Small λ

High variance (overfit)

$\lambda = 0$

Regularization and Bias/Variance

Bias vs. Variance

Choosing the regularization parameter λ

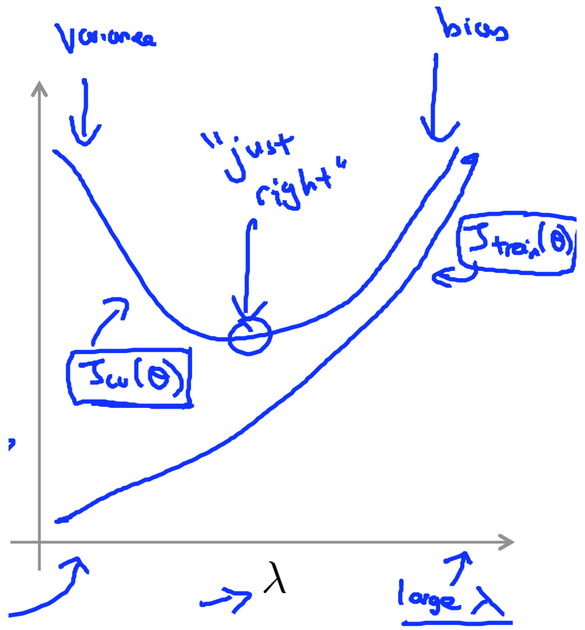
$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m}\sum_{j=1}^{m}\theta_j^2$$

$$J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$$J_{cv}(\theta) = \frac{1}{2m_{cv}}\sum_{i=1}^{m_{cv}}(h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$$

$$J_{test}(\theta) = \frac{1}{2m_{test}}\sum_{i=1}^{m_{test}}(h_\theta(x_{test}^{(i)}) - y_{test}^{(i)})^2$$



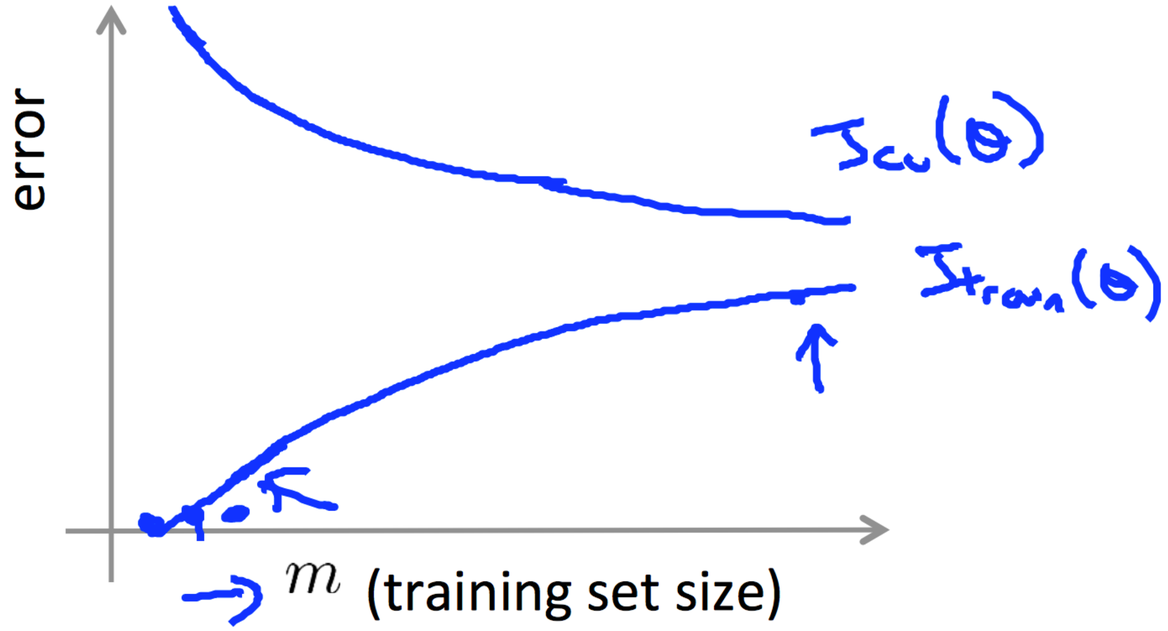
Learning Curves

Evaluating a Learning Algorithm

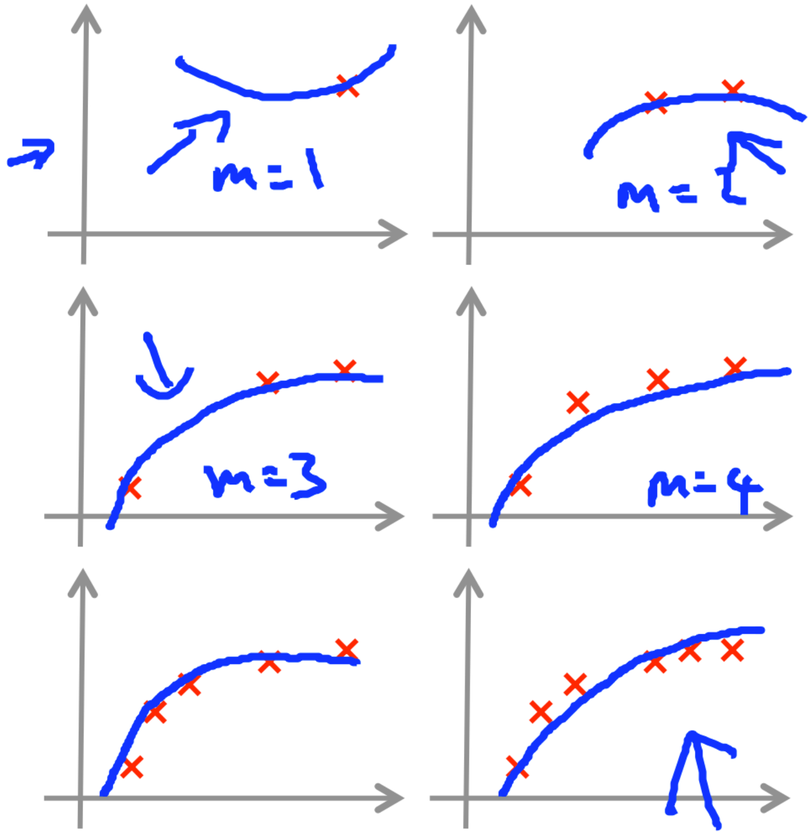
Learning curves

$$J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$$J_{cv}(\theta) = \frac{1}{2m_{cv}}\sum_{i=1}^{m_{cv}}(h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$$



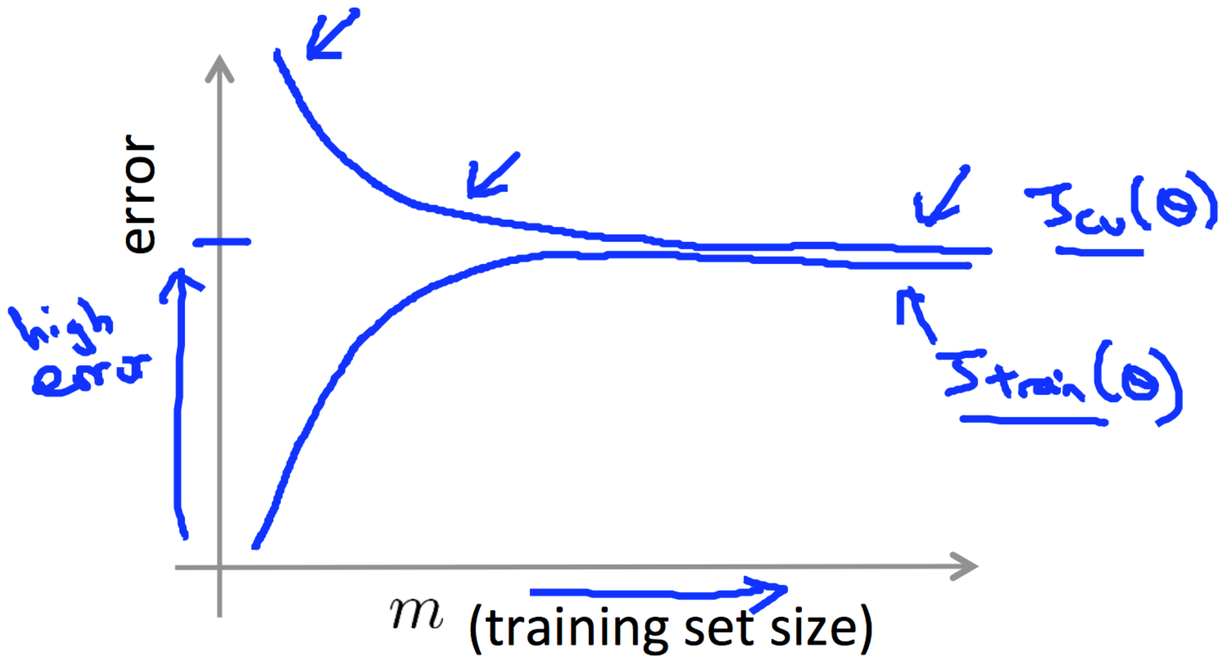
$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$

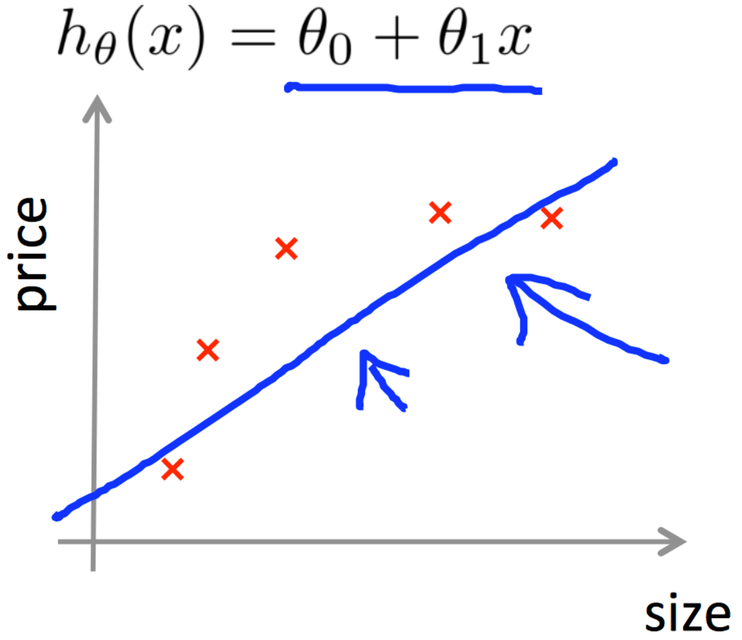


Learning Curves

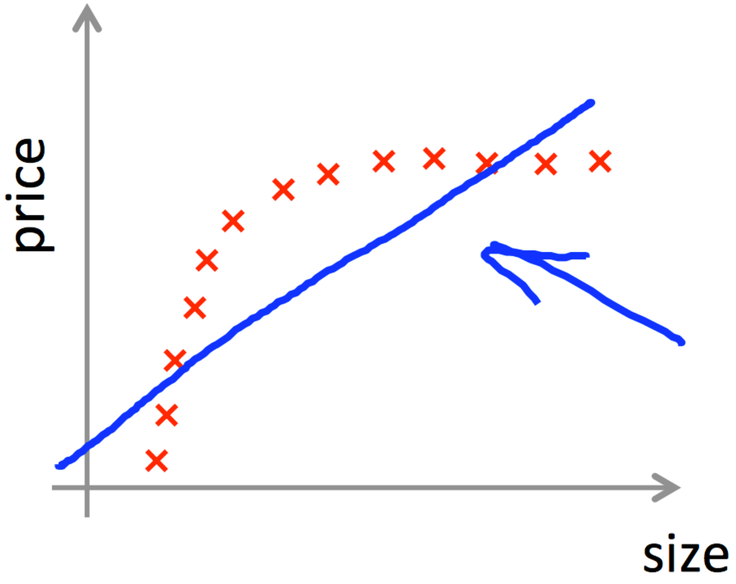
Evaluating a Learning Algorithm

High bias





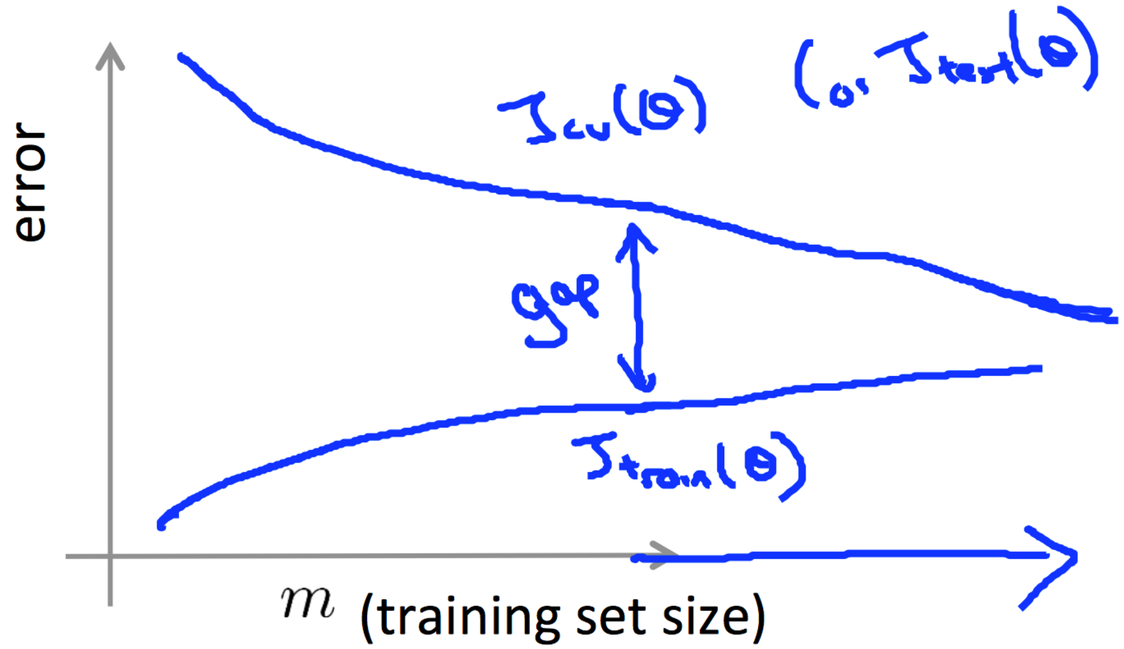
If a learning algorithm is suffering from high bias, getting more training data will not (by itself) help much.



Learning Curves

Evaluating a Learning Algorithm

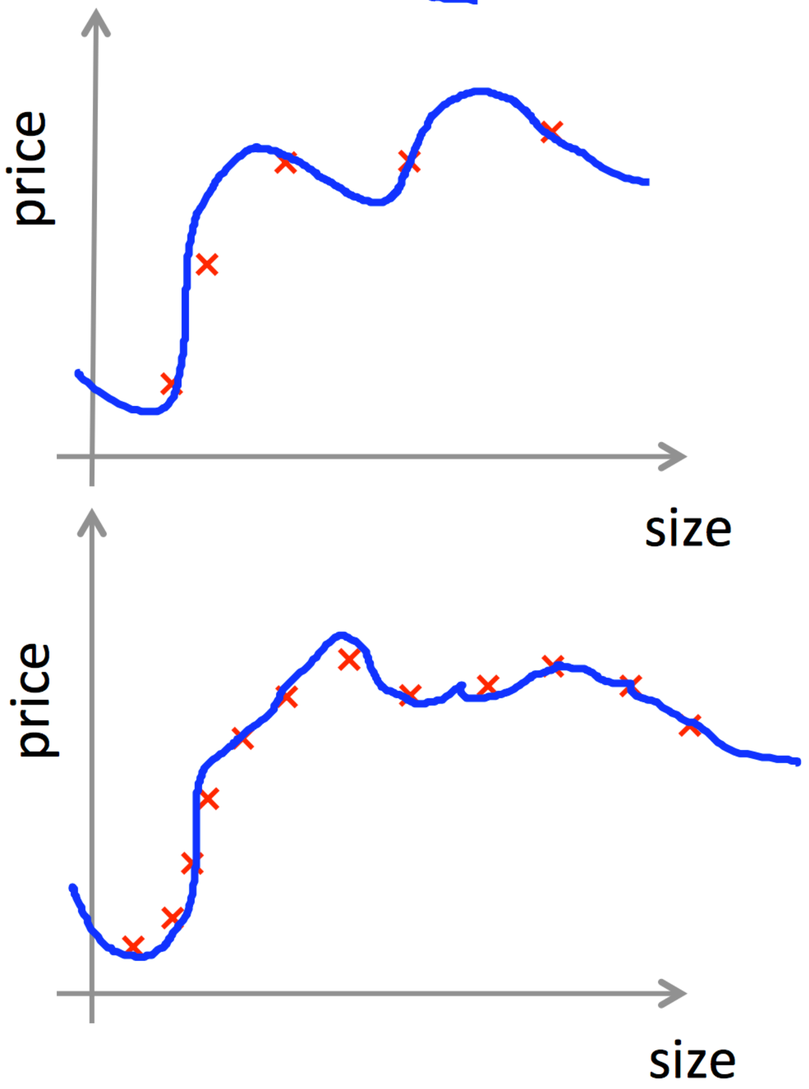
High variance



If a learning algorithm is suffering from high variance, getting more training data is likely to help.

$$h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_{100} x^{100}$$

(and small λ)



Evaluating a Learning Algorithm

Deciding what to do next revisited

Debugging a learning algorithm:

Suppose you have implemented regularized linear regression to predict housing prices. However, when you test your hypothesis in a new set of houses, you find that it makes unacceptably large errors in its prediction. What should you try next?

Evaluating a Learning Algorithm

Deciding what to do next revisited

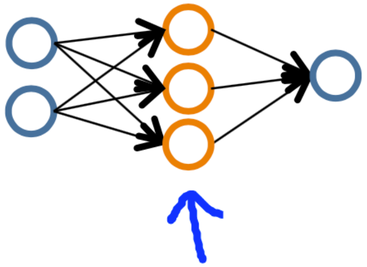
Debugging a learning algorithm:

Suppose you have implemented regularized linear regression to predict housing prices. However, when you test your hypothesis in a new set of houses, you find that it makes unacceptably large errors in its prediction. What should you try next?

Evaluating a Learning Algorithm

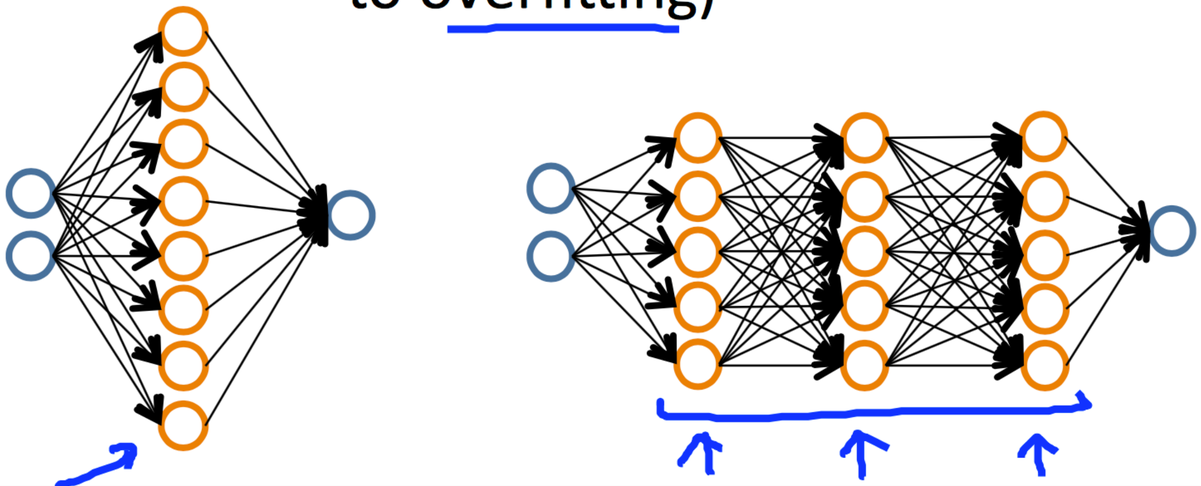
Deciding what to do next revisited

Neural networks and overfitting



Computationally cheaper

"Large" neural network
(more parameters; more prone to overfitting)



$J_{cv}(\Theta)$

Machine Learning System Design

Example Building a spam classifier

Building a spam classifier

From: cheapsales@buystufffromme.com
To: ang@cs.stanford.edu
Subject: Buy now!

Deal of the week! Buy now!
Rolex w4tchs - $100
[redacted](any kind) - $50
Also low cost M0rgages
available.

Spam (1)

From: Alfred Ng
To: ang@cs.stanford.edu
Subject: Christmas dates?

Hey Andrew,
Was talking to Mom about plans
for Xmas. When do you get off
work. Meet Dec 22?
Alf

Non-spam (0)

Machine Learning System Design

Example Building a spam classifier

Building a spam classifier

Supervised learning. $x =$ features of email. $y =$ spam (1) or not spam (0).

Features x: Choose 100 words indicative of spam/not spam.

E.g. deal, buy, discount, andrew, now, ...

$x_j = \begin{cases} 1 & \text{if word } j \text{ appears in email} \\ 0 & \text{otherwise.} \end{cases}$

```
From: cheapsales@buystufffromme.com
To: ang@cs.stanford.edu
Subject: Buy now!

Deal of the week! Buy now!
```

Note: In practice, take most frequently occurring n words (10,000 to 50,000) in training set, rather than manually pick 100 words.

Machine Learning System Design

Example Building a spam classifier

Building a spam classifier

[redacted]

- [redacted]
  - E.g. “honeypot” project.
- [redacted] based on email routing information ([redacted]).
- [redacted] for [redacted] e.g. should “discount” and “discounts” be treated as the same word? How about “deal” and “Dealer”? Features about punctuation?
- [redacted] to [redacted] (e.g. m0rtgage, med1cine, w4tches.)

Machine Learning System Design

Error analysis

Recommended approach

- [redacted]
  Implement it and test it on your cross-validation data.
- [redacted] to decide if more data, more features, etc. are likely to help.
- [redacted] Manually examine the examples (in cross validation set) that your algorithm made errors on. See if you spot any systematic trend in what type of examples it is making errors on.

Machine Learning System Design

Error analysis

Error Analysis

$m_{CV} =$ 500 examples in cross validation set

Algorithm misclassifies 100 emails.

[redacted], and categorize them based on:

→ (i) What type of email it is — pharma, replica, steal passwords, ...

→ (ii) What cues (features) you think would have helped the algorithm classify them correctly.

Pharma: 12

Replica/fake: 4

Steal passwords: 53

Other: 31

Machine Learning System Design

Handling skewed data

Cancer classification example

Train logistic regression model $h_\theta(x)$. ($y = 1$ if cancer, $y = 0$ otherwise)

Find that you got 1% error on test set.

(99% correct diagnoses)

Only 0.50% of patients have cancer.

→ skewed classes.

```
function y = predictCancer(x)
  → y = 0; %ignore x!
return
```

→ 0.5% error

→ 99.2% accuracy (0.8% error)

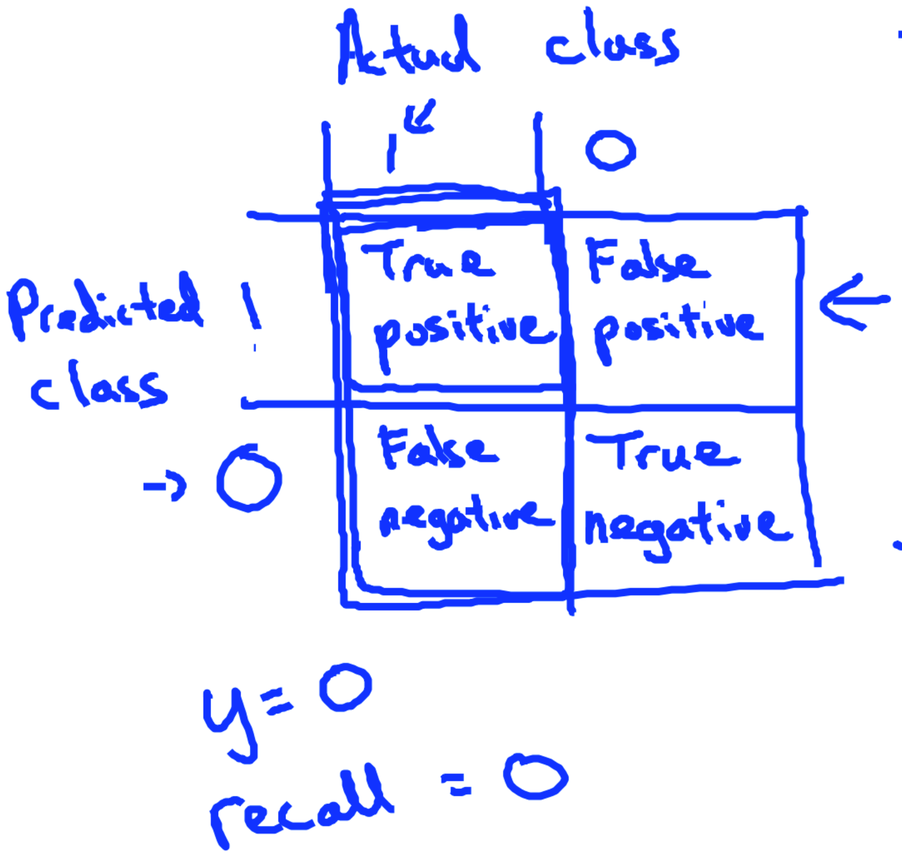
→ 99.5% accuracy (0.5% error)

Machine Learning System Design

Handling skewed data

Precision/Recall

$y = 1$ in presence of rare class that we want to detect



→ **Precision**

(Of all patients where we predicted $y = 1$, what fraction actually has cancer?)

$$\frac{\text{True positives}}{\#\text{predicted positive}} = \frac{\text{True positive}}{\text{True pos + False pos}}$$

→ **Recall**

(Of all patients that actually have cancer, what fraction did we correctly detect as having cancer?)

$$\frac{\text{True positives}}{\#\text{actual positives}} = \frac{\text{True positives}}{\text{True pos + False neg}}$$

Machine Learning System Design

Trading off precision and recall

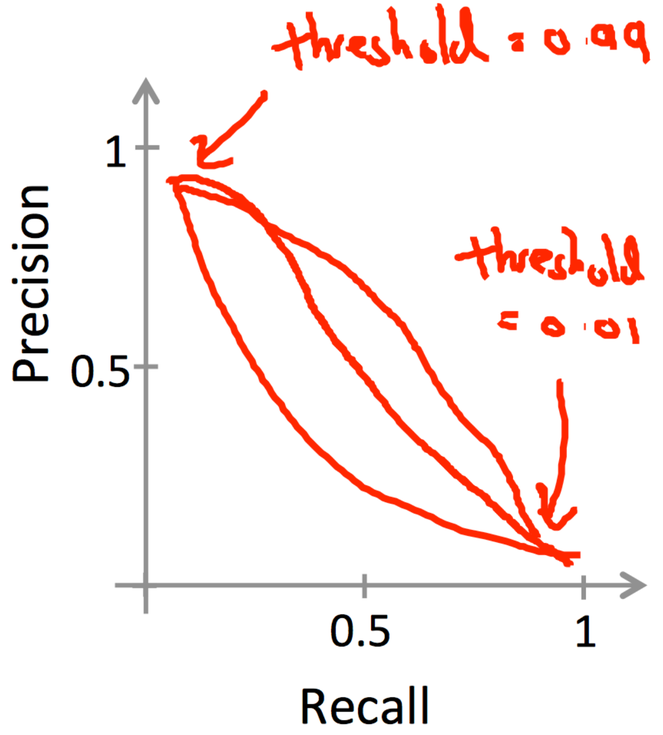
Trading off precision and recall

- Logistic regression: $0 \leq h_\theta(x) \leq 1$
  Predict 1 if $h_\theta(x) \geq 0.5$ 0.7 0.9 0.3
  Predict 0 if $h_\theta(x) < 0.5$ 0.7 0.9 0.3
- Suppose we want to predict $y = 1$ (cancer) only if very confident.
  → Higher precision, lower recall
- Suppose we want to avoid missing too many cases of cancer (avoid false negatives).
  → Higher recall, lower precision.

More generally: Predict 1 if $h_\theta(x) \geq$ threshold.

$$\text{precision} = \frac{\text{true positives}}{\text{no. of predicted positive}}$$

$$\text{recall} = \frac{\text{true positives}}{\text{no. of actual positive}}$$



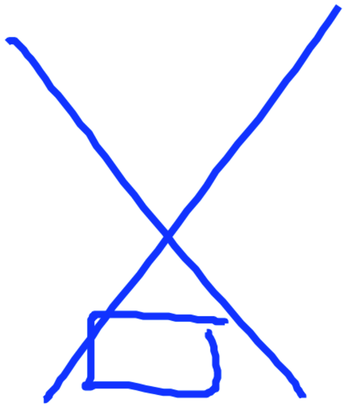
Machine Learning System Design

Trading off precision and recall

F_1 Score (F score)

How to compare precision/recall numbers?

	Precision(P)	Recall (R)
Algorithm 1	0.5	0.4
Algorithm 2	0.7	0.1
Algorithm 3	0.02	1.0



Predict y=1 all the time

Average: $\frac{P+R}{2}$

F_1 Score: $2\frac{PR}{P+R}$

P=0 or R=0 $\Rightarrow$ F-score = 0

P=1 and R=1 $\Rightarrow$ F-score = 1

Using large data sets

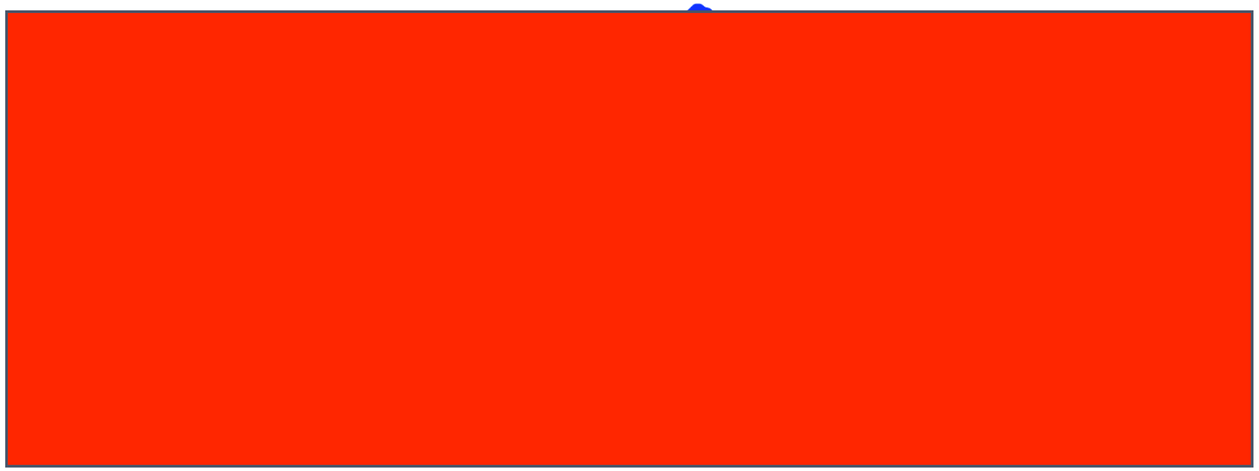
Machine Learning System Design

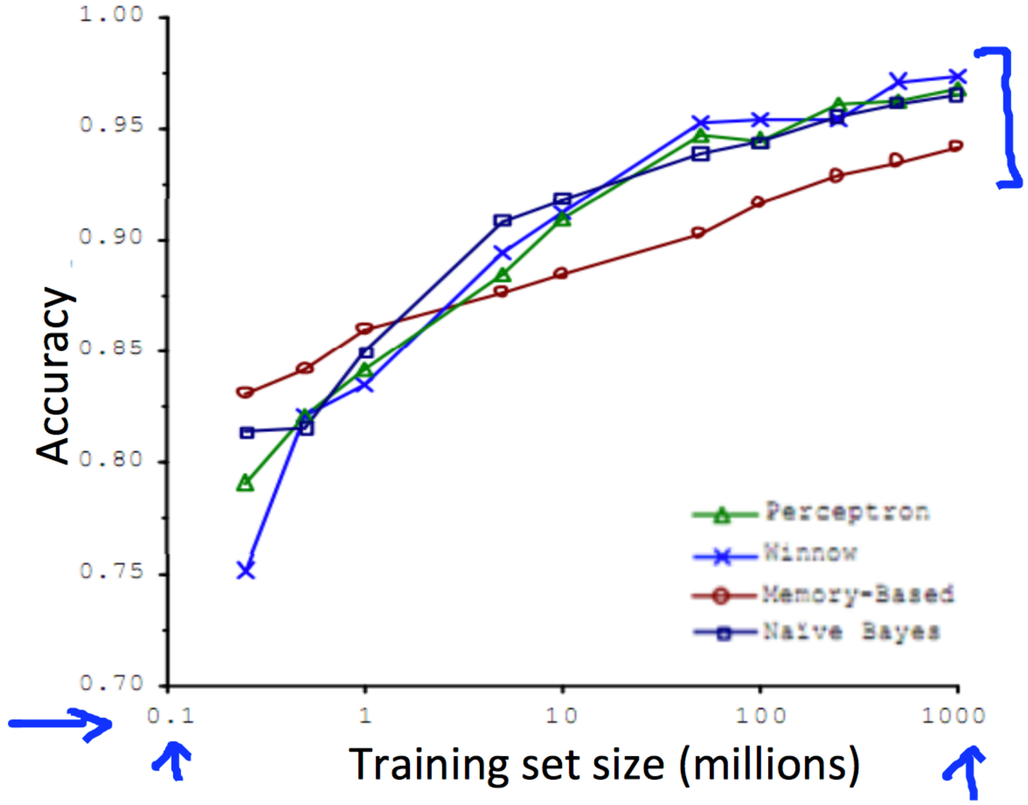
Designing a high accuracy learning system

E.g. Classify between confusable words.

{to, two, too}, {then, than}

For breakfast I ate two eggs.

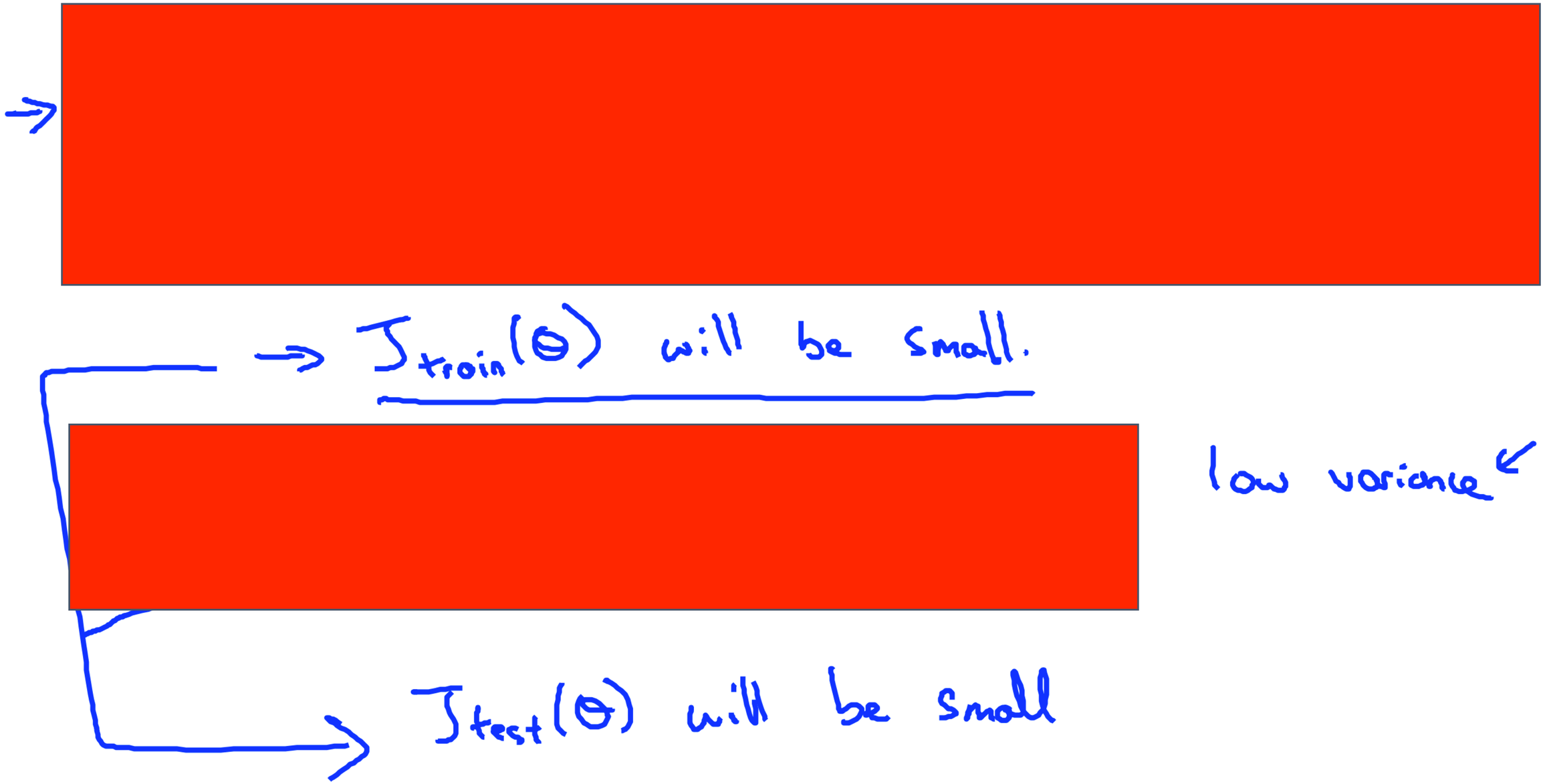




"It's not who has the best algorithm that wins.
It's who has the most data."

Using large data sets

Machine Learning System Design



Unsupervised Learning

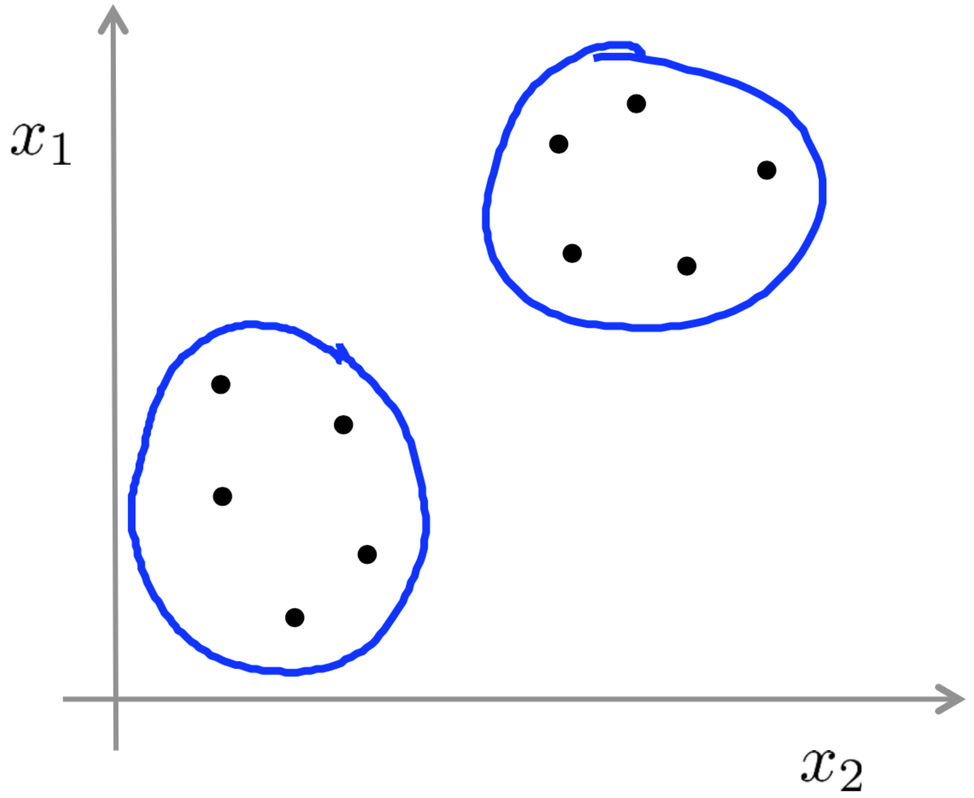
Unsupervised learning, on the other hand, allows us to approach problems with little or no idea what our results should look like. We can derive structure from data where we don't necessarily know the effect of the variables.

We can derive this structure by clustering the data based on relationships among the variables in the data.

With unsupervised learning there is no feedback based on the prediction results, i.e., there is no teacher to correct you.

Clustering

Unsupervised learning

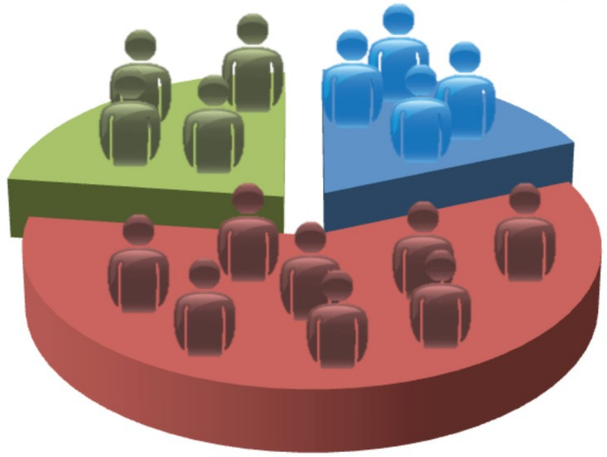


Clustering algorithm

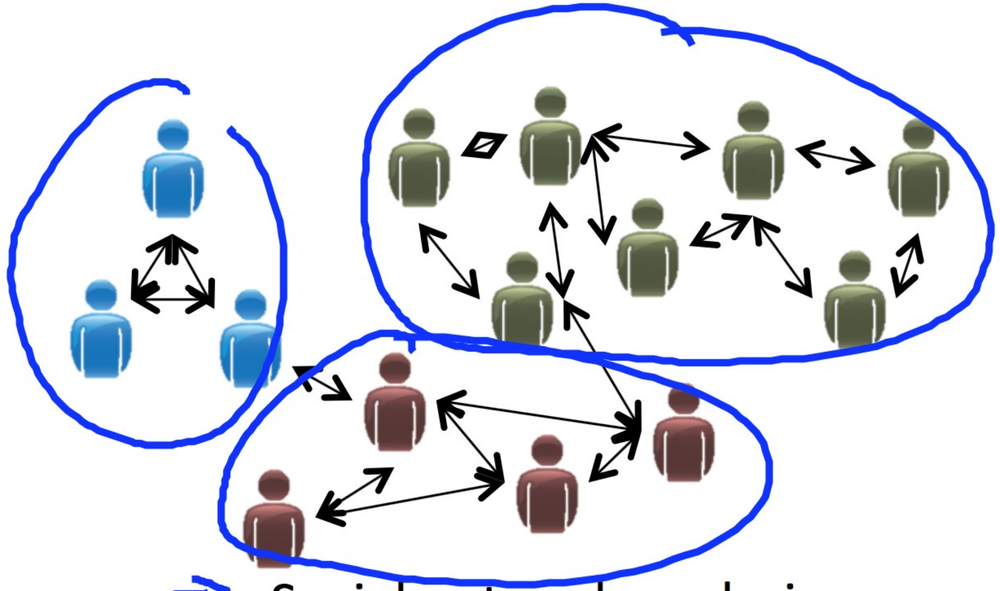
Training set: $\{x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(m)}\}$

Clustering

Applications of clustering



→ Market segmentation



→ Social network analysis

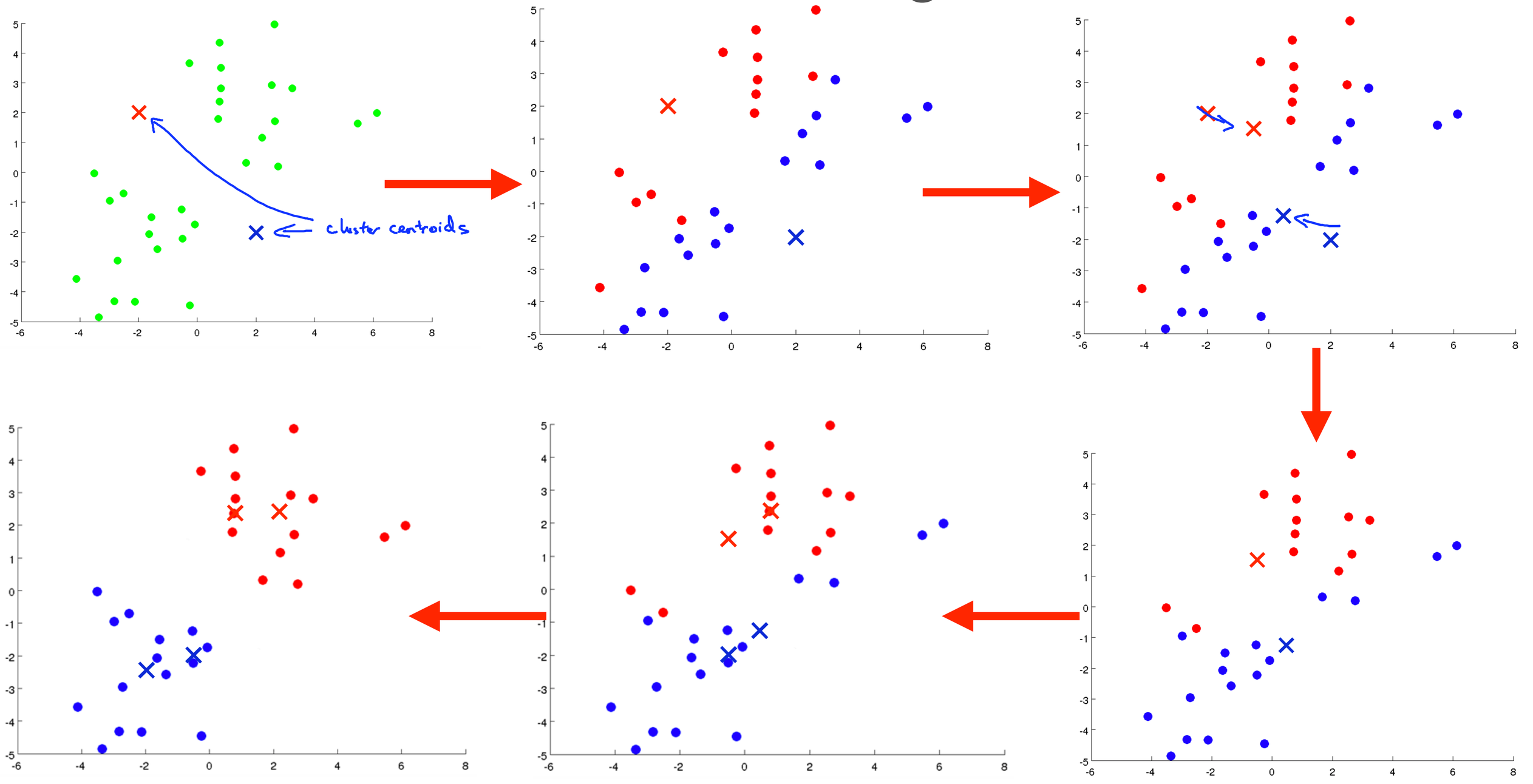
→ Organize computing clusters

Image credit: NASA/JPL-Caltech/E. Churchwell (Univ. of Wisconsin, M

→ Astronomical data analysis

K-Means Algorithm

Clustering



Clustering

K-means algorithm

μ_1 μ_2

Randomly initialize K cluster centroids $\mu_1, \mu_2, \ldots, \mu_K \in \mathbb{R}^n$

Repeat {

Cluster assignment step

for i = 1 to m

$c^{(i)}$:= index (from 1 to K) of cluster centroid closest to $x^{(i)}$

$\min_k \|x^{(i)} - \mu_k\|^2$ → $c^{(i)}$

Move centroid

for k = 1 to K

→ μ_k := average (mean) of points assigned to cluster k

$x^{(1)}, x^{(5)}, x^{(6)}, x^{(10)}$ → $c^{(1)}=2$, $c^{(5)}=2$, $c^{(6)}=2$, $c^{(10)}=2$

$$\mu_2 = \frac{1}{4}\left[x^{(1)} + x^{(5)} + x^{(6)} + x^{(10)}\right] \in \mathbb{R}^n$$

}

Clustering

Optimization objective

K-means optimization objective

→ $c^{(i)}$ = index of cluster (1,2,…,K) to which example $x^{(i)}$ is currently assigned

→ μ_k = cluster centroid k ($\mu_k \in \mathbb{R}^n$)

$\mu_{c^{(i)}}$ = cluster centroid of cluster to which example $x^{(i)}$ has been assigned

K $\quad k \in \{1,2,\dots,K\}$

$x^{(i)} \rightarrow 5$ $\quad c^{(i)} = 5$ $\quad \mu_{c^{(i)}} = \mu_5$

Clustering

K-means algorithm

Randomly initialize K cluster centroids $\mu_1, \mu_2, \ldots, \mu_K \in \mathbb{R}^n$

Repeat {

for i = 1 to m

$c^{(i)}$:= index (from 1 to K) of cluster centroid closest to $x^{(i)}$

for k = 1 to K

μ_k := average (mean) of points assigned to cluster k

}

Random initialization

Random initialization

Should have $K < m$

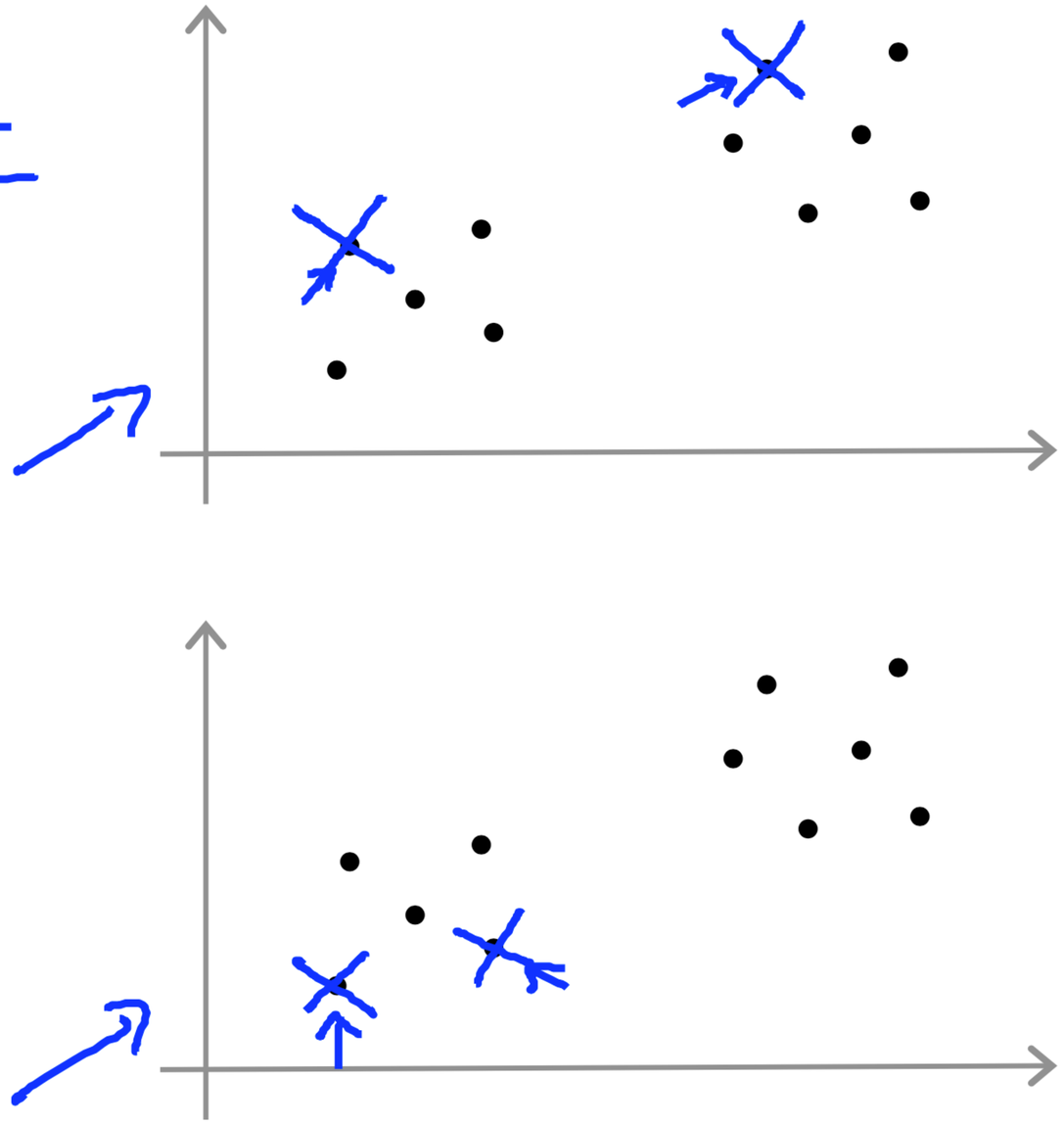
$K = 2$

Randomly pick K training examples.

Set $\mu_1, \ldots, \mu_K$ equal to these K examples.

$\mu_1 = x^{(i)}$

$\mu_2 = x^{(j)}$



Clustering

Random initialization

For i = 1 to 100 {

Randomly initialize K-means.

Run K-means. Get $c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K$.

Compute cost function (distortion)

$J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K)$

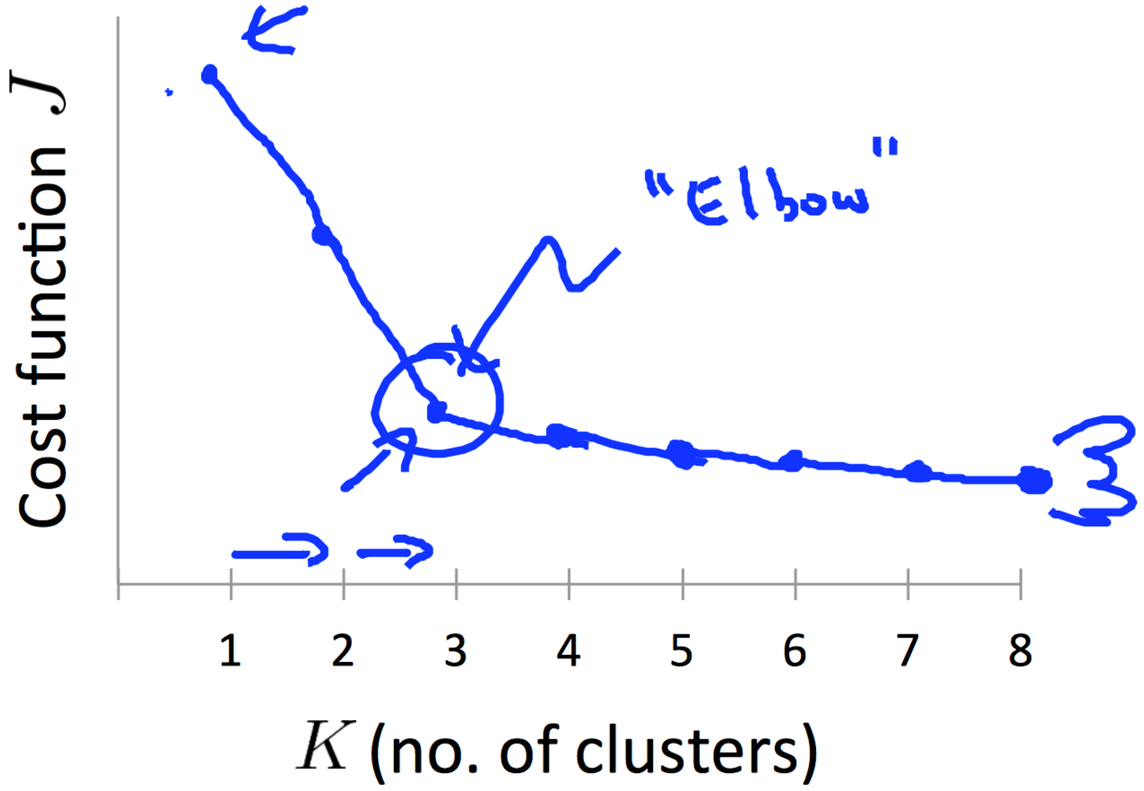
}

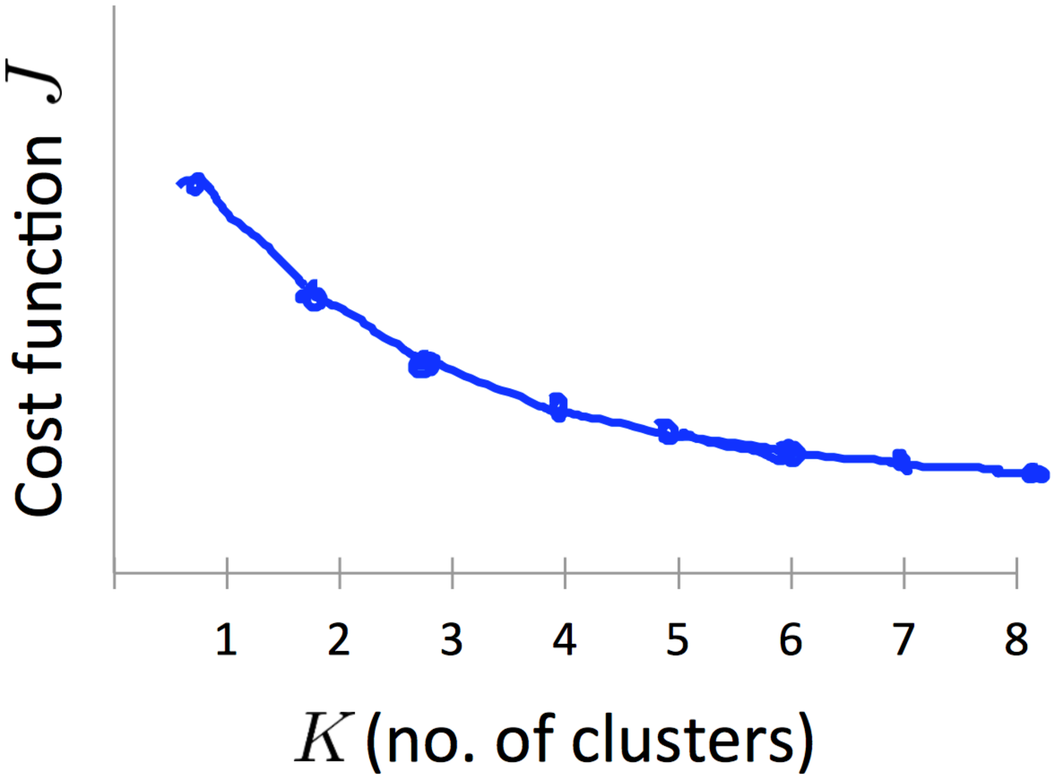
Pick clustering that gave lowest cost $J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K)$

Clustering

Choosing the value of K

Elbow method:





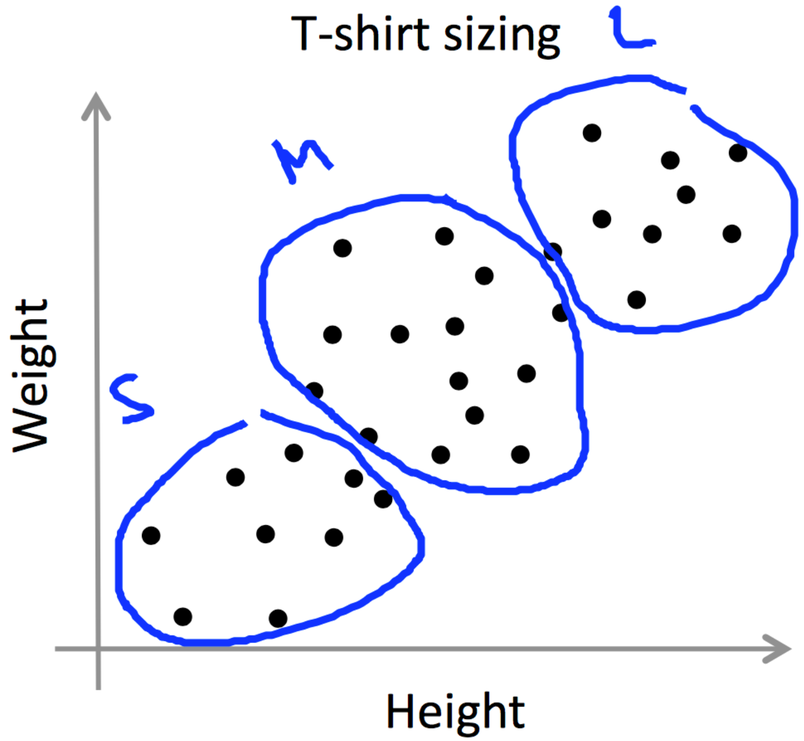
Clustering

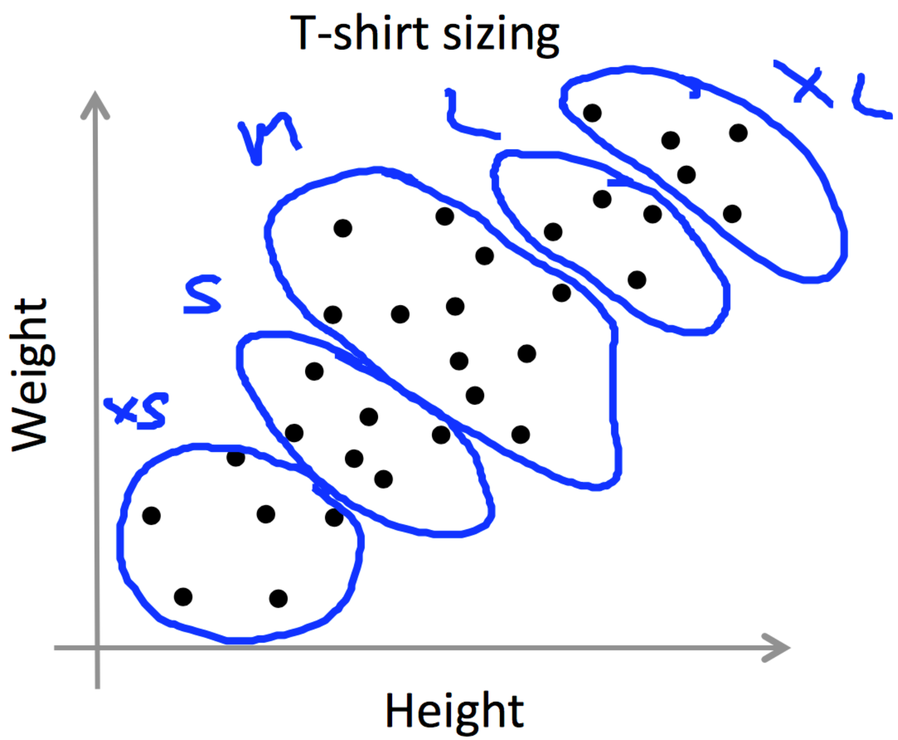
Choosing the number of clusters

K=3 S, M, L

K=5 XS, S, M, L, XL

E.g.





Dimensionality Reduction

In this module, we introduce Principal Components Analysis, and show how it can be used for data compression to speed up learning algorithms as well as for visualizations of complex datasets.

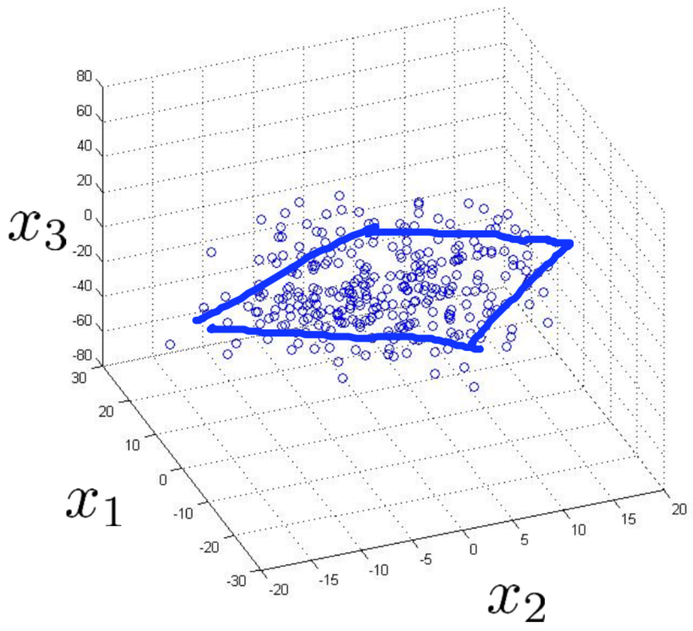
Dimensionality Reduction

Motivation I: Data Compression

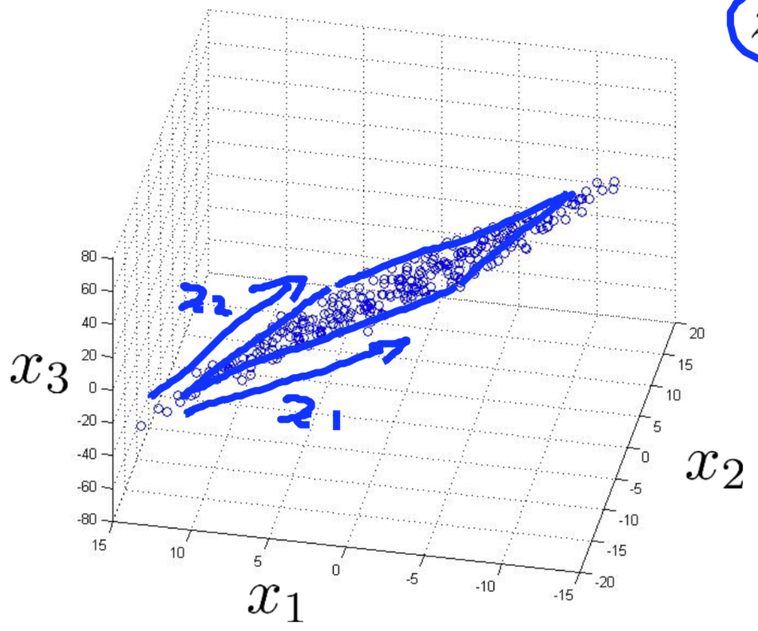
Data Compression

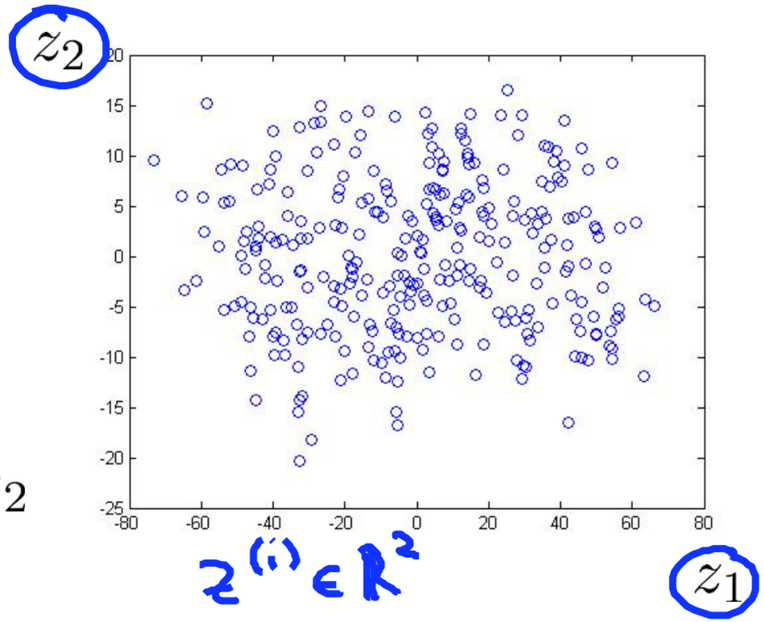
1000D → 100D

Reduce data from 3D to 2D



$x^{(i)} \in \mathbb{R}^3$





$z^{(i)} \in \mathbb{R}^2$

$$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \qquad z^{(i)} = \begin{bmatrix} z_1^{(i)} \\ z_2^{(i)} \end{bmatrix}$$

Dimensionality Reduction

Motivation II: Visualization

Data Visualization

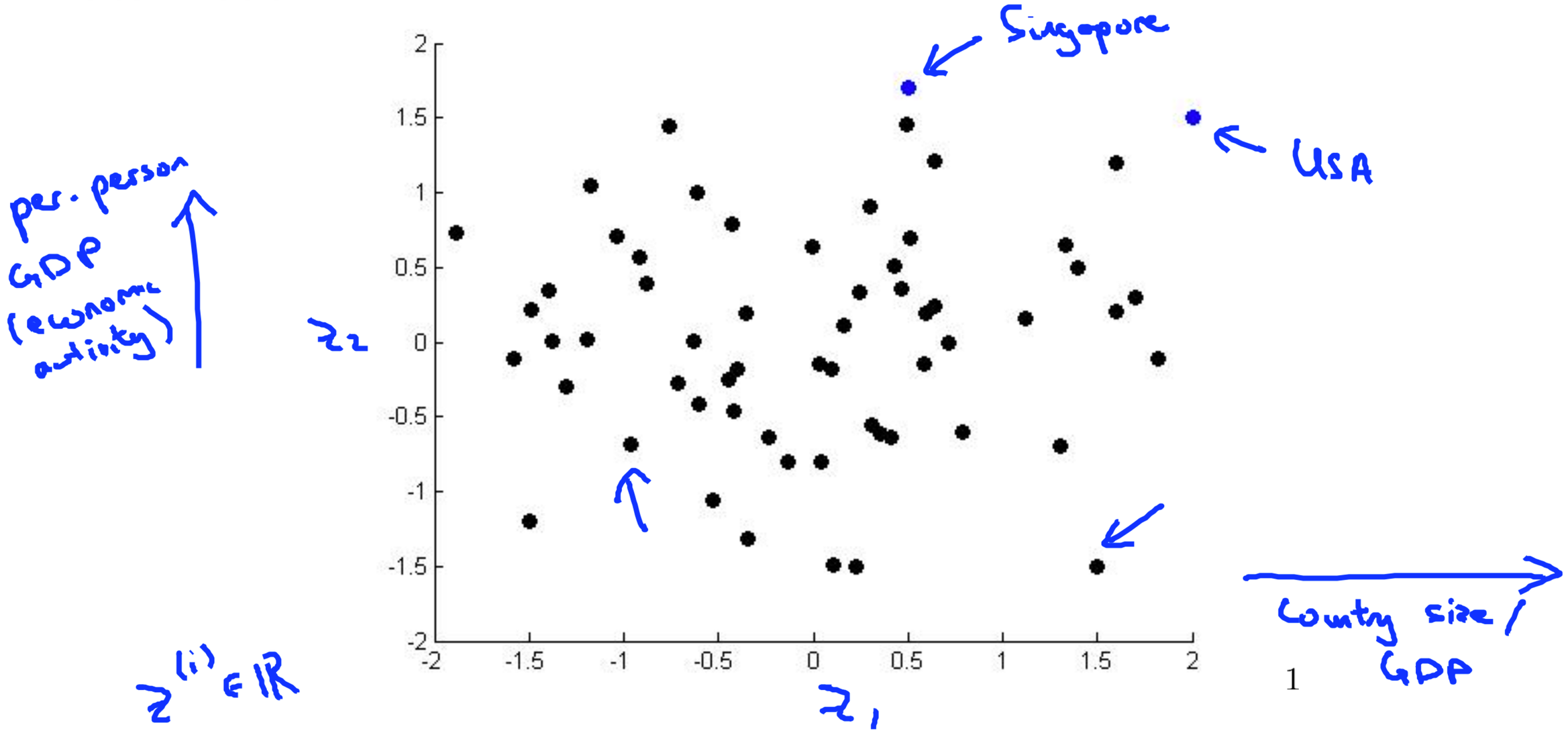
$x \in \mathbb{R}^{50}$ $x^{(i)} \in \mathbb{R}^{50}$

Country	x_1 GDP (trillions of US$)	x_2 Per capita GDP (thousands of intl. $)	x_3 Human Develop-ment Index	x_4 Life expectancy	x_5 Poverty Index (Gini as percentage)	x_6 Mean household income (thousands of US$)	…
Canada	1.577	39.17	0.908	80.7	32.6	67.293	…
China	5.878	7.54	0.687	73	46.9	10.22	…
India	1.632	3.41	0.547	64.7	36.8	0.735	…
Russia	1.48	19.84	0.755	65.5	39.9	0.72	…
Singapore	0.223	56.69	0.866	80	42.5	67.1	…
USA	14.527	46.86	0.91	78.3	40.8	84.3	…
…	…	…	…	…	…	…	

Dimensionality Reduction

Motivation II: Visualization

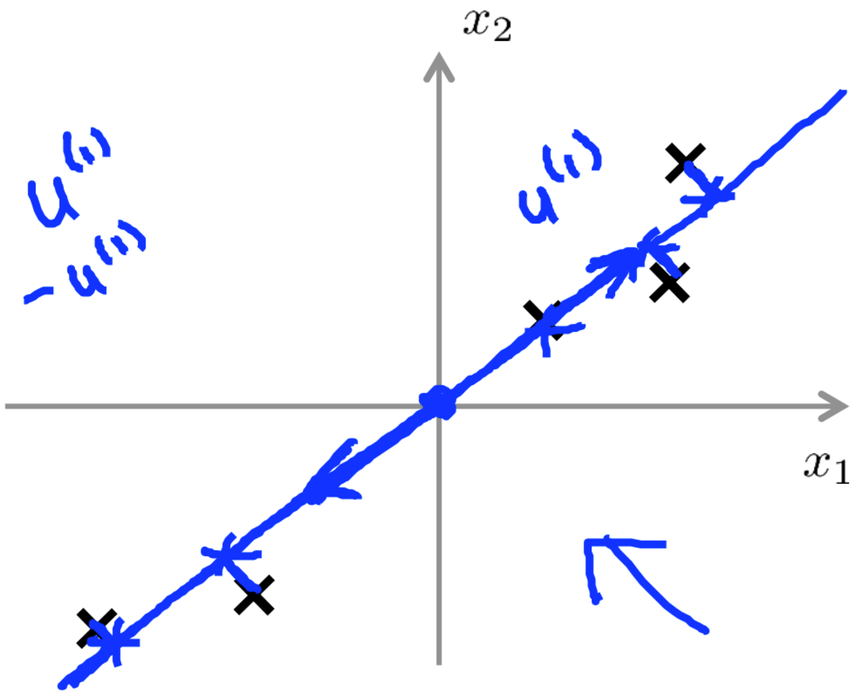
Data Visualization



Dimensionality Reduction

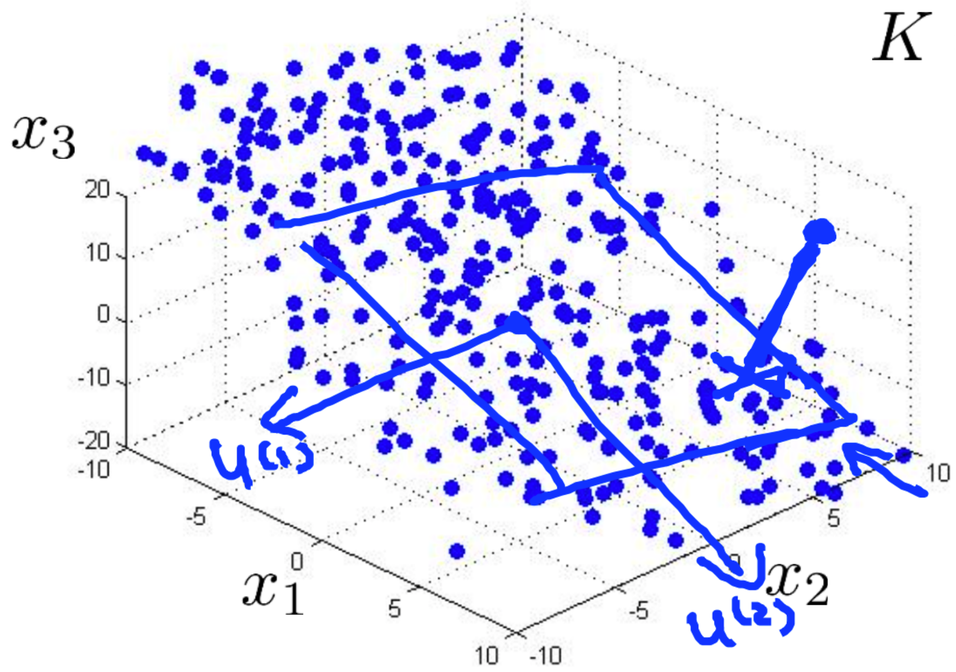
Principle component analysis

Principal Component Analysis (PCA) problem formulation



$3D \rightarrow 2D$

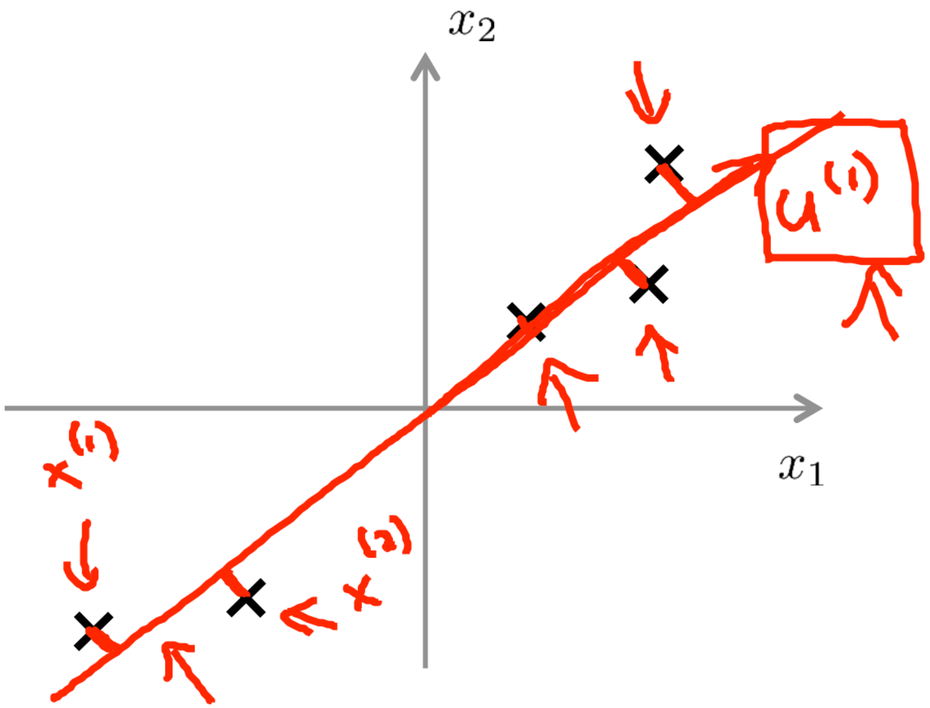
$K = 2$



Dimensionality Reduction

Principle component analysis

Principal Component Analysis (PCA) algorithm

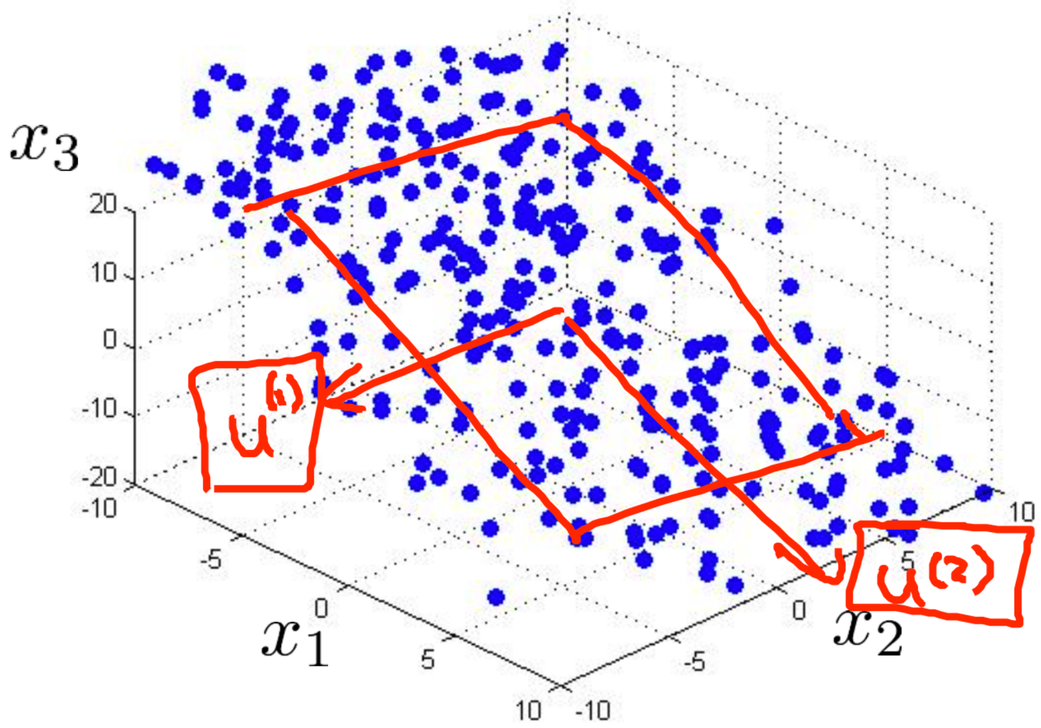


Reduce data from 2D to 1D

$z^{(1)}$ $x^{(i)} \in \mathbb{R}^2 \rightarrow z^{(i)} \in \mathbb{R}$

$x_2^{(i)}$

z_1



Reduce data from 3D to 2D

$x^{(i)} \in \mathbb{R}^3 \rightarrow z^{(i)} \in \mathbb{R}^2$

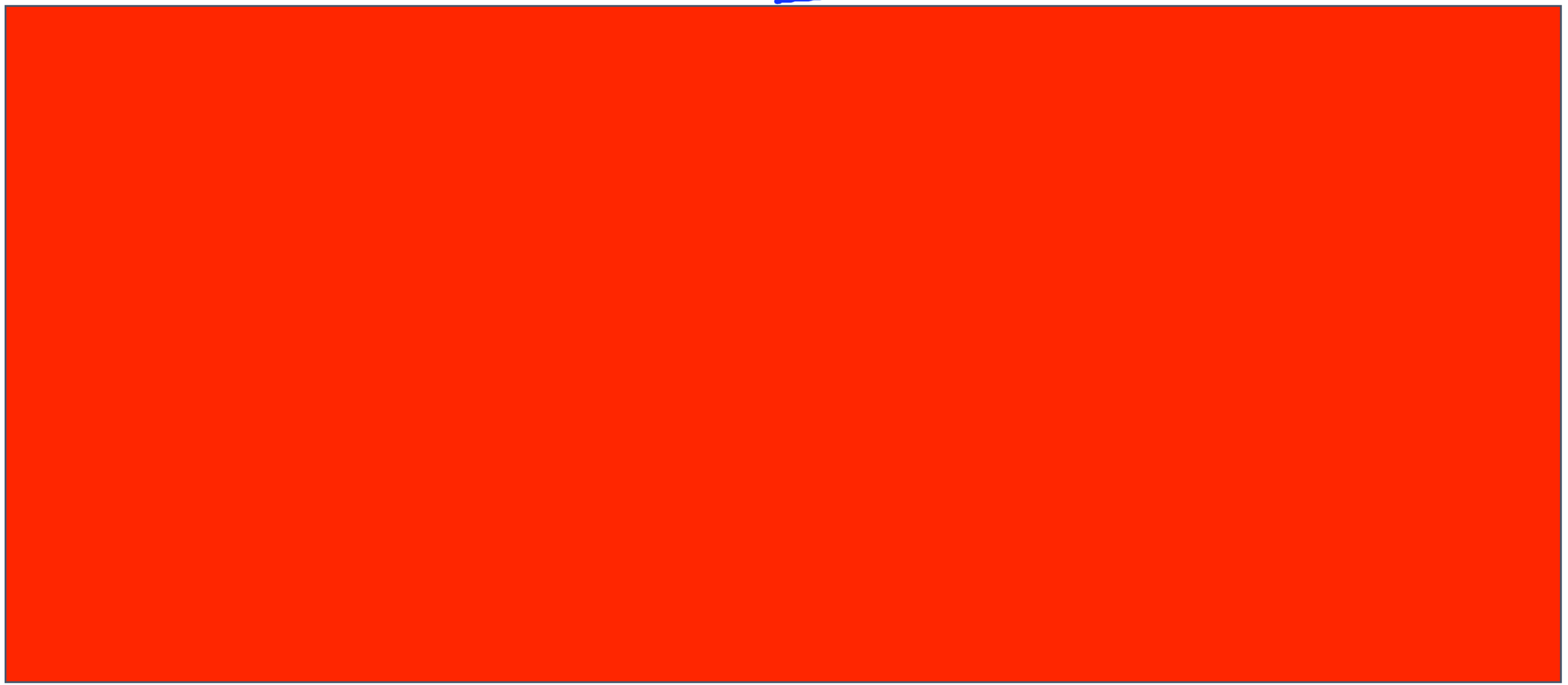
$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$

Dimensionality Reduction

Principle component analysis: Algorithm

Principal Component Analysis (PCA) algorithm

Reduce data from n-dimensions to k-dimensions

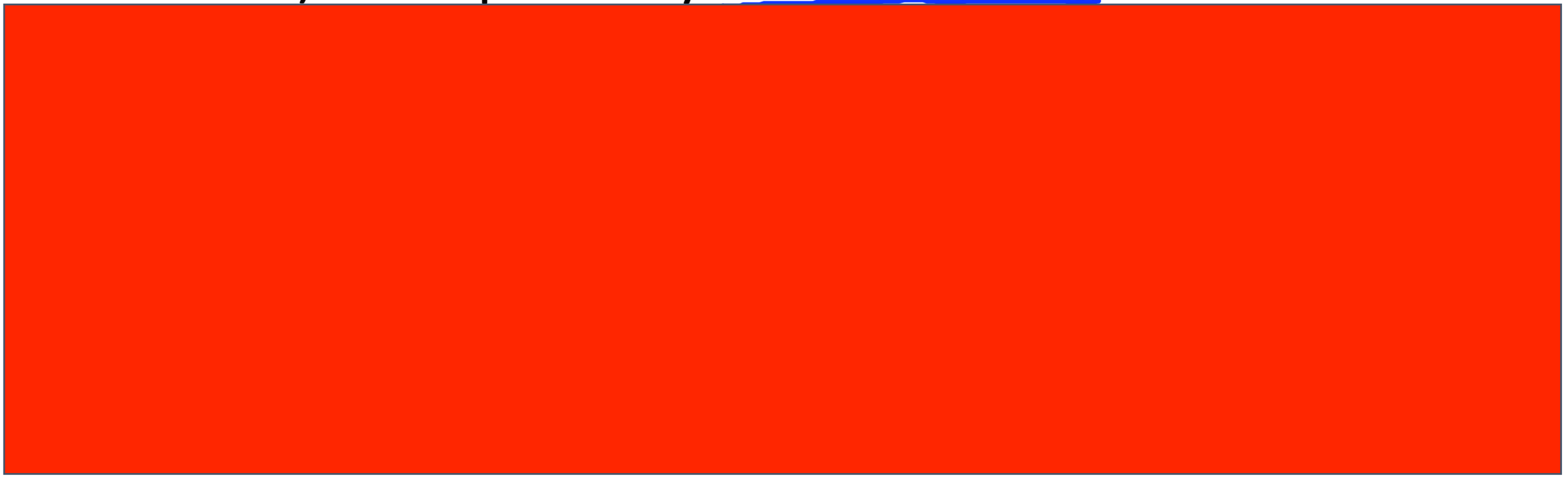


Dimensionality Reduction

Principle component analysis: Algorithm

Principal Component Analysis (PCA) algorithm summary

After mean normalization (ensure every feature has zero mean) and optionally feature scaling:

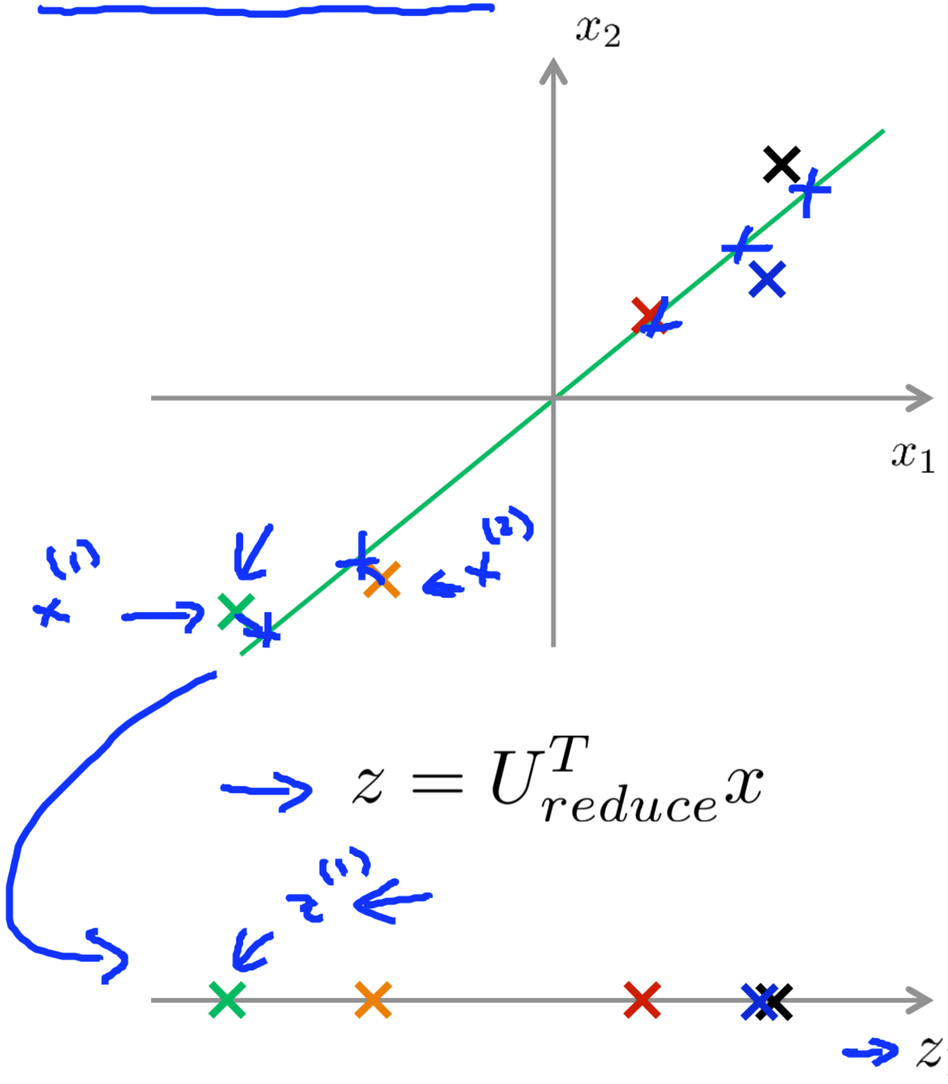


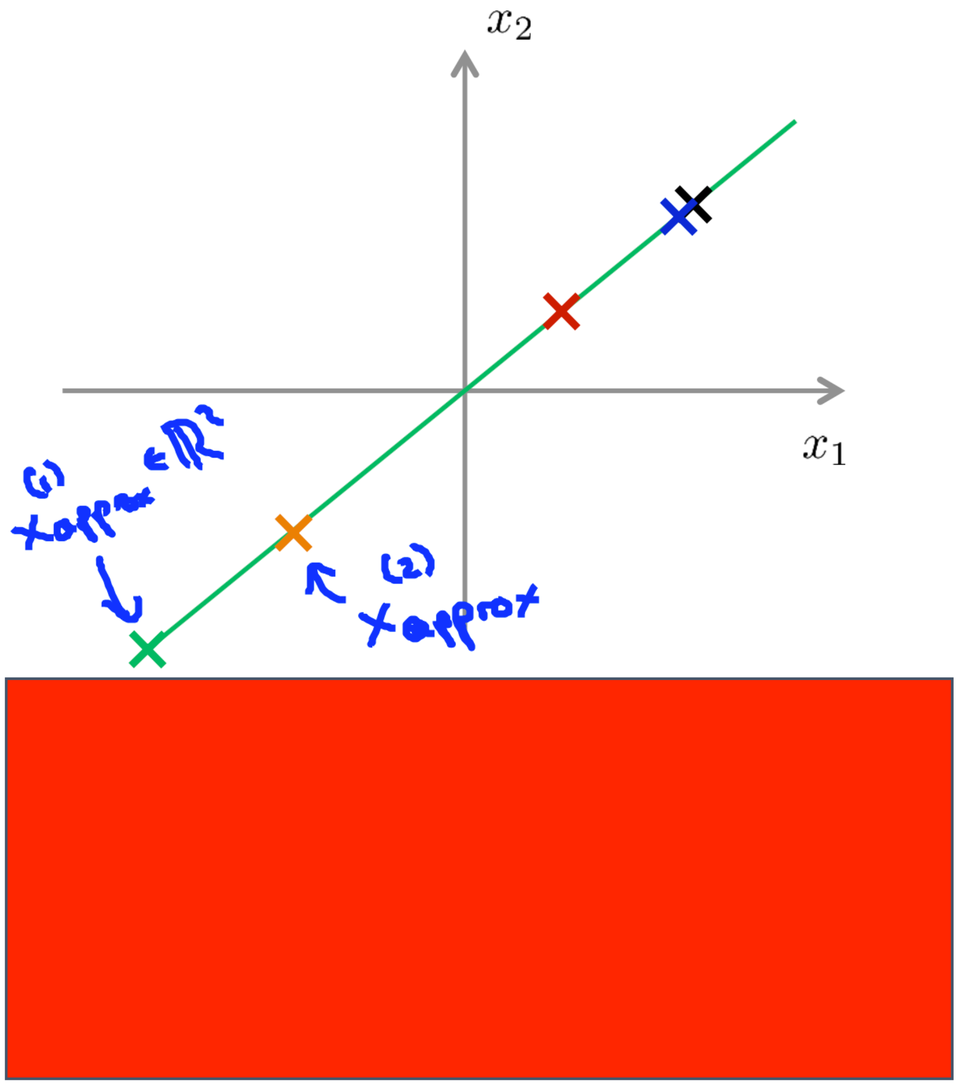
Applying PCA

Dimensionality Reduction

Reconstruction from Compressed Representation

Reconstruction from compressed representation





Applying PCA

Dimensionality Reduction

Choosing the number of principal components

Choosing k (number of principal components)

Average squared projection error: $\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)} - x_{approx}^{(i)}\|^2$

Total variation in the data: $\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)}\|^2$

- "99% of variance is retained"

95 to 90%

Applying PCA

Dimensionality Reduction

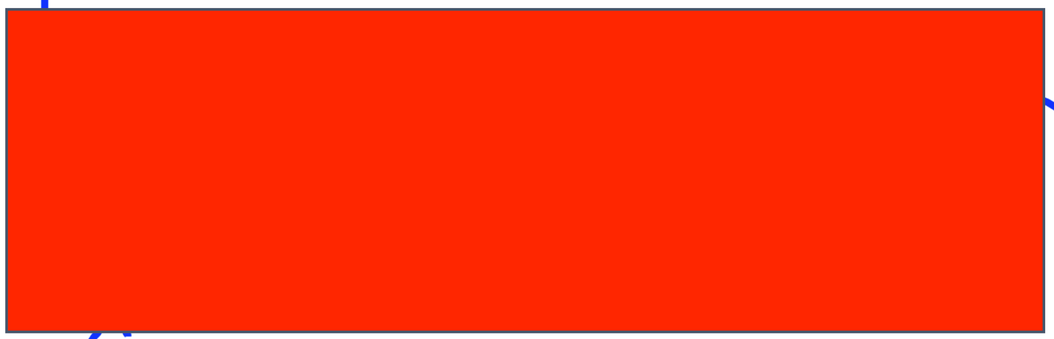
Choosing the number of principal components

Choosing k (number of principal components)

Algorithm:

Try PCA with ~~$k = 1$~~ ~~$k = 2$~~ $k = 3$ ... $k = 9$

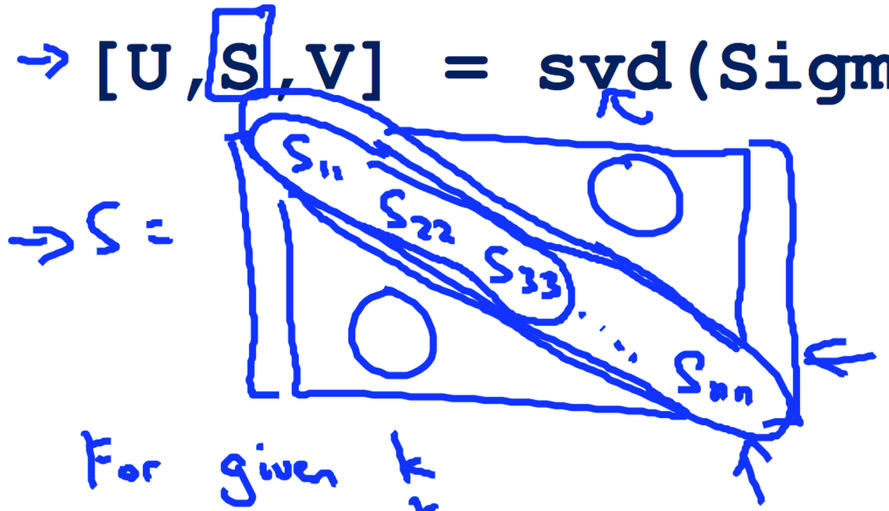
Compute $U_{reduce}, z^{(1)}, z^{(2)}, \ldots, z^{(m)}, x^{(1)}_{approx}, \ldots, x^{(m)}_{approx}$



$k = 17$

The singular values for every matrix, sorted in descending order.

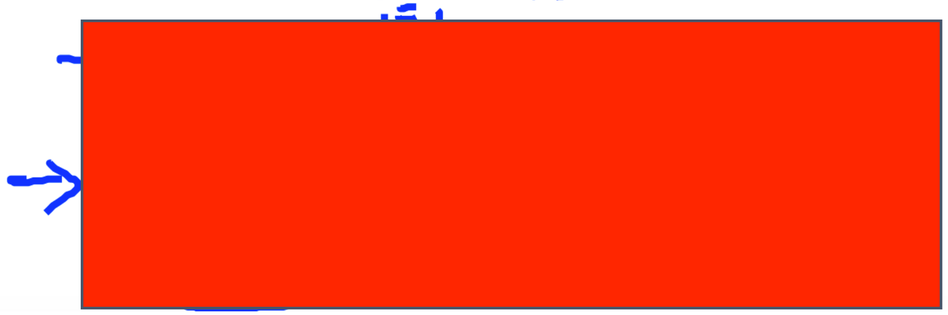
→ [U, S, V] = svd(Sigma)



For given k

$$1 - \frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}} \leq 0.01$$

$k = 3$



https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.linalg.svd.html

Andrew

Applying PCA

Advice for applying PCA

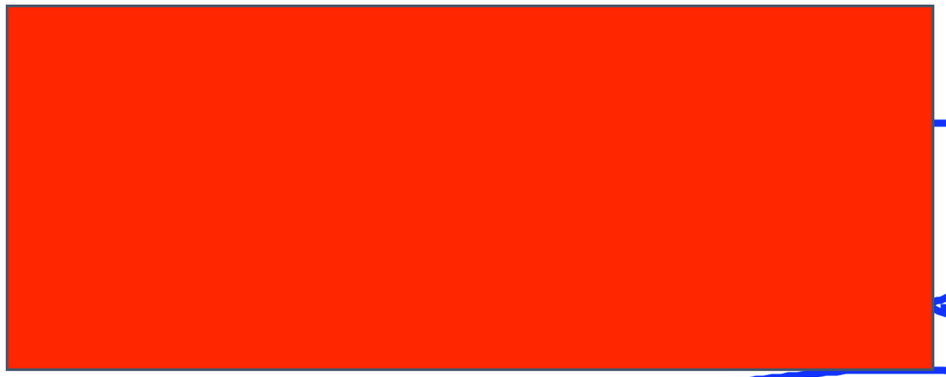
Dimensionality Reduction

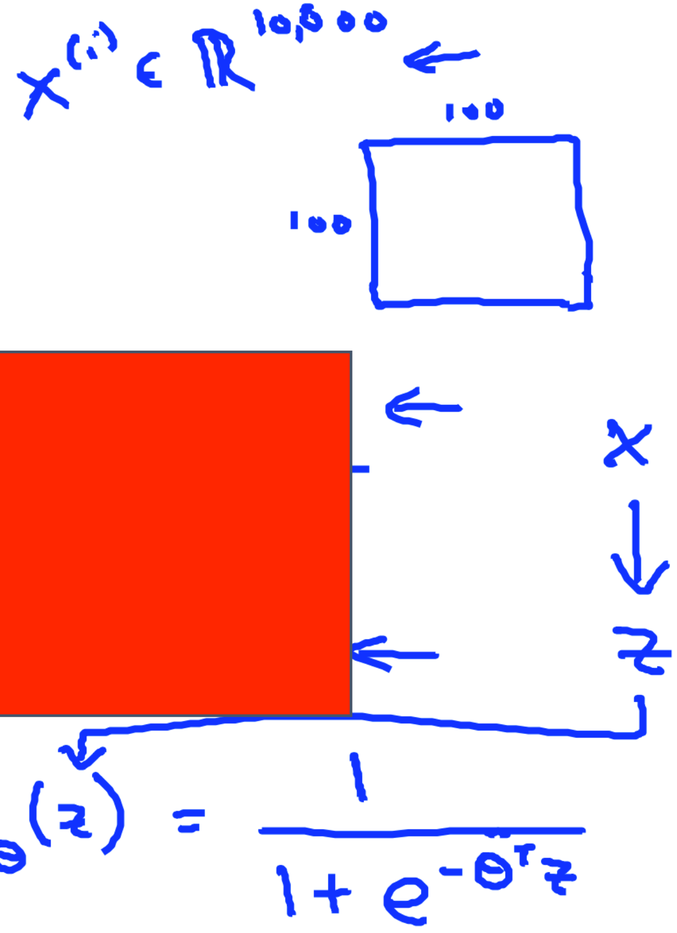
Supervised learning speedup

$(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$

Extract inputs:

Unlabeled dataset:





U reduce

New training set:

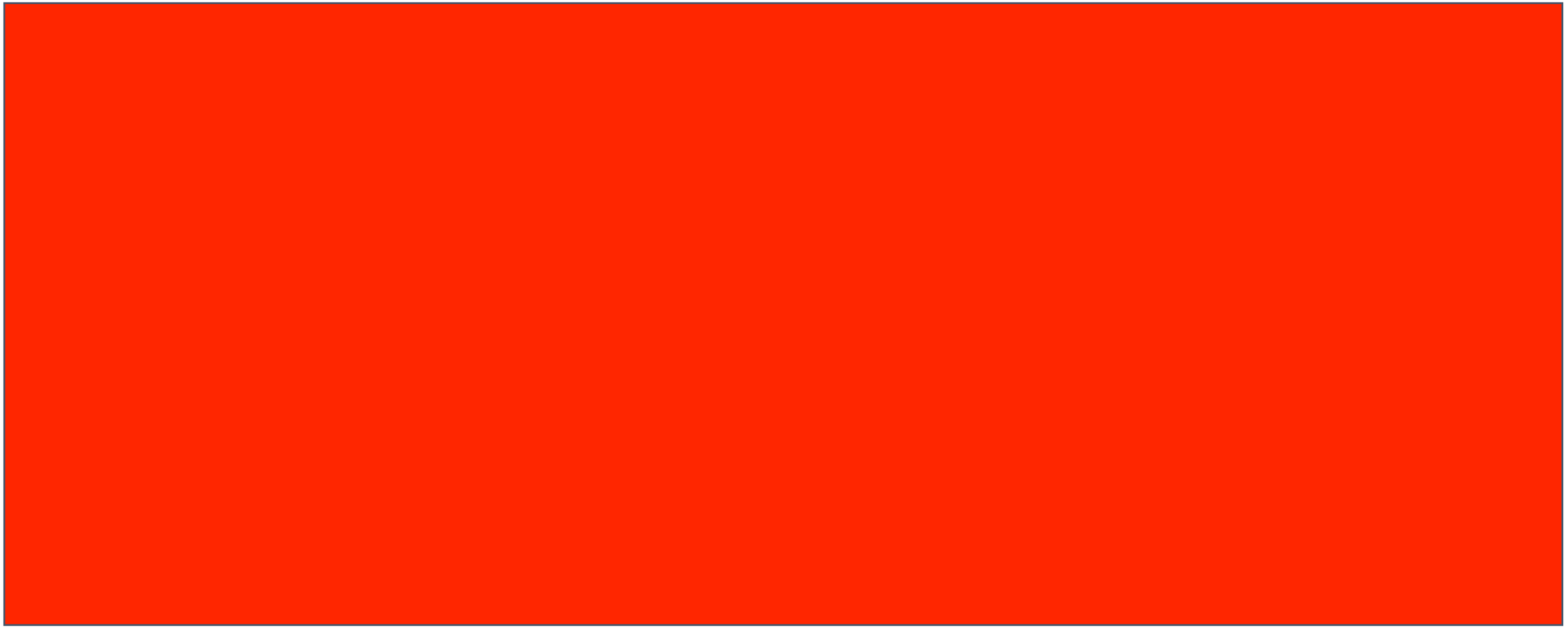
$(z^{(1)}, y^{(1)}), (z^{(2)}, y^{(2)}), \ldots, (z^{(m)}, y^{(m)})$

Note: Mapping $x^{(i)} \rightarrow z^{(i)}$ should be defined by running PCA only on the training set. This mapping can be applied as well to the examples $x_{cv}^{(i)}$ and $x_{test}^{(i)}$ in the cross validation and test

x → z

Dimensionality Reduction

Application of PCA



Applying PCA

Advice for applying PCA

Dimensionality Reduction

[redacted]

Use $z^{(i)}$ instead of $x^{(i)}$ to reduce the number of features to $k < n$.

1000

10000

Thus, fewer features, less likely to overfit.

Bad!

[redacted]

Applying PCA

Advice for applying PCA

Dimensionality Reduction

PCA is sometimes used where it shouldn't be

Design of ML system:

- Get training set $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$
- ~~Run PCA to reduce $x^{(i)}$ in dimension to get $z^{(i)}$~~
- Train logistic regression on $\{(z^{(1)}, y^{(1)}), \ldots, (z^{(m)}, y^{(m)})\}$
- Test on test set: Map $x_{test}^{(i)}$ to $z_{test}^{(i)}$. Run $h_\theta(z)$ on $\{(z_{test}^{(1)}, y_{test}^{(1)}), \ldots, (z_{test}^{(m)}, y_{test}^{(m)})\}$

Recommender Systems

Recommender systems, which are used by companies like Amazon, Netflix and Apple to recommend products to their users. Recommender systems look at patterns of activities between different users and different products to produce these recommendations. In these lessons, we introduce recommender algorithms such as the collaborative filtering algorithm and low-rank matrix factorization.

Introduction to Python Recommendation System

1. Popularity-Based Recommender Systems
2. Correlation-Based Recommender Systems
3. Classification-Based Collaborative Filtering
4. Model-Based Collaborative Filtering
5. Content-Based Recommender Systems
6. Evaluating of Recommender Systems

Large Scale Machine Learning

Machine learning works best when there is an abundance of data to leverage for training. With the amount data that many websites/companies are gathering today, knowing how to handle ‘big data’ is one of the most sought after skills in Silicon Valley.

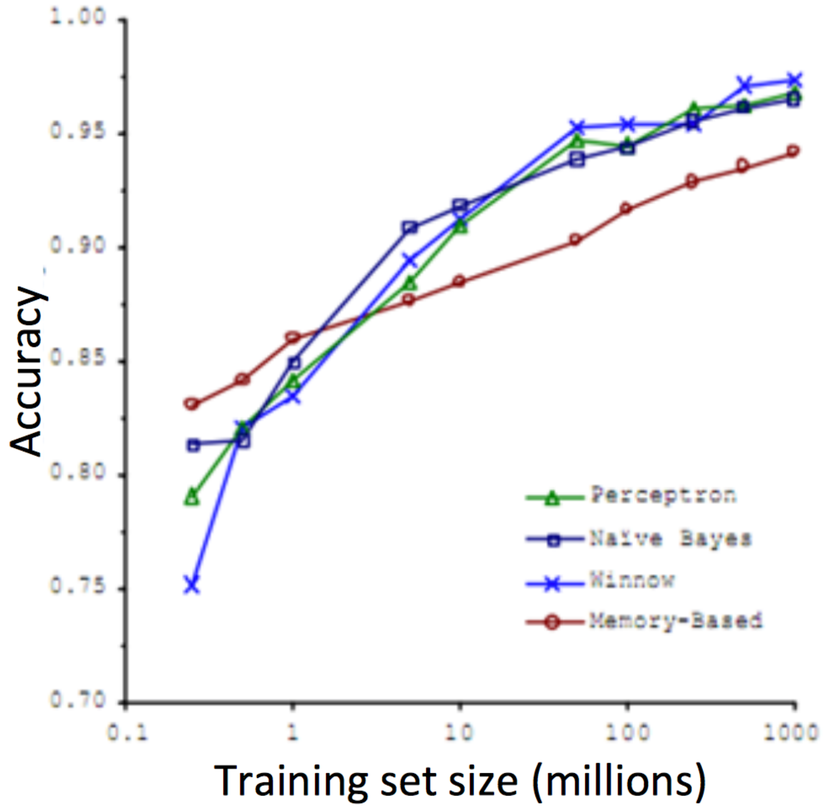
Gradient Descent with Large Datasets

Learning with Large Datasets

Machine learning and data

Classify between confusable words.
E.g., {to, two, too}, {then, than}.

For breakfast I ate two eggs.



"It's not who has the best algorithm that wins.
It's who has the most data."

Gradient Descent with Large Datasets

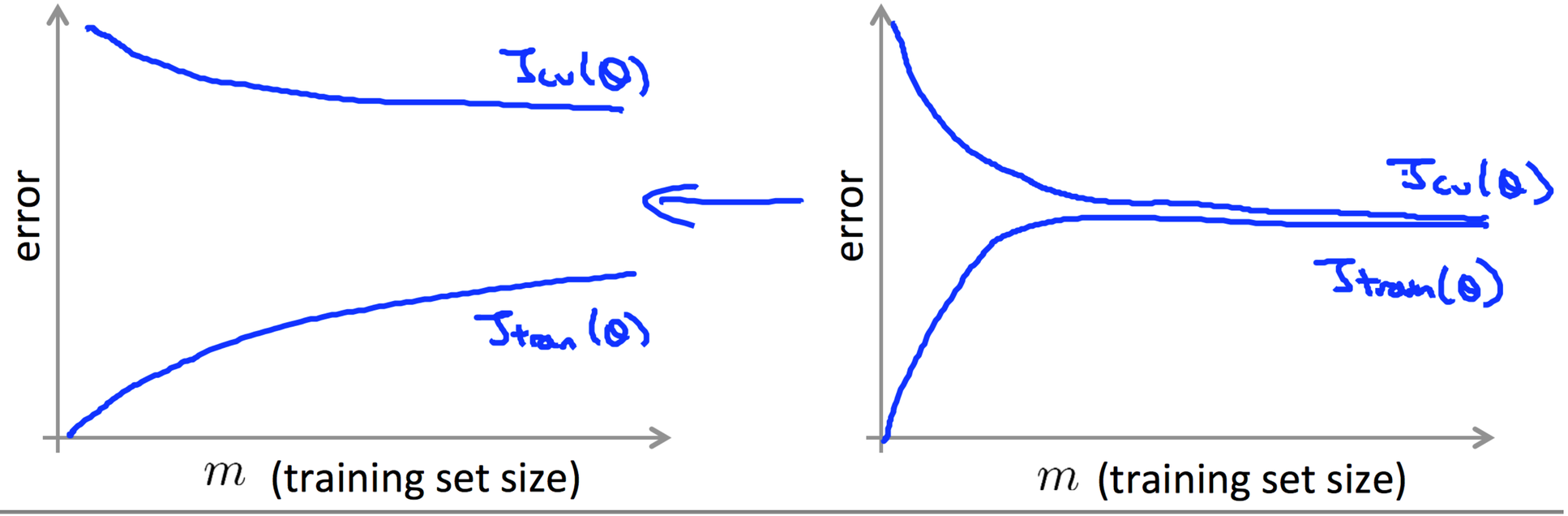
Learning with Large Datasets

Learning with large datasets

$m = 100,000,000$

$m = 1,000?$

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$



Gradient Descent with Large Datasets

Stochastic Gradient Descent

Linear regression with gradient descent

Recap

$$h_\theta(x) = \sum_{j=0}^{n} \theta_j x_j$$

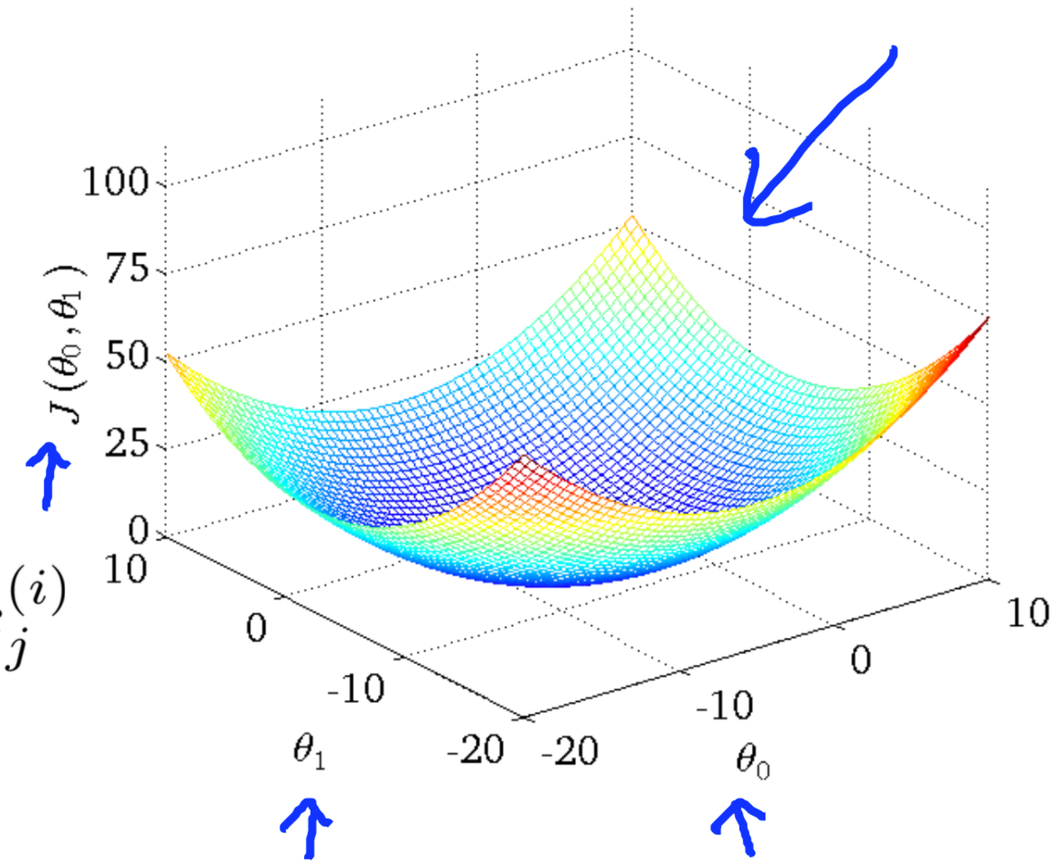
$$J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Repeat {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(for every $j = 0, \ldots, n$)

}



Gradient Descent with Large Datasets

Stochastic Gradient Descent

Linear regression with gradient descent

Recap

$$h_\theta(x) = \sum_{j=0}^{n} \theta_j x_j$$

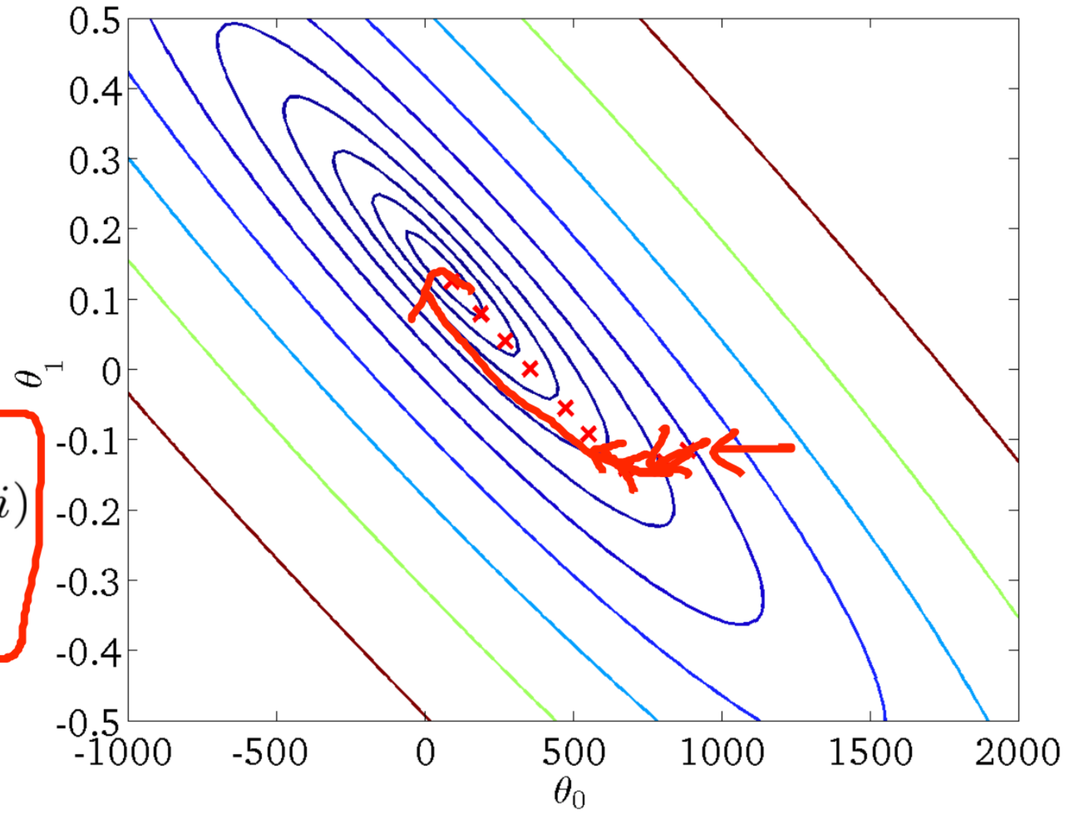
$$J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Repeat {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(for every $j = 0, \ldots, n$)

}



$m = 300,000,000$

Batch gradient descent

Gradient Descent with Large Datasets

Stochastic Gradient Descent

Batch gradient descent

$$J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

Repeat {

$$\theta_j := \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

$\frac{\partial}{\partial \theta_j} J_{train}(\theta)$

(for every $j = 0, \ldots, n$)

}

$m = 300{,}000{,}000$

Stochastic gradient descent

$$cost(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$$J_{train}(\theta) = \frac{1}{m}\sum_{i=1}^{m} cost(\theta, (x^{(i)}, y^{(i)}))$$

Gradient Descent with Large Datasets

Stochastic Gradient Descent

Stochastic gradient descent

1. Randomly shuffle (reorder) training examples

2. Repeat { 1-10×

   for $i := 1, \ldots, m$ {

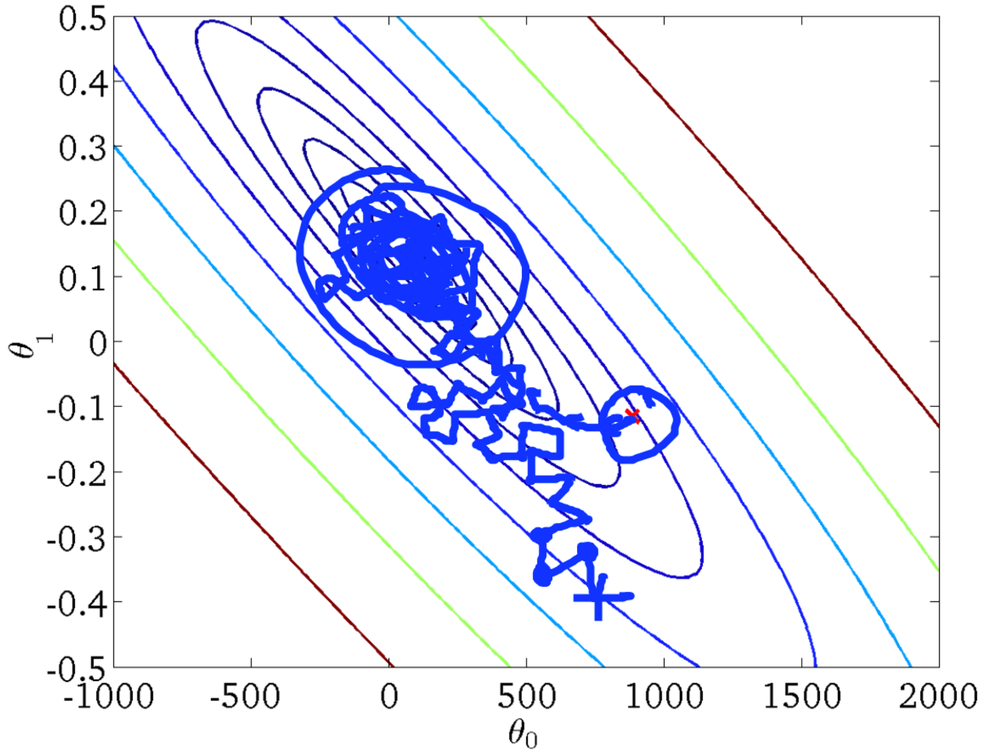
   $\theta_j := \theta_j - \alpha(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$

   (for $j = 0, \ldots, n$)

   } every

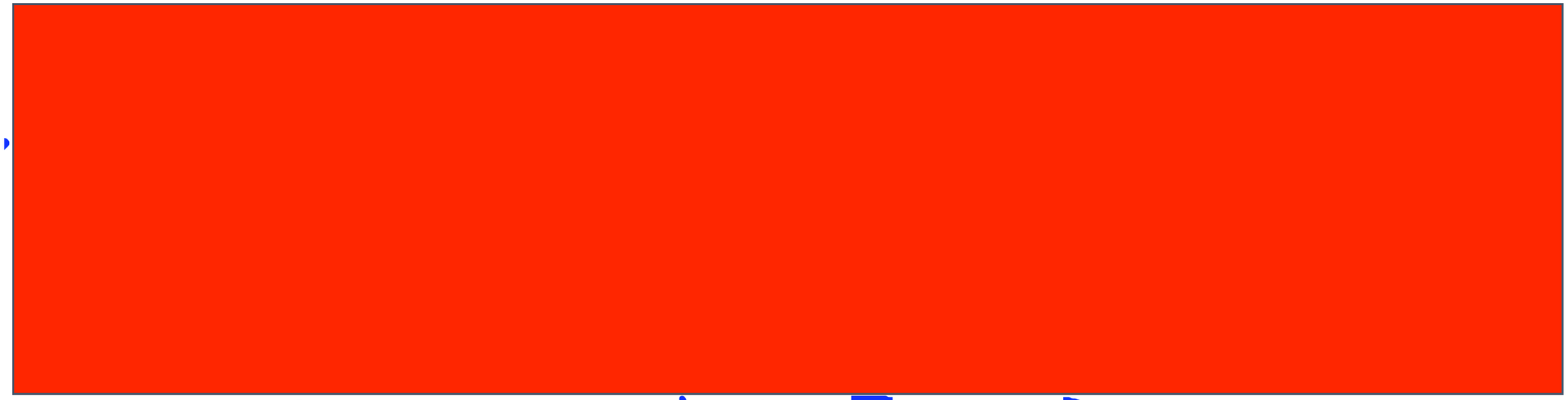
   → $m = 300,000,000$

}



Gradient Descent with Large Datasets

Mini-Batch Gradient Descent



b = mini-batch size. $b = 10$. $2 - 100$

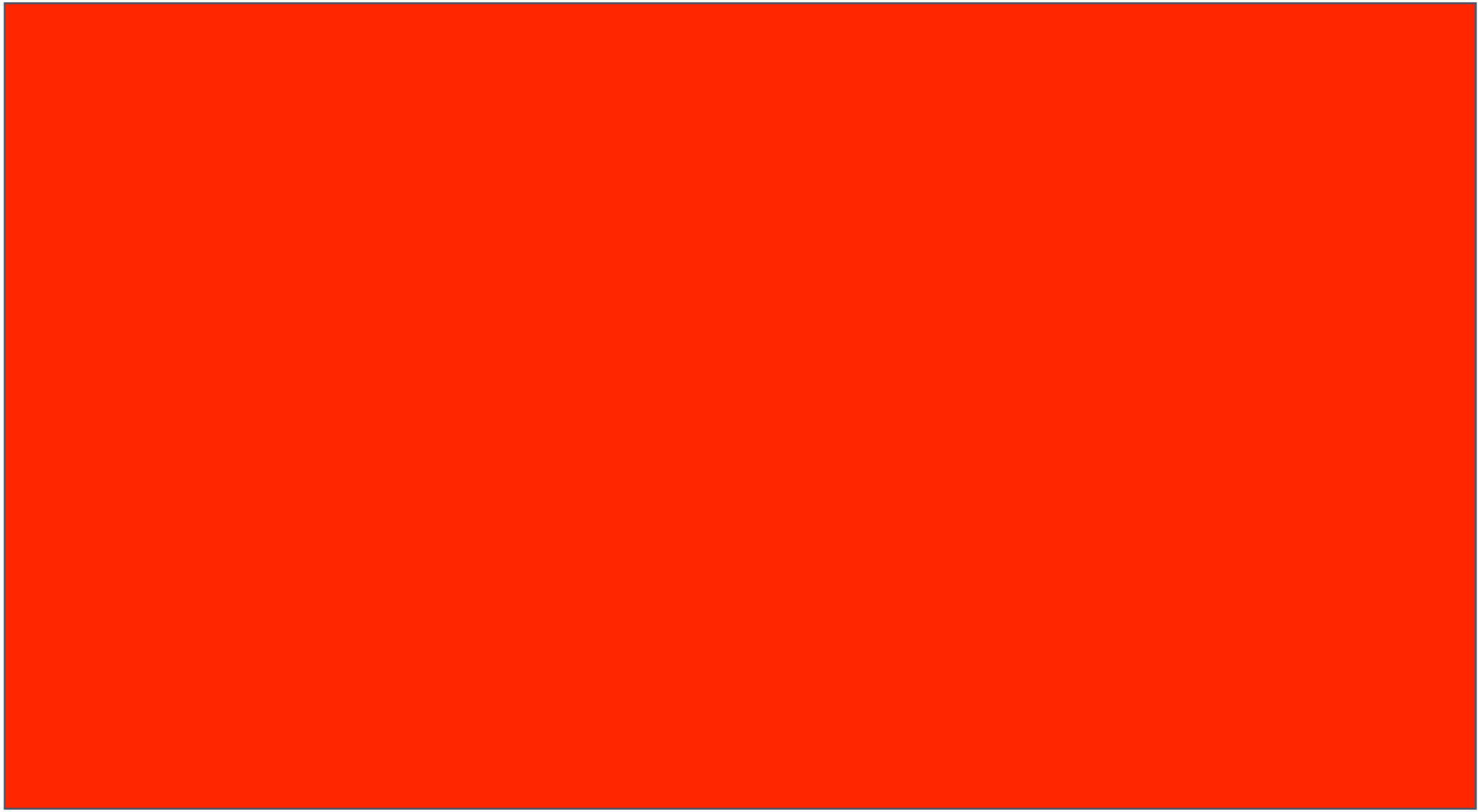
Get $b = 10$ examples $(x^{(i)}, y^{(i)}) \dots (x^{(i+9)}, y^{(i+9)})$

$$\theta_j := \theta_j - \alpha \frac{1}{10} \sum_{k=i}^{i+9} (h_\theta(x^{(k)}) - y^{(k)}) \cdot x_j^{(k)}$$

$i := i + 10$

Gradient Descent with Large Datasets

Mini-Batch Gradient Descent

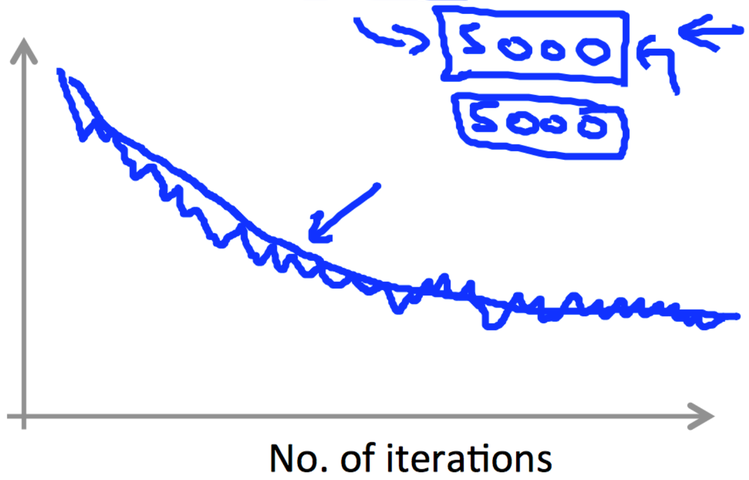


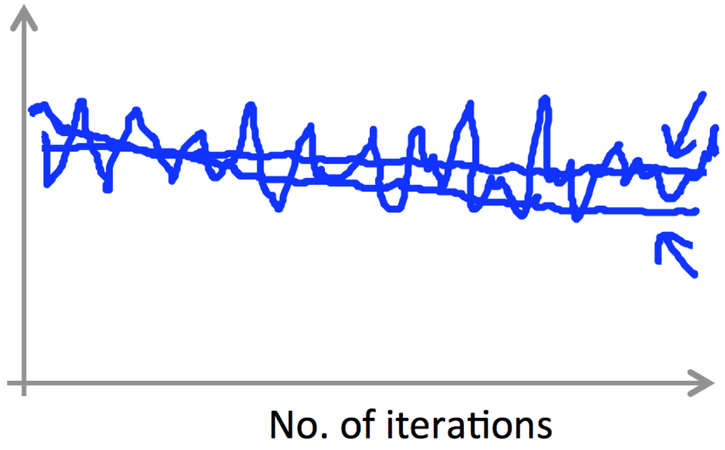
Gradient Descent with Large Datasets

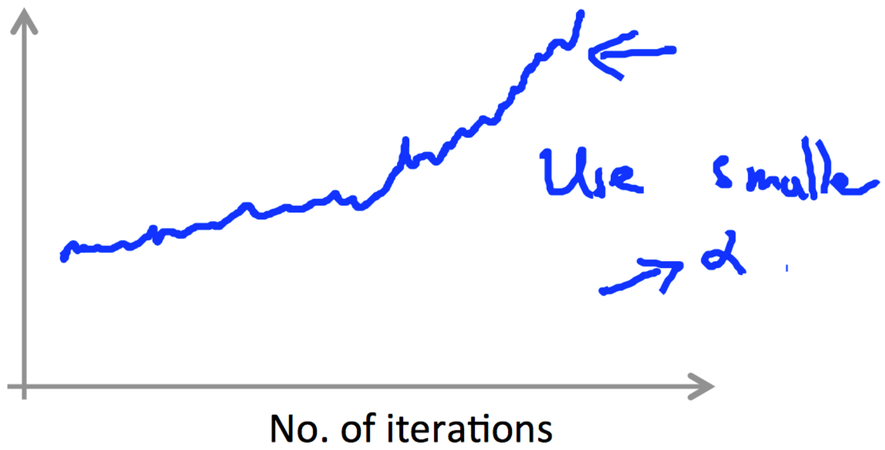
Stochastic Gradient Descent Convergence

Checking for convergence

Plot $cost(\theta, (x^{(i)}, y^{(i)}))$, averaged over the last 1000 (say) example







Gradient Descent with Large Datasets

Stochastic Gradient Descent Convergence

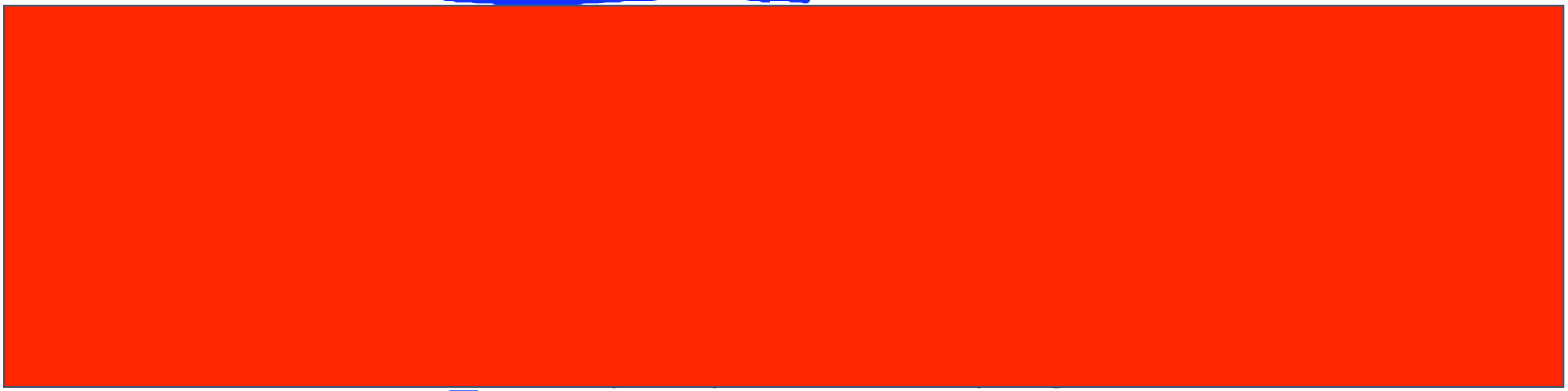
Checking for convergence

Batch gradient descent:

- Plot $J_{train}(\theta)$ as a function of the number of iterations of gradient descent.

$$J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$m = 300,000,000$



Gradient Descent with Large Datasets

Stochastic Gradient Descent Convergence

Stochastic gradient descent

$$cost(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$$J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m} cost(\theta, (x^{(i)}, y^{(i)}))$$

1. Randomly shuffle dataset.
2. Repeat {

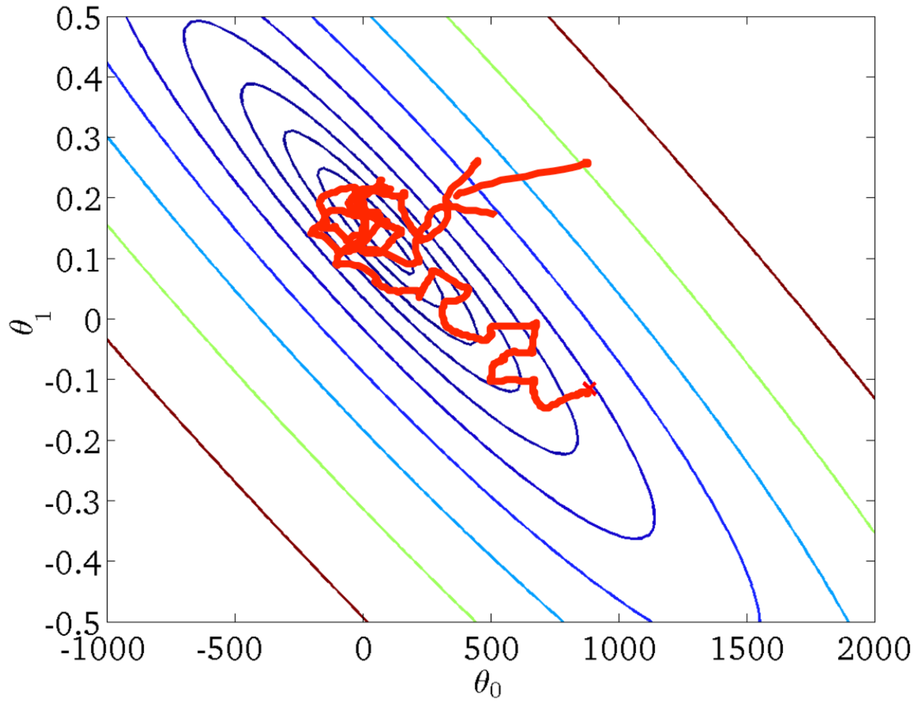
   for $i := 1, \ldots, m$ {

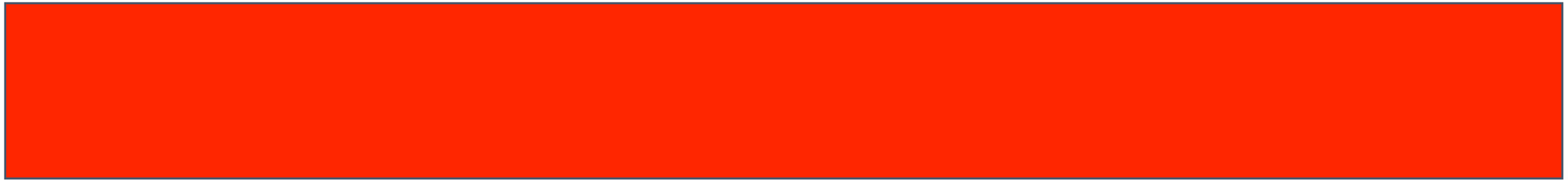
   $\theta_j := \theta_j - \alpha(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$

   (for $j = 0, \ldots, n$)

   }

}





Gradient Descent with Large Datasets

Stochastic Gradient Descent Convergence

Stochastic gradient descent

$$cost(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$$J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m} cost(\theta, (x^{(i)}, y^{(i)}))$$

1. Randomly shuffle dataset.
2. Repeat {

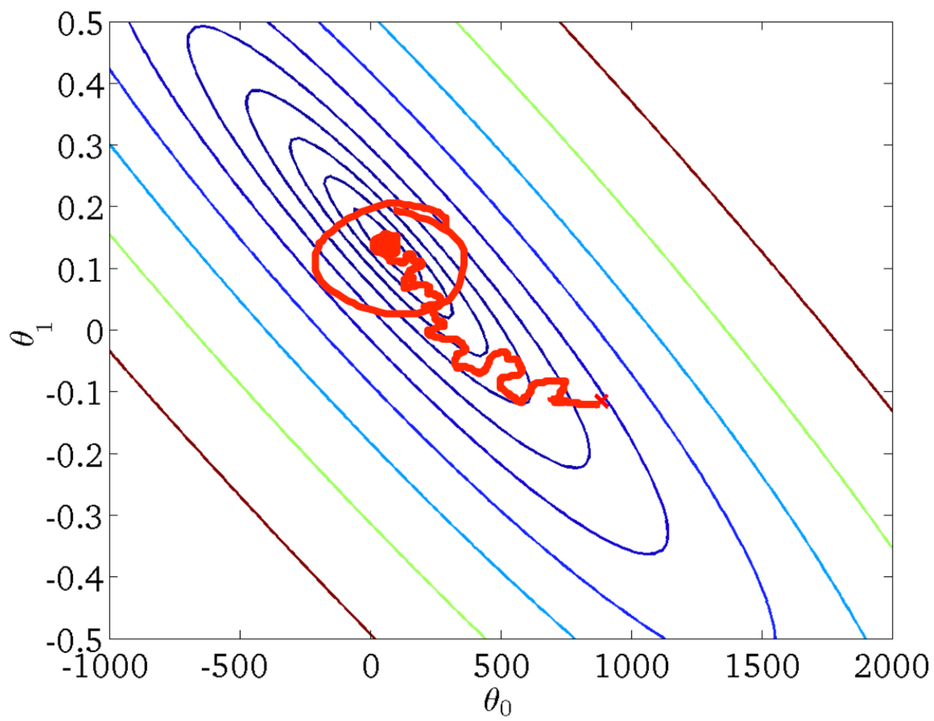
   for $i := 1, \ldots, m$ {

   $\theta_j := \theta_j - \alpha(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$

   (for $j = 0, \ldots, n$)

   }

}



Advanced Topics

Online learning

Shipping service website where user comes, specifies origin and destination, you offer to ship their package for some asking price, and users sometimes choose to use your shipping service ($y = 1$), sometimes not ($y = 0$).

Features x capture properties of user, of origin/destination and asking price. We want to learn $p(y = 1|x; \theta)$ to optimize price.

price

logistic regression

Repeat forever {

Get (x, y) corresponding to user.

Update θ using (x, y).

$\rightarrow \theta_j := \theta_j - \alpha (h_\theta(x) - y) \cdot x_j \quad (j = 0, \ldots, n)$

}

$\rightarrow$ Can adapt to changing user preference.

Advanced Topics

Other online learning example:

Product search (learning to search)

User searches for "Android phone 1080p camera"

Have 100 phones in store. Will return 10 results.

→ $x =$ features of phone, how many words in user query match name of phone, how many words in query match description of phone, etc.

→ $y = 1$ if user clicks on link. $y = 0$ otherwise.

→ Learn $p(y = 1|x; \theta)$. predicted CTR

→ Use to show user the 10 phones they're most likely to click on.

Other examples: Choosing special offers to show user; customized selection of news articles; product recommendation; …

Advanced Topics

Map-reduce

Batch gradient descent: $\theta_j := \theta_j - \alpha \frac{1}{400} \sum_{i=1}^{400} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$

$m = 400$ $\quad$ $m = 400{,}000{,}000$

Machine 1: Use $(x^{(1)}, y^{(1)}), \ldots, (x^{(100)}, y^{(100)})$.

$temp_j^{(1)} = \sum_{i=1}^{100} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$

Machine 2: Use $(x^{(101)}, y^{(101)}), \ldots, (x^{(200)}, y^{(200)})$.

$temp_j^{(2)} = \sum_{i=101}^{200} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$

Machine 3: Use $(x^{(201)}, y^{(201)}), \ldots, (x^{(300)}, y^{(300)})$.

$temp_j^{(3)} = \sum_{i=201}^{300} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$

Machine 4: Use $(x^{(301)}, y^{(301)}), \ldots, (x^{(400)}, y^{(400)})$.

$temp_j^{(4)} = \sum_{i=301}^{400} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$

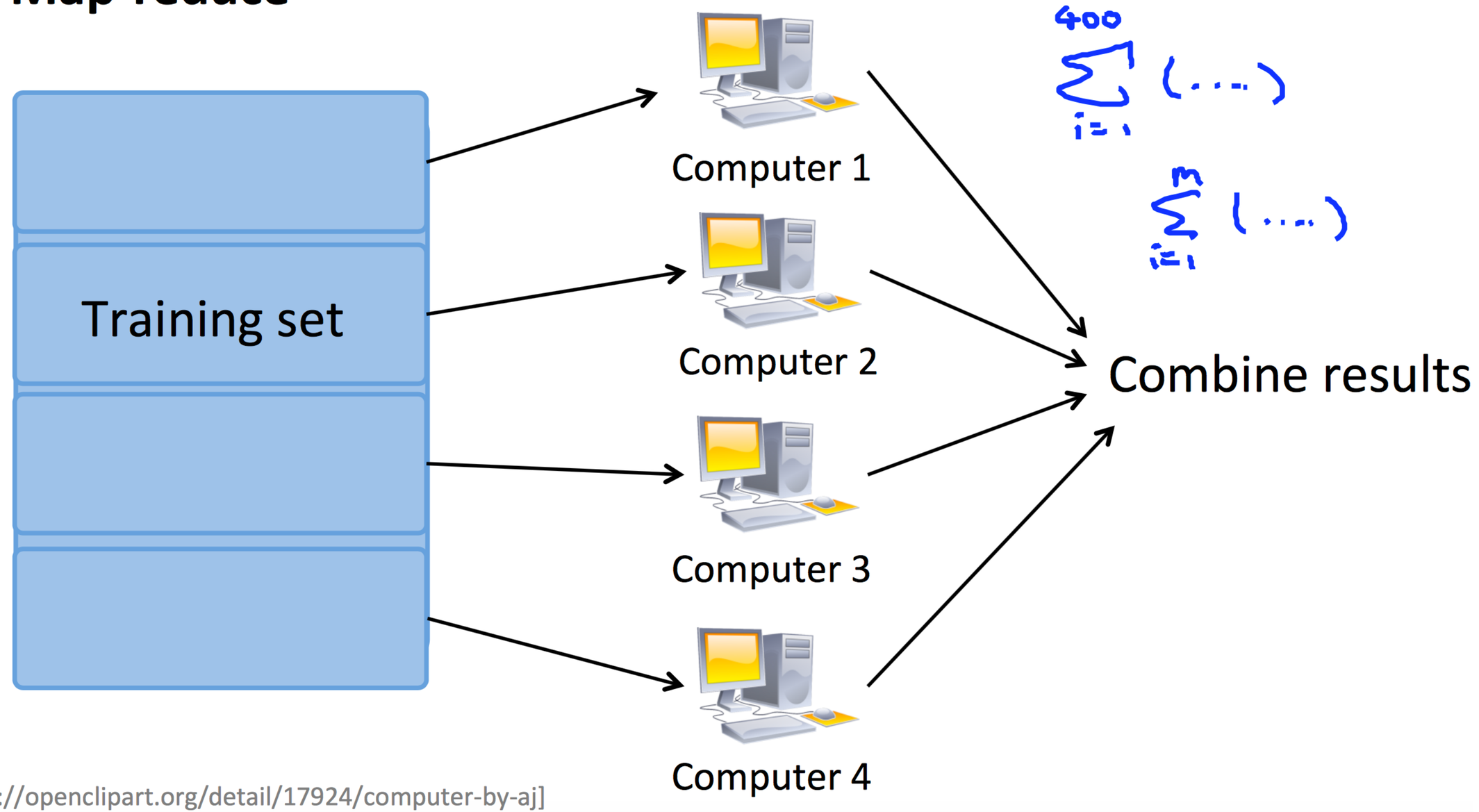
Combine:

$\theta_j := \theta_j - \alpha \frac{1}{400} (temp_j^{(1)} + temp_j^{(2)} + temp_j^{(3)} + temp_j^{(4)})$

$(j = 0, \ldots, n)$

Advanced Topics

Map-reduce



[http://openclipart.org/detail/17924/computer-by-aj]

Advanced Topics

Map-reduce and summation over the training set

Many learning algorithms can be expressed as computing sums of functions over the training set.

E.g. for advanced optimization, with logistic regression, need:

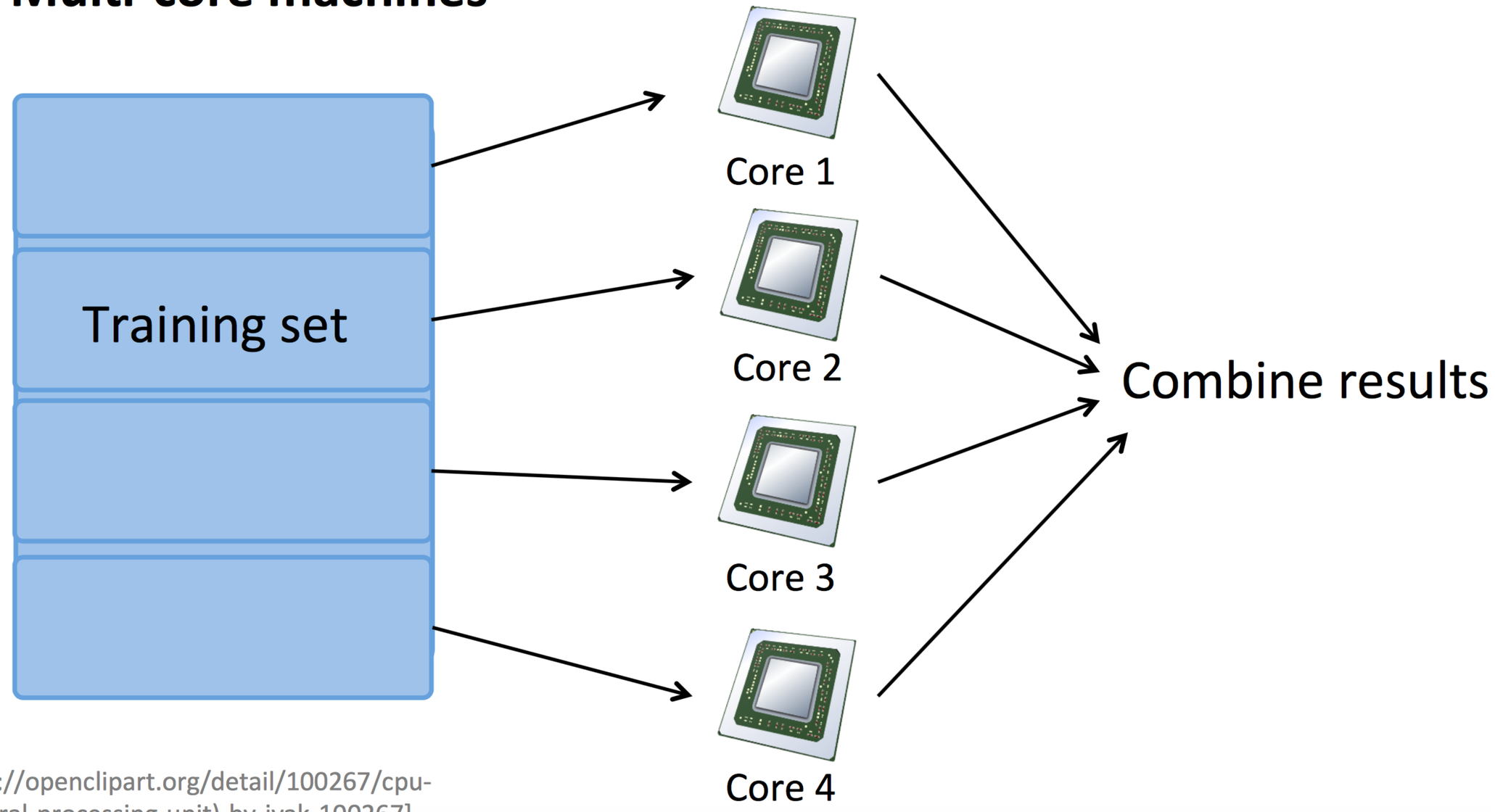
$$J_{train}(\theta) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) - (1-y^{(i)}) \log(1 - h_\theta(x^{(i)}))$$

$$\frac{\partial}{\partial \theta_j} J_{train}(\theta) = \frac{1}{m}\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$$

$temp^{(i)}$ $temp_j^{(i)} \leftarrow$

Advanced Topics

Multi-core machines



[http://openclipart.org/detail/100267/cpu-(central-processing-unit)-by-ivak-100267]

Advanced Topics

Recursive Feature Elimination

The Recursive Feature Elimination (RFE) method is a feature selection approach. It works by recursively removing attributes and building a model on those attributes that remain. It uses the model accuracy to identify which attributes (and combination of attributes) contribute the most to predicting the target attribute.

This recipe shows the use of RFE on the iris dataset to select 3 attributes.

```
# Recursive Feature Elimination
from sklearn import datasets
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
# load the iris datasets
dataset = datasets.load_iris()
# create a base classifier used to evaluate a subset of attributes
model = LogisticRegression()
# create the RFE model and select 3 attributes
rfe = RFE(model, 3)
rfe = rfe.fit(dataset.data, dataset.target)
# summarize the selection of the attributes
print(rfe.support_)
print(rfe.ranking_)
```

Feature Importance

Advanced Topics

Feature Importance

Methods that use ensembles of decision trees (like Random Forest and Extra Trees) can also compute the relative importance of each attribute. These importance values can be used to inform a feature selection process.

This recipe shows the construction of an Extra Trees ensemble of the iris dataset and the display of the relative feature importance.

```
# Feature Importance
from sklearn import datasets
from sklearn import metrics
from sklearn.ensemble import ExtraTreesClassifier
# load the iris datasets
dataset = datasets.load_iris()
# fit an Extra Trees model to the data
model = ExtraTreesClassifier()
model.fit(dataset.data, dataset.target)
# display the relative importance of each attribute
print(model.feature_importances_)
```

Cross Validation

Advanced Topics

Cross Validation

Cross validation is a way of evaluating a model on a dataset. It provides an estimation of the accuracy of the model if it were to make predictions on previously unseen data. Cross validation estimations are used to aid in the selection of a robust model that is fit for purpose.

This recipe estimates the performance of logistic regression on the iris dataset using k-fold cross validation with 10 folds.

```
# Cross Validation Classification
import numpy as np
from sklearn import datasets
from sklearn.linear_model import LogisticRegression
from sklearn import cross_validation
# load the iris datasets
dataset = datasets.load_iris()
# prepare cross validation folds
num_folds = 10
num_instances = len(dataset.data)
kfold = cross_validation.KFold(n=num_instances, n_folds=num_folds)
# prepare a Logistic Regression model
model = LogisticRegression()
# evaluate the model k-fold cross validation
results = cross_validation.cross_val_score(model, dataset.data, dataset.target, cv=kfold)
# display the mean classification accuracy on each fold
print(results)
# display the mean and stdev of the classification accuracy
print(results.mean())
print(results.std())
```

Grid Search Parameter Tuning

Advanced Topics

Grid Search Parameter Tuning

Machine learning models are parameterized so that their behavior can be tuned for a given problem. Models can have many parameters and finding the best combination of parameters can be treated as a search problem.

Grid search is an approach to parameter tuning that will methodically build and evaluate a model for each combination of algorithm parameters specified in a grid.

This recipe evaluates different alpha values for the Ridge Regression algorithm on the diabetes dataset.

```
# Grid Search for Algorithm Tuning
import numpy as np
from sklearn import datasets
from sklearn.linear_model import Ridge
from sklearn.grid_search import GridSearchCV
# load the diabetes datasets
dataset = datasets.load_diabetes()
# prepare a range of alpha values to test
alphas = np.array([1,0.1,0.01,0.001,0.0001,0])
# create and fit a ridge regression model, testing each alpha
model = Ridge()
grid = GridSearchCV(estimator=model, param_grid=dict(alpha=alphas))
grid.fit(dataset.data, dataset.target)
print(grid)
# summarize the results of the grid search
print(grid.best_score_)
print(grid.best_estimator_.alpha)
```

Advanced Topics

Random Search Parameter Tuning

Random search is an approach to parameter tuning that will sample algorithm parameters from a random distribution (i.e. uniform) for a fixed number of iterations. A model is constructed and evaluated for each combination of parameters chosen.

This recipe evaluates different alpha random values between 0 and 1 for the Ridge Regression algorithm on the diabetes dataset.

```
# Randomized Search for Algorithm Tuning
import numpy as np
from scipy.stats import uniform as sp_rand
from sklearn import datasets
from sklearn.linear_model import Ridge
from sklearn.grid_search import RandomizedSearchCV
# load the diabetes datasets
dataset = datasets.load_diabetes()
# prepare a uniform distribution to sample for the alpha parameter
param_grid = {'alpha': sp_rand()}
# create and fit a ridge regression model, testing random alpha values
model = Ridge()
rsearch = RandomizedSearchCV(estimator=model, param_distributions=param_grid, n_iter=100)
rsearch.fit(dataset.data, dataset.target)
print(rsearch)
# summarize the results of the random parameter search
print(rsearch.best_score_)
print(rsearch.best_estimator_.alpha)
```

Advanced Topics

Random Search Parameter Tuning

Random search is an approach to parameter tuning that will sample algorithm parameters from a random distribution (i.e. uniform) for a fixed number of iterations. A model is constructed and evaluated for each combination of parameters chosen.

This recipe evaluates different alpha random values between 0 and 1 for the Ridge Regression algorithm on the diabetes dataset.

```
# Randomized Search for Algorithm Tuning
import numpy as np
from scipy.stats import uniform as sp_rand
from sklearn import datasets
from sklearn.linear_model import Ridge
from sklearn.grid_search import RandomizedSearchCV
# load the diabetes datasets
dataset = datasets.load_diabetes()
# prepare a uniform distribution to sample for the alpha parameter
param_grid = {'alpha': sp_rand()}
# create and fit a ridge regression model, testing random alpha values
model = Ridge()
rsearch = RandomizedSearchCV(estimator=model, param_distributions=param_grid, n_iter=100)
rsearch.fit(dataset.data, dataset.target)
print(rsearch)
# summarize the results of the random parameter search
print(rsearch.best_score_)
print(rsearch.best_estimator_.alpha)
```

Advanced Topics

Computer Science > Learning

API design for machine learning software: experiences from the scikit-learn project

Lars Buitinck (ILPS), Gilles Louppe, Mathieu Blondel, Fabian Pedregosa (INRIA Saclay - Ile de France), Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort (INRIA Saclay - Ile de France, LTCI), Jaques Grobler (INRIA Saclay - Ile de France), Robert Layton, Jake Vanderplas, Arnaud Joly, Brian Holt, Gaël Varoquaux (INRIA Saclay - Ile de France)

(Submitted on 1 Sep 2013)

Scikit-learn is an increasingly popular machine learning li- brary. Written in Python, it is designed to be simple and efficient, accessible to non-experts, and reusable in various contexts. In this paper, we present and discuss our design choices for the application programming interface (API) of the project. In particular, we describe the simple and elegant interface shared by all learning and processing units in the library and then discuss its advantages in terms of composition and reusability. The paper also comments on implementation details specific to the Python ecosystem and analyzes obstacles faced by users and developers of the library.

Subjects: **Learning (cs.LG)**; Mathematical Software (cs.MS)
Journal reference: European Conference on Machine Learning and Principles and Practices of Knowledge Discovery in Databases (2013)
Cite as: **arXiv:1309.0238 [cs.LG]**
(or **arXiv:1309.0238v1 [cs.LG]** for this version)

Submission history

From: Gael Varoquaux [view email]
[v1] Sun, 1 Sep 2013 16:22:48 GMT (28kb)

https://arxiv.org/abs/1309.0238

Deep Learning

Upload your happiest moments in FY18 Asia GBB Ready - #AsiaICGBB

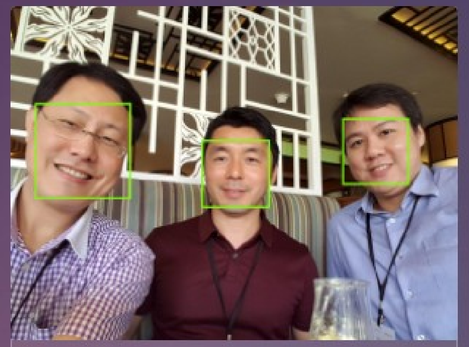
Dont' have ? Attach the photos and wely.lau@microsoft.com with subject containing #AsiaICGBB

Top 20 the happiest photos Top 20 the unhappiest photos

RANK 1

Christina Peh (Adecco) (OneDrive)

0.99999765 3



RANK 5

TakayukiHoshino (Twitter)

0.99994497 3

RANK 9

Christina Peh (Adecco) (OneDrive)

0.99815755 3

RANK 13

Margaret Synan (OneDrive)

0.97417196 8

RANK 17

Genevieve Ang (Adecco) (OneDrive)

0.89848932 10

RANK 2

Genevieve Ang (Adecco) (OneDrive)

RANK 6

Genevieve Ang (Adecco) (OneDrive)

RANK 10

Genevieve Ang (Adecco) (OneDrive)

RANK 14

Yusuf Rangwala (OneDrive)

0.94711960 7

RANK 18

Danette Seward (OneDrive)

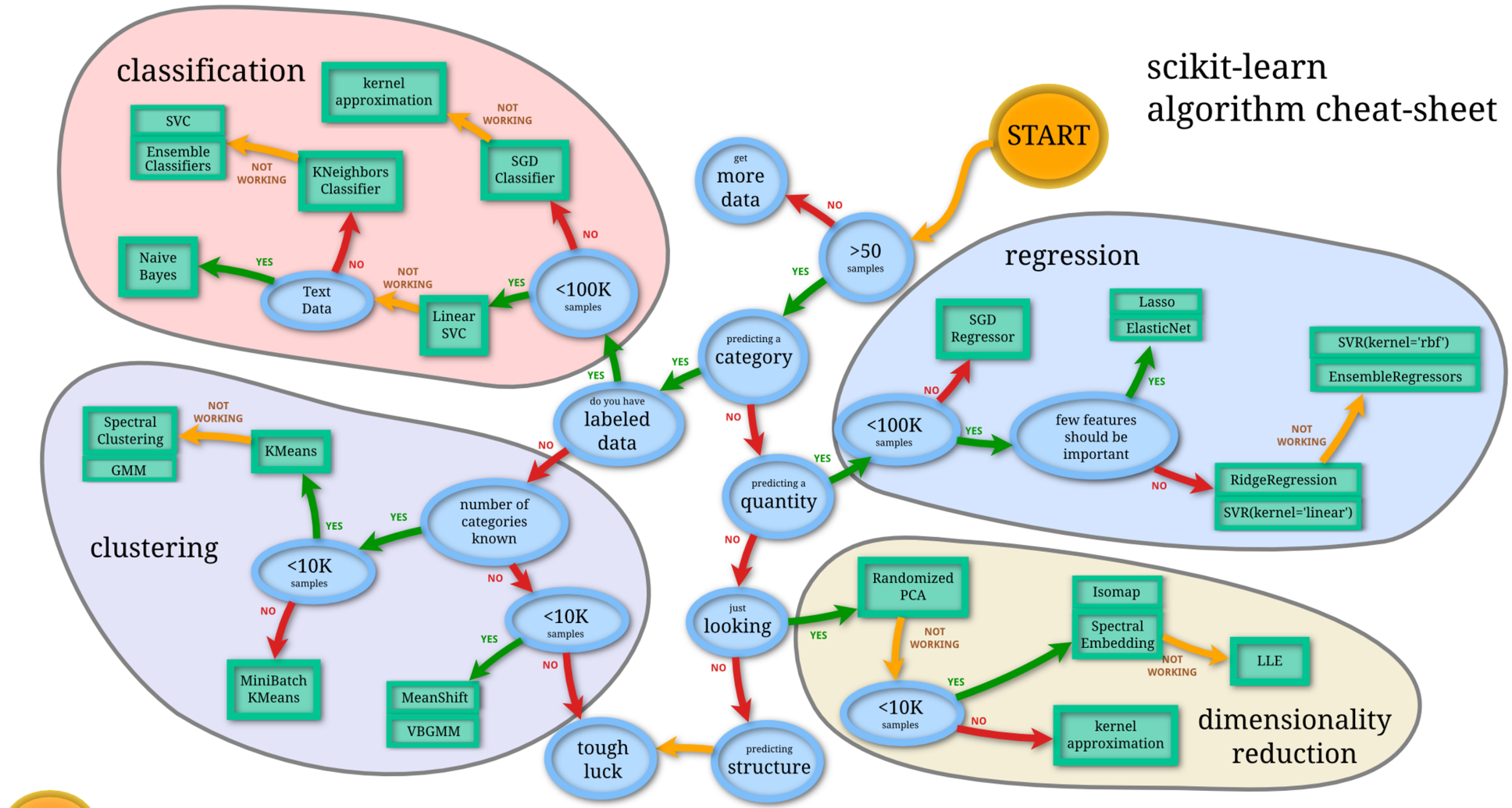
OPEN-SOURCE ML TOOLS

What is scikit-learn?

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. This is stack that includes:

- **NumPy**: Base n-dimensional array package
- **SciPy**: Fundamental library for scientific computing
- **Matplotlib**: Comprehensive 2D/3D plotting
- **IPython**: Enhanced interactive console
- **Sympy**: Symbolic mathematics
- **Pandas**: Data structures and analysis

OPEN-SOURCE ML TOOLS



OPEN-SOURCE ML TOOLS

Regression

Models	• Linear regression • Regularization: Ridge (L2), Lasso (L1)
Algorithms	• Gradient descent • Coordinate descent
Concepts	• Loss functions, bias-variance tradeoff, cross-validation, sparsity, overfitting, model selection

scikits learn

machine learning in Python

Source:Machine Learning Specialization , Emily Fox & Carlos Guestrin

OPEN-SOURCE ML TOOLS

Classification

Models	• Linear classifiers (logistic regression, SVMs, perceptron) • Kernels • Decision trees
Algorithms	• Stochastic gradient descent • Boosting
Concepts	• Decision boundaries, MLE, ensemble methods, random forests, CART, online learning

machine learning in Python

Source:Machine Learning Specialization , Emily Fox & Carlos Guestrin

OPEN-SOURCE ML TOOLS

Clustering and Retrieval

Models

- Nearest neighbors
- Clustering, mixtures of Gaussians
- Latent Dirichlet allocation (LDA)

Algorithms

- KD-trees, locality-sensitive hashing (LSH)
- K-means
- Expectation-maximization (EM)

Concepts

- Distance metrics, approximation algorithms, hashing, sampling algorithms, scaling up with map-reduce

scikits learn
machine learning in Python

Source:Machine Learning Specialization , Emily Fox & Carlos Guestrin

GETTING STARTED WITH SCIKIT-LEARN

ANACONDA®

ACCELERATE

Time-to-Value

INNOVATE faster through managed agile experimentation

MOVE from analysis to deployment immediately

DELIVER high performance analytics processing

CONNECT

Data, Analytics & Compute

LEVERAGE innovative open source analytics to extract value from data

MAXIMIZE your computational power to easily analyze all your data

CONNECT and integrate all your data sources for predictive models

EMPOWER

Data Science Teams

ITERATE quickly to create powerful analysis and predictive models

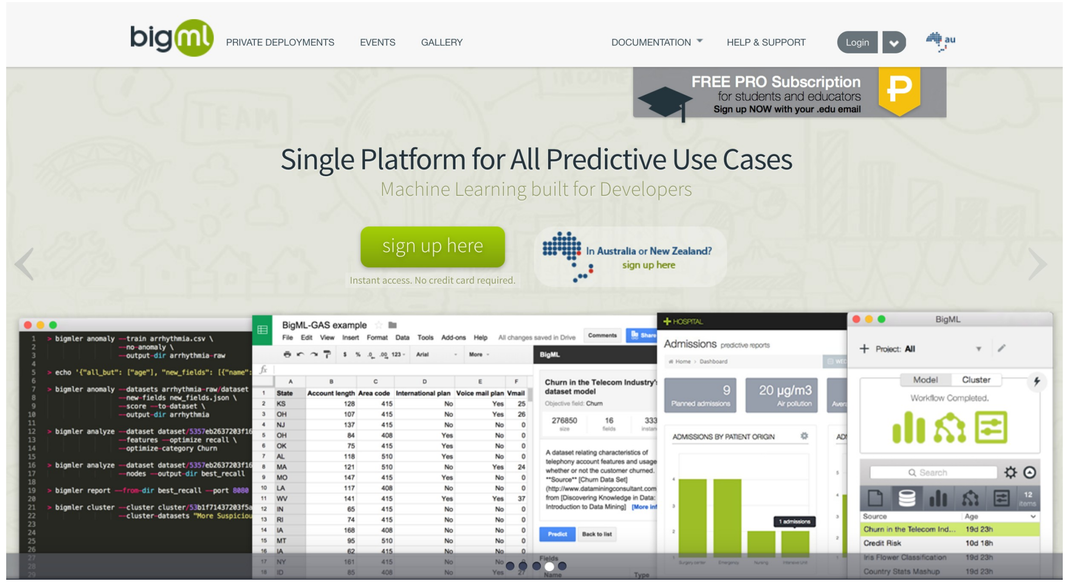
COLLABORATE and share with your data science team

PUBLISH interactive results to the business

source: https://www.continuum.io

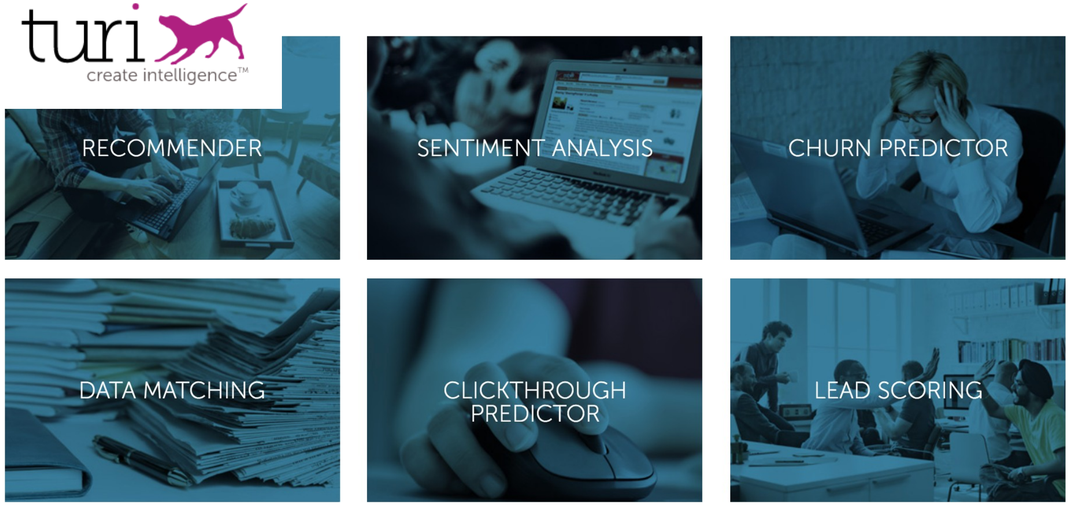
CLOUD-BASED ML TOOLS

AWS ML



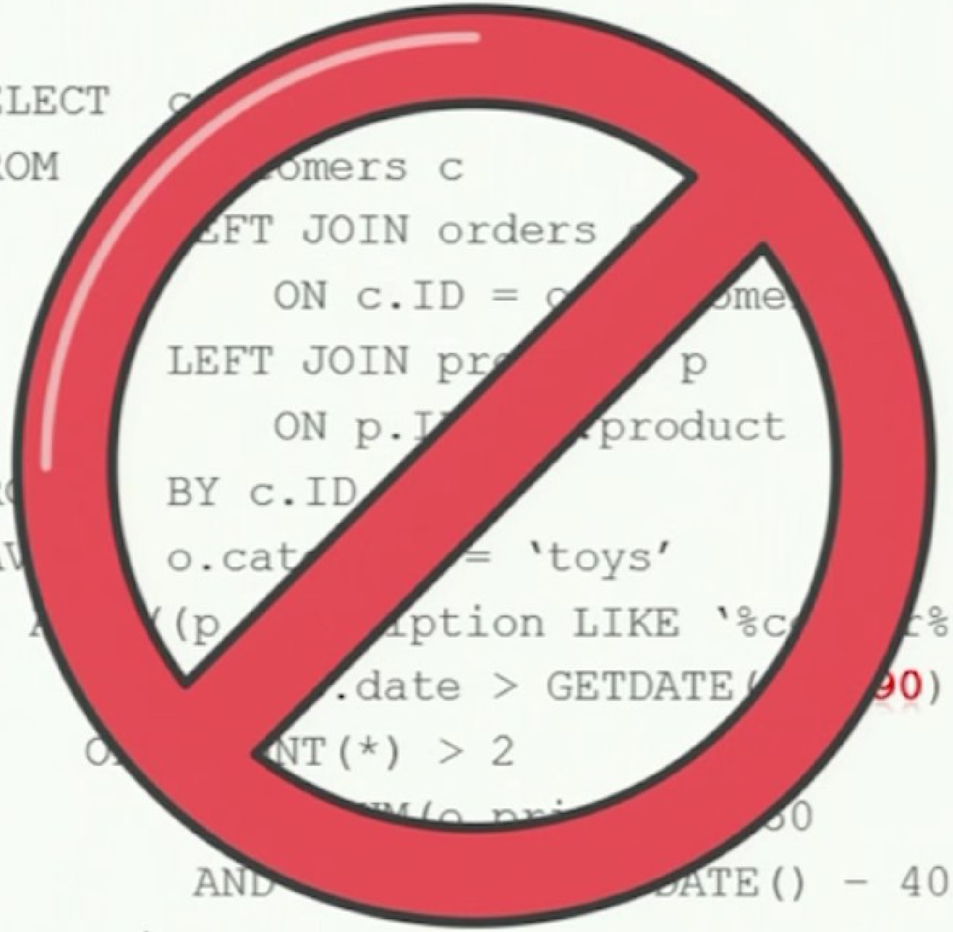
BIG ML

MS Azure ML



TURI ML

Smart applications by *counterexample*



```
SELECT  c
FROM        omers c
        LEFT JOIN orders
            ON c.ID = c    ome
        LEFT JOIN pr       p
            ON p.I       product
GR      BY c.ID
HAV     o.cat      = 'toys'
    A   (p      iption LIKE '%c      t%'
            .date > GETDATE(      90)
    O       NT(*) > 2
                M(o.pri          0
        AND          DATE() - 40)
    )
```

Use machine learning technology to **learn** your business rules from data!

CLOUD-BASED ML TOOLS

Three types of data-driven development

Retrospective
analysis and
reporting

Amazon Redshift,
Amazon RDS
Amazon S3
Amazon EMR

Here-and-now
real-time processing
and dashboards

Amazon Kinesis
Amazon EC2
AWS Lambda

Predictions
to enable smart
applications

And a few more examples…

Fraud detection	Detecting fraudulent transactions, filtering spam emails, flagging suspicious reviews,…
Personalization	Recommending content, predictive content loading, improving user experience,…
Targeted marketing	Matching customers and offers, choosing marketing campaigns, cross-selling and up-selling,…
Content classification	Categorizing documents, matching hiring managers and resumes,…
Churn prediction	Finding customers who are likely to stop using the service, upgrade targeting,…
Customer support	Predictive routing of customer emails, social media listening,…

CLOUD-BASED ML TOOLS

Three supported types of predictions

Binary classification

Predict the answer to a Yes/No question

Multiclass classification

Predict the correct category from a list

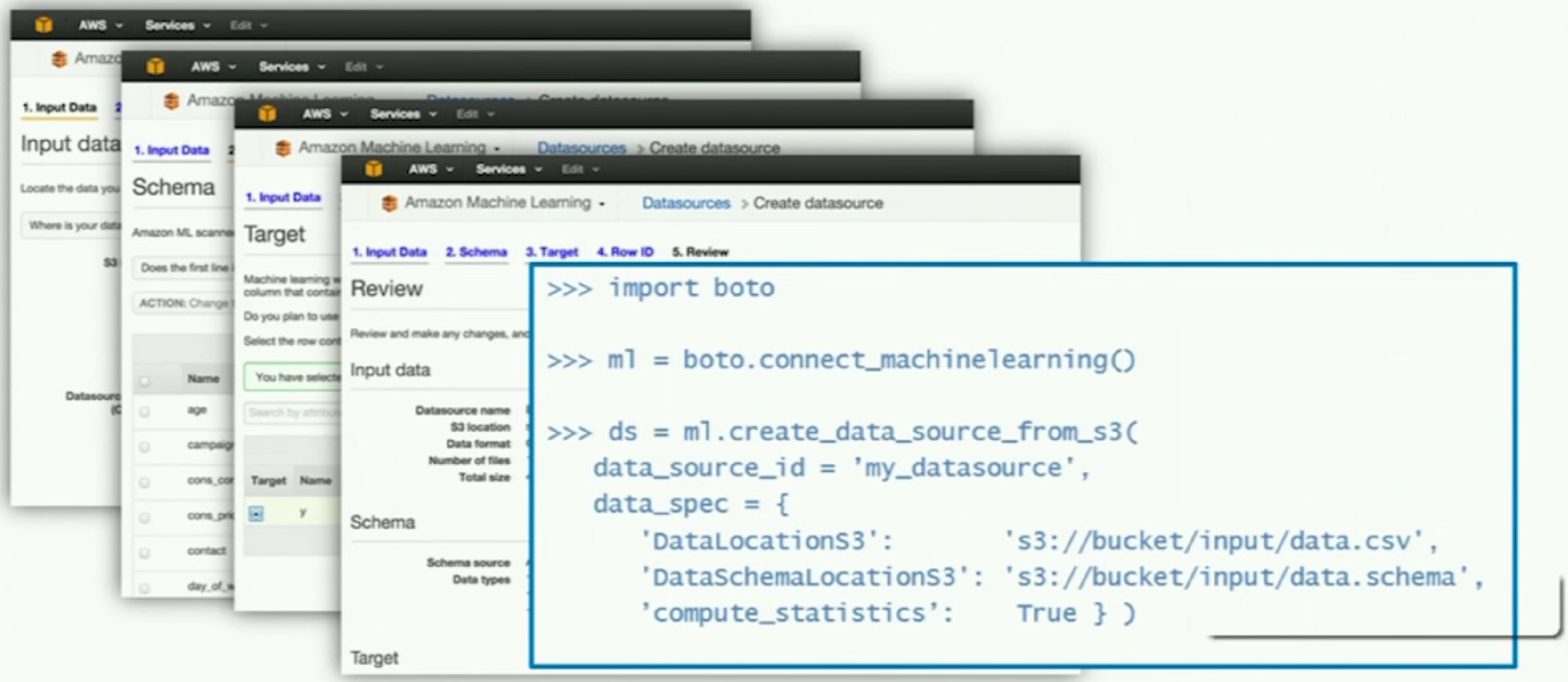
Regression

Predict the value of a numeric variable

Building smart applications with Amazon ML

1. Train model
2. Evaluate and optimize
3. Retrieve predictions

Create a datasource object



CLOUD-BASED ML TOOLS

Machine

gement,

Explore and understand your data

AWS Services Edit

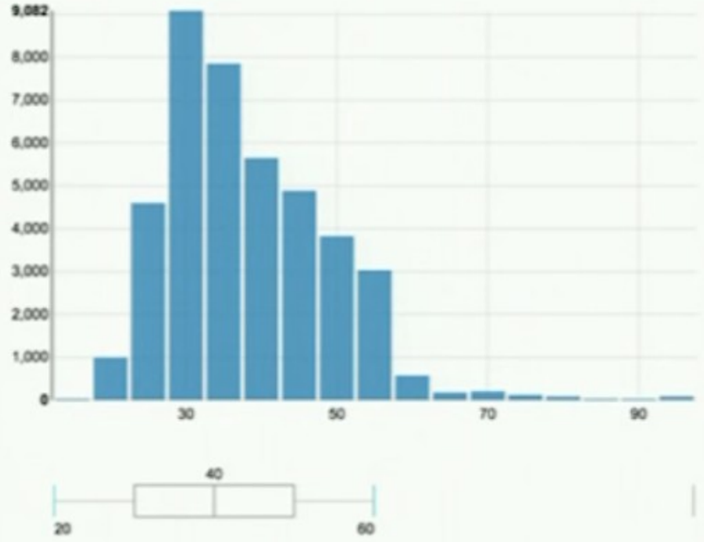
Target distributions: y

Categorical attributes

Categorical attributes: job

Numeric attributes: age

Select bin width: 10 5 2 1 0.5



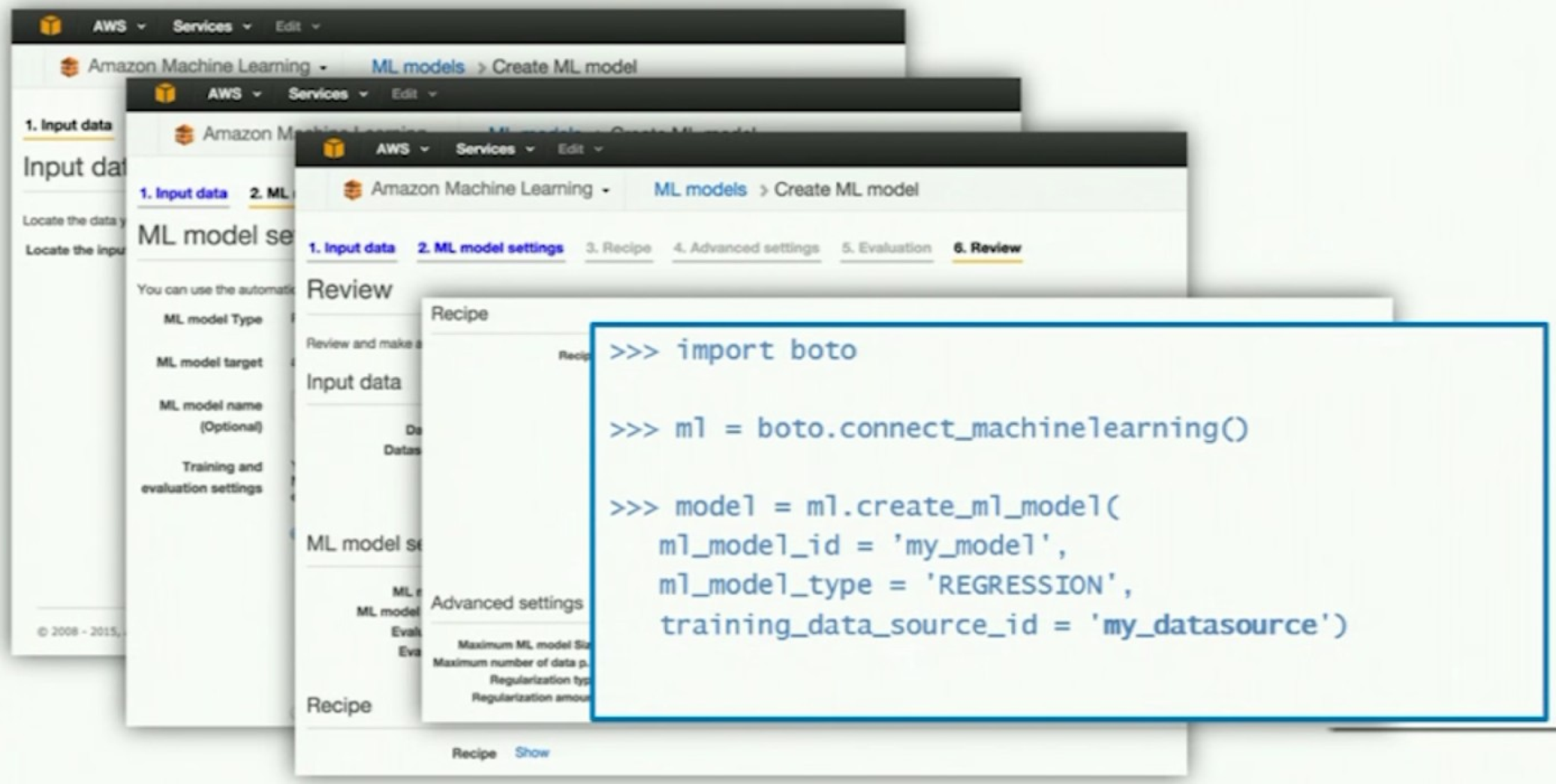
Min: 17.00 Mean: 40.02 Max: 98.00

job	using	loan	contact	month	frequent	Preview
ent		no	telephone	may		
ent		no	telephone	may		
ent		no	telephone	may		
ent		no	telephone	may		
ent		no	telephone	may		

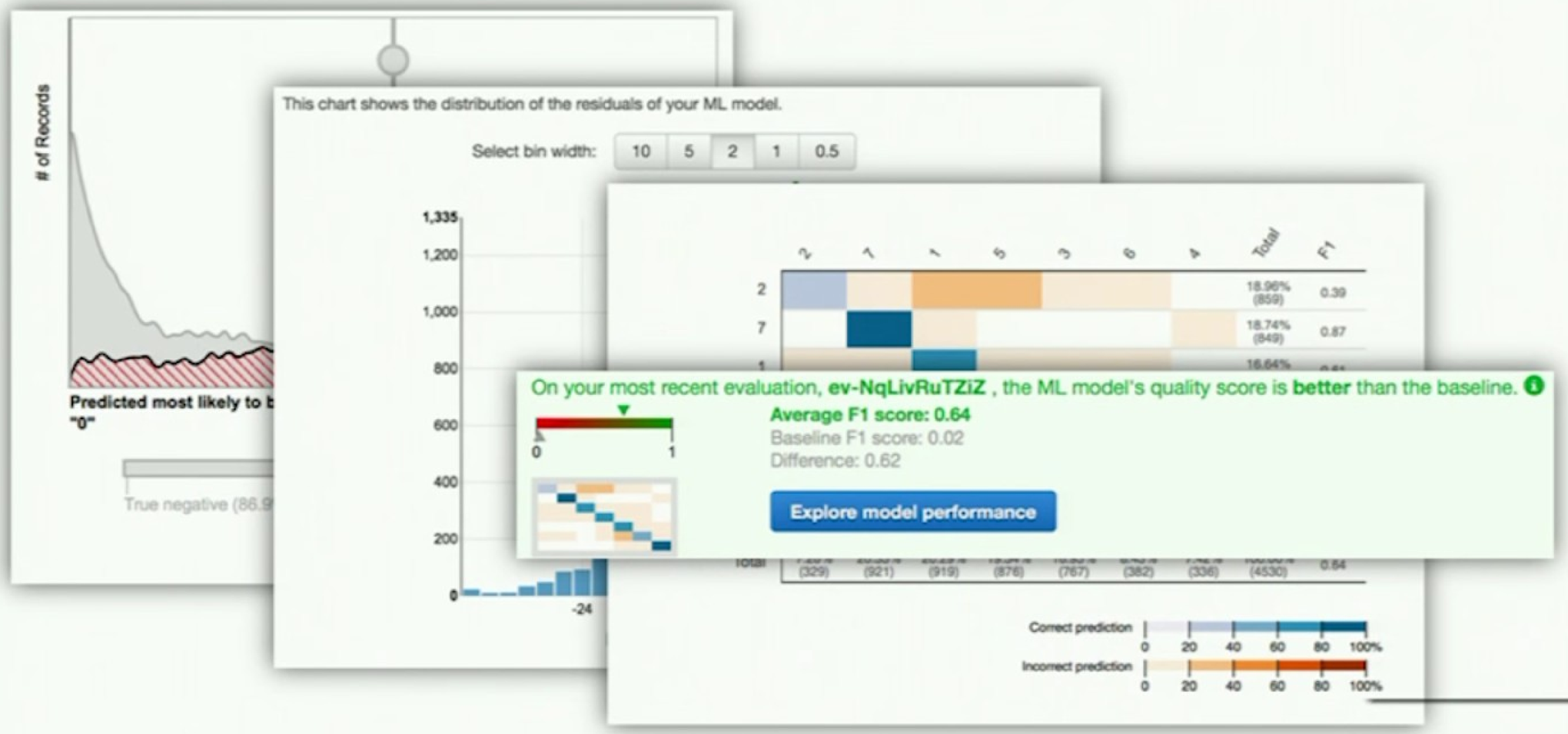
Close

CLOUD-BASED ML TOOLS

Train your model



Explore model quality



CLOUD-BASED ML TOOLS

Accuracy

Accuracy (ACC) measures the fraction of correct predictions. The range is 0 to 1. A larger value indicates better predictive accuracy:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

Precision

Precision measures the fraction of actual positives among those examples that are predicted as positive. The range is 0 to 1. A larger value indicates better predictive accuracy:

$$Precision = \frac{TP}{TP + FP}$$

Recall

Recall measures the fraction of actual positives that are predicted as positive. The range is 0 to 1. A larger value indicates better predictive accuracy:

$$Recall = \frac{TP}{TP + FN}$$

False Positive Rate

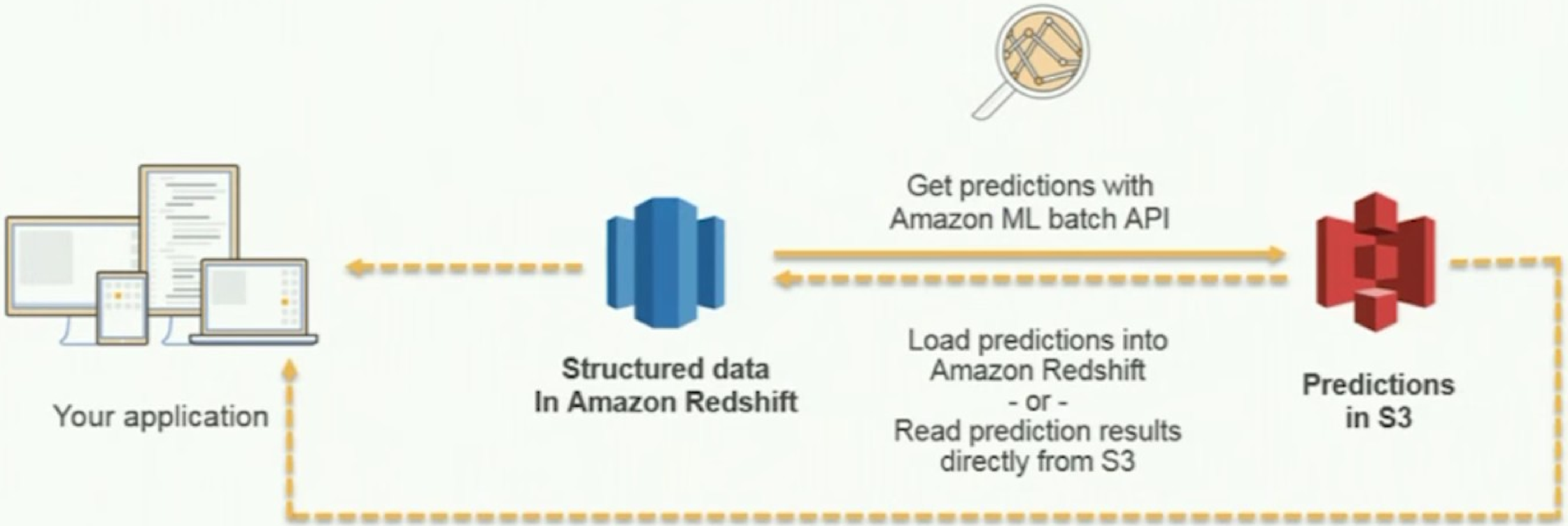
The *false positive rate* (FPR) measures the false alarm rate or the fraction of actual negatives that are predicted as positive. The range is 0 to 1. A smaller value indicates better predictive accuracy:

$$FPR = \frac{FP}{FP + TN}$$

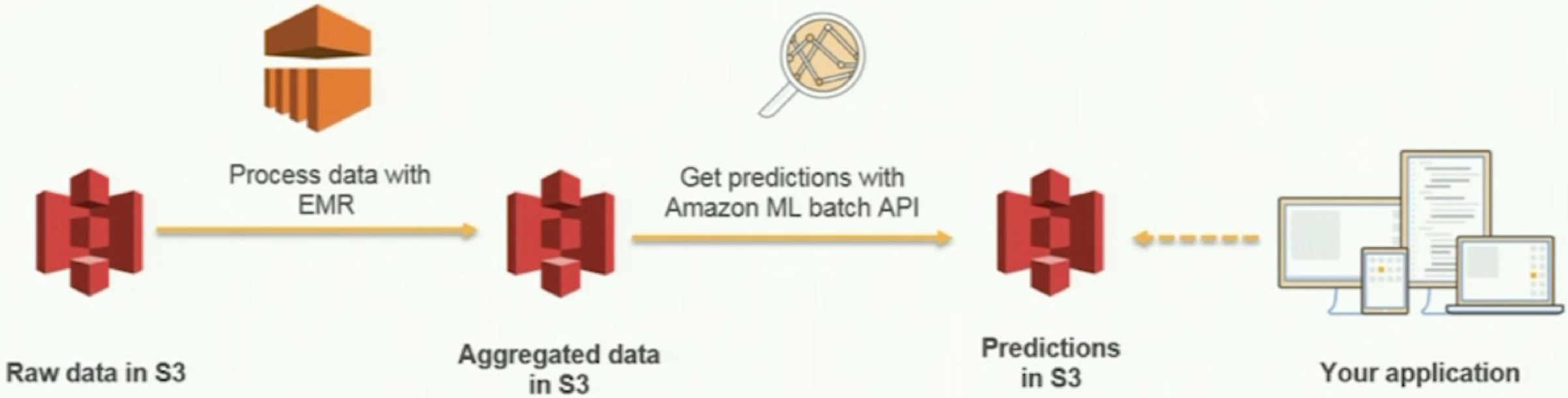
		Prediction	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

Ref: http://docs.aws.amazon.com/machine-learning/latest/dg/binary-model-insights.html?icmpid=docs_machinelearning_console

Batch predictions with Amazon Redshift



Batch predictions with EMR

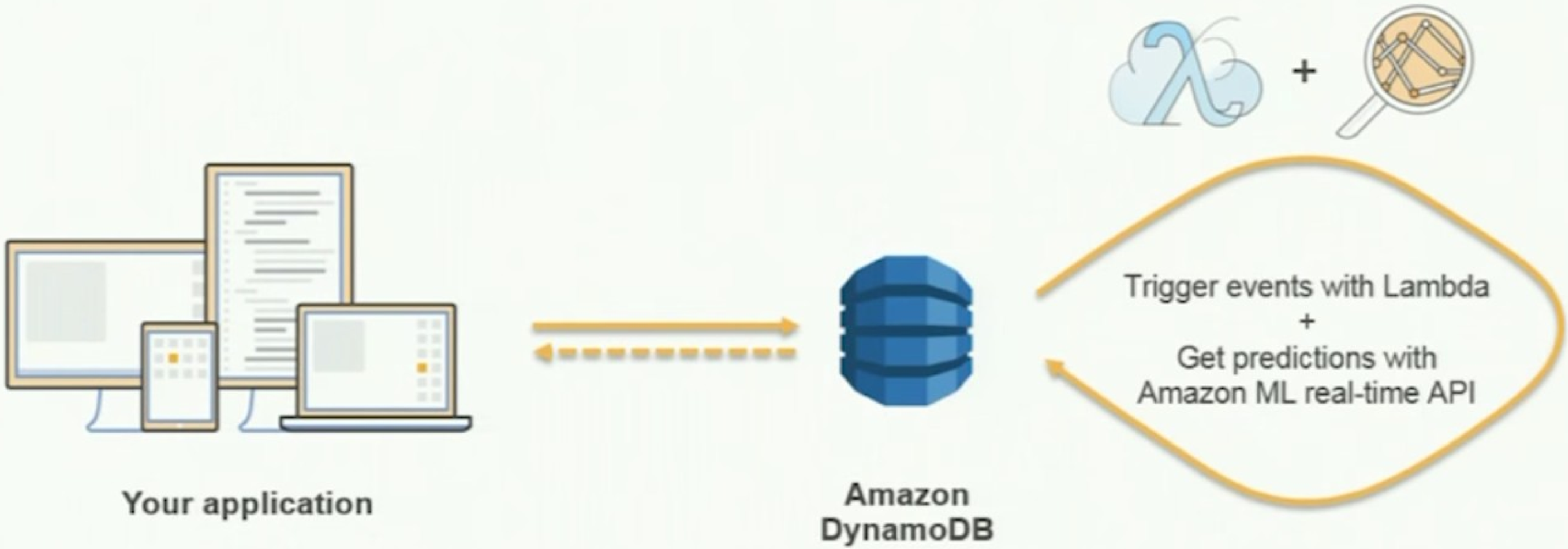


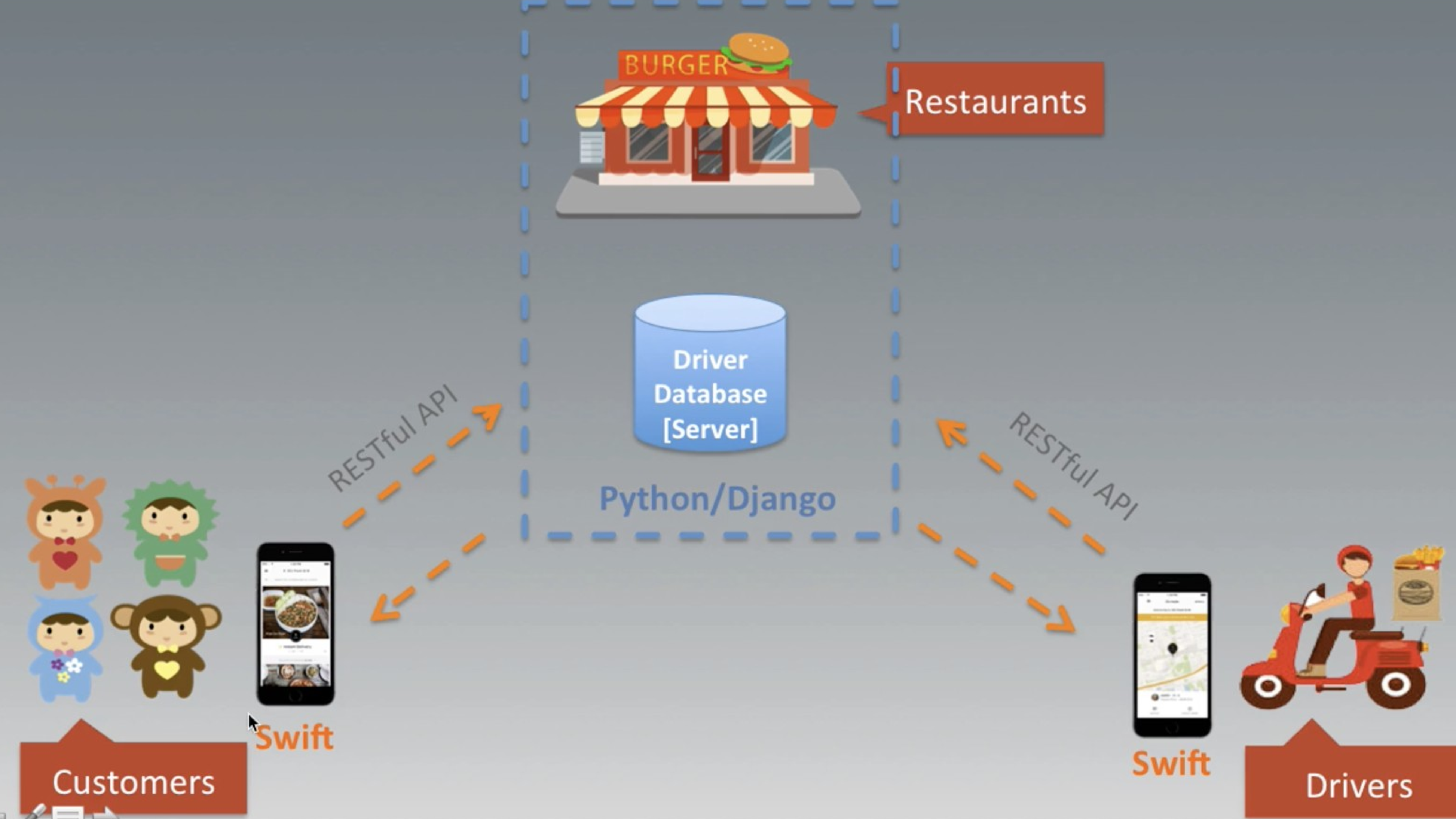
Real-time predictions for interactive applications

Your application

Amazon ML service

Adding predictions to an existing data flow




BURGER
Restaurants
Driver Database [Server]
Python/Django
RESTful API
RESTful API
Swift
Swift
Customers
Drivers

SO, WHAT IF I WANT TO LEARN HOW TO BUILD DATA

Code4Startup

One Month

lynda.com

udemy

treehouse™

UDACITY

coursera

Meetup

Code School
a Pluralsight company

codecademy

CHALLENGE YOURSELF WITH REAL-WORLD ML

kaggle Search kaggle Competitions Datasets Kernels Discussion Jobs Sign Up Log In

Welcome to Kaggle Competitions

Challenge yourself with real-world machine learning problems

New to Data Science?
Get started with a tutorial on our most popular competition for beginners, Titanic: Machine Learning from Disaster.

Build a Model
Get the data & use whatever tools or methods you prefer to make predictions.

Make a Submission
Upload your prediction file for real-time scoring & a spot on the leaderboard.