NLP and Dimensionality Reduction

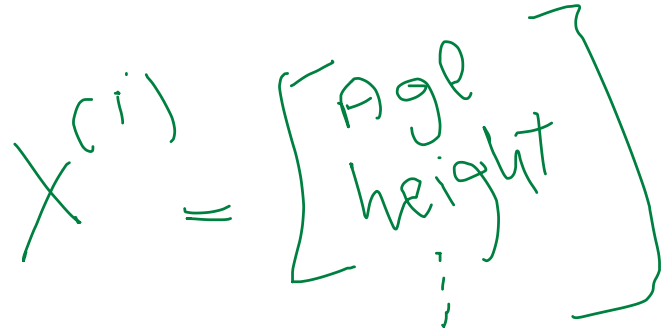
Puya Vahabi

2018-August

Reminder

- Results of the survey on syllabus and course complexity.
- Survey on what to do as last lesson.
- How is it going with the project?
- Each action is graded, but its voluntary.

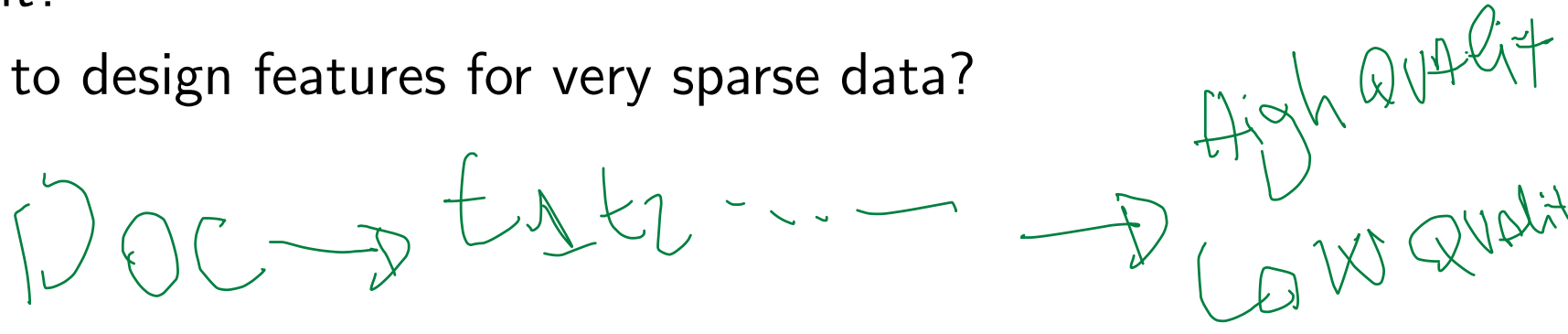
Sparse Data



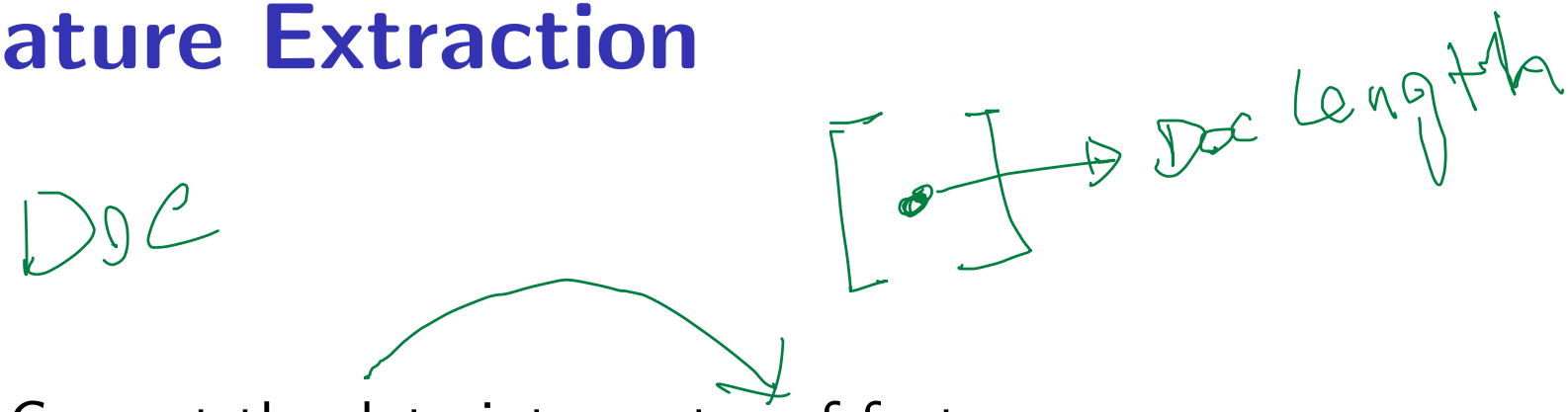
Feature engineering: it is the process of feature types later used for classification purpose.

- Text Data. How would you extract features/attributes to be used in learning phase?#words? sequence of words? #phrases?
- Image Data. Pixels of a certain color? Objects inside of it?

How to design features for very sparse data?



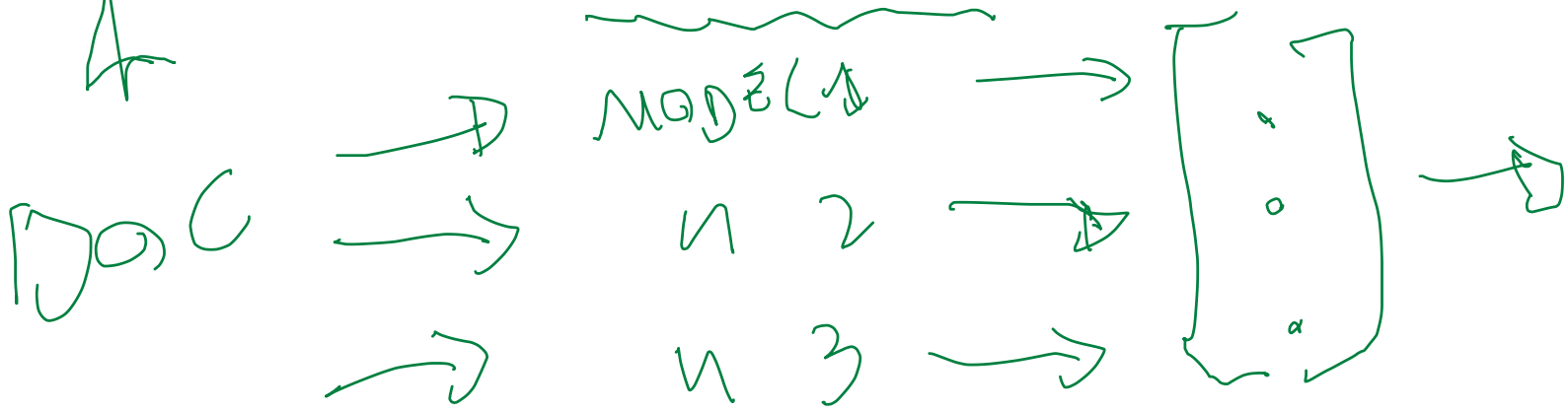
Feature Extraction



Convert the data into vector of features.

Consider *efficiency* of extracting features:

- For each instance on training and test you need to extract them.
- Complex features. This is what you have in complex problems such as Uber self driving car.



Feature Selection

Select a set of features that are important from the initial set of features.

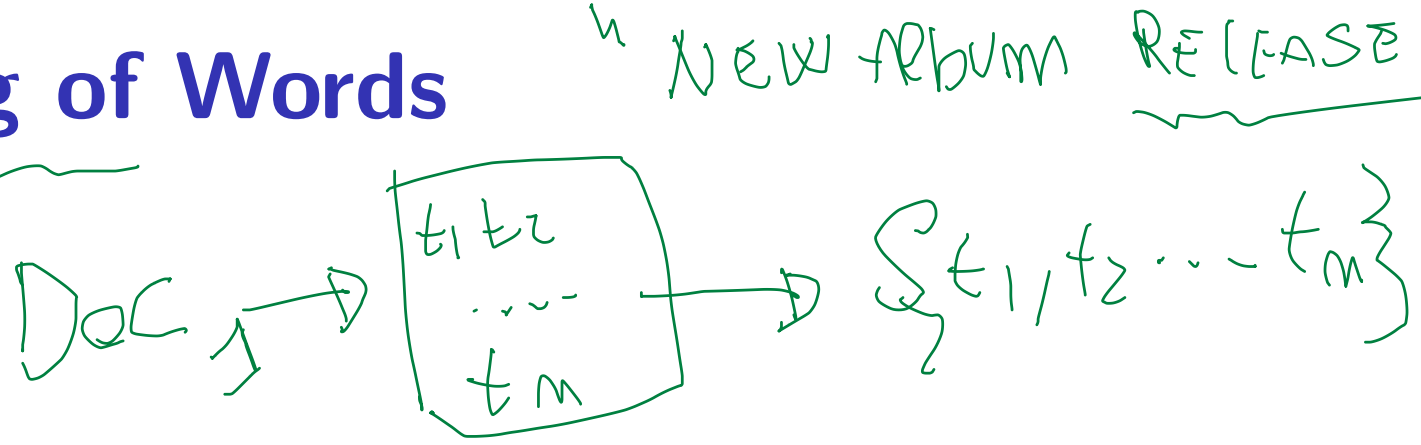
- Can make the learning process faster.
- Can reduce the complexity of the learned model.

Dimensionality reduction is often used to have a reduced feature space.

Feature Extraction

Text Data

Bag of Words

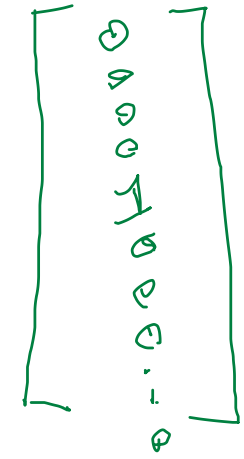


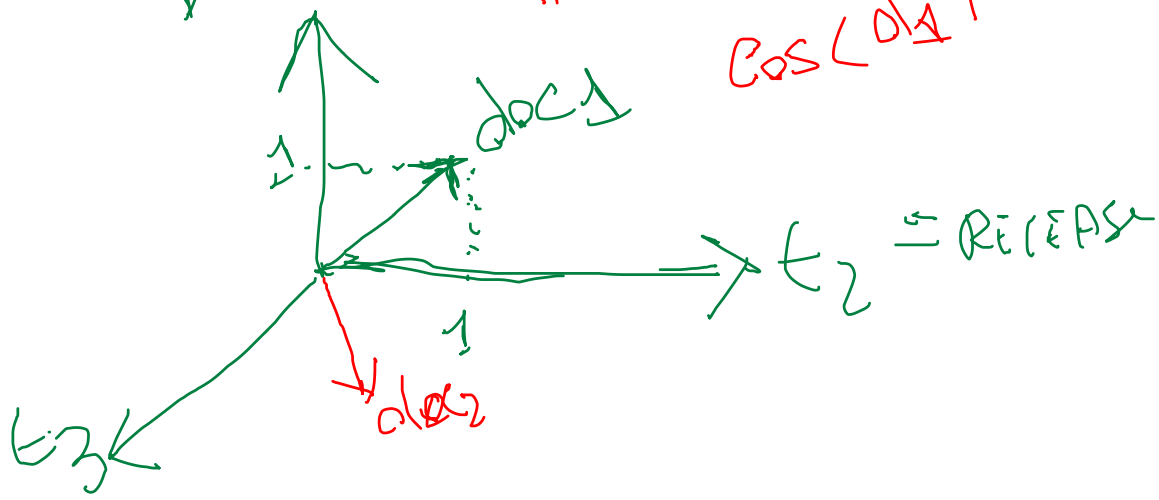
The most common feature encoding for text is called a **bag of words** representation

- Words are represented as sets, their order or position is not important.
  - If "Michael" is at the 2nd word in the document is treated the same as is in any other position in the doc.

Definitions

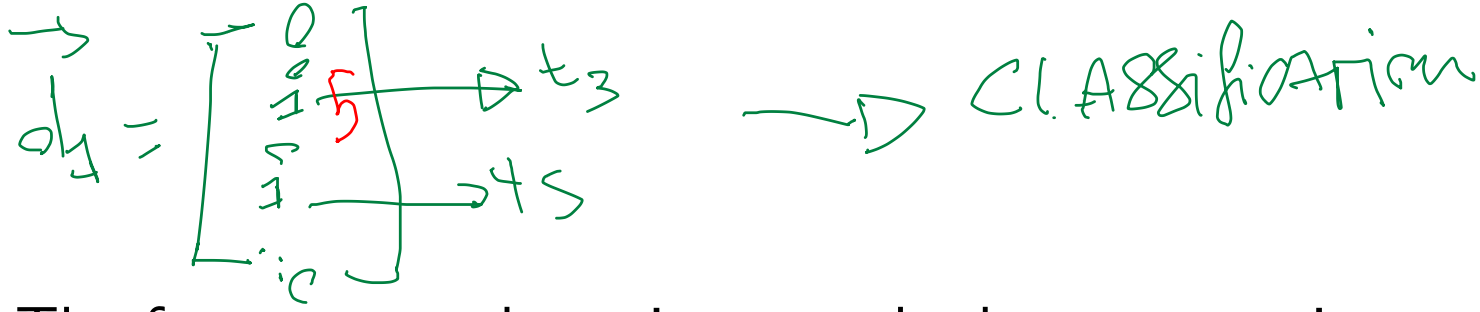
- 1-Hot-Vector
- Term Vector Space
- Document as vector
- Bag of Words





Notes

BOW as features

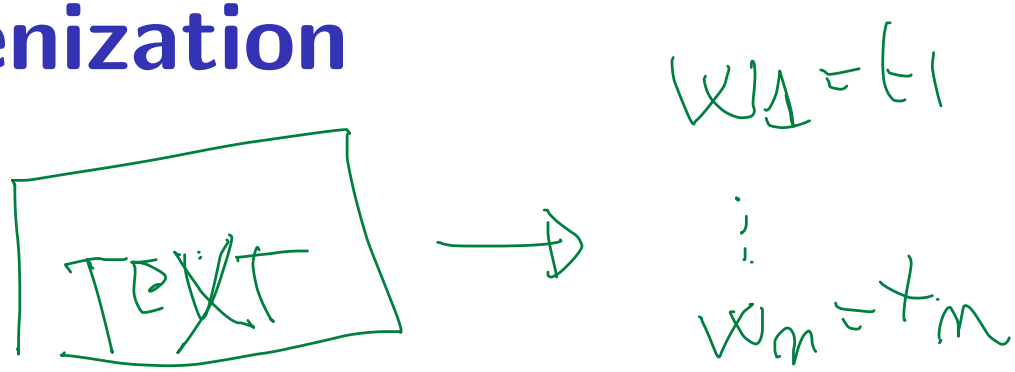


BOW. The features are the unique words that appears in vocabulary.

Can they have a weight?

- #times the word is appearing in a particular document.
- 0 means the word is not in the document
- You can also use binary values.

Tokenization



Tokenization. Split a string into a set of strings or words.

- Space based? (For Chinese language would not work)
- Vocabulary matching?

Whats after you find the atomic unit of language you want to work on?

Text Pre-Processing

- Lower case everything. Problems: "they went to the White House" vs "they lived in a white house".
- Handling Abbreviations. "the dr. lives in a blue box"
  - I.O.U.
  - M.D.
  - N.B.A.
  - U.S.A.
  - mr.
  - .sh
  - .java

Text Pre-Processing

- Remove punctuations.
  - Treat “blue,” differently from “blue”?
  - “?”, “!” these puntuactions can be very useful.
  - Emojis :) :(?
  - Apostrophes
  - x
- Handling hyphenated words. Should I consider one space separated word or two?
  - “self-assessment”, “Los Angeles-based”
  - “the New York-based co-operative was fine- tuning forty-two K-9-like models”

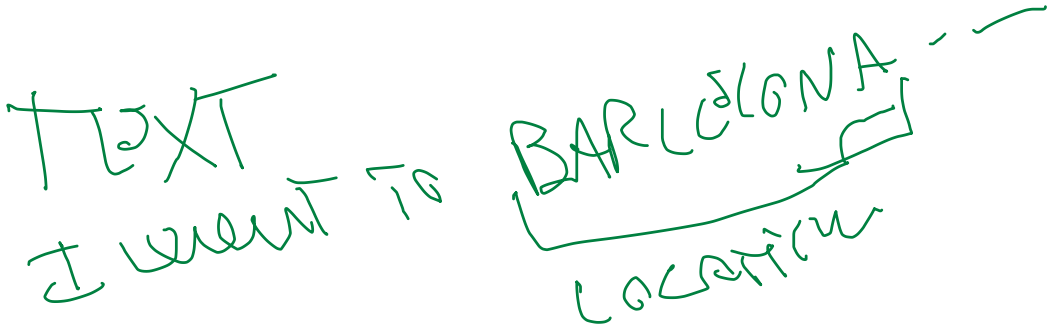
Text Pre-Processing

- Numerical and special expressions
  - Email addresses
  - URLs
  - Complex enumeration of items
  - Telephone Numbers
    - $123 - 456 - 7890$
    - $(123) - 456 - 7890$
  - Dates
    - July 10th, 2020
    - 10/07/2020
  - Time
  - Measures
  - Vehicle Licence Numbers
  - Paper and book citations

The same thing

Text Pre-Processing

- Entity Extractions
  - Person
  - Date
  - Location
  - Organization
- Few tokenizers tools
  - Apache Open NLP
  - Stanford Parser
- English Enclitics
  - He's $\longrightarrow$ He has (highly present)
  - You're $\longrightarrow$ You are



Text Pre-Processing

Stemming is a technique to convert a word to its "root" or "base" form

- This forces different forms of a word to be treated as the same feature

Word	Stem
fish	fish
fishes	fish
fished	fish
fishing	fish

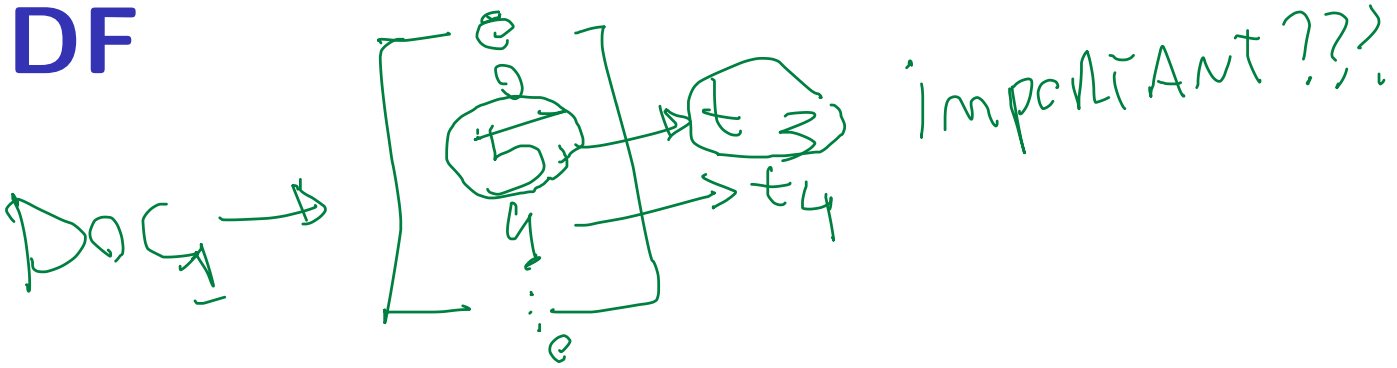
Text Pre-Processing

Stemming limitations.

- We are loosing info on the meaning of the sentence
- Stemming algorithms are making mistakes
  - university, universe, universal

Other things such as stop words removal.

TF-IDF



Common words like "the" and "and" have low information, and needs to be weighted.

$$\mathrm{tf-idf}(t,d) = tf(t,d) \times \mathrm{idf}(t,d)$$

$$\mathrm{idf}(t,d) = \log \frac{n_d}{1 + \mathrm{df}(d,t)}$$

where n_d is the total number of documents, and $\mathrm{df}(d,t)$ is the number of documents that contain the term t.

N-Grams

An extension of the the bag of words features:

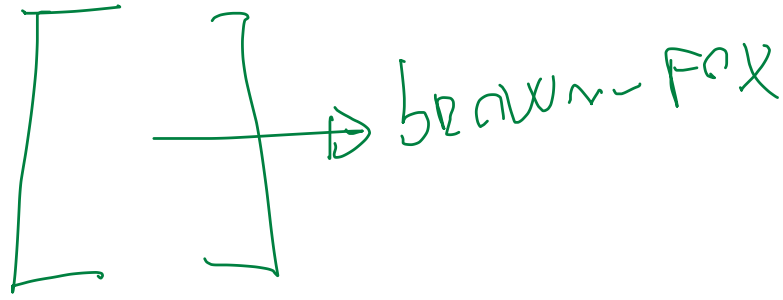
- n words in sequence considered as a single token
- 3-grams. All the 3-words in a document

The BOW is 1-gram.

"the quick brown fox jumped over the lazy dog" **1-grams (unigrams):**

```
the
quick
brown
fox
over
lazy
dog
jumped
```

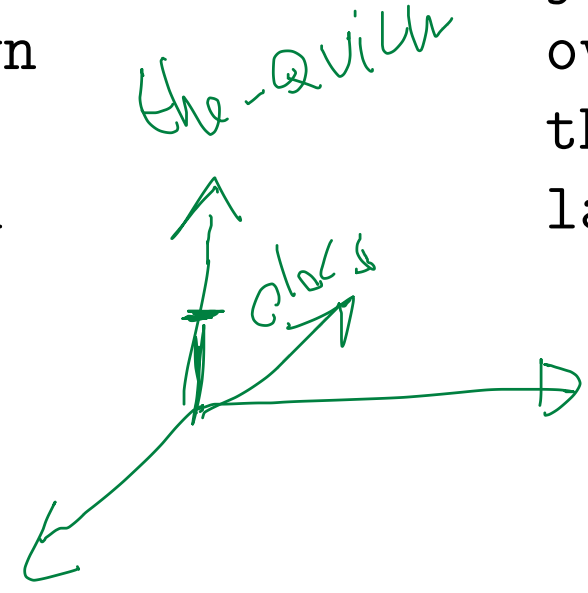
2-Grams



"the quick brown fox jumped over the lazy dog"

2-grams (bigrams):

`the_quick`	`jumped_over`
`quick_brown`	`over_the`
`brown_fox`	`the_lazy`
`fox_jumped`	`lazy_dog`



3-Grams

"the quick brown fox jumped over the lazy dog"

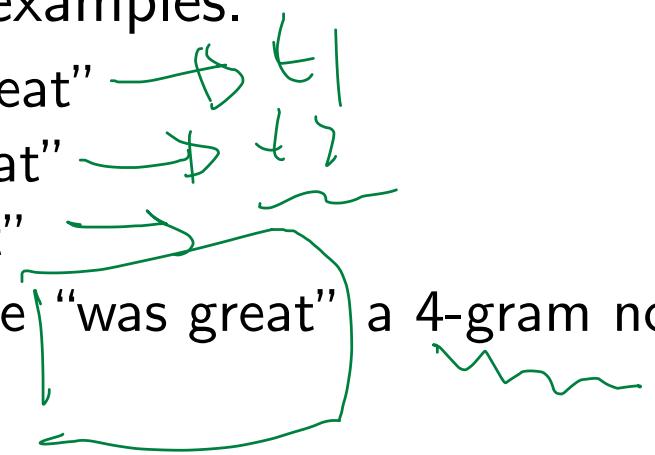
3-grams (trigrams):

```
the_quick_brown
quick_brown_fox
brown_fox_jumped
fox_jumped_over
jumped_over_the
over_the_lazy
the_lazy_dog
```

N-Grams

There is a trade-off.

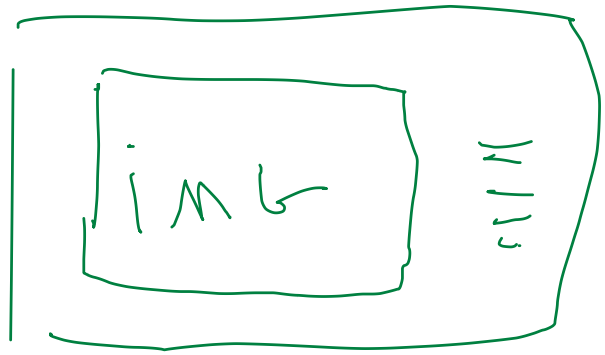
- Too short. We are missing important sequential detail "great" vs "NOT great"
- Too long. Not enough examples.
  - "the sausage was great"
  - "the cheese was great"
  - "the crust was great"
  - a bigram can capture "was great" a 4-gram no!



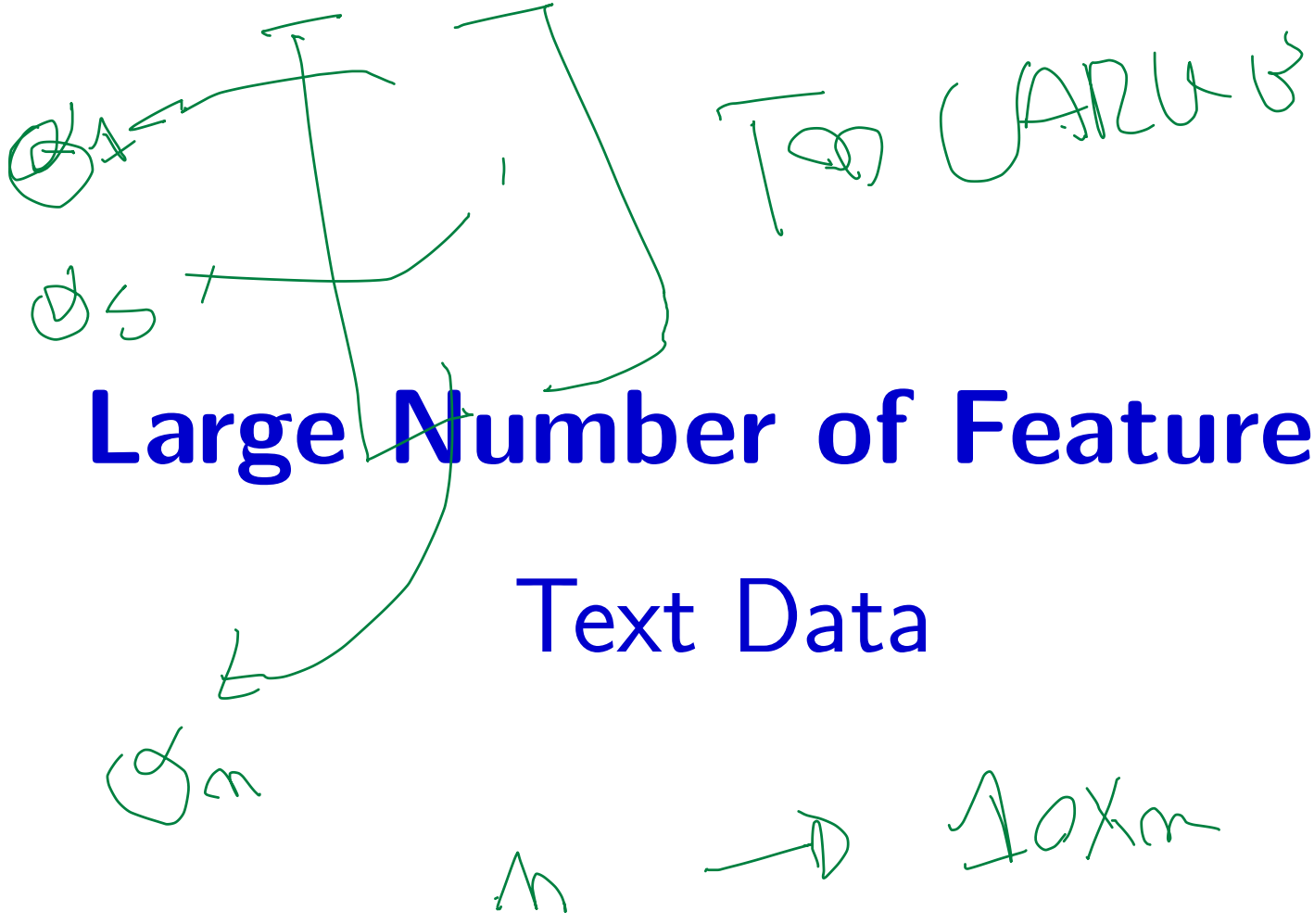
Other Features in documents?

Examples:

- Document length
- H1 position
- Image ratio w.r.t. text
- HTML parsing



Notes



Large Number of Features

Text Data

Feature Selection

We can use for instance L1 regularization.

- Useful for feature selection
- The "selection" is happening during training (is not a pre-processing)

Often worse than L2 in prediction, but it can improve the prediction time.

Sequential algorithms (slow):

- Remove 1 feature at a time and see the performance.
- remove the feature that is less important.

Feature Selection

Other methods:

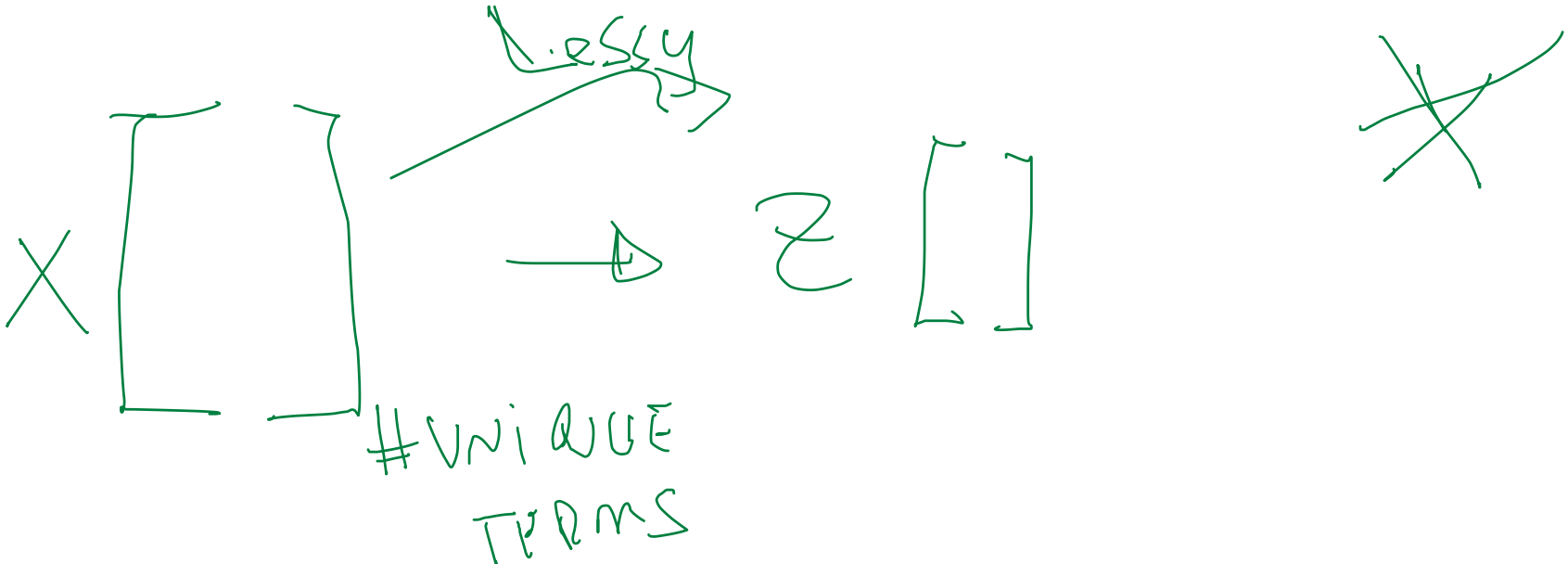
- Remove features based on statistical correlation.
- Remove the long tail.

Dimensionality Reduction

Dimensionality Reduction

Dimensionality reduction means reducing the number of features in your data

- We want to transform the feature space, creating features that are different from the original features.



Dimensionality Reduction

Example: two different types of blood pressure usually correlated

BP(S)	BP(D)	Heart Rate	Temperature
120	80	75	98.5
125	82	78	98.7
140	93	95	98.5
112	74	80	98.6

Dimensionality Reduction

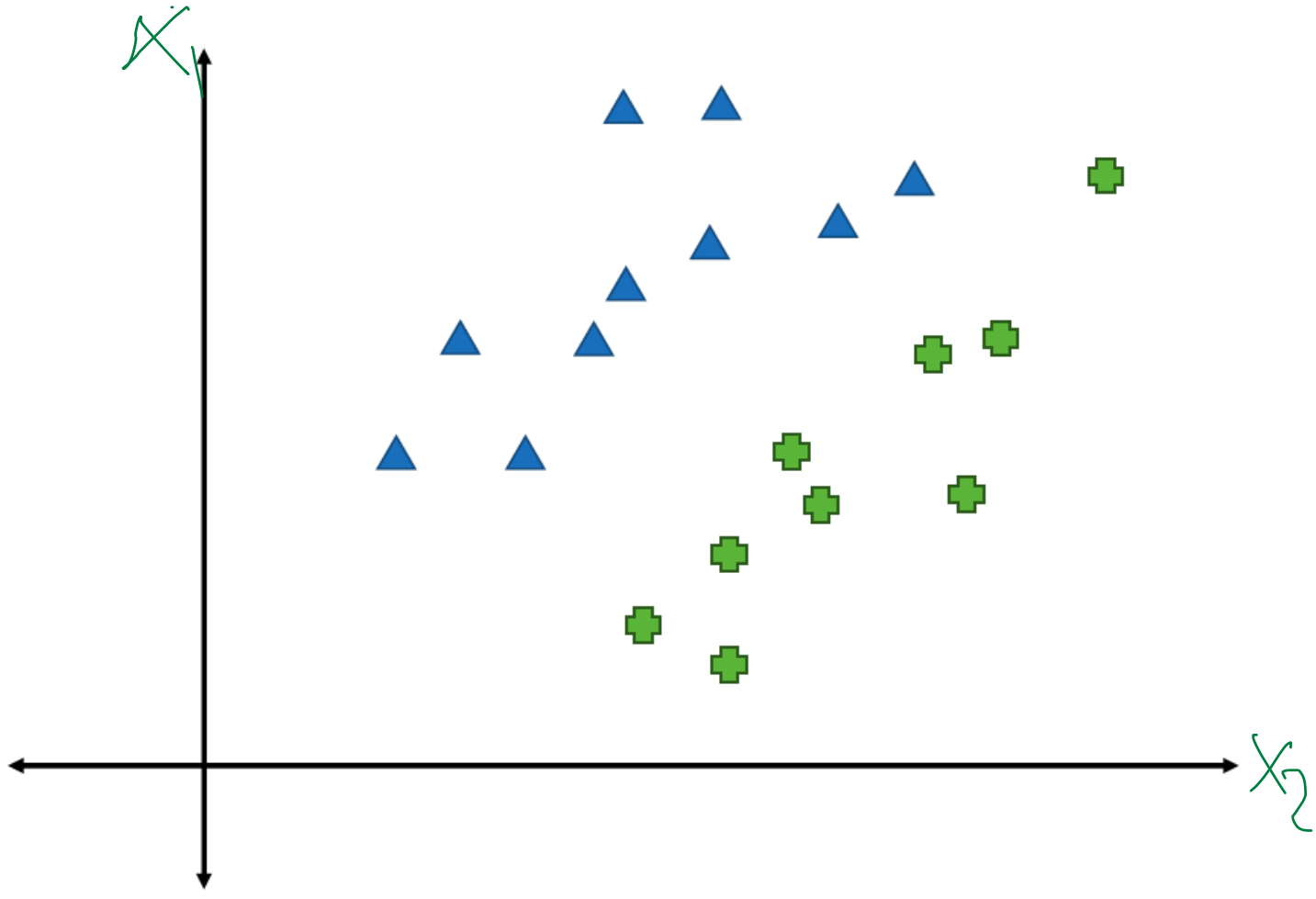
Example: two different types of blood pressure, usually correlated

BP(S)	BP(D)	BP-Avg	Heart Rate	Temperature
120	80	100	75	98.5
125	82	104	78	98.7
140	93	117	95	98.5
112	74	93	80	98.6

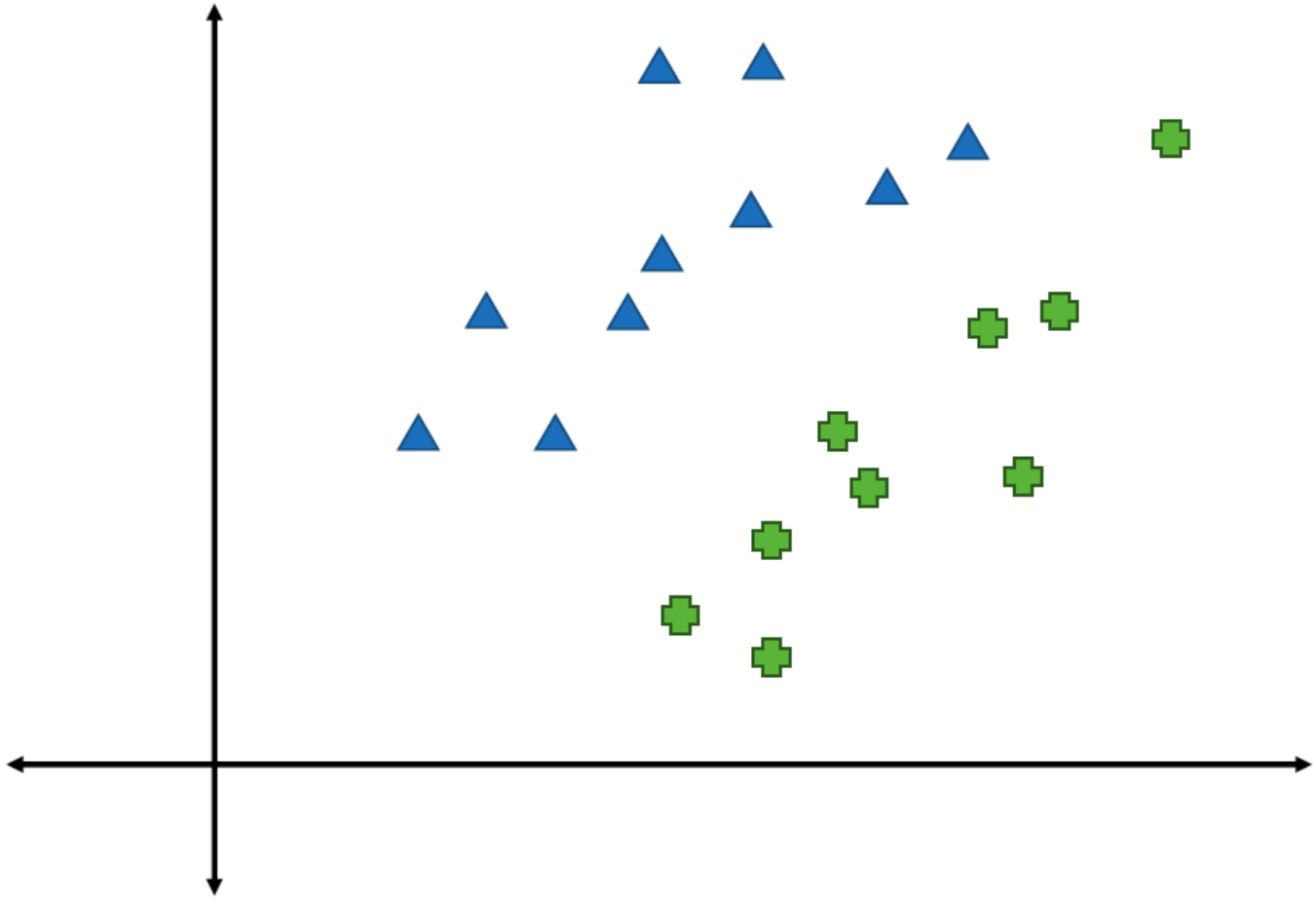
Replace the two BP features with the average of them.

- We are reducing the features in a lossy way, its not selection.

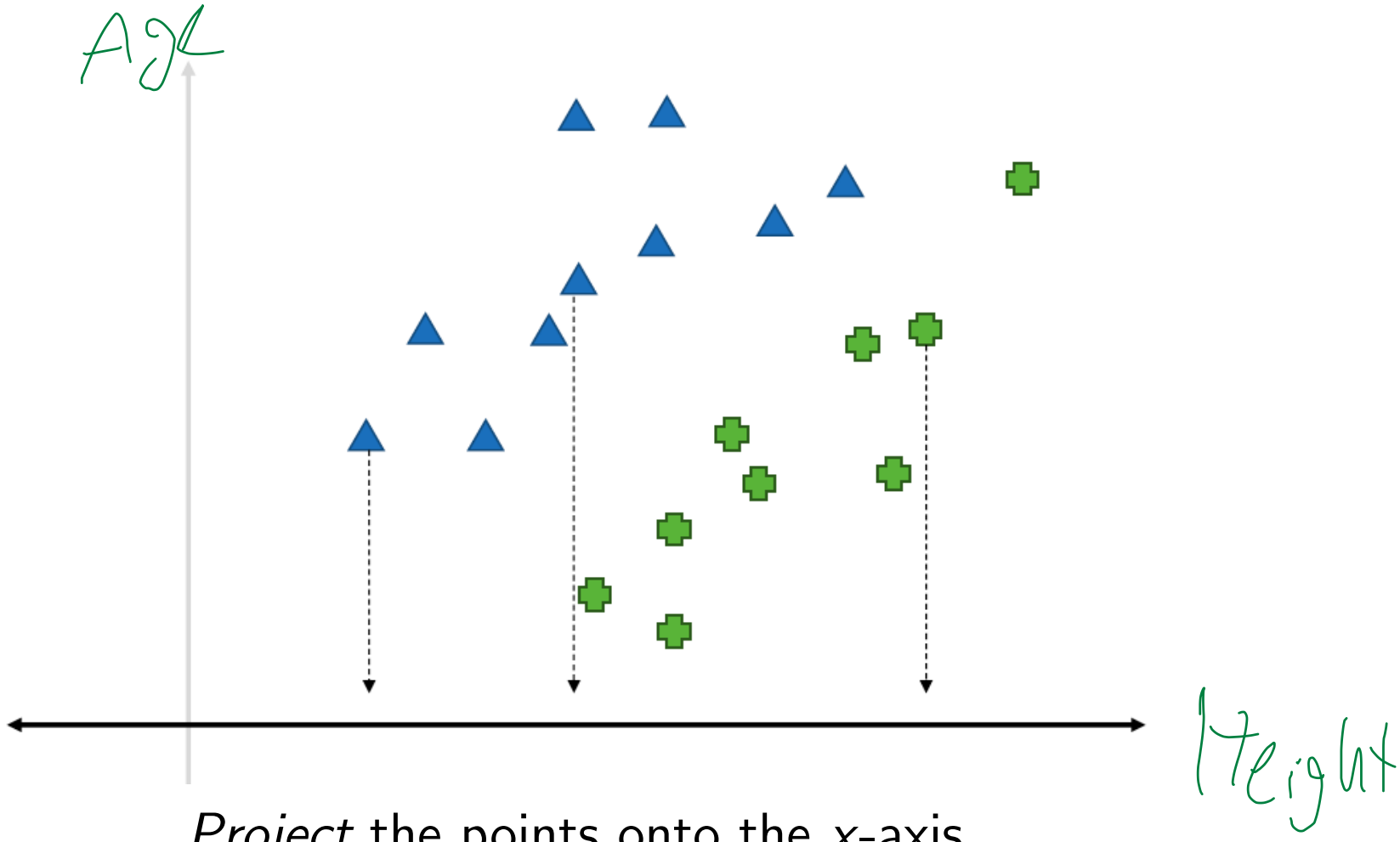
Suppose we have two dimensions (two features)



Feature selection: choose one of the two features to keep

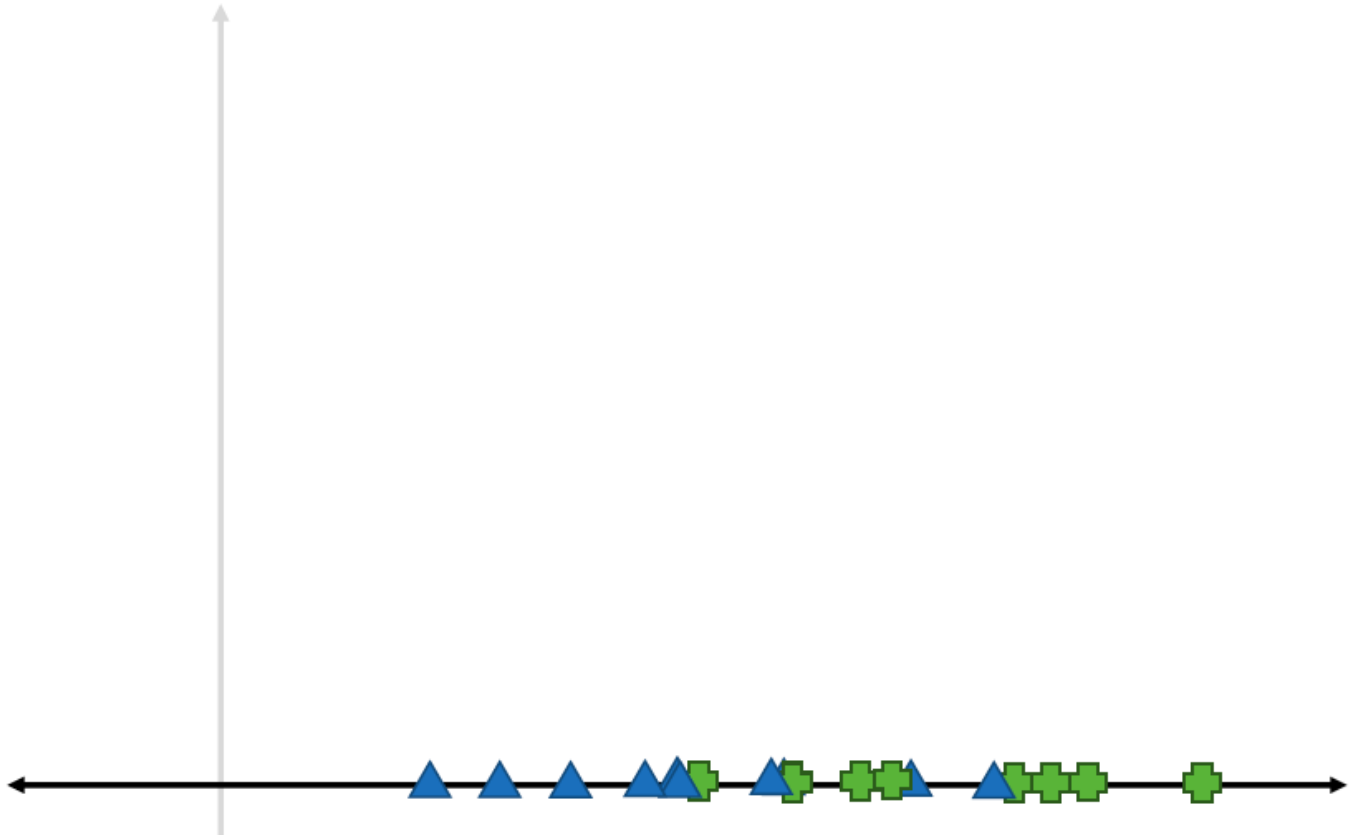


Suppose we choose the feature represented by the x-axis



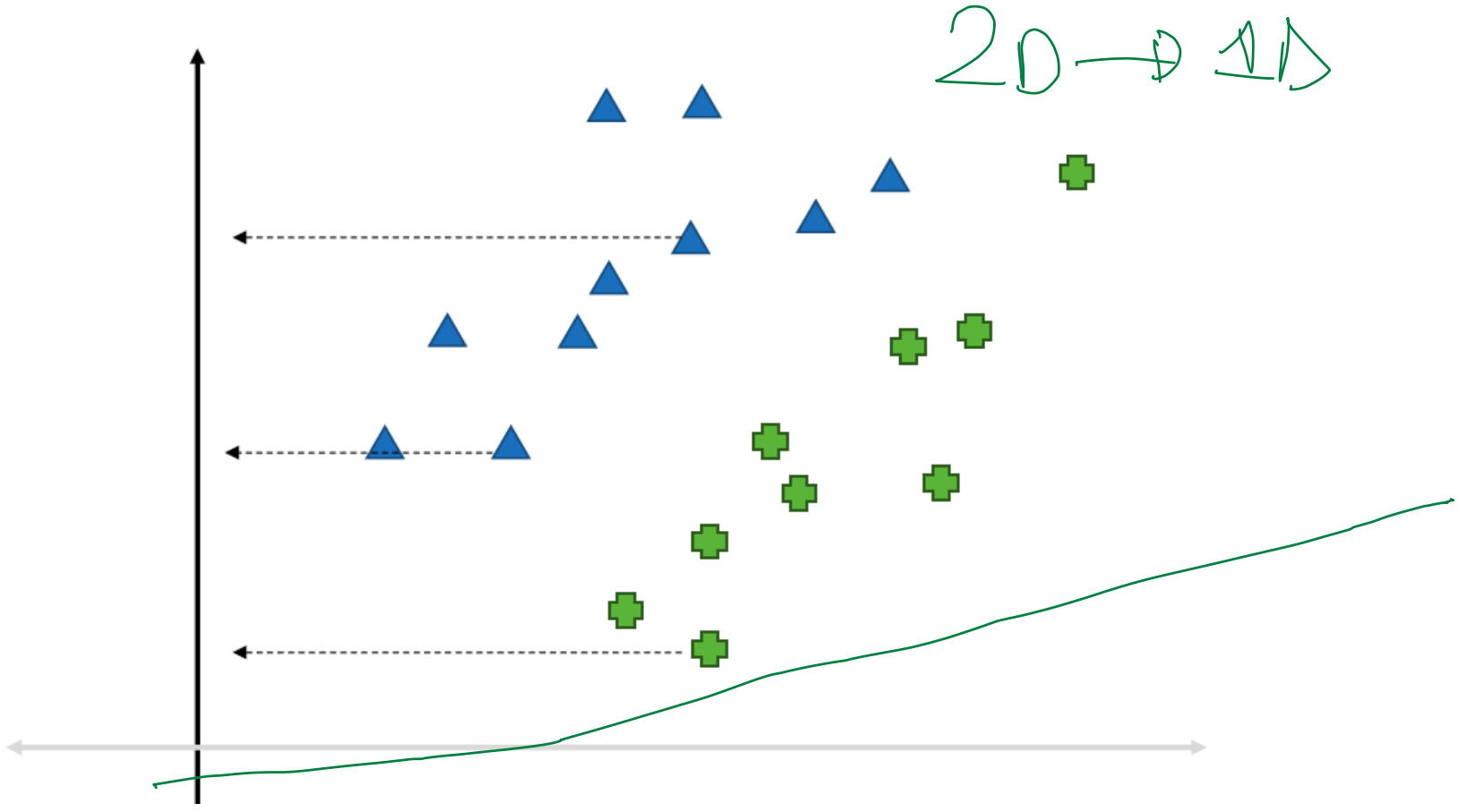
Project the points onto the x-axis

Suppose we choose the feature represented by the x-axis



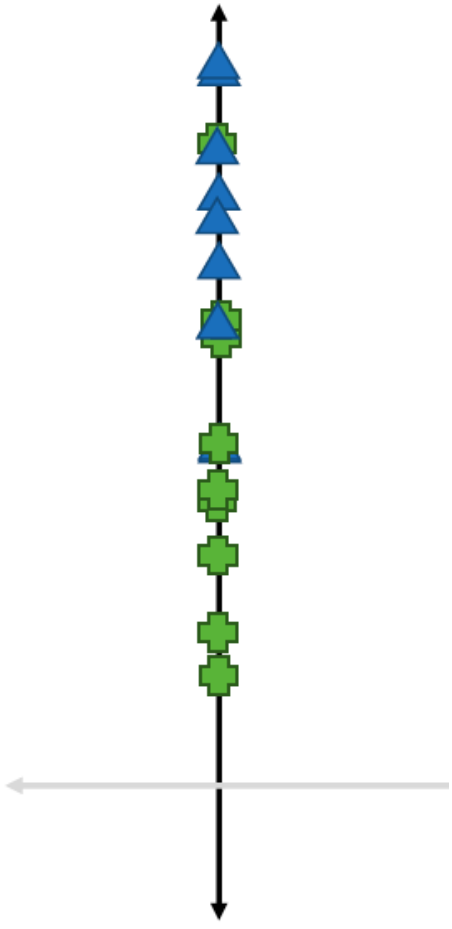
The positions along the x-axis now represent the feature values of each instance

Suppose we choose the feature represented by the y-axis



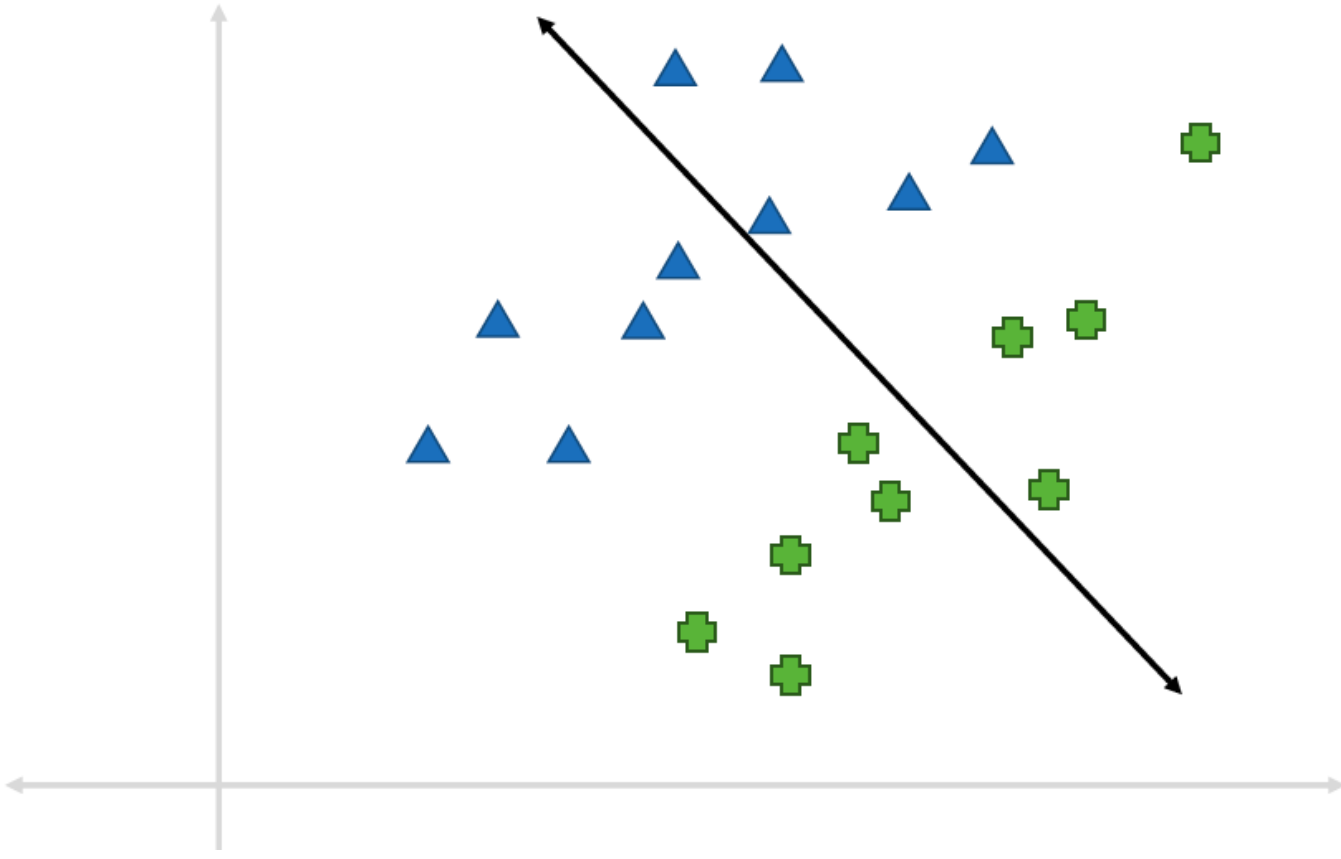
Project the points onto the y-axis

Suppose we choose the feature represented by the y-axis



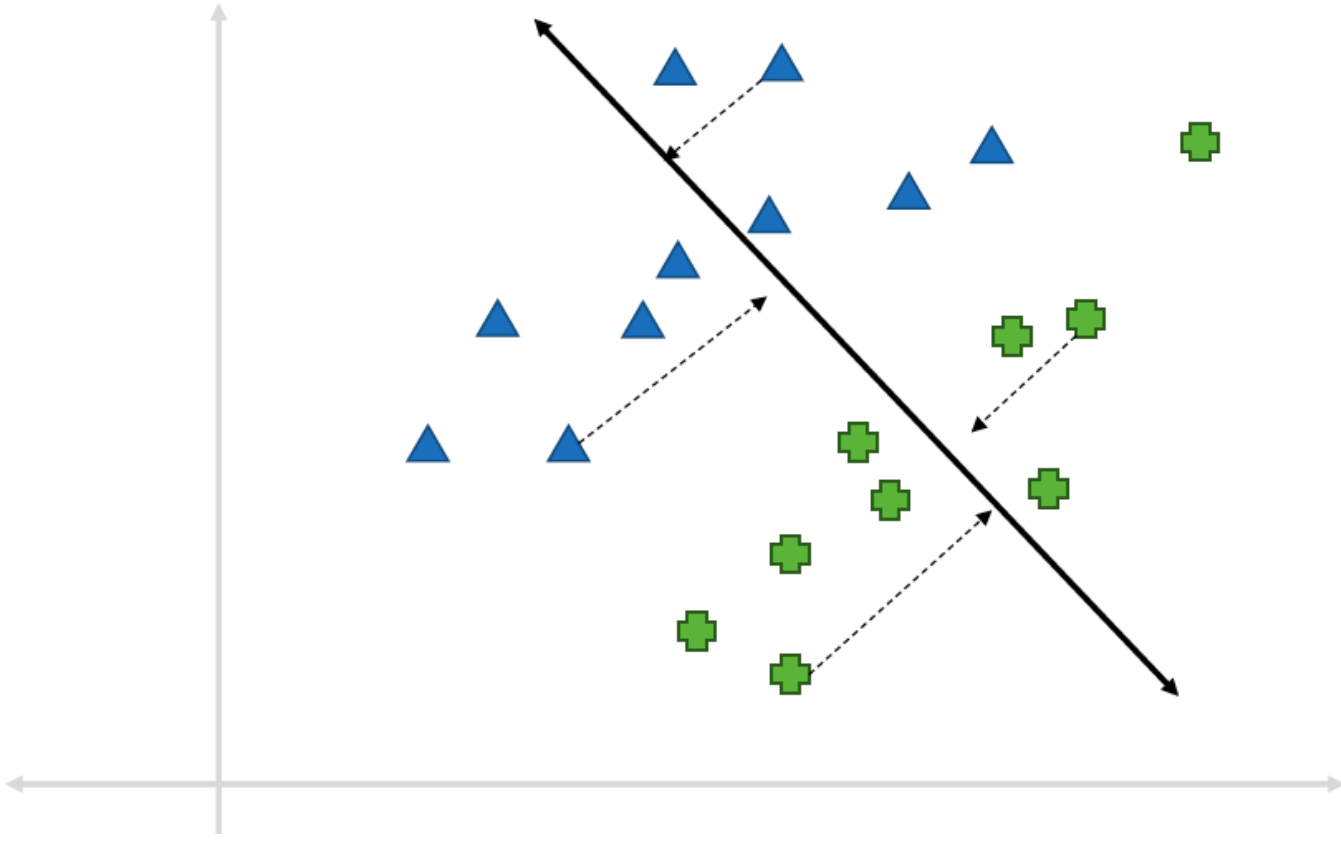
The positions along the y-axis now represent the feature values of each instance

We don't have to restrict ourselves to picking either the x-axis or y-axis



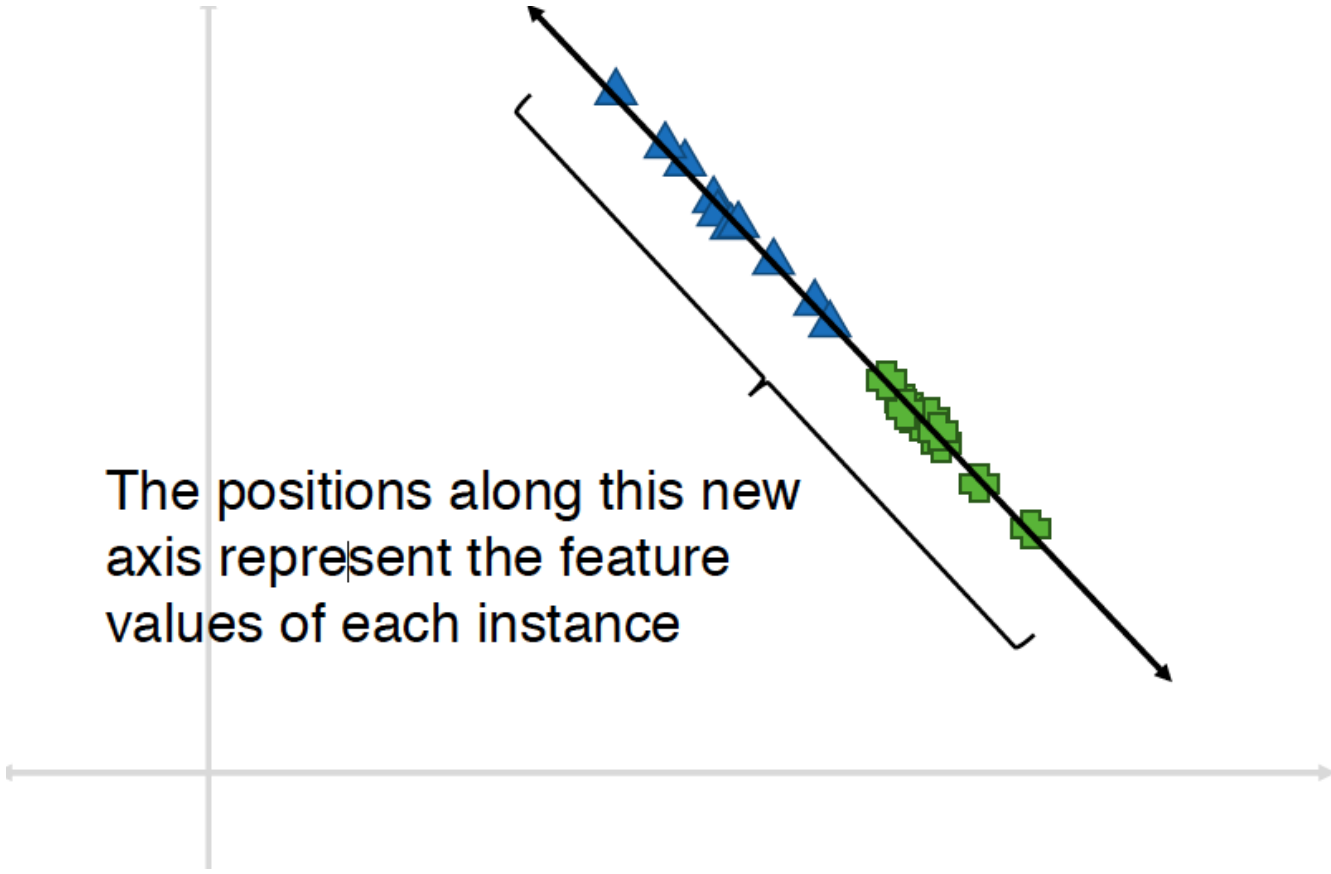
We could create a new axis!

We don't have to restrict ourselves to picking either the x-axis or y-axis

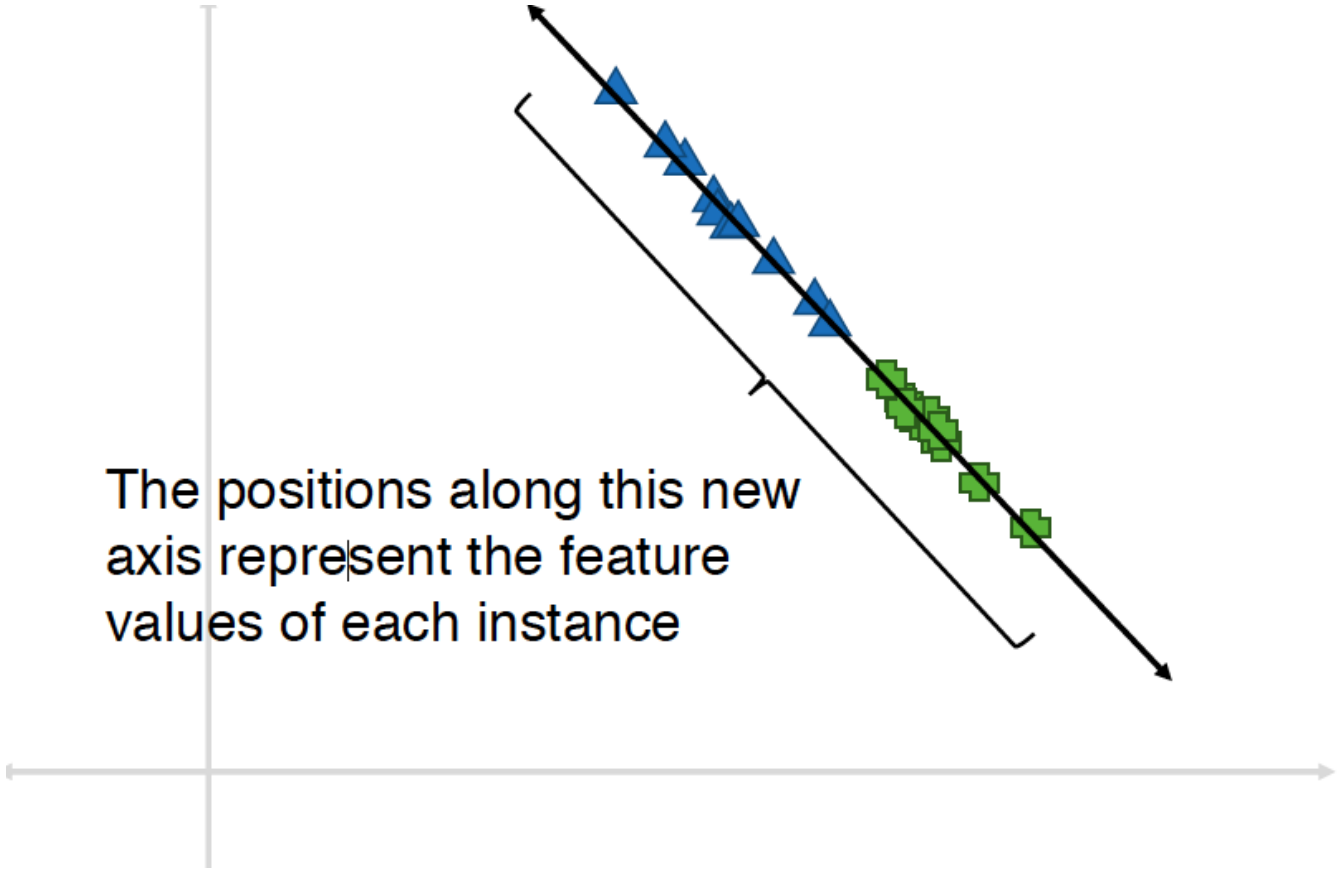


Project points onto this new axis

We don't have to restrict ourselves to picking either the x-axis or y-axis



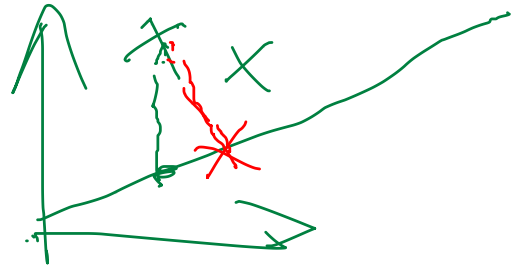
This is an example of transforming the feature space (as opposed to *selecting* a subset of features)

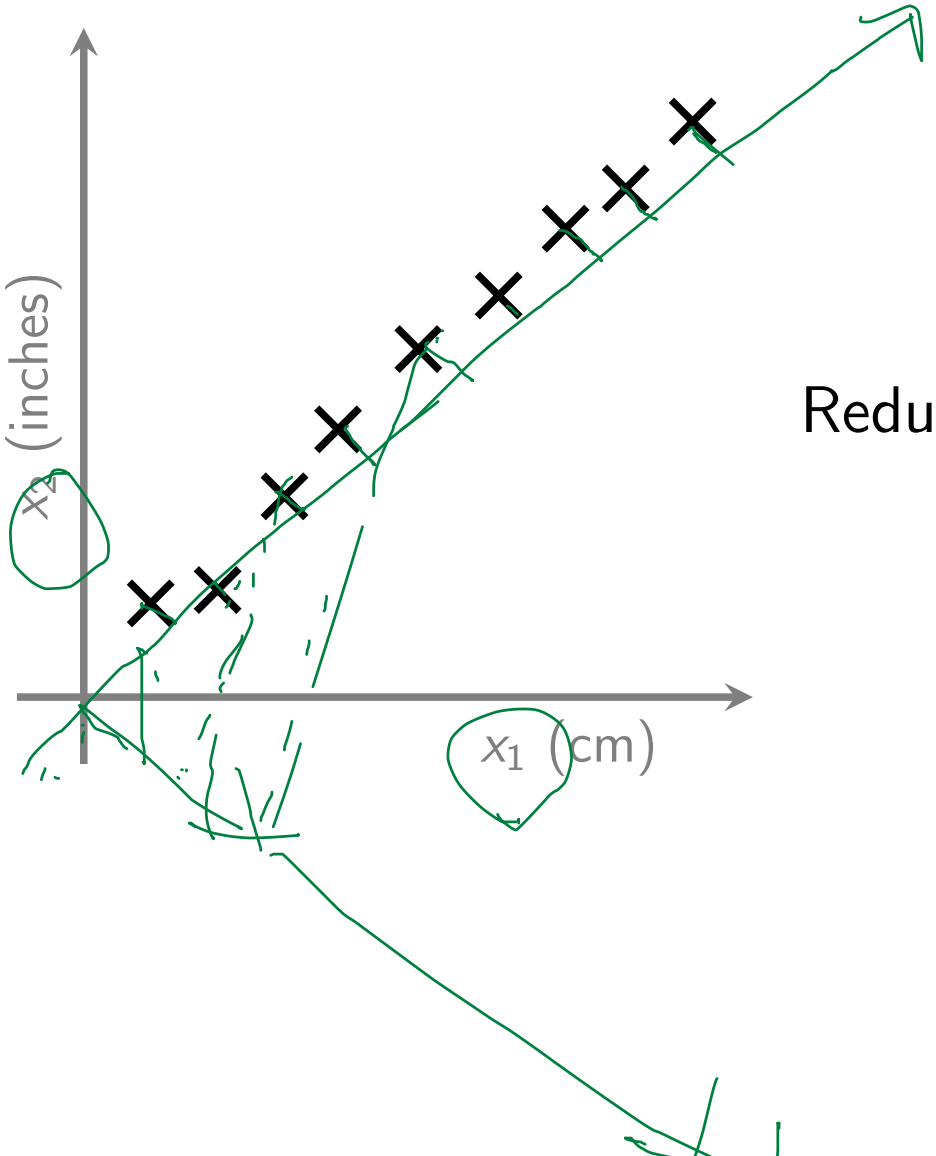


Dimensionality Reduction

Data Compression

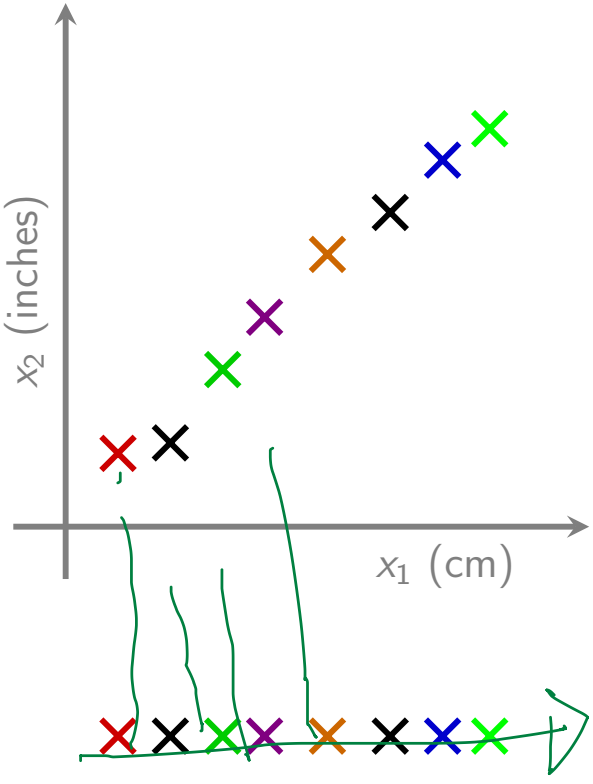
Data Compression





Reduce data from 2D to 1D

Data Compression

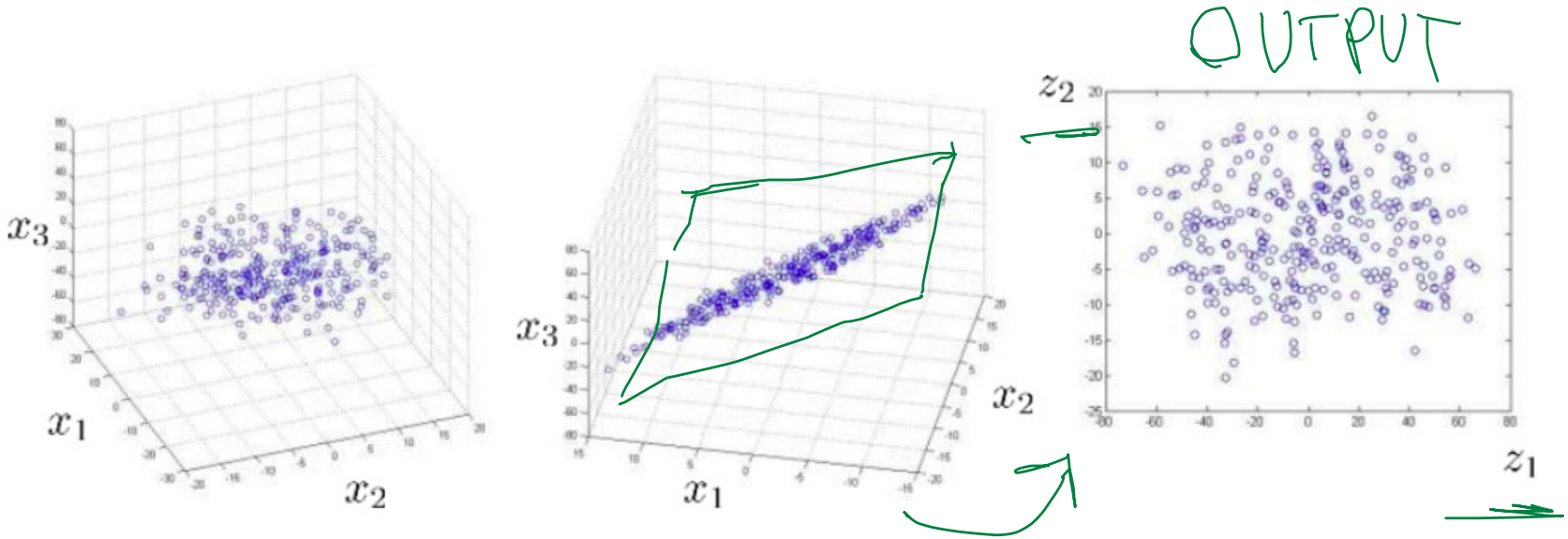


Reduce data from 2D to 1D

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

Data Compression

Reduce data from 3D to 2D



Dimensionality Reduction

Data Visualization

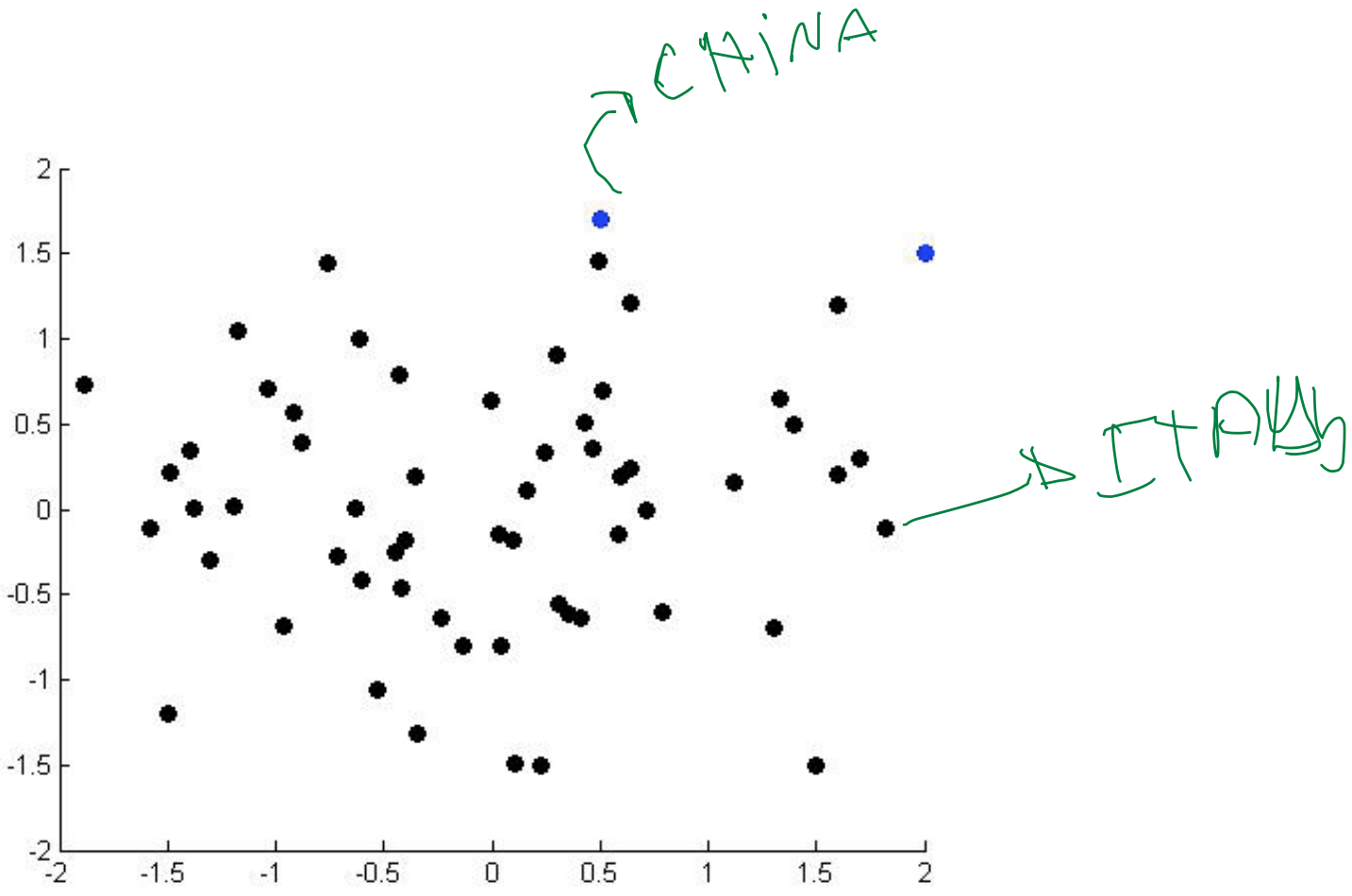
Data Visualization

Country	GDP (trillions of US$)	Per capita GDP (thousands of intl. $)	Human Development Index	Life expectancy	Poverty Index (Gini as percentage)	Mean household income (thousands of US$)	...
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	...
...	...	...	...	...	...	...	...

Data Visualization

Country	z_1	z_2
Canada	1.6	1.2
China	1.7	0.3
India	1.6	0.2
Russia	1.4	0.5
Singapore	0.5	1.7
USA	2	1.5
...	...	...

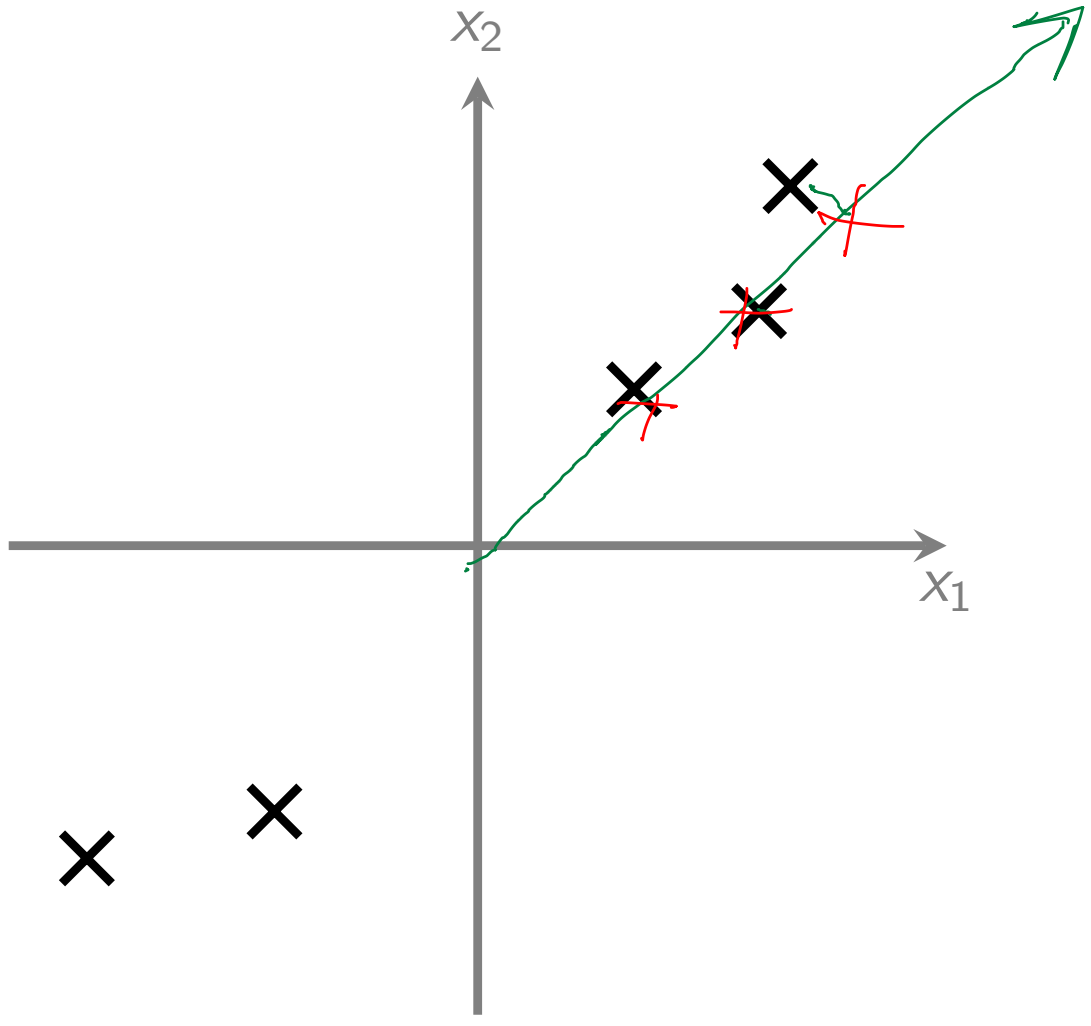
Data Visualization



Dimensionality Reduction

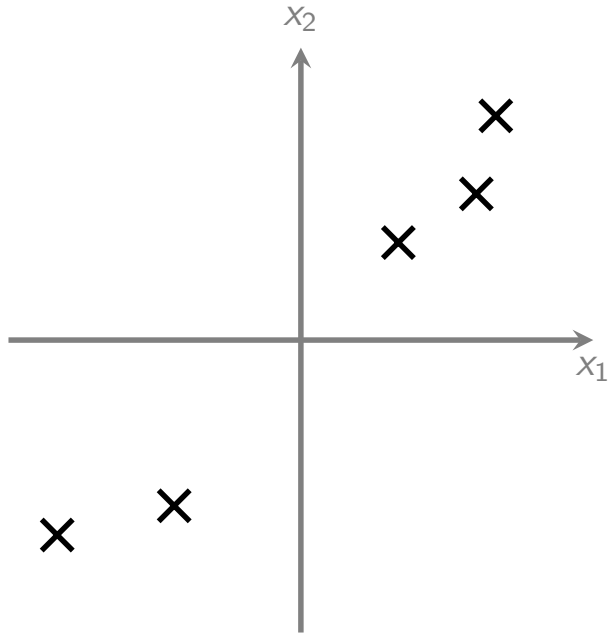
Principal Component Analysis problem formulation

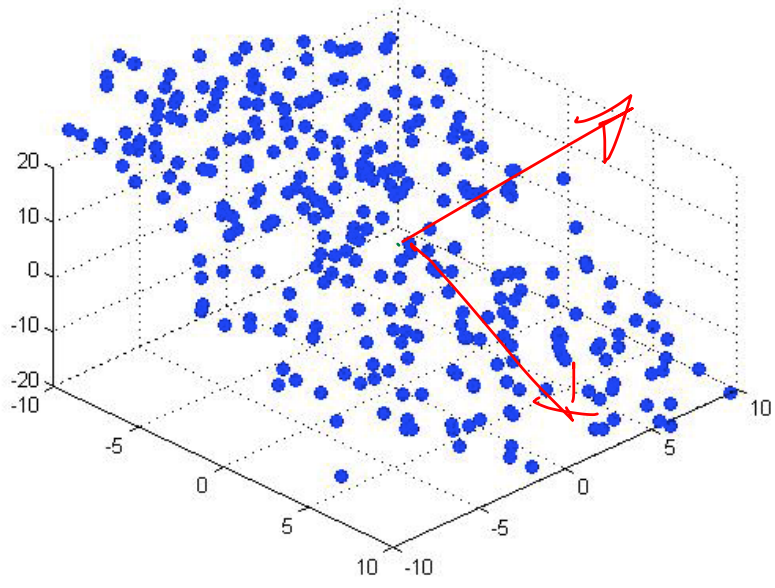
Principal Component Analysis (PCA) problem formulation



ERROR is minimal

Principal Component Analysis (PCA) problem formulation

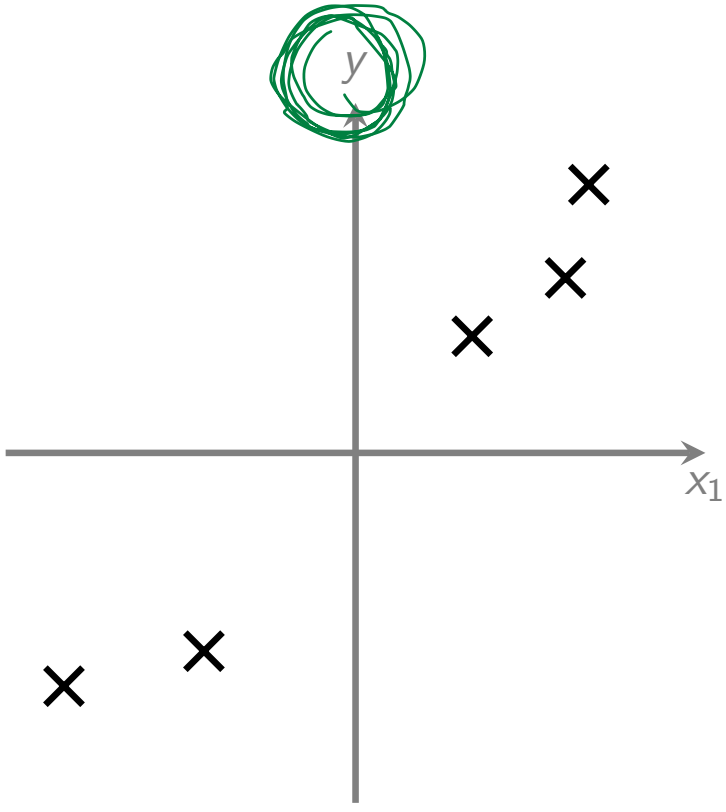


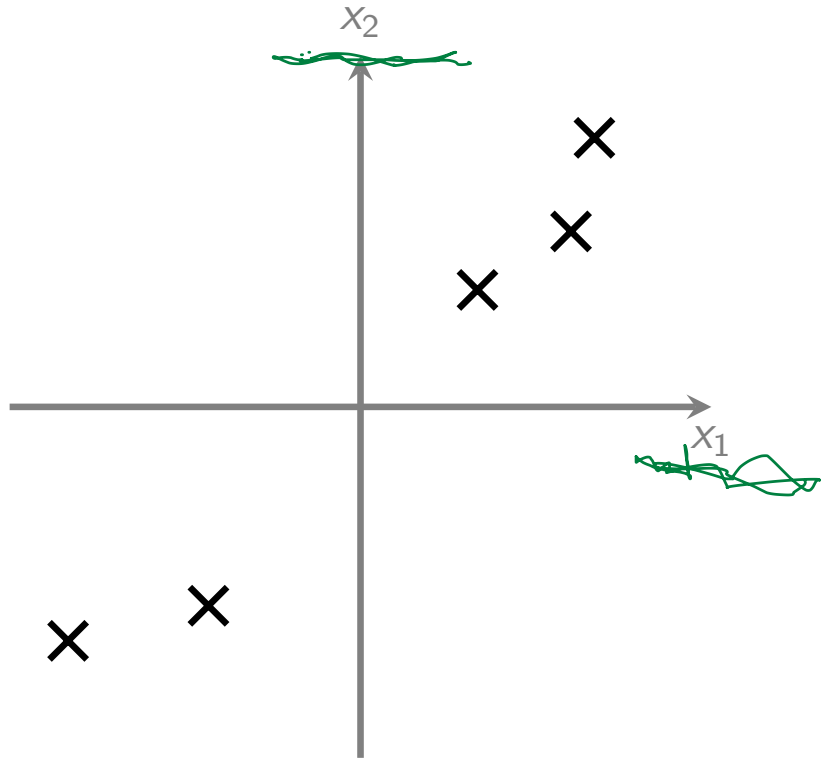


Reduce from 2-dimension to 1-dimension: Find a direction (a vector $u^{(1)} \in \mathbb{R}$) onto which to project the data so as to minimize the projection error.

Reduce from n-dimension to k-dimension: Find k vectors $u^{(1)}, u^{(2)}, \ldots, u^{(k)}$ onto which to project the data, so as to minimize the projection error.

PCA is not linear regression





Dimensionality Reduction

Principal Component Analysis algorithm

Data preprocessing

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

Preprocessing (feature scaling/mean normalization):

$$\mu_j = \frac{1}{m} \sum_{i=1}^{m} x_j^{(i)}$$

Replace each $x_j^{(i)}$ with $x_j - \mu_j$.

If different features on different scales (e.g., x_1 = size of house, x_2 =number of bedrooms), scale features to have comparable range of values.

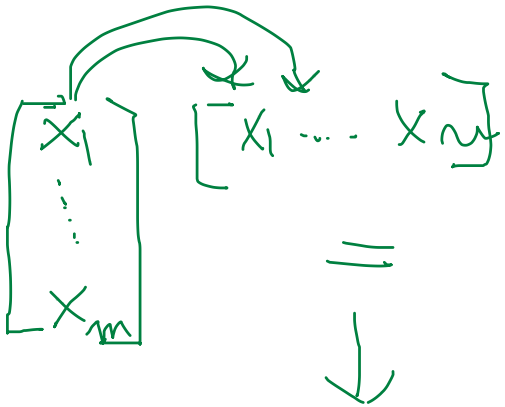
Principal Component Analysis (PCA) algorithm

$k < n$

Reduce data from n-dimensions to k-dimensions

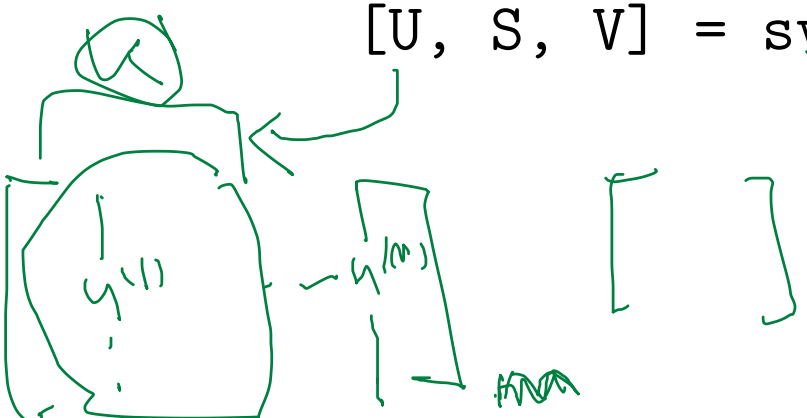
Compute "covariance matrix":

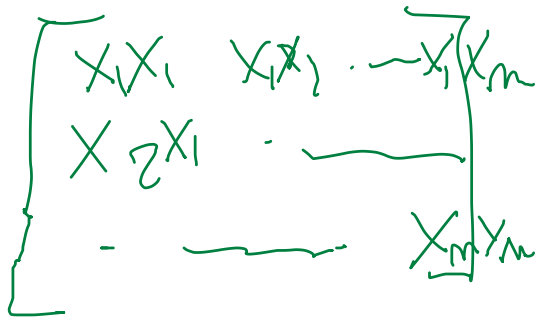
$$\Sigma = \frac{1}{m} \sum_{i=1}^{n} \left(x^{(i)}\right) \left(x^{(i)}\right)^T$$



Compute "eigenvectors" of matrix Σ:

```
[U, S, V] = svd(Sigma);
```

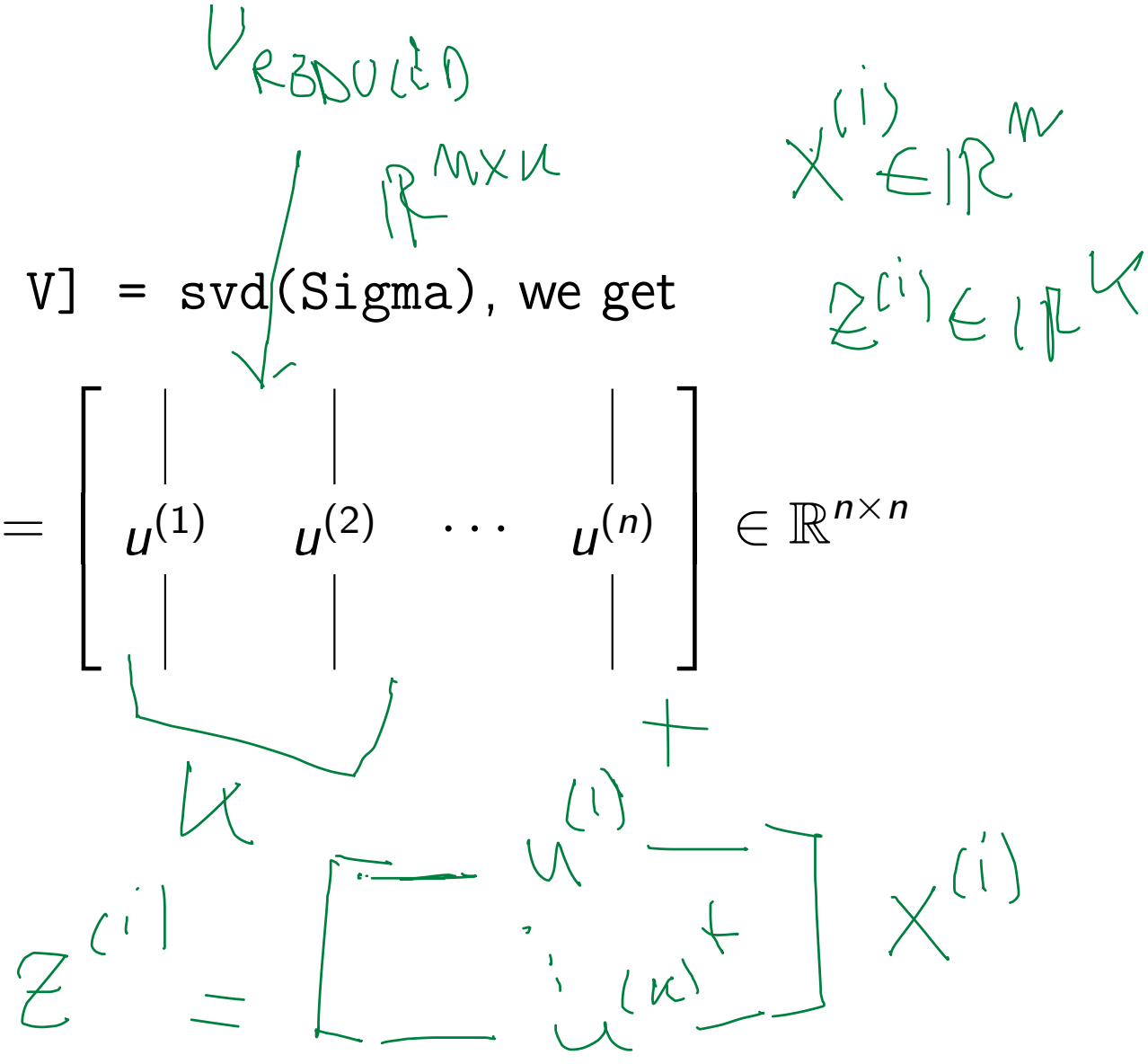




Principal Component Analysis (PCA) algorithm

From `[U, S, V] = svd(Sigma)`, we get

$$U = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \cdots & u^{(n)} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times n}$$



Principal Component Analysis (PCA) algorithm summary

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

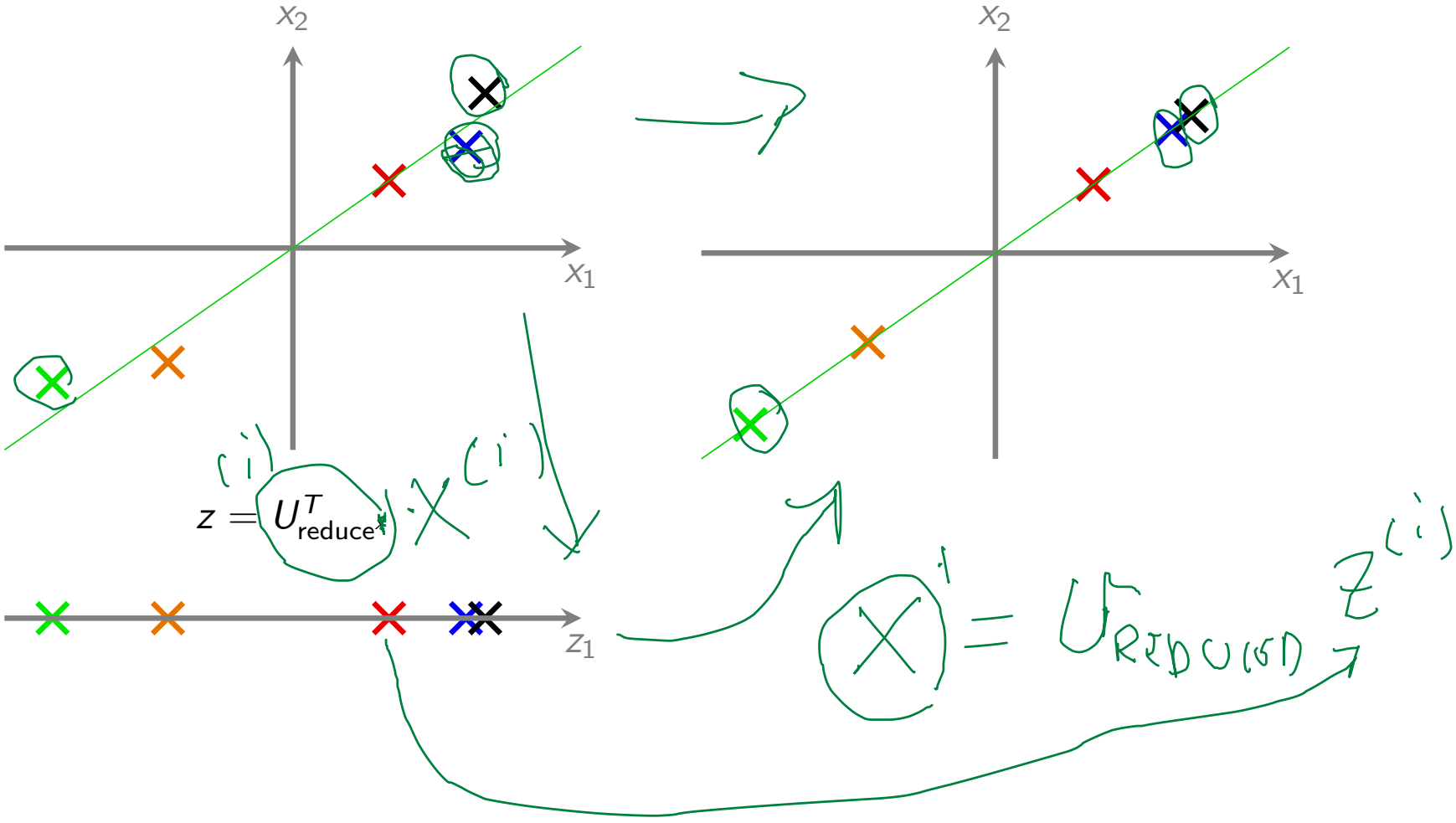
Sigma = $\frac{1}{m}\sum_{i=1}^{m}\left(x^{(i)}\right)\left(x^{(i)}\right)^{T}$

```
[U,S,V]=svd(Sigma);
Ureduce = U(:,1:k);
z=Ureduce'*x;
```

Dimensionality Reduction

Reconstruction from compressed representation

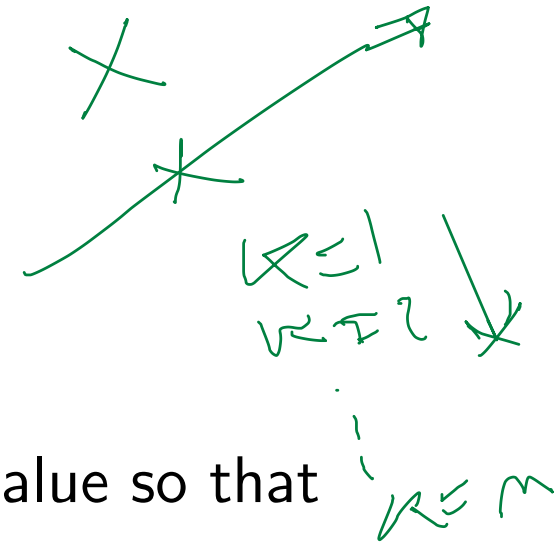
Reconstruction from compressed representation



Dimensionality Reduction

Choosing the number of principal components

Choosing k (number of principal components)



Average squared projection error:
Total variation in the data:
Typically, choose k to be smallest value so that

$$\frac{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)} - x_{\text{approx}}^{(i)}\|^2}{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}\|^2} \leq 0.01 \quad (1\%)$$

"99% of variance is retained"

Choosing k (number of principal components)

Algorithm:

`[U,S,V]=svd(Sigma)`

Try PCA with $k = 1$

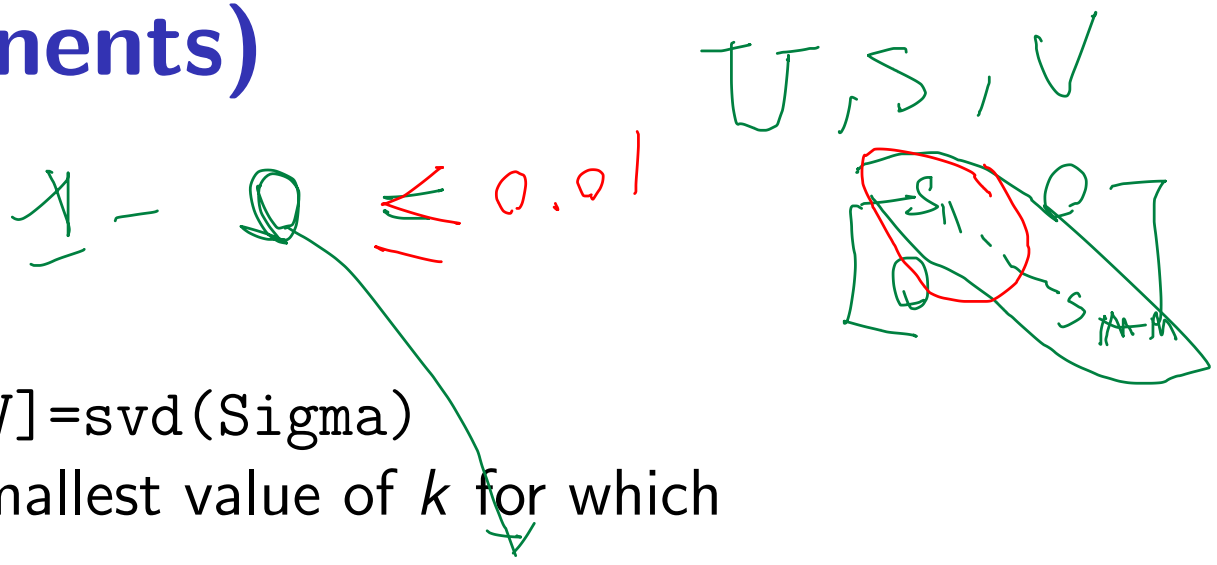
Compute U_{reduce}, $z^{(1)}$, $z^{(2)}$, $\ldots$, $z^{(m)}$, $x^1_{\text{approx}}, \ldots, x^m_{\text{approx}}$

Check if

$$\frac{\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)} - x^{(i)}_{\text{approx}}\|^2}{\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)}\|^2} \leq 0.01$$

$$1 - \frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{m} S_{ii}}$$

Choosing k (number of principal components)



`[U,S,V]=svd(Sigma)`

Pick smallest value of k for which

$$\frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{m} S_{ii}} \geq 0.99$$

(99% of variance retained)

Dimensionality Reduction

Advice for applying PCA

Supervised learning speedup

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

Extract inputs:

Unlabeled dataset: $x^{(1)}, x^{(2)}, \ldots, x^{(m)} \in \mathbb{R}^{10000}$

$\downarrow PCA$

$z^{(1)}, z^{(2)}, \ldots, z^{(m)} \in \mathbb{R}^{1000}$

New training set:

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

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 sets.

Application of PCA

- Compression
  - Reduce memory/disk needed to store data
  - Speed up learning algorithm
- Visualization

Bad use of PCA: To prevent overfitting

Use $z^{(i)}$ instead of $x^{(i)}$ to reduce the number of features to $k < n$.
Thus, fewer features, less likely to overfit.

This might work OK, but isn't a good way to address overfitting. Use regularization instead.

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

PCA is sometimes used where it shouldn't be

Design of ML system:

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

How about doing the whole thing without using PCA? Before implementing PCA first try running whatever you want to do with the original/raw data $x^{(i)}$. Only if that doesn't do what you want, then implement PCA and consider using $z^{(i)}$.