Dimensionality Reduction

INFO-4604, Applied Machine Learning

University of Colorado Boulder

October 26, 2017

Prof. Michael Paul

Dimensionality

The **dimensionality** of data is the number of variables

- Usually this refers to the number of input variables
- In other words, the number of features

The **curse of dimensionality** refers to challenges that arise when data has many dimensions

- Training: the more features you have, the more data you need to learn
- Distance: all points are far apart in high-dimensional space, harder to define “close” vs “far”

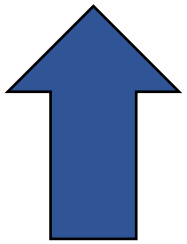
Dimensionality Reduction

Dimensionality reduction refers to the process of reducing the number of features in your data

- Last time we saw *feature selection* as one approach that reduces the number of features
- Today, we'll see methods that *transform* the feature space, creating features that are different from the original features
  - Called feature *extraction* in the textbook, though will not usually use that term in this class

Dimensionality Reduction

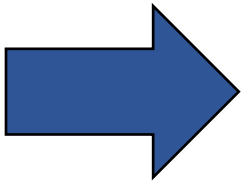
Dimensionality Reduction



“go forward”

or

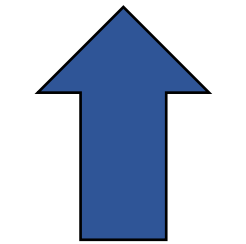
“go up”



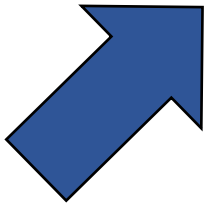
“go right”

Going from 3D to 2D: can lose information, create ambiguity

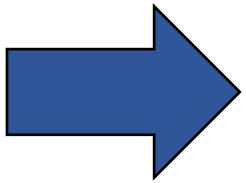
Dimensionality Reduction



“go forward”



“go up”



“go right”

Can adjust the 2D values to carry over 3D meaning

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

You might replace the two BP features with a new feature that simply averages the two original BP values

- This reduces the number of features, but different from feature selection (not just selecting existing features)

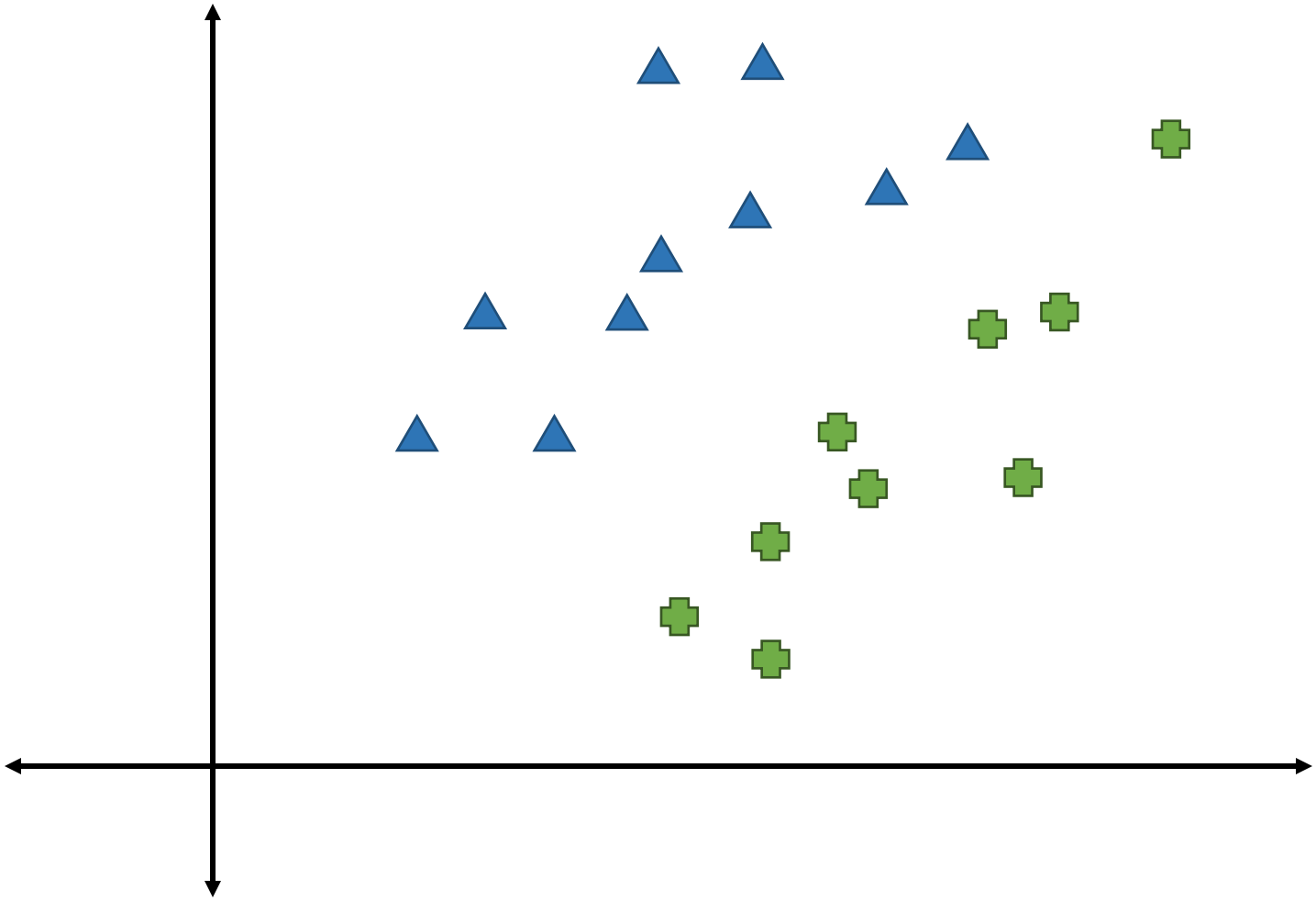
Geometric Intuition

Like feature selection, transformation-based dimensionality reduction is usually automated

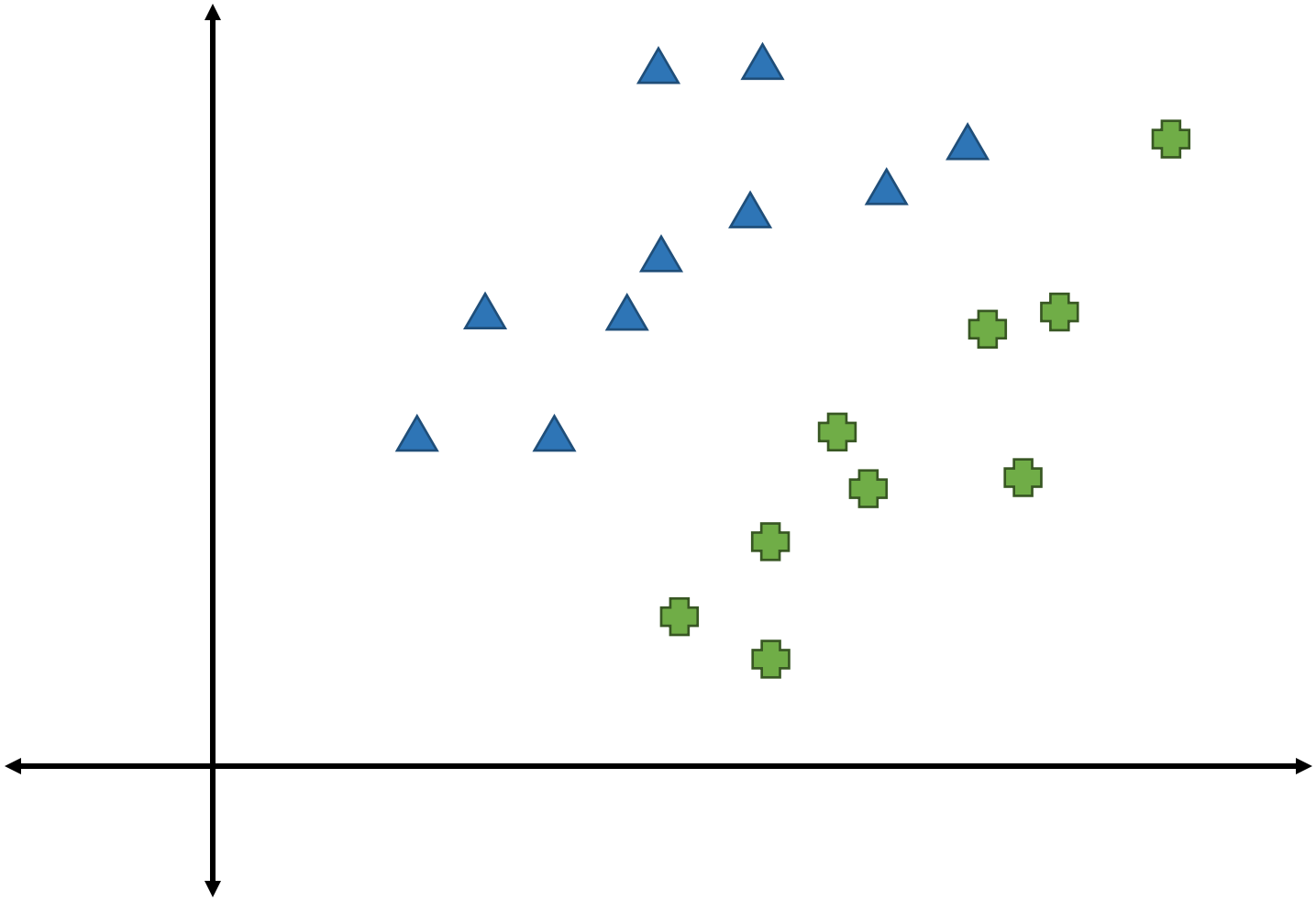
- You don't manually specify rules
  (like averaging BP in the previous example)

In general, what does it mean to transform or change the features?

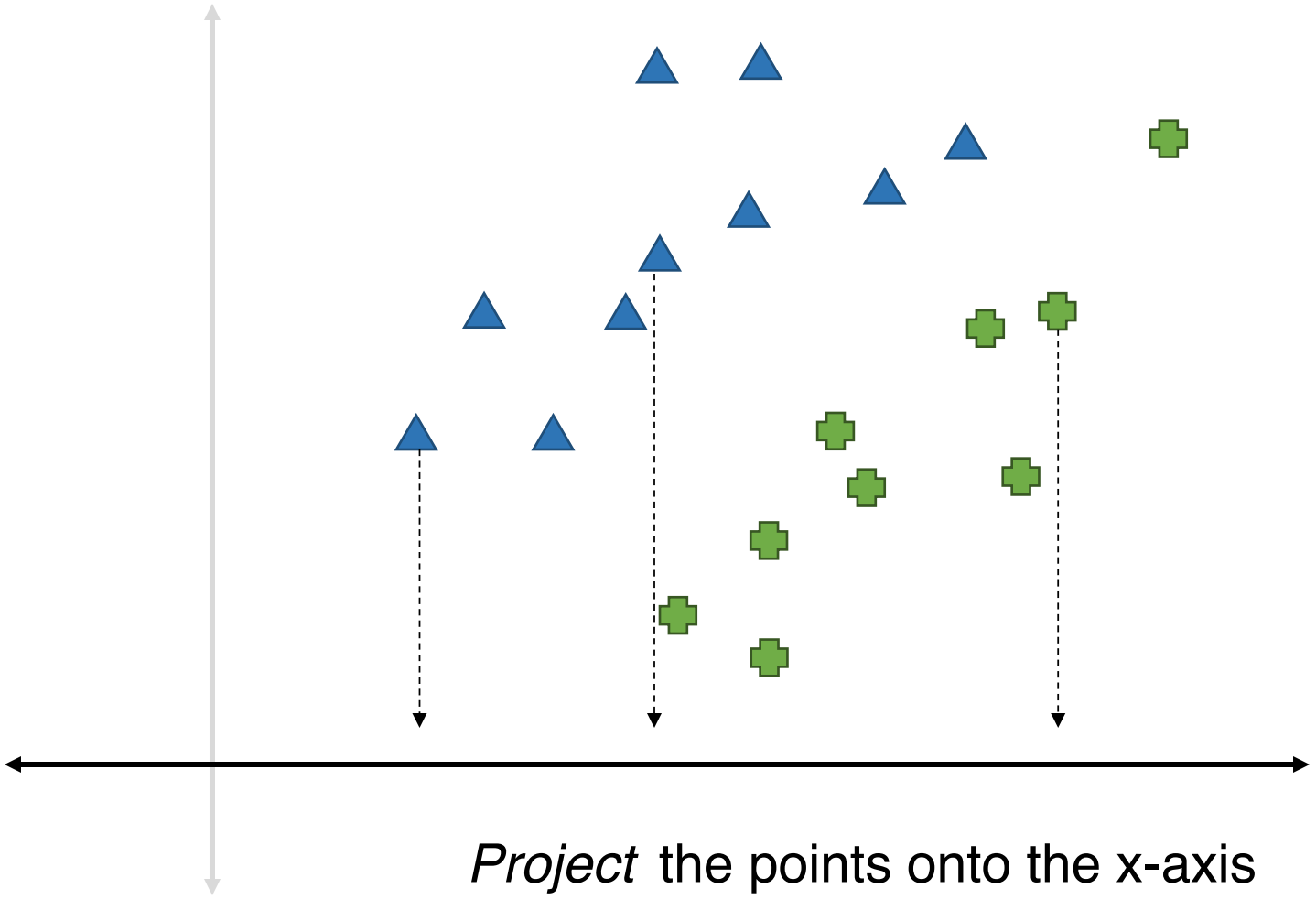
Suppose we have two dimensions (two features)



Feature selection: choose one of the two features to keep

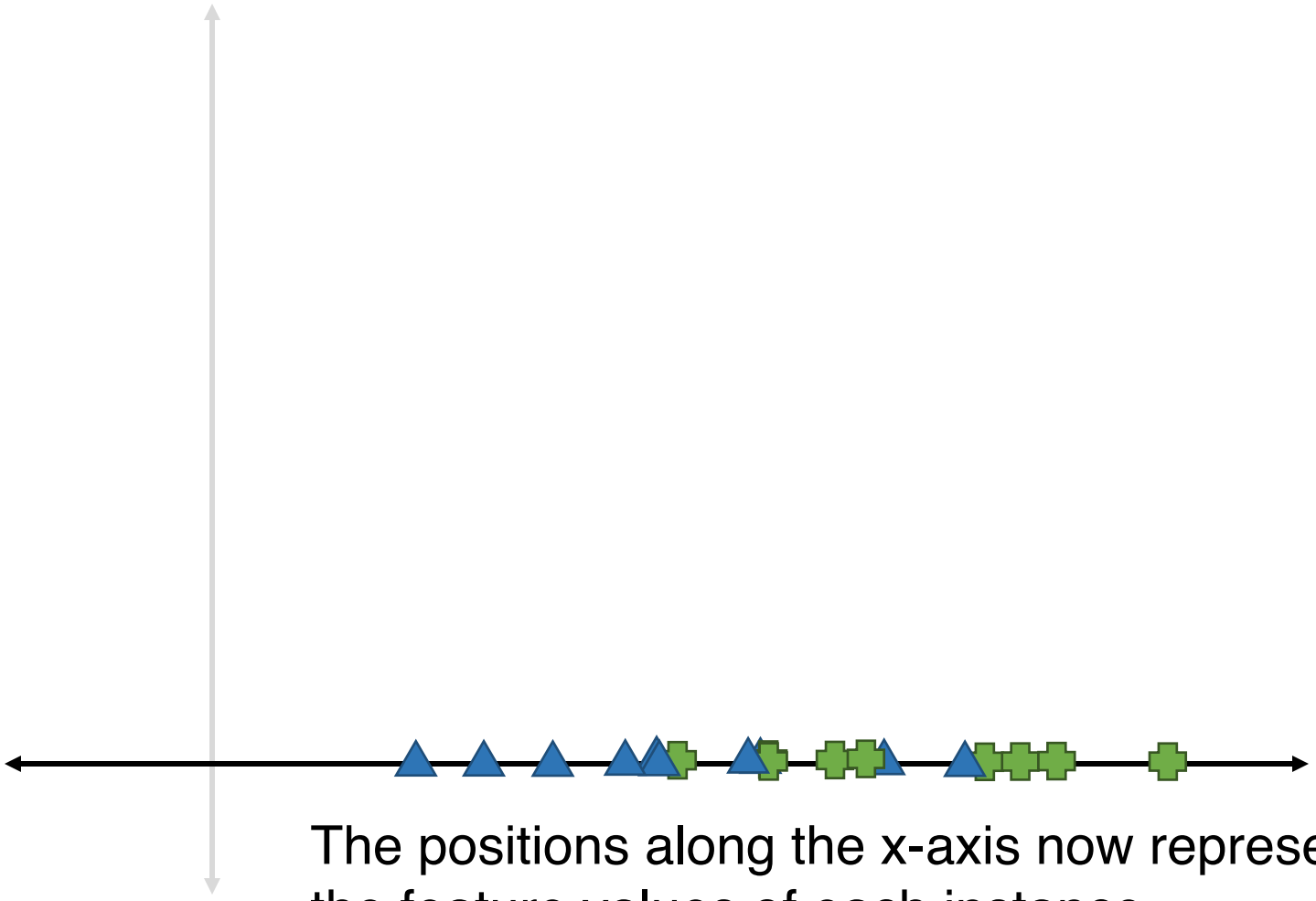


Suppose we choose the feature represented by the x-axis



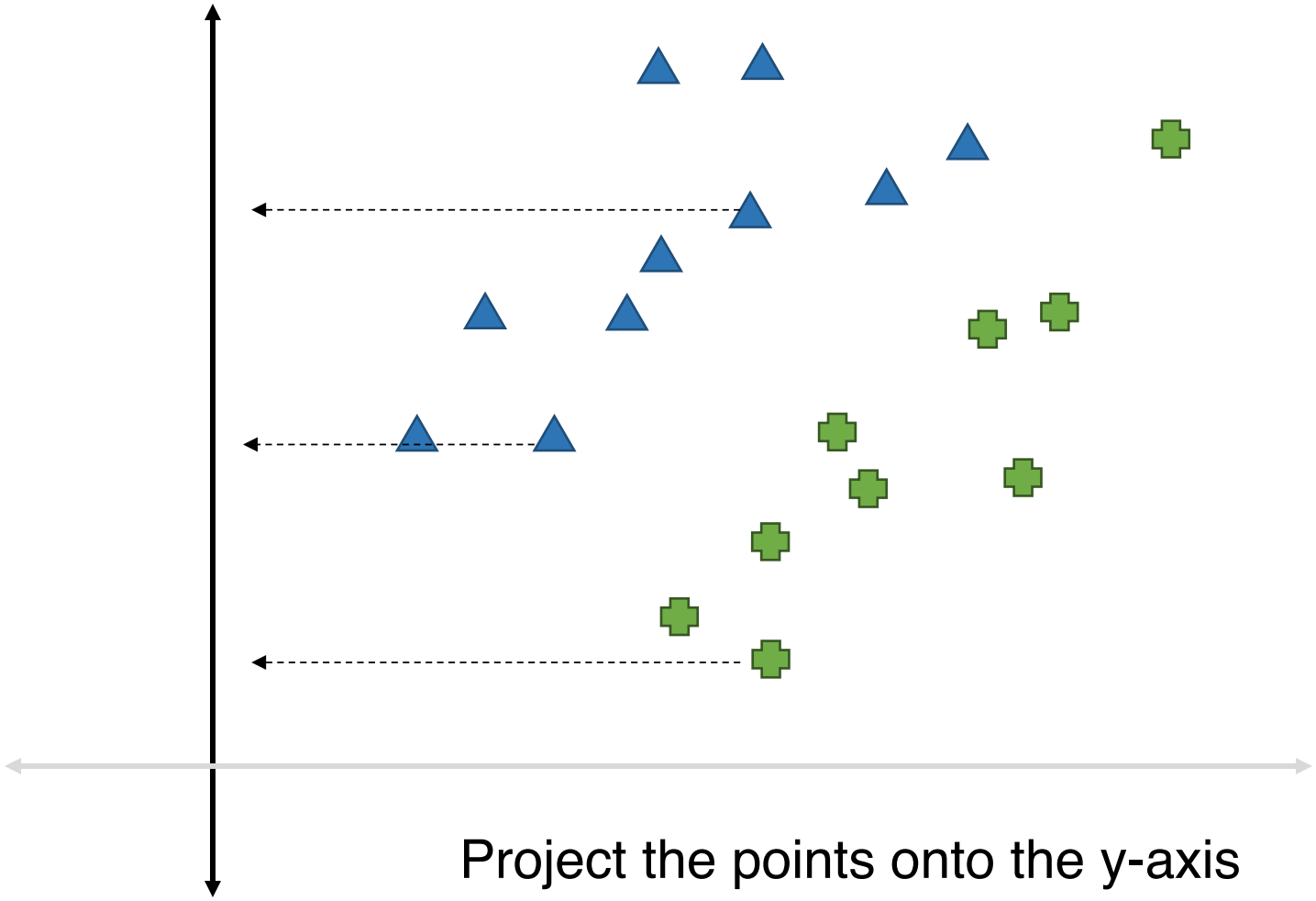
Project the points onto the x-axis

Suppose we choose the feature represented by the x-axis



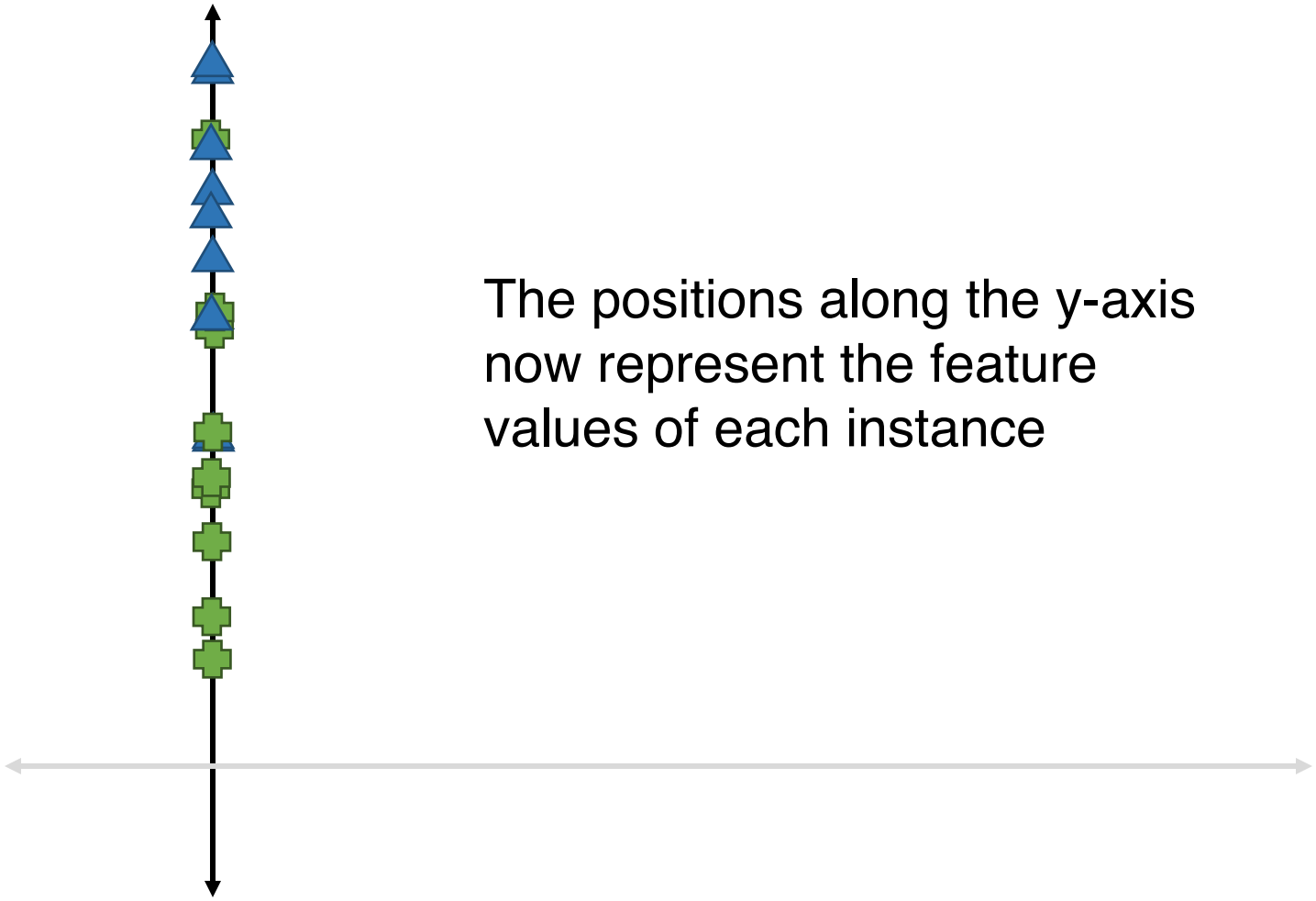
The positions along the x-axis now represent the feature values of each instance

Suppose we choose the feature represented by the y-axis



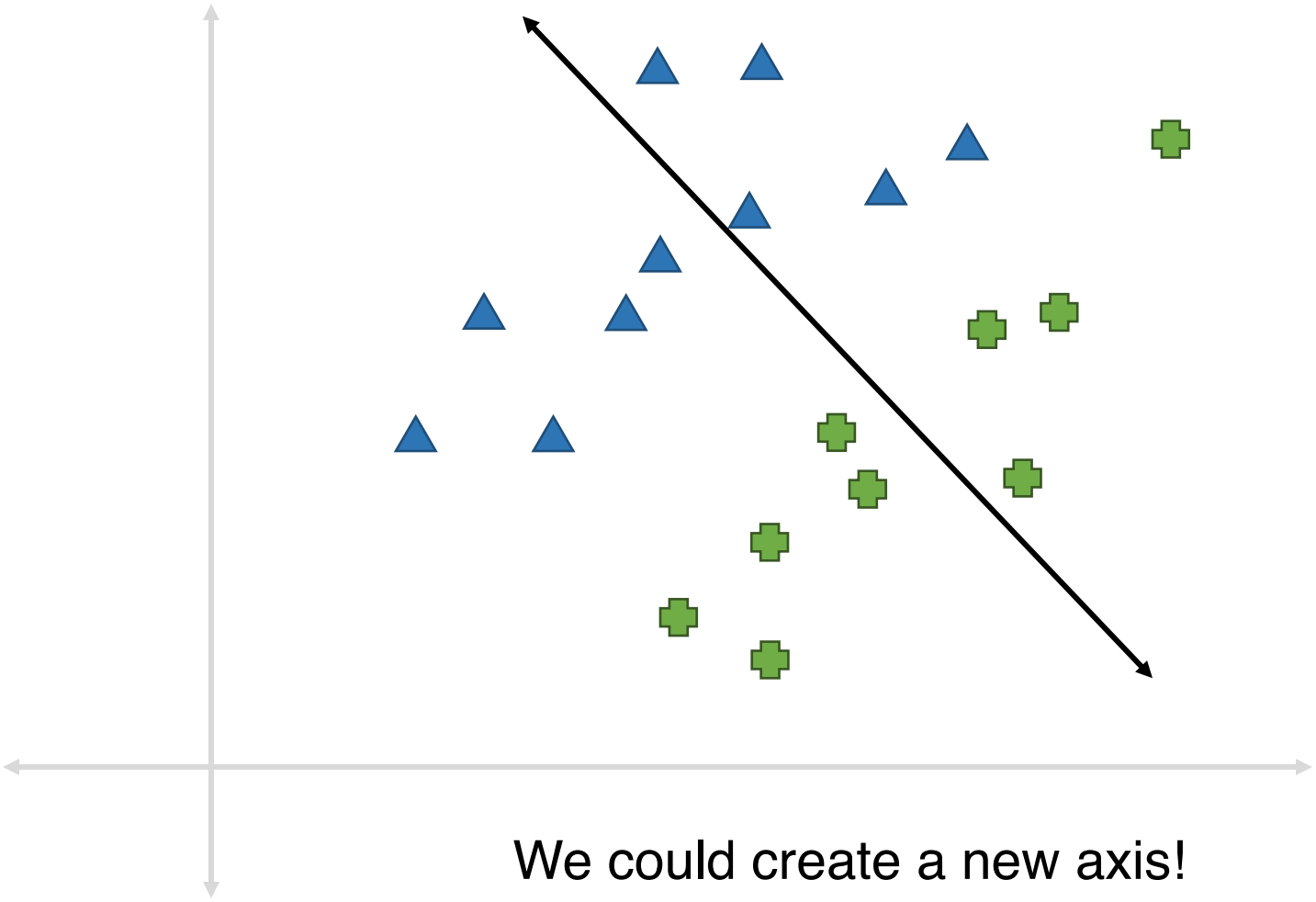
Project the points onto the y-axis

Suppose we choose the feature represented by the y-axis



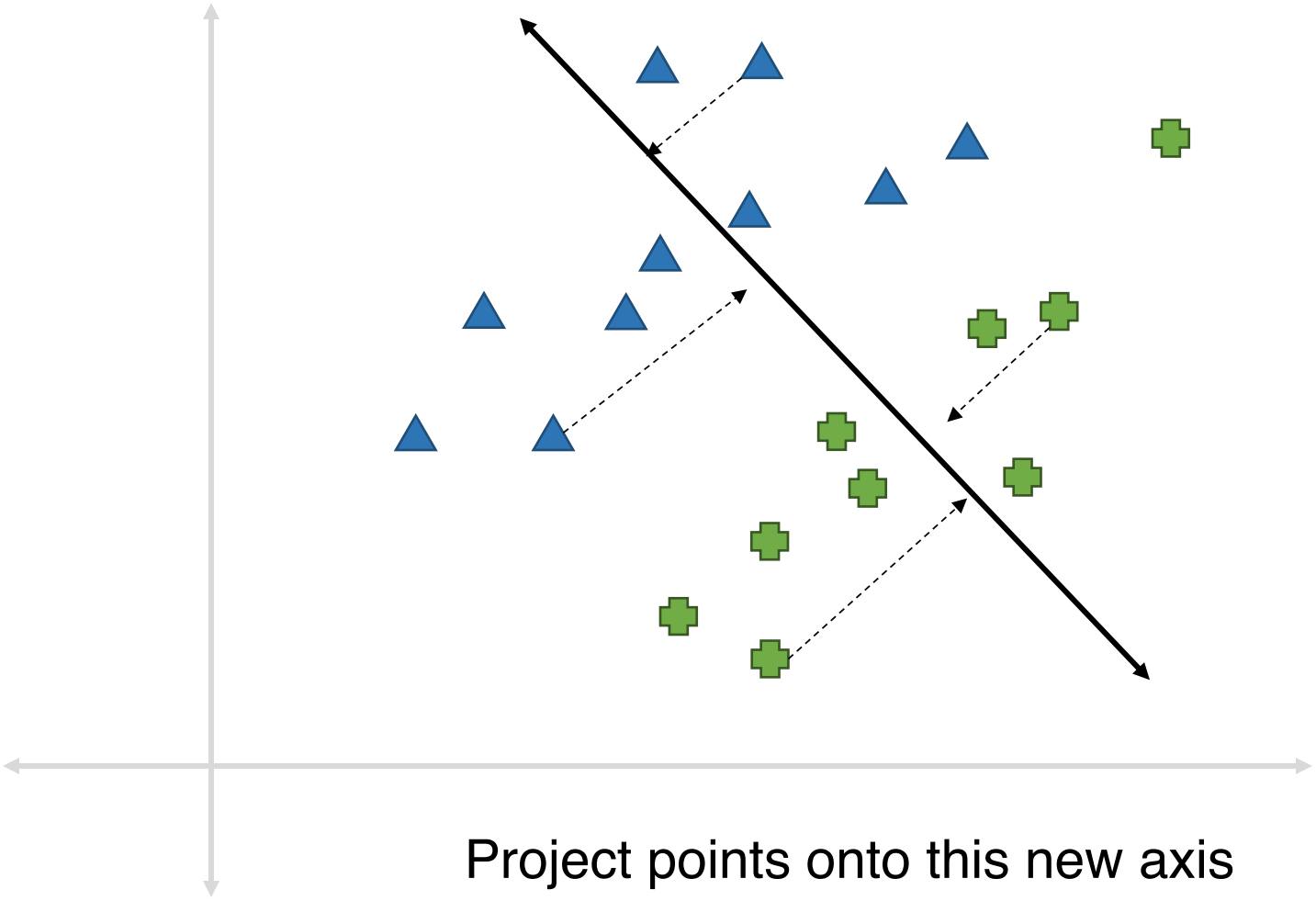
The positions along the y-axis now represent the feature values of each instance

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



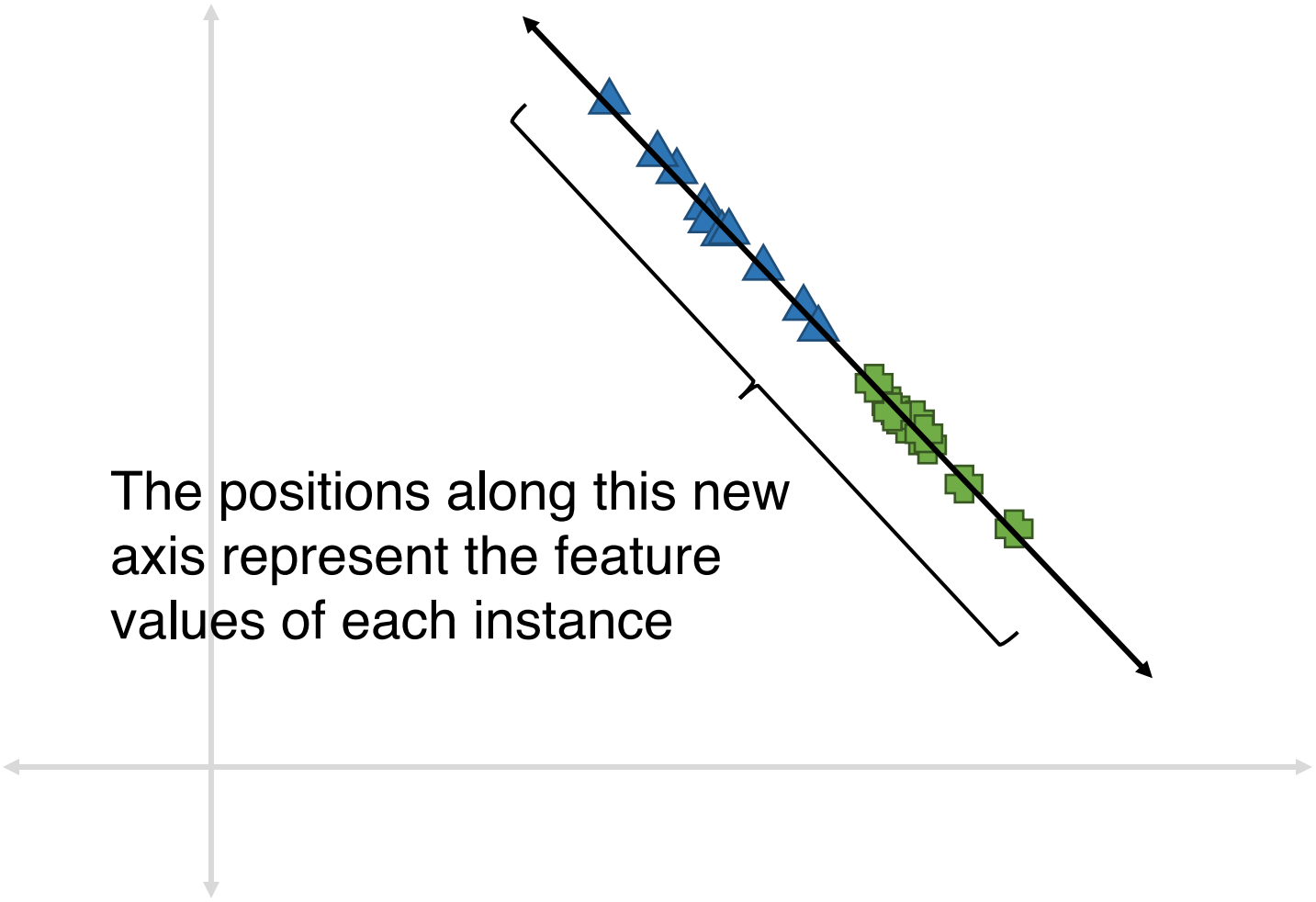
We could create a new axis!

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



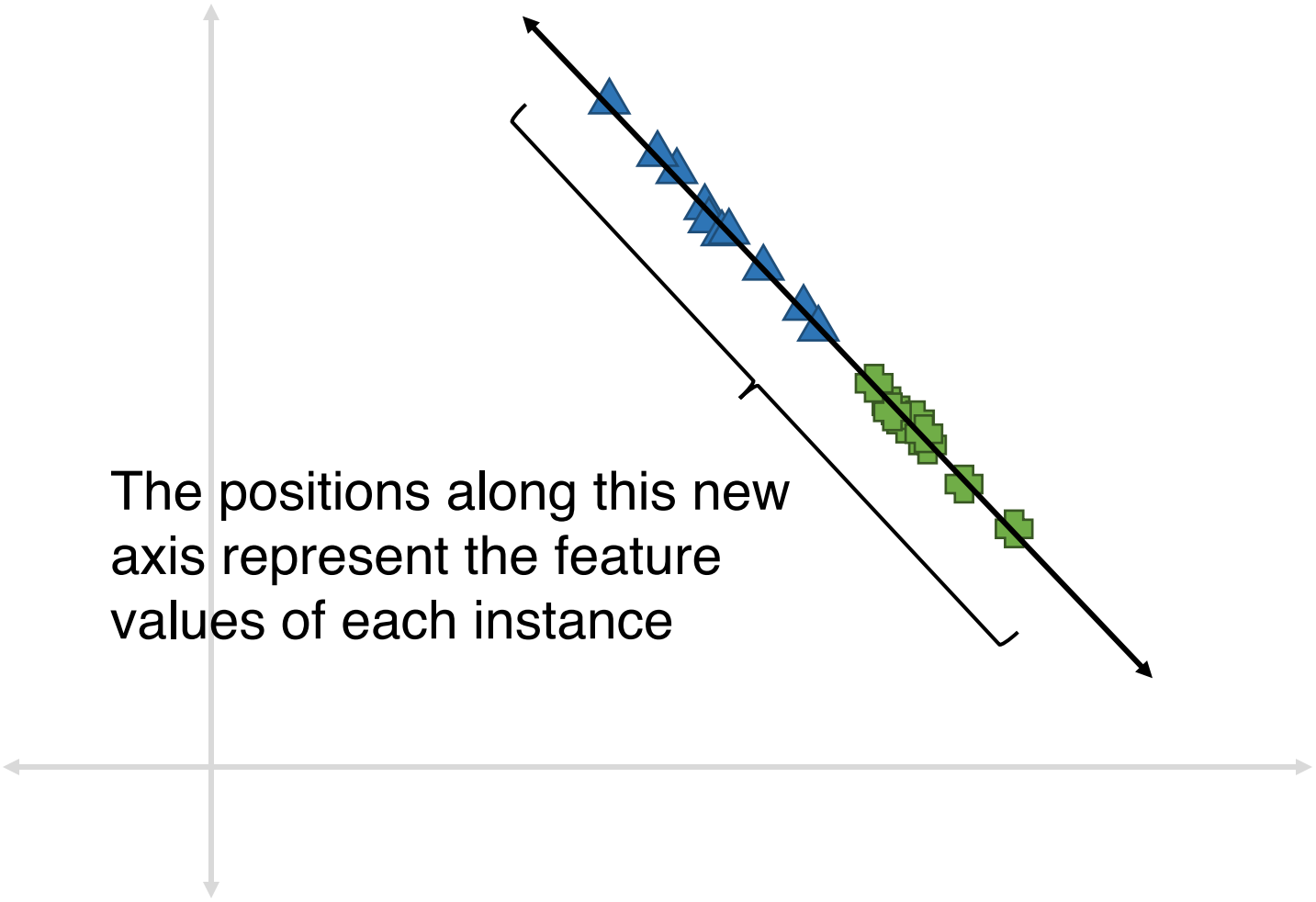
Project points onto this new axis

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



The positions along this new axis represent the feature values of each instance

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



The positions along this new axis represent the feature values of each instance

Dimensionality Reduction

In general:

- Original feature space has D dimensions (axes)
- New feature space has K dimensions (axes), $K < D$

The transformed feature vectors are sometimes called **embeddings**

Dimensionality Reduction

Why does dimensionality reduction work?

Why can it be automated?

Dimensionality Reduction

One intuition:

If multiple features are *correlated*, they can often be mapped to the same feature without losing a lot of information

- The blood pressure example applies here

Dimensionality Reduction

Another intuition:

It's possible to change the dimensionality so that instances that were similar to each other (or close to each other) in the original feature space are still similar to each other in the reduced space

- and instances that were previously dissimilar should be dissimilar in the new space

You might imagine automating this, trying to learn a reduction that minimizes the difference between the original similarities and new similarities

Dimensionality Reduction

Another intuition:

Think about dimensionality reduction as a *compression* problem (lossy compression)

- Want to compress the data as much as possible while retaining “most” information
- If you transform your feature vectors into new feature vectors with fewer dimensions, how well could you reconstruct the original vectors from the smaller vectors?

Dimensionality Reduction

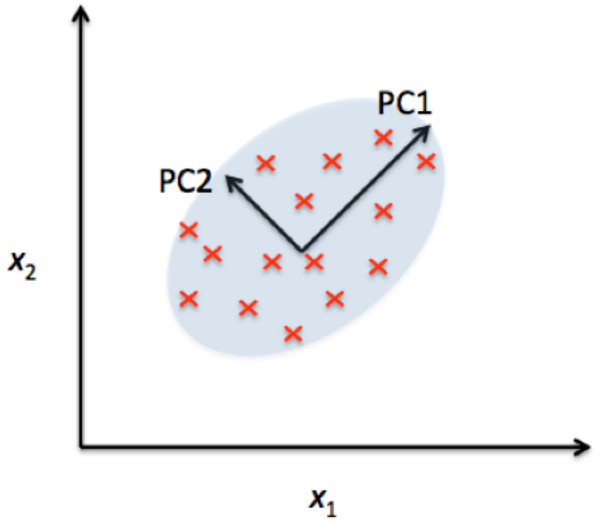
Most dimensionality reduction techniques work based on some of these intuitions (maybe not all of them, though they are related)

- If you don't know a particular technique, you can probably still assume it is taking advantage of these general principles

We'll look at two of the more common techniques today, but won't cover a lot of technical detail

Principle Component Analysis

Principle component analysis (**PCA**) is a widely used technique that chooses new axes to project the data onto



The new axes are called **principle components**

Principle Component Analysis

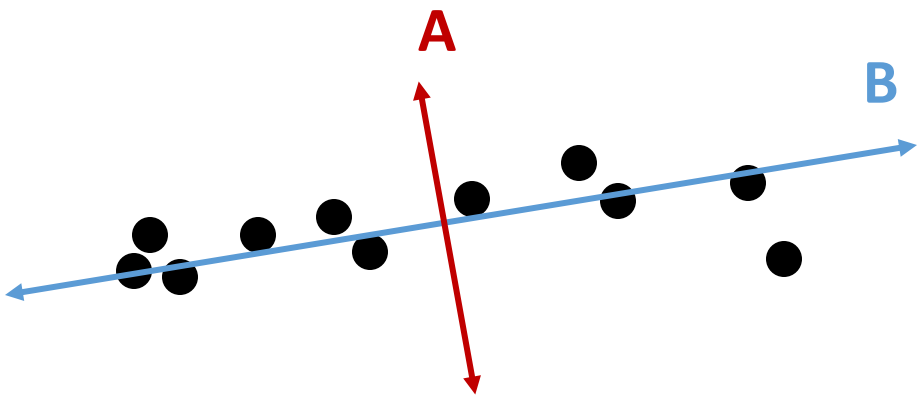
PCA does not use any information about the class labels

- *Unsupervised* dimensionality reduction
- This has advantages and disadvantages (we'll discuss later)

So how does PCA decide how to choose axes?

- Idea: pick an axis so that the values will have high *variance* once projected onto it

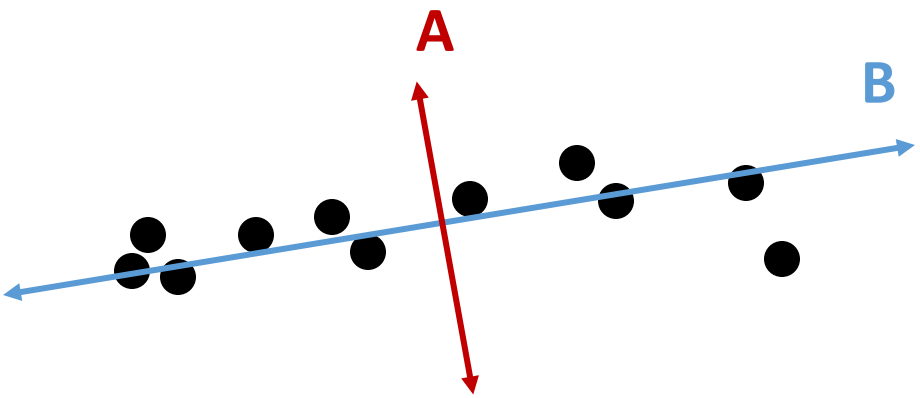
Principle Component Analysis

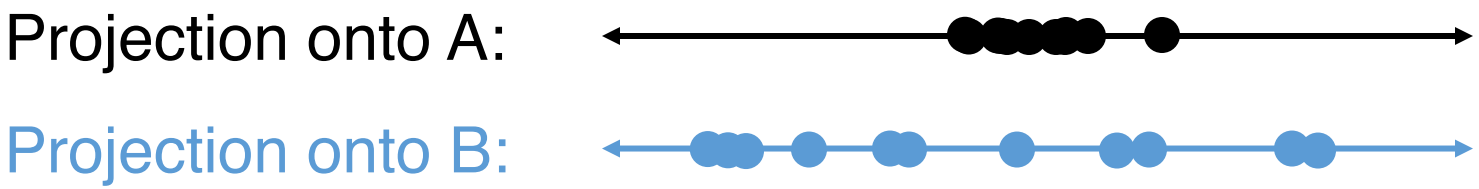


Projection onto A:

Projection onto B:

Principle Component Analysis





B yields higher variance (points are more spread out)

- More “informative”; separates the points better
- More likely to be useful for separating by class

Principle Component Analysis

Overview of PCA algorithm (details in book):

- Start by identifying the principle component (axis) with the highest variance
- Pick another principle component that is orthogonal (forms a right angle with) the previous component(s) with the next-highest variance
  - Why orthogonal? Don't want to just pick another nearly identical component, or it won't give you much information beyond what the other component is already providing
  - Point is to reduce redundancy
- Keep repeating until K principle components have been chosen

Principle Component Analysis

Overview of PCA algorithm (details in book):

- The algorithm will also learn a function that transforms an instance **x** into the projected space
- Then you can use the transformed instances in your prediction algorithm

Note: variance depends on the scale of the values

- For PCA to work, important that all features are on the same scale
- Perform normalization/standardization first

Principle Component Analysis

What is the dimensionality, K?

This is a hyperparameter you have to provide

- Common values are on the order of 100
- But you can tune this, like other hyperparameters
  - Next week

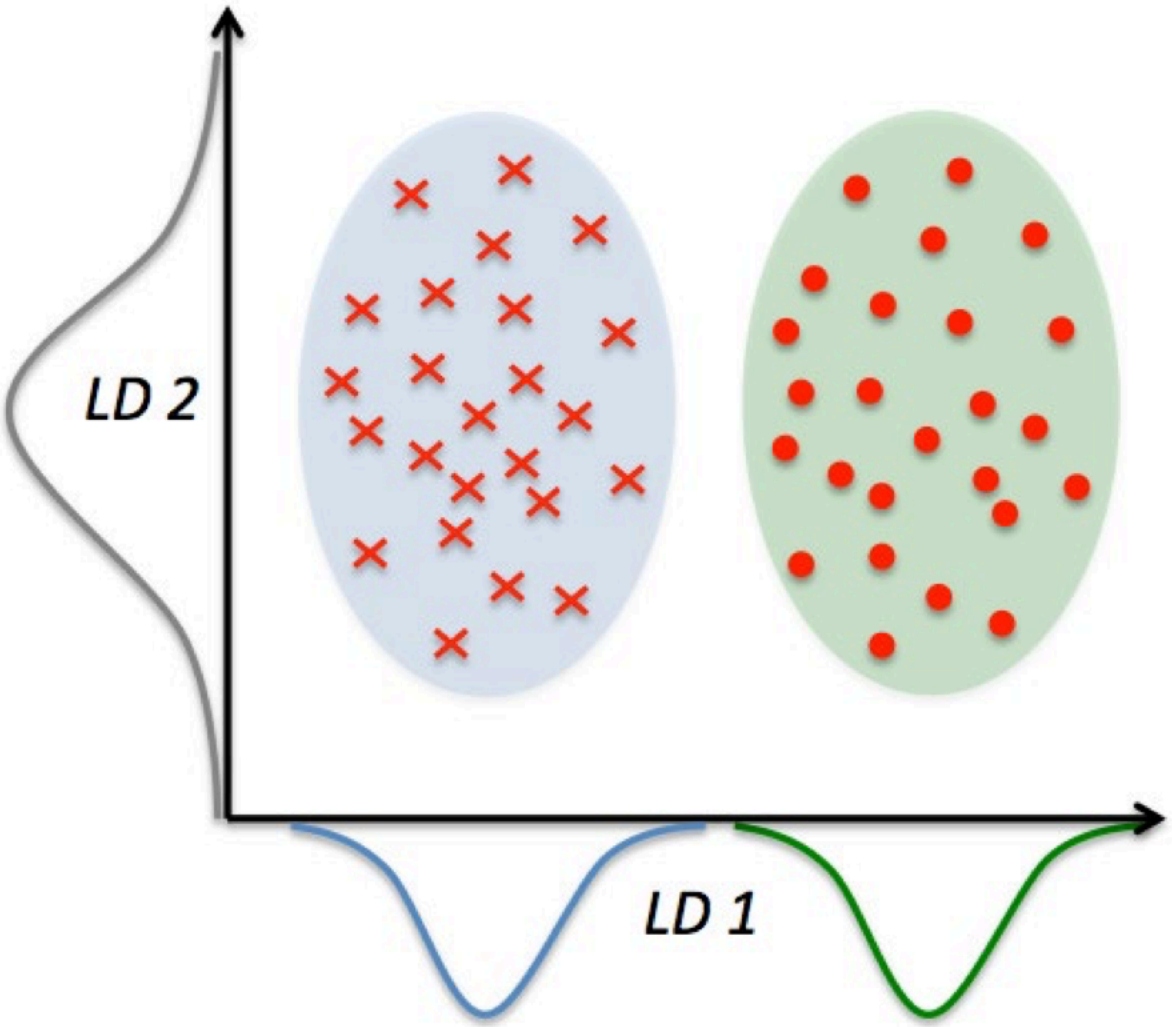
Linear Discriminant Analysis

Linear discriminant analysis (**LDA***) is another technique that works similarly to PCA

- Key difference: instead of choosing axes that have high variance, LDA chooses axes that best separate the class labels
- *Supervised* dimensionality reduction

* not to be confused with Latent Dirichlet Allocation (also abbreviated LDA), a topic modeling algorithm that is also sometimes used for dimensionality reduction

Linear Discriminant Analysis



Linear Discriminant Analysis

LDA uses a metric called *scatter* that measures how separated the class labels are along an axis

- See book for more detail (not needed in this class)

Similar idea to how decision trees choose features at each node

- Want features that will separate the classes

Supervised or Unsupervised?

Supervised reduction (LDA)

- Changes the feature space in a way that directly optimizes for the prediction task

Unsupervised reduction (PCA)

- Can take advantage of unlabeled data
- Potentially advantageous when you have a small amount of training data but a large amount of unlabeled data from the same domain

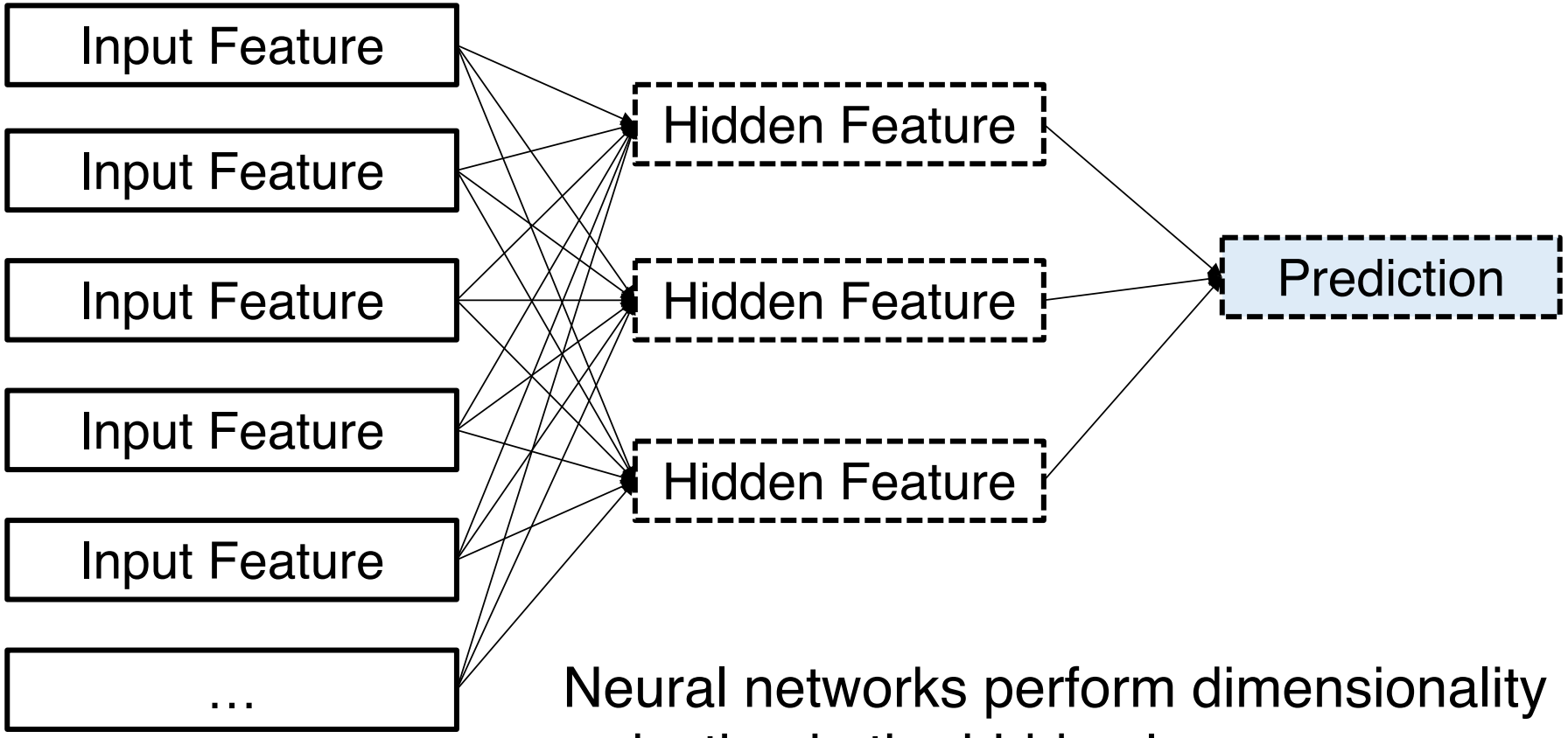
Supervised or Unsupervised?

How can unlabeled data help?

Example: Classifying news articles

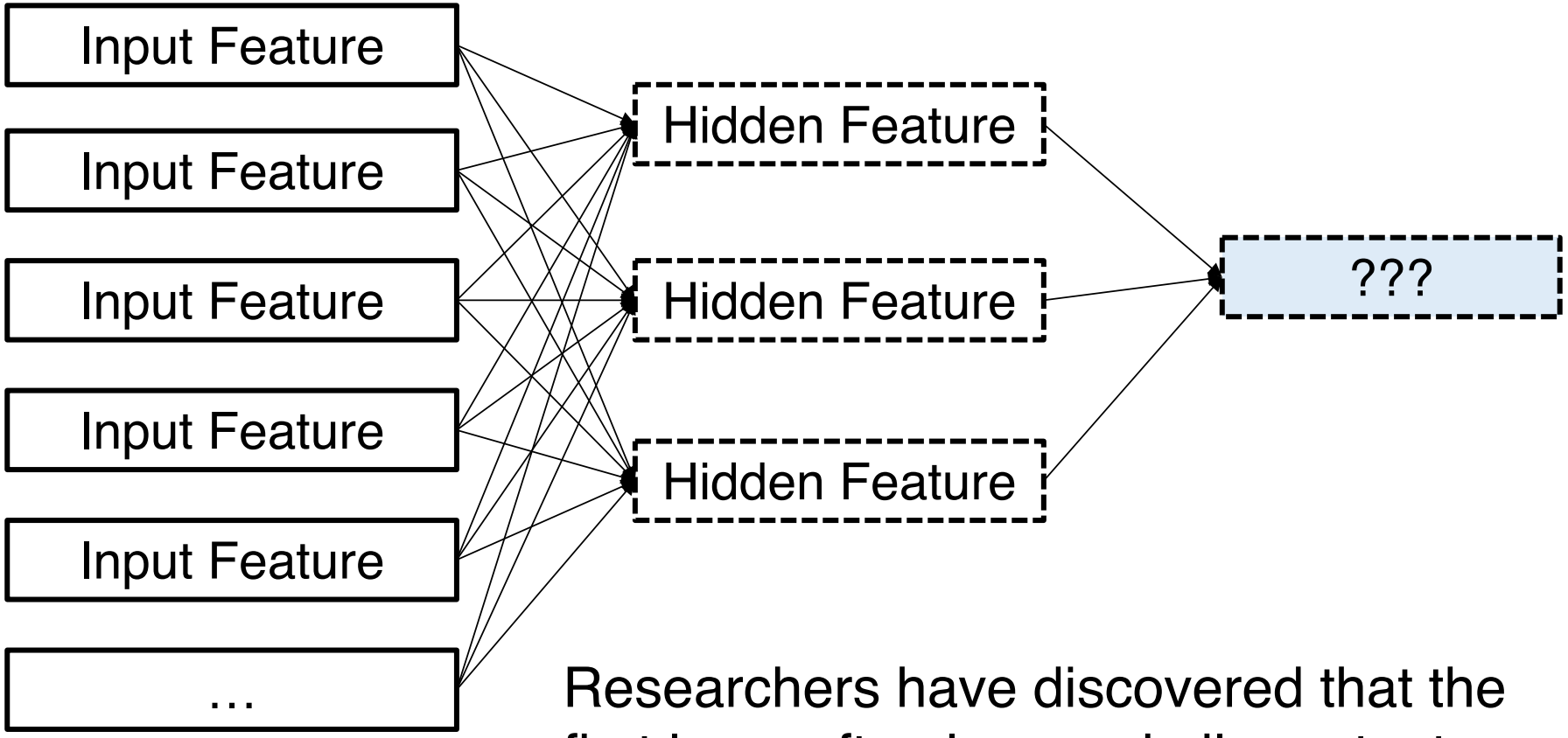
- Suppose you never see the word “iPad” in your training set
- But it occurs plenty of times in your entire collection of news articles (just not labeled)
- This word is probably correlated with other words like “iPhone”, “tablet”, “Apple”, etc.
- Most techniques will pick up on this correlation, and instances containing the word “iPad” will get transformed similarly to instances containing these other words

Revisiting Neural Networks



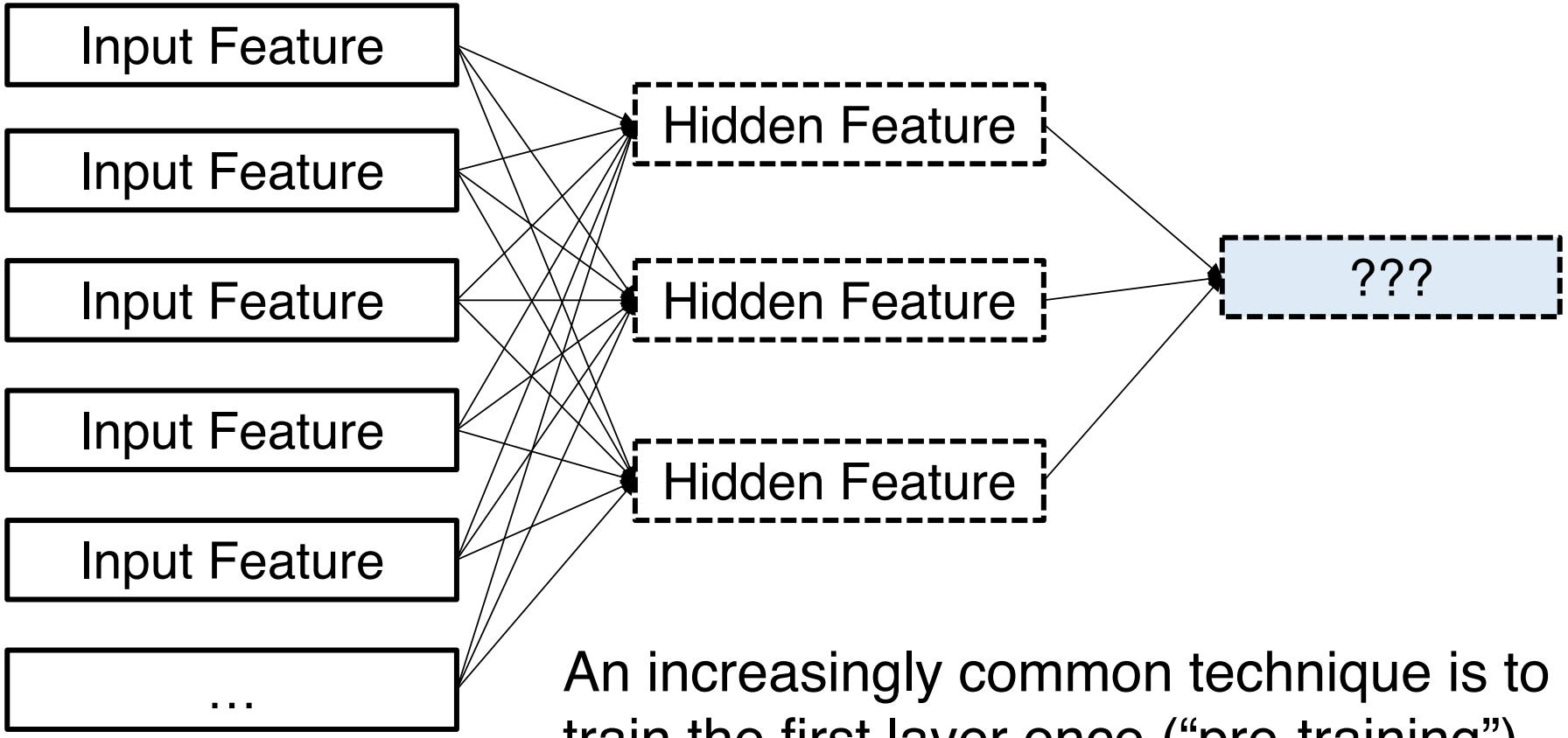
Neural networks perform dimensionality reduction in the hidden layers

Revisiting Neural Networks



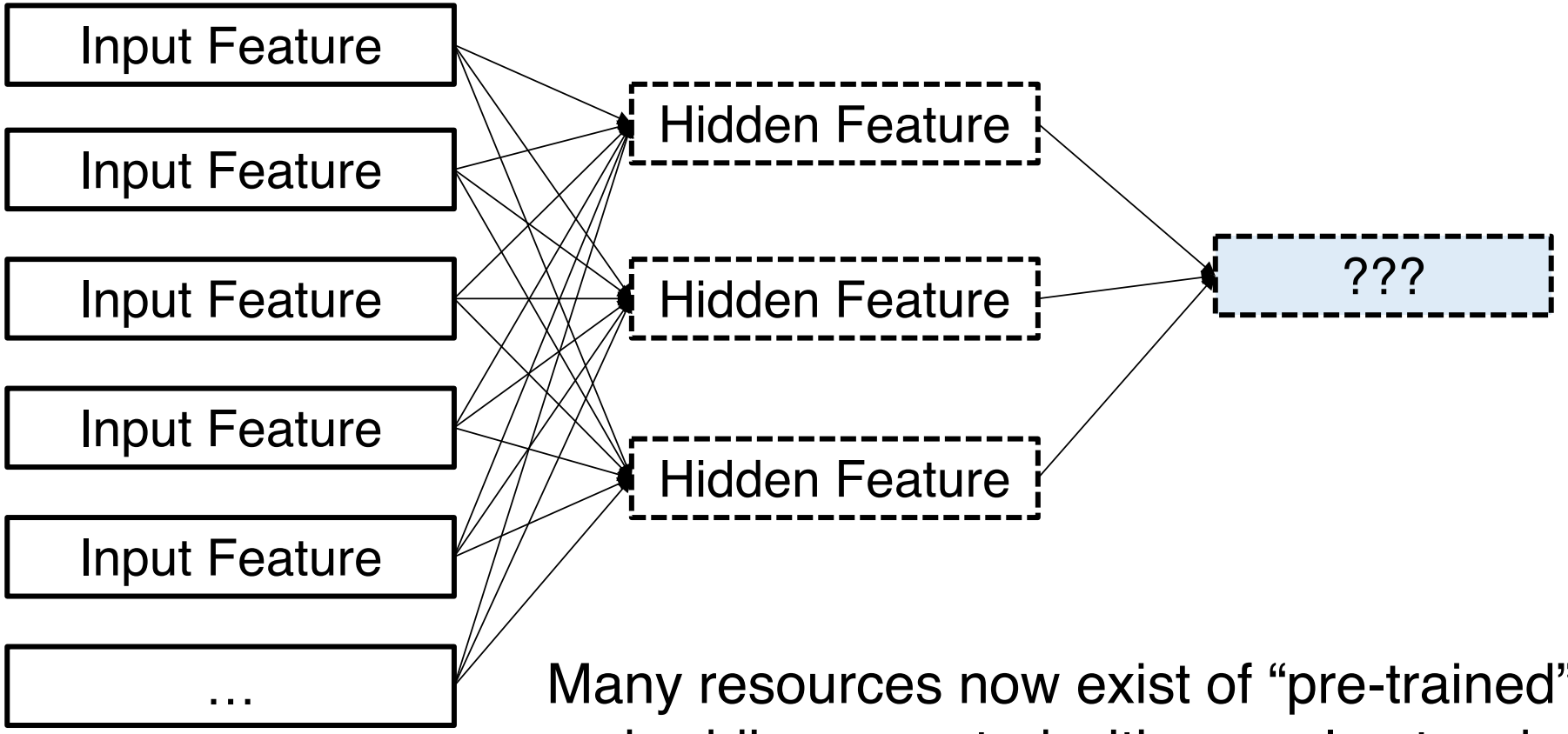
Researchers have discovered that the first layer often learns similar outputs even when the data and task change

Revisiting Neural Networks



An increasingly common technique is to train the first layer once (“pre-training”) and keep reusing it, doing most experimentation in the hidden layers

Revisiting Neural Networks



Many resources now exist of “pre-trained” embeddings created with neural networks that can be used for various problems