Data Foundations: Exam Review

Instructor: Anthony Rios

Outline

Review

The Exam

Basic Coding Concepts

Machine Learning (Features and Evaluation)

Data Annotation

Feature Selection/Transformation and Missing Data

Missing Data

Review

Any Questions?

Review

The Exam

Basic Coding Concepts

Machine Learning (Features and Evaluation)

Data Annotation

Feature Selection/Transformation and Missing Data

Missing Data

Exam

The Exam is available **NOW**. It is due next Tuesday!

- The exam has 20 questions
- There are two coding questions. They are slightly harder than Midterm 1. Come prepared
- There is one regex problem

  The rest of the questions focus on everything else covered that was taught after Exam 1
- The web scraping material is **NOT** on the exam.
- Most of the questions are multiple choice, some of the questions will be short answer, and there will be a few coding questions.
- Once you start the exam, you have **2 hours** to finish it.

Basic Coding Concepts and FileIO

- Basic data types and data structures (floats/ints, strings, lists, sets, dicts, ...)
- Conditional Statements (if, elif, else) and boolean expressions (and, not/!, in, or)
- Looping constructs (for, while), as well as how to use range().
- File IO (Text files, CSVs, XML, JSON, JSONL)

Object Oriented Programming

Object oriented programming is a way to arrange your code so that you can **zoom into 50 lines** of the code and **understand it** while **ignoring the other 999,950 lines** of code for the moment.

Methods vs Functions

A method is a **function** that is contained **within a class** and the objects that are constructed from the class.

An **object** contains both data and **methods** that manipulate that data.

- An object is active, not passive; it does things
- An object is responsible for its own data
  - But: it can expose that data to other objects

An object has state

An object contains both **data** and methods that manipulate that data

- The data represent the **state** of the object
- Data can also describe the relationships between this object and other objects
- Data is also known as “attributes”

Example: A “rabbit” object

Assume we want to create an object that represents a rabbit.

It would have **data**:

- How hungry it is
- How frightened it is
- Where it is

And **methods**:

- eat, hide, run, dig

Creating Objects in Python

example.py

```
class PartyAnimal:
    x = 0 # data/attribute
    def party(self): # Method
        self.x += 1
        print("So far", self.x)
```

Creating Objects in Python

example.py

```
class PartyAnimal:
        x = 0 # data/attribute
        def party(self): # Method
                self.x += 1
                print("So far", self.x) an = PartyAnimal()
an.party()
an.party()
an.party()
```

anthony@MacBook:~$ python example.py

```
So far 1
So far 2
So far 3
```

Object life-cycle

example.py

```
class PartyAnimal:
    x = 0
    def __init__(self): # Called when object is created/initialized
        print('I am constructed')
    def party(self):
        self.x = self.x + 1
        print('So far',self.x)
    def __del__(self): # Called when object is destroyed
        print('I am destructed', self.x)
an = PartyAnimal()
an.party()
an.party()
an = 42
print('an contains',an)
```

Object life-cycle

example.py

```
...
an = PartyAnimal()
an.party()
an.party()
an = 42
print('an contains',an)
```

anthony@MacBook:~$ python example.py

```
I am constructed
So far 1
So far 2
I am destructed 2
an contains 42
```

Inheritance

example.py

```
from party import PartyAnimal
class CricketFan(PartyAnimal):
        points = 0
        def six(self):
                self.points = self.points + 6
                self.party()
                print(self.name,"points",self.points)
s = PartyAnimal("Sally")
s.party()
j = CricketFan("Jim")
j.party()
j.six()
```

Inheritance

example.py

```
s = PartyAnimal("Sally")
s.party()
j = CricketFan("Jim")
j.party()
j.six()
```

anthony@MacBook:~$ python example.py

```
Sally constructed
Sally party count 1
Jim constructed
Jim party count 1
Jim party count 2
Jim points 6
```

Machine Learning

Main Topics:

- Data Annotation
- Feature Engineering
- Model Selection and Evaluation

Data Objects and Attribute Types

A **data object** represents an entity. Examples include...

- Customers
- Students/Professors/Courses
- Tweets
- Images
- Genes, Drugs, Procedures

A **attribute** is a data field (synonyms dimension/feature/variable)

Types of Attributes

`https://www.youtube.com/watch?v=N9fDIAflCMY`

- **Nominal** - hair_color (black, blonde, red, green?), martial_status (single married, divorced, widowed,...), occupation (teacher, dentist, programmer,...)
- **Binary (Boolean)** - smoker (yes or no), medical tests (positive or negative)
- **Ordinal** - drink_size (short, tall, grande, venti), grade (A, B, C, D, F), professional_rank (assistant, associate, full)
- **Numeric** - temperature (70°F), speed (400 mph)

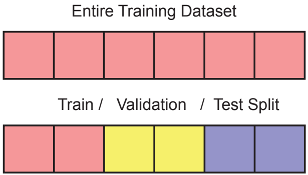
Model Evaluation

Model Selection: estimate the performance of different models in order to choose the best one.

Model Assessment: having chosen a final model, estimate its prediction error on new data.

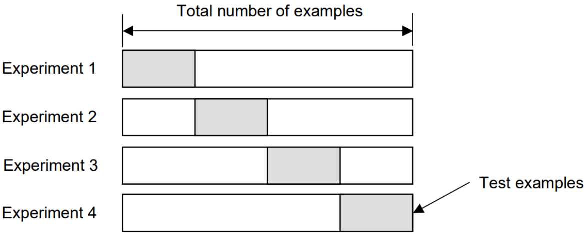
Model Selection Assessment

- Three-way data splits
  - **Training set**: A set of examples for learning
  - **Validation Set**: A set of examples used to tune the parameters/model selection
  - **Test set**: A set of examples **only** to assess the performance of the fully trained model.
    - After assessing the final model with the test set, **YOU MUST NOT** further tune your model.



K-Fold Cross-Validation

- Create K-fold partition of the dataset
  - For each of the K experiments, use K-1 folds for training and the remaining one for testing.



- The final error (evaluation measure) can be estimated as

$$E = \frac{1}{K} \sum_{i=1}^{K} E_i$$

How many folds are needed?

- With a large number of folds
  - $+$ The bias of true error rate is small
  - - Variance of true error rate will be large
  - - Large computational time (many experiments)
- With a small number of folds
  - $+$ Computation time reduced
  - $+$ Variance of estimator will be small
  - - The bias of estimator will be large (Conservative or higher than true error rate)
- In practice, the choice of the number of folds depends on the size of the dataset
  - For large datasets, 3-fold CV will be quite accurate
  - For very small datasets, we may have to use leave-one-out in order to train on as many examples as possible
- A common choice for K-Fold CV is K=10

Review

The Exam

Basic Coding Concepts

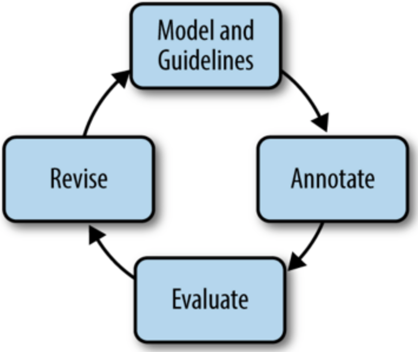
Machine Learning (Features and Evaluation)

Data Annotation

Feature Selection/Transformation and Missing Data

Missing Data

Annotation Pipeline



Pustejovsky and Stubbs (2012), Natural Language Annotation for Machine Learning

Annotation Process

1. Determine what to annotate.
2. Formalize the instructions for the annotation task
3. Perform a pilot annotation
4. Annotate the data
5. Compute and report inter-annotator agreement, and release the data.

Annotation Guidelines

Our goal: Given the constraints of our problem, how can we formalize our descriptions of the annotation process **to encourage multiple annotators to provide the same judgment?**

Annotation Guidelines

- What is the goal of the project?
- What is each class called and how is it used? (Be specific: provide examples and discuss gray areas)
- What **exactly** should be annotated and what should be left alone?

Pustejovsky and Stubbs (2012), Natural Language Annotation for Machine Learning

Practicalities

- Annotation takes time/concentration (can't do it 8 hours a day)

- Annotators get better as they annotate (earlier annotations not as good as later ones)

Why not do it yourself?

- Expensive/time-consuming
- Multiple people provide a measure of consistency: is the task well enough defined?
- Low agreement = not enough training, guidelines not well enough defined, task is bad.

Adjudication

- Adjudication is the process of deciding on a single annotation for a piece of text, using information about the **independent annotations**.
- Can be **time-consuming** (or more so) as primary annotation.
- Does **NOT** need to be identical with the primary annotation. (both annotators can be wrong by chance)

Fleiss' kappa

- Same fundamental idea of measuring the observed agreement compared to the agreement we would expect by chance.

$$\kappa = \frac{P_o - P_e}{1 - P_e}$$

- With N $>$ 2, we calculate agreement among **pairs** of annotators.

Fleiss' Kappa

n_{ij} is the number of annotators that agree on assigning the i-th class to the j-th item.

o is the total number of annotators

K is the number of classes

For item i with n annotations, how many annotators agree, among all n(n-1) possible pairs.

$$P_i = \frac{1}{o(o-1)} \sum_{j=1}^{K} n_{ij}(n_{ij} - 1)$$

Fleiss' Kappa

[redacted]	Positive	Negative	Neutral	P_i
Tweet 1	3	1	6	**0.4**
Tweet 2	9	1	0	
Tweet 3	3	5	2	
Tweet 4	2	0	8	
p_j				[redacted]

$$P_1 = \frac{1}{10(10-1)} * (3*2 + 1*0 + 6*5) = \mathbf{0.4}$$

Fleiss' Kappa

[redacted]	Positive	Negative	Neutral	P_i
Tweet 1	3	1	6	0.4
Tweet 2	9	1	0	0.8
Tweet 3	3	5	2	0.3111
Tweet 4	2	0	8	**0.6444**
p_j				[redacted]

$$P_4 = \frac{1}{10(10-1)} * (2 * 1 + 0 * -1 + 8 * 7) = \mathbf{0.6444}$$

Fleiss' Kappa

[redacted]	Positive	Negative	Neutral	P_i
Tweet 1	3	1	6	0.4
Tweet 2	9	1	0	0.8
Tweet 3	3	5	2	0.3111
Tweet 4	2	0	8	0.6444
p_j				[redacted]

N is the total number of items (Total Tweets in this example)

Average observed agreement among all items

$$P_o = \frac{1}{N}\sum_{i=1}^{N} P_i = \frac{1}{4} * (0.4 + 0.8 + 0.3111 + 0.6444) = 0.5388$$

Fleiss' Kappa

[redacted]	Positive	Negative	Neutral	P_i
Tweet 1	3	1	6	0.4
Tweet 2	9	1	0	0.8
Tweet 3	3	5	2	0.3111
Tweet 4	2	0	8	0.6444
p_j	**0.425**			[redacted]

N is the total number of items (Total Tweets in this example)
o is the total number of annotators
Probability of category j

$$p_j = \frac{1}{N * o} \sum_{i=1}^{N} n_{ij}$$

$$p_{positive} = \frac{1}{4 * 10} * (3 + 9 + 3 + 2) = \mathbf{0.425}$$

Fleiss' Kappa

[redacted]	Positive	Negative	Neutral	P_i
Tweet 1	3	1	6	0.4
Tweet 2	9	1	0	0.8
Tweet 3	3	5	2	0.3111
Tweet 4	2	0	8	0.6444
p_j	0.425	0.175	**0.4**	[redacted]

N is the total number of items (Total Tweets in this example)
o is the total number of annotators
Probability of category j

$$p_j = \frac{1}{N * o} \sum_{i=1}^{N} n_{ij}$$

$$p_{neutral} = \frac{1}{4 * 10} * (6 + 0 + 2 + 8) = \mathbf{0.4}$$

Fleiss' Kappa

[redacted]	Positive	Negative	Neutral	P_i
Tweet 1	3	1	6	0.4
Tweet 2	9	1	0	0.8
Tweet 3	3	5	2	0.3111
Tweet 4	2	0	8	0.6444
p_j	0.425	0.175	0.4	[redacted]

Expected agreement by chance – joint probability two raters pick the same label is the product of their independent probabilities of picking that label
K is the number of classes

$$P_e = \sum_{j=1}^{K} p_j * p_j = 0.425 * 0.425 + 0.175 * 0.175 + 0.4 * 0.4 = \mathbf{0.3715}$$

Fleiss' kappa

- Same fundamental idea of measuring the observed agreement compared to the agreement we would expect by chance.

$$\kappa = \frac{P_o - P_e}{1 - P_e} = \frac{0.5388 - 0.3715}{1 - 0.3715} = \mathbf{0.2662}$$

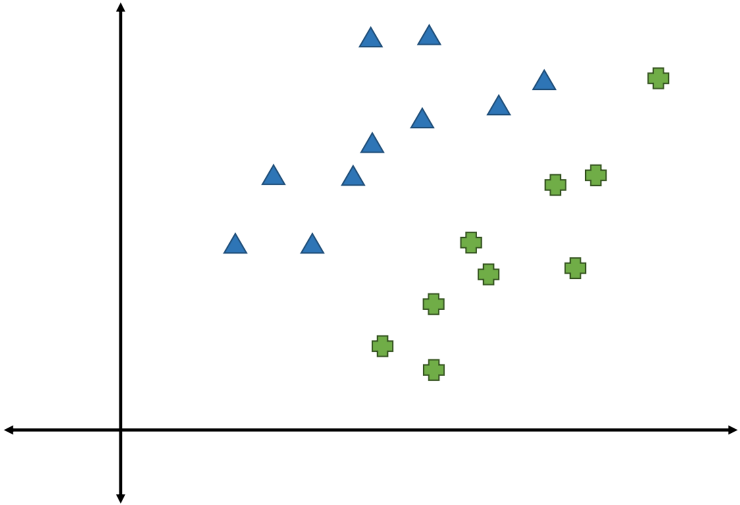
Fleiss' Kappa

"Good" values are subject to interpretation, but rule of thumb

Score Range	Interpretation
0.81 - 1.00	Almost Perfect
0.61 - 0.80	Substantial agreement
0.41 - 0.60	Moderate agreement
0.21 - 0.40	Fair agreement
0.01 - 0.20	Slight agreement
< 0.0	Poor agreement

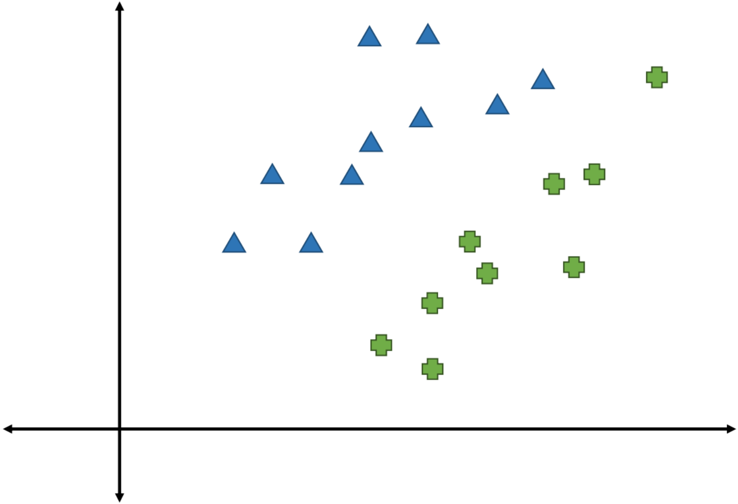
Geometric Intuition of Feature Selection and PCA

Suppose we have two dimensions (two features)



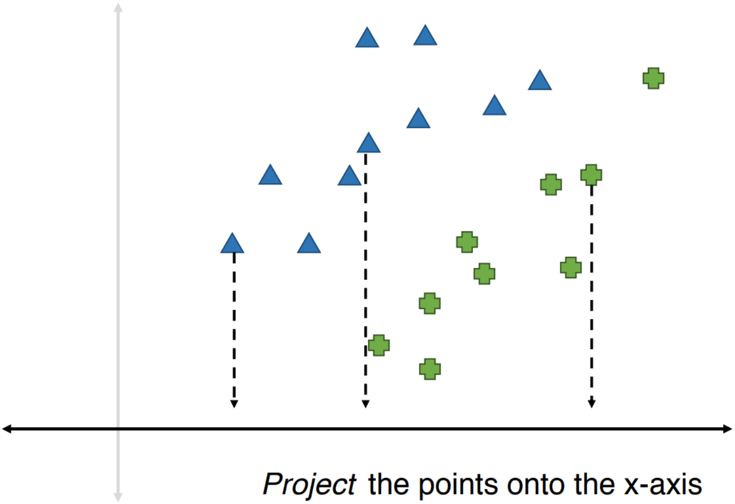
Geometric Intuition: Feature Selection

Feature selection: choose one of the two features to keep



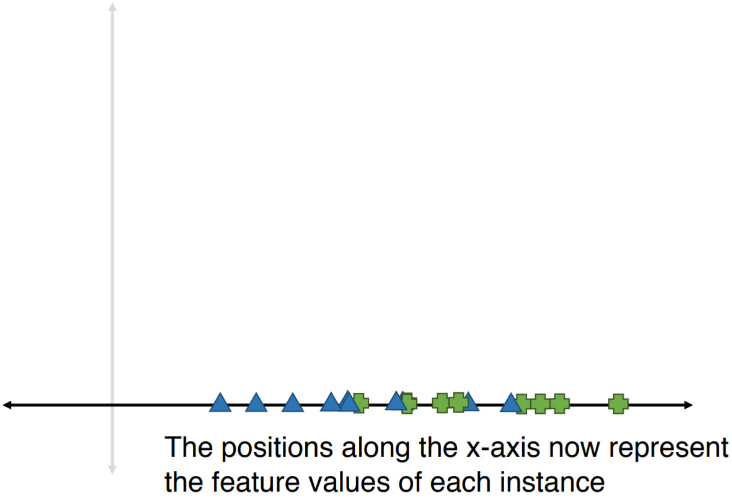
Geometric Intuition: Feature Selection

Suppose we choose the feature represented by the x-axis



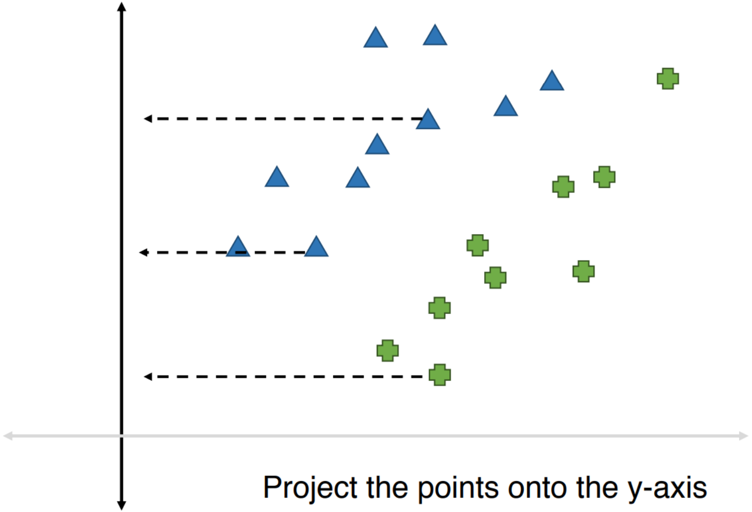
Geometric Intuition: Feature Selection

Suppose we chose the feature represented by the x-axis



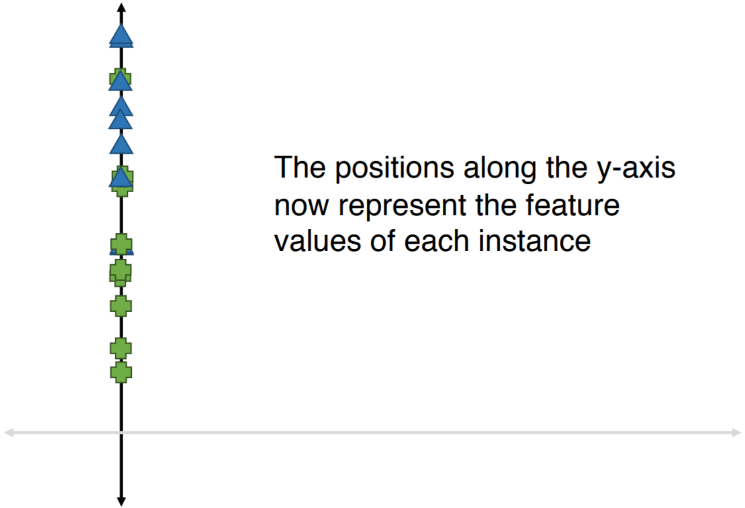
Geometric Intuition: Feature Selection

Suppose we choose the feature represented by the y-axis



Geometric Intuition: Feature Selection

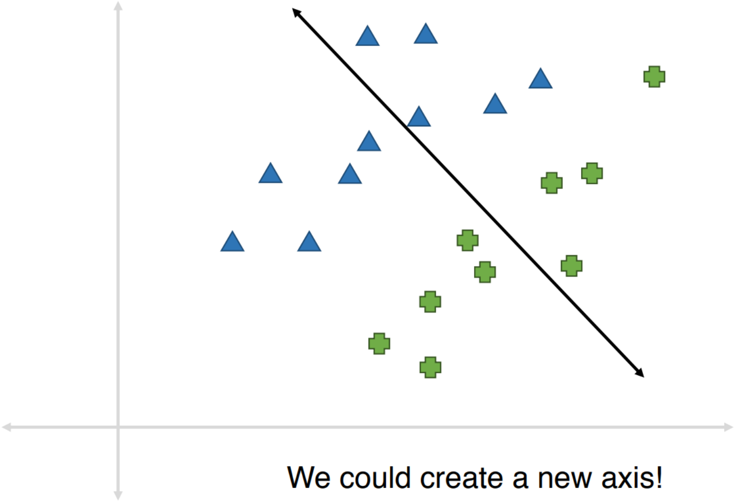
Suppose we choose the feature represented by the y-axis



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

Geometric Intuition: Feature Transformation

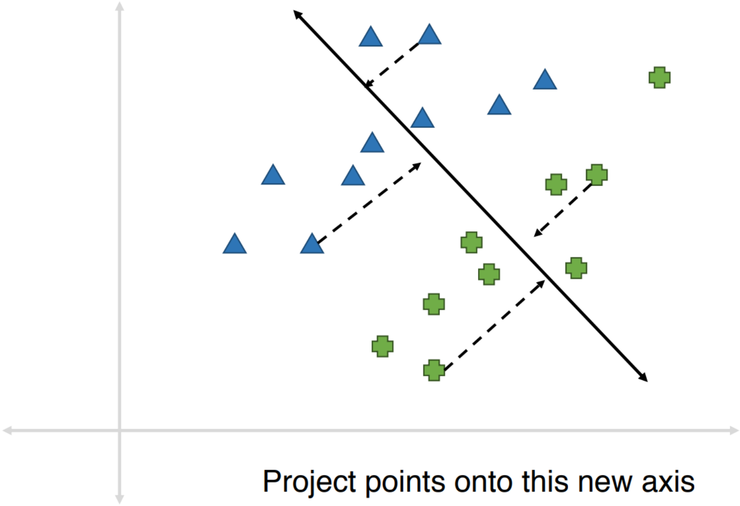
We don't have to restrict ourselves to picking either the x-axis or the y-axis



We could create a new axis!

Geometric Intuition: Feature Transformation

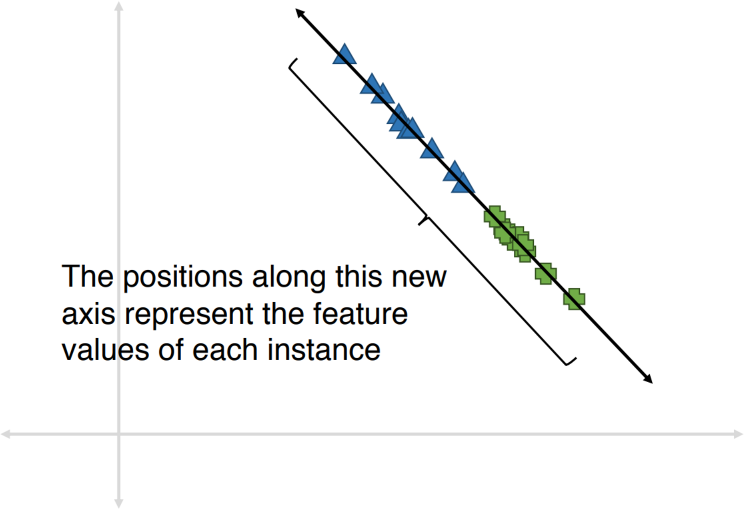
We don't have to restrict ourselves to picking either the x-axis or the y-axis



Project points onto this new axis

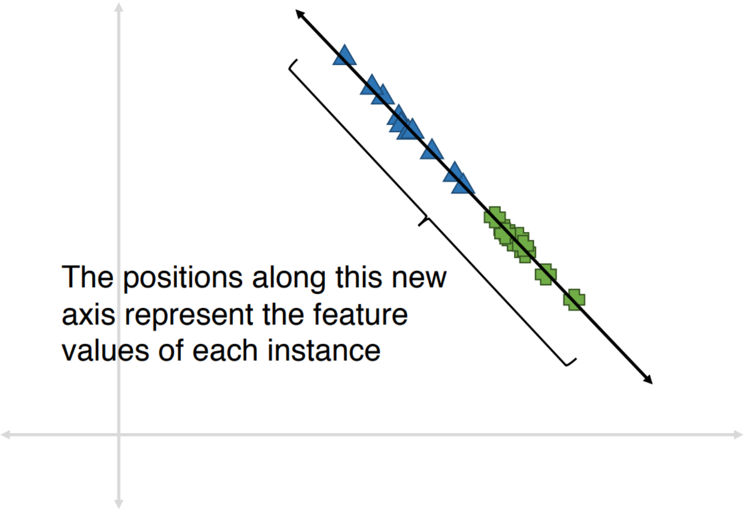
Geometric Intuition: Feature Transformation

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



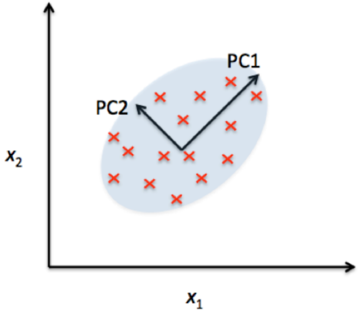
Geometric Intuition: Feature Transformation

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



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 the **principle components**

Principle Component Analysis

PCA does not use any information about the class labels

- **Unsupervised** dimensionality reduction
- This has advantages and disadvantages

So, how does PCA decide how to choose axes?

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

Types of Missing Data

- Missing Completely at Random (MCAR)
  - P(missing) is unrelated to the process under study
- Missing at Random (MAR)
  - P(missing) depends only on **observed data**
- Missing Not at Random (MNAR)
  - P(missing) depends on both observed and **unobserved data**.

Type type of missing data drastically affects what we can ultimately do to compensate for missing-ness

Complete Case Analysis

Delete all rows with **any missing values** at all, so you are left only with observations where all variables are observed.

This is the **easiest way to handle missing data**. In some cases it will work fine; **in others, ????**

- Loss of sample will lead to variance larger than reflected by the size of your data

- May bias your sample

Missing Value Skipping: Pros and Cons

Pros

- Easy to understand and implement
- Can be applied to any model (decision trees, logistic regression, linear regression, ...)

Cons

- Removing data points and features may remove important information from the data
- Unclear when it's better to remove data points versus features
- Doesn't help if data is missing at test time

Mode

Credit	Term	Income	y
Excellent	3 yrs	High	safe
Fair	?	Low	Risky
Fair	3 yrs	High	Safe
Poor	5 yrs	High	Risky
Excellent	3 yrs	Low	Risky
Fair	5 yrs	High	Safe
Poor	3 yrs	Low	Risky
Poor	?	Low	Safe
Fair	?	High	Safe

Credit	Term	Income	y
Excellent	3 yrs	High	safe
Fair	3 yrs	Low	Risky
Fair	3 yrs	High	Safe
Poor	5 yrs	High	Risky
Excellent	3 yrs	Low	Risky
Fair	5 yrs	High	Safe
Poor	3 yrs	Low	Risky
Poor	3 yrs	Low	Safe
Fair	3 yrs	High	Safe

Impute each feature with missing values:

- **Categorical features use mode**: Most popular value (mode) of non-missing x_i
- **Numerical features use average or median**: Average or median value of non-missing x_i

There are other methods, e.g., expectation-maximization algorithm, regressing/classifying on missing columns

Missing Value Imputation: Pros and Cons

Pros

- Easy to understand and implement
- Can be applied to any model (decision trees, logistic regression, linear regression, ...)
- Can be used at prediction time using the same imputation rules

Cons

- May result in systematic errors
  - Example: Feature "age" missing in all banks in Washington by state law

The End

Good Luck! Email me if you have any questions or concerns.