# Will you like it?

Training a classifier with only one class

## Introduction

**Subset of restaurants I have visited in Boston:**

Panda Express, Gyu-kaku, Pikaichi, Brown Sugar Cafe

**Can you predict which I have also visited and which I am not interested in?**

**-** Chipotle, Uberger, Nud Pob Thai Cusine, McDonald’s, Raising Cane’s

## Introduction

**Subset of restaurants I have visited in Boston:**

Panda Express, Gyu-kaku, Pikaichi, Brown Sugar Café

**Can you predict which I have also visited and which I am not interested in?**

- Chipotle, ~~Uberger~~, ~~Nud Pob Thai Cuisine~~, McDonald’s, Raising Cane’s
- On average, it’s a 50/50 guess

## Introduction

- How to predict whether I will try a new restaurant based on my favorites?
  - Similarity between restaurants
    - How close is close?
    - Where's the boundary?

## Introduction

- How to predict whether I will try a new restaurant based on my favorites?
  - Similarity between restaurants
    - How close is close?
    - Where's the boundary?
  - How to train such a predictor?
    - Basic classifiers don't work with single class

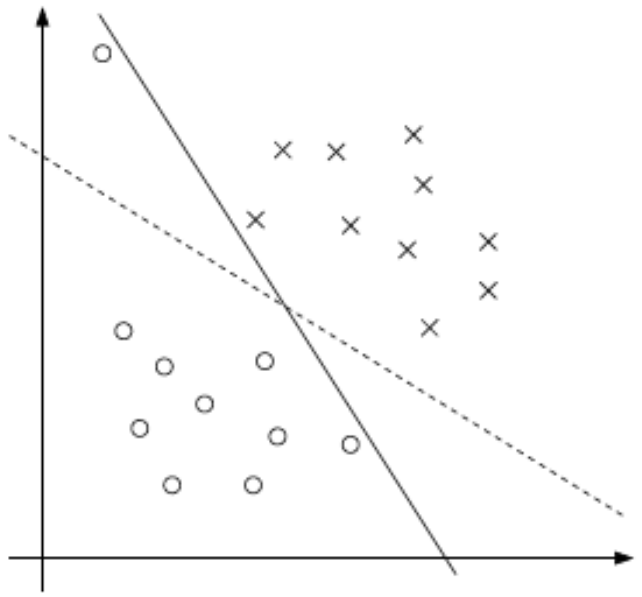
# SVM revisit

**Support Vector Machine** – a regularized large margin classifier implemented with kernel trick.

$$\min_{w,b,\xi_i} \frac{1}{2}\|w\|^2 + \underbrace{C\sum_{i=1}^{m}\xi_i}_{regularization}$$

$$y^{(i)}(w^T\phi(x^{(i)})+b) \geq 1-\xi_i, i=1,\ldots,m$$

$$\xi_i \geq 0, i=1,\ldots,m$$



C: hyperparameter, cost of misclassification.
Determined by cross-validation.

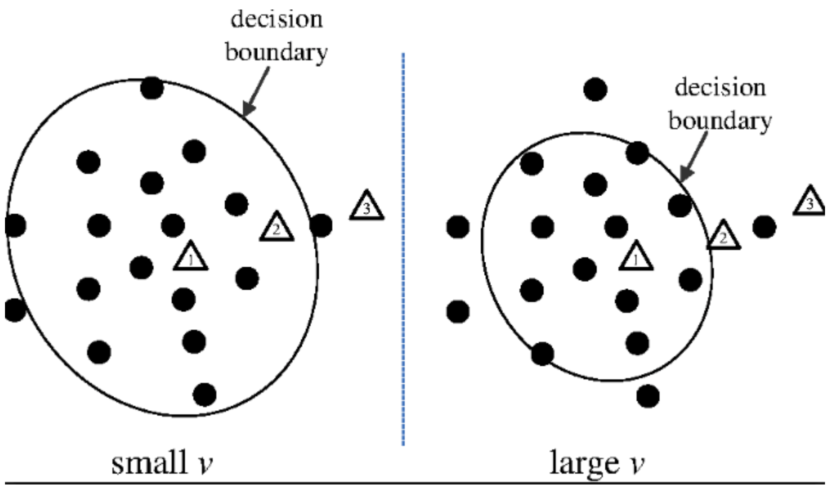
# One-class SVM, an outlier detector

- When finding negative class is costly, but finding positive class is cheap

$$\min_{w,\xi_i,\rho} \frac{1}{2}\|w\|^2 + \underbrace{\frac{1}{\nu n}\sum_{i=1}^{m}\xi_i - \rho}_{regularization}$$

$$y^{(i)}(w^T\phi(x^{(i)})+b) \geq \rho - \xi_i, i = 1,\ldots,m$$

$$\xi_i \geq 0, i = 1,\ldots,m$$



v: hyperparameter, (0,1], similar to "C", controls the radius of the circle. Determined by cross-validation.

## One-class SVM with “pure” positive data

- Training data: $x_{train}$, $y_{train} = 1$
- Testing data: $X_{train}$, $y_{train} = 1$
- Best One-class SVM solution:
  - Set ν -> 0, i.e. make a huge circle, assign $X_{train} = 1$
  - 100% accuracy on testing set!

## One-class SVM with “pure” positive data

- Training data: $x_{train}$, $y_{train} = 1$
- Testing data: $X_{train}$, $y_{train} = 1$
- Best One-class SVM solution:
  - Set ν -> 0, i.e. make a huge circle, assign $X_{train} = 1$
  - 100% accuracy on testing set. **Meaningless!!!**

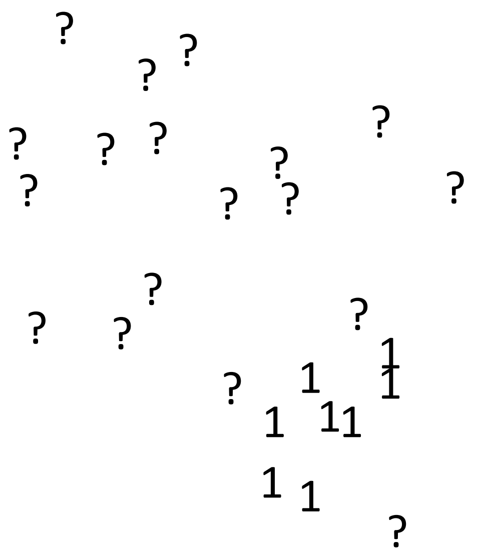
## One-class SVM with “pure” positive data

- Training data: $x_{train}$, $y_{train}$ = 1
- Testing data: $X_{train}$, $y_{train}$ = 1
- Best One-class SVM solution:
  - Set ν = 1, i.e. make a huge circle, assign $X_{train}$= 1
  - 100% accuracy on testing set. **Meaningless!!!**
  - Make ν bigger? How big?
    - We don’t know because cross-validation only works with at 1 negative class.

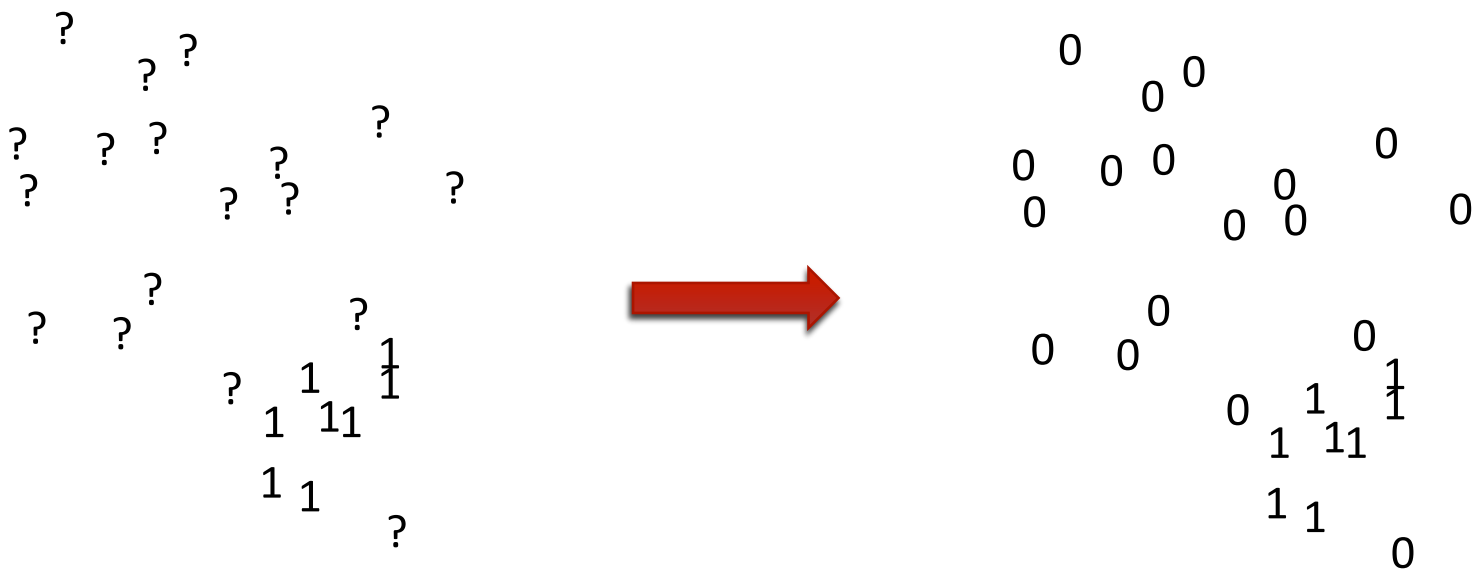
## Circumvention

- Nearest Centroid – SVM combination
  - Assumption
    - Outliers exists
    - Outliers should be far away from positive class, i.e. somehow separable in a high dimension feature space
- Nearest Centroid
  - Belongs to nearest neighbor family
  - It finds the centroid(mean) of every class in training data, and assign a new point to class i if it's closest to centroid $c_i$

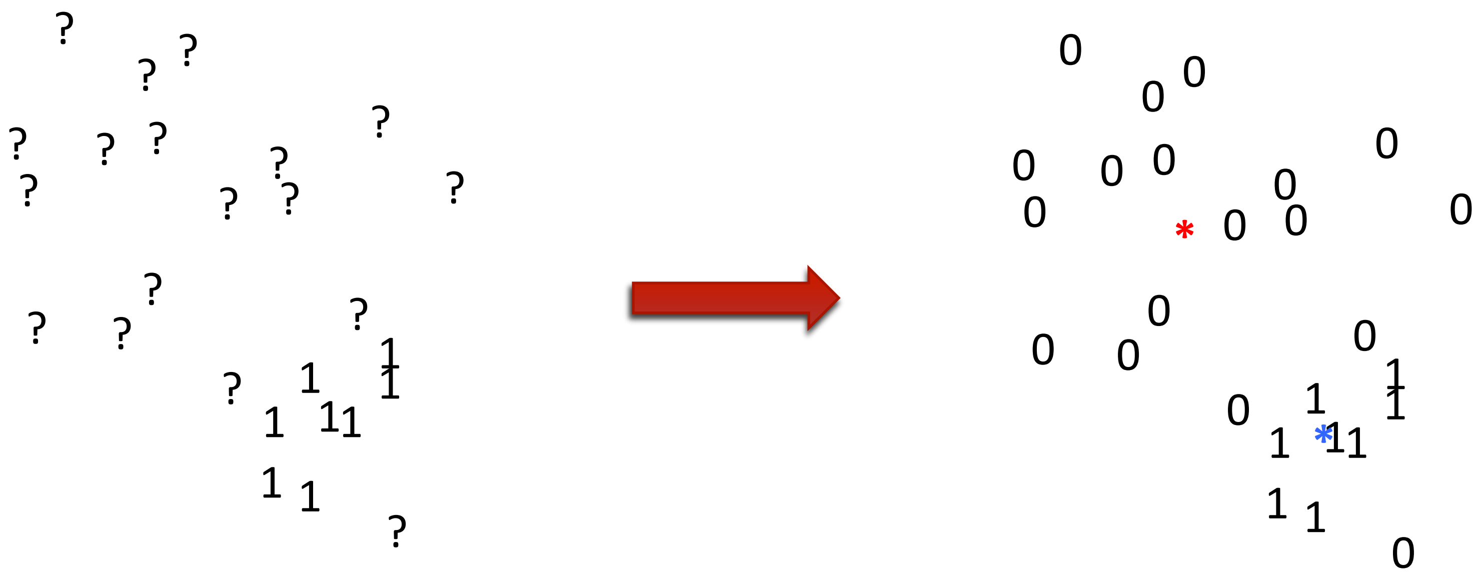
# Nearest Centroid – SVM combination



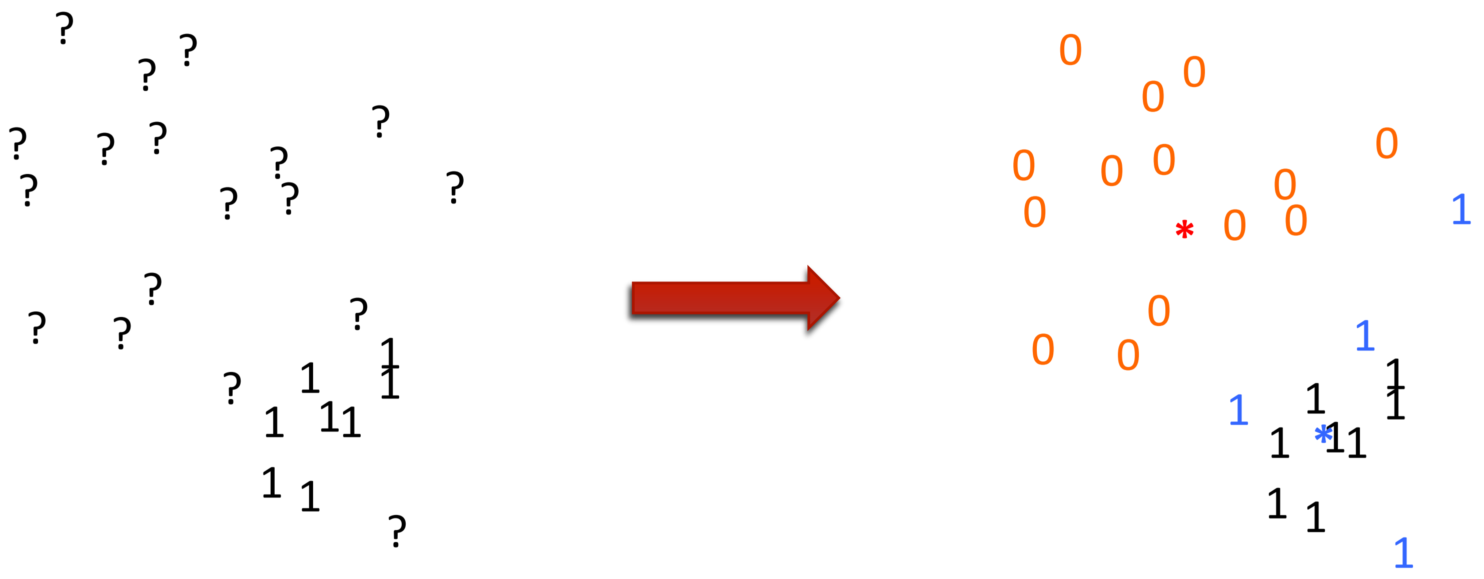
# Nearest Centroid – SVM combination



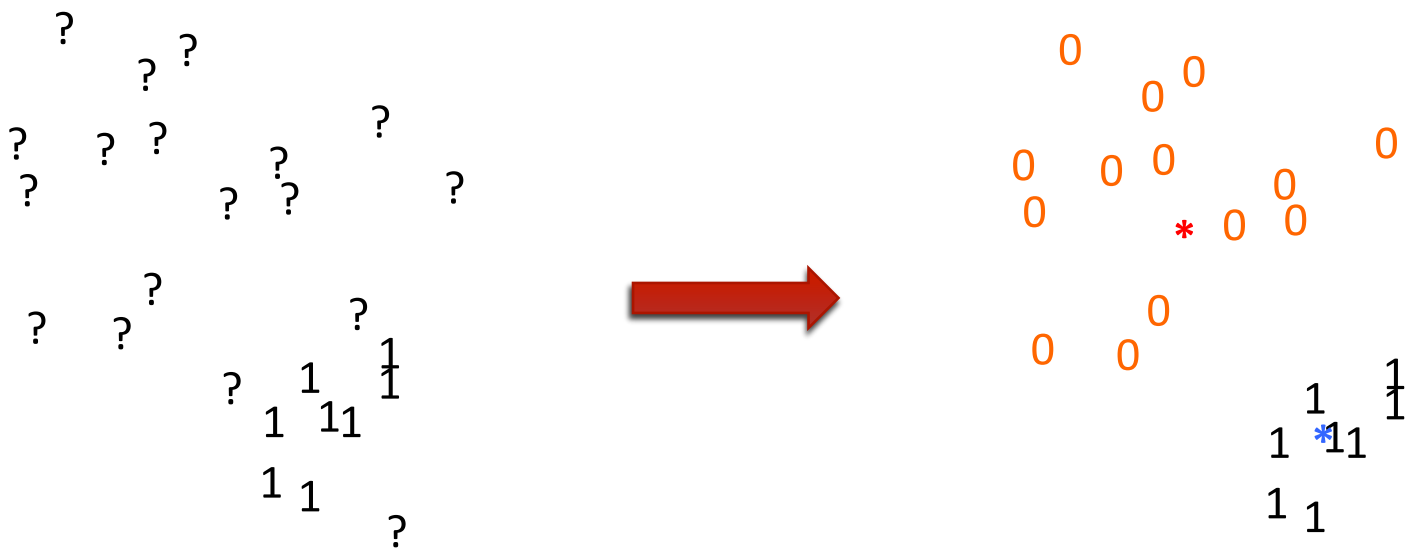
# Nearest Centroid – SVM combination



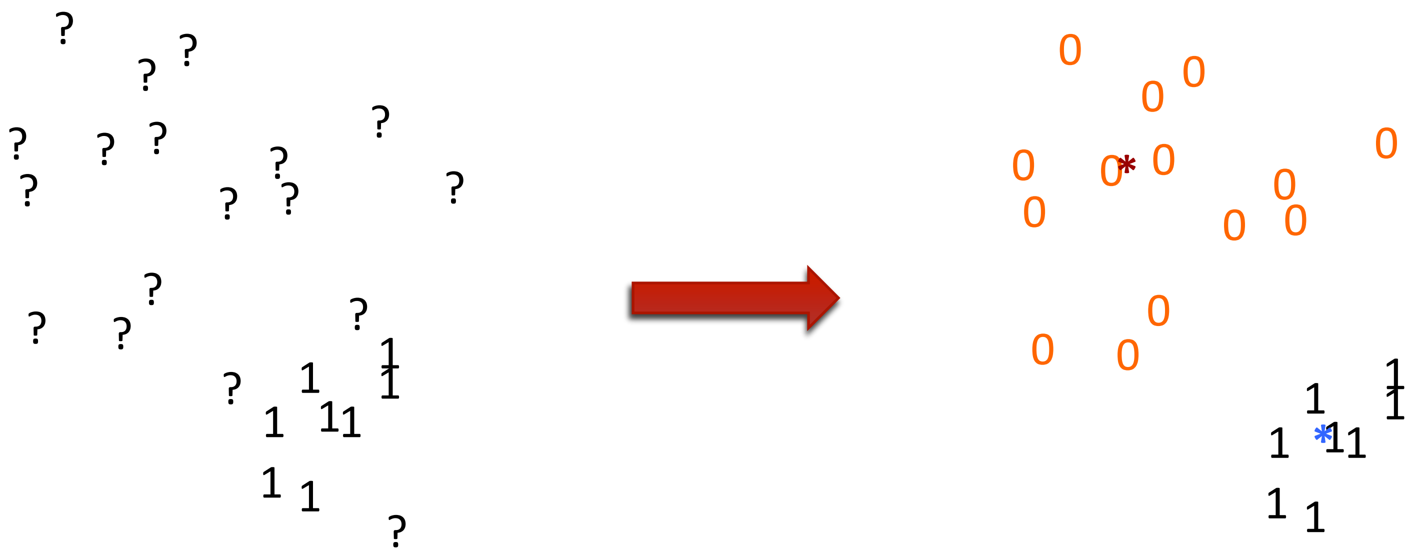
# Nearest Centroid – SVM combination



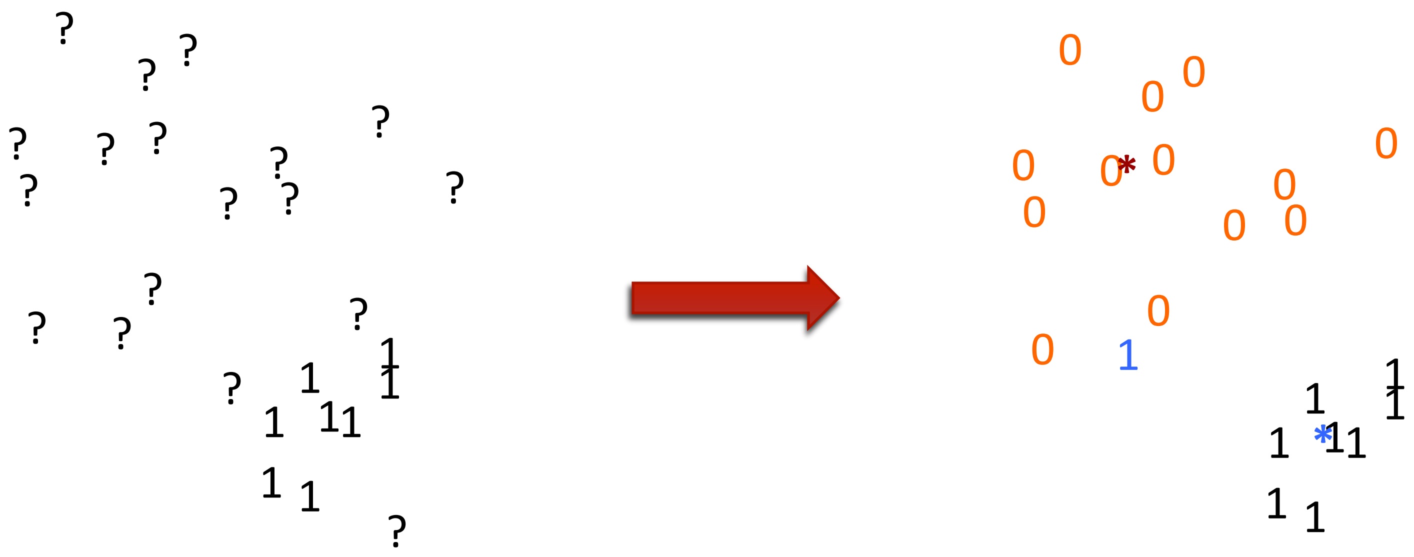
# Nearest Centroid – SVM combination



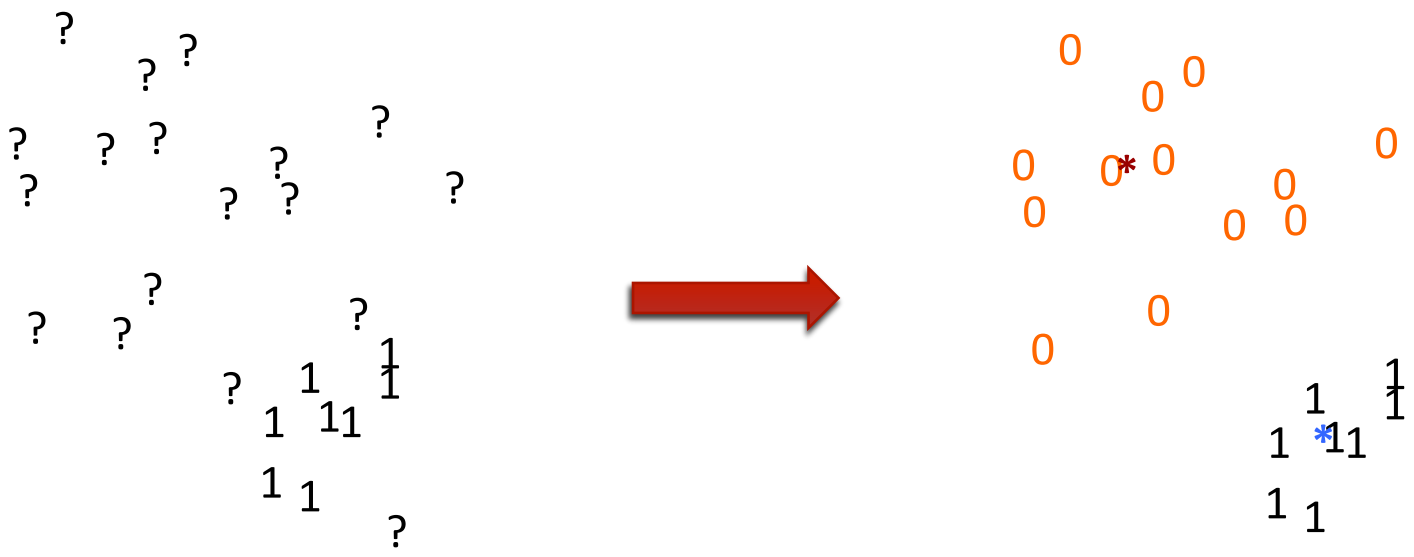
# Nearest Centroid – SVM combination



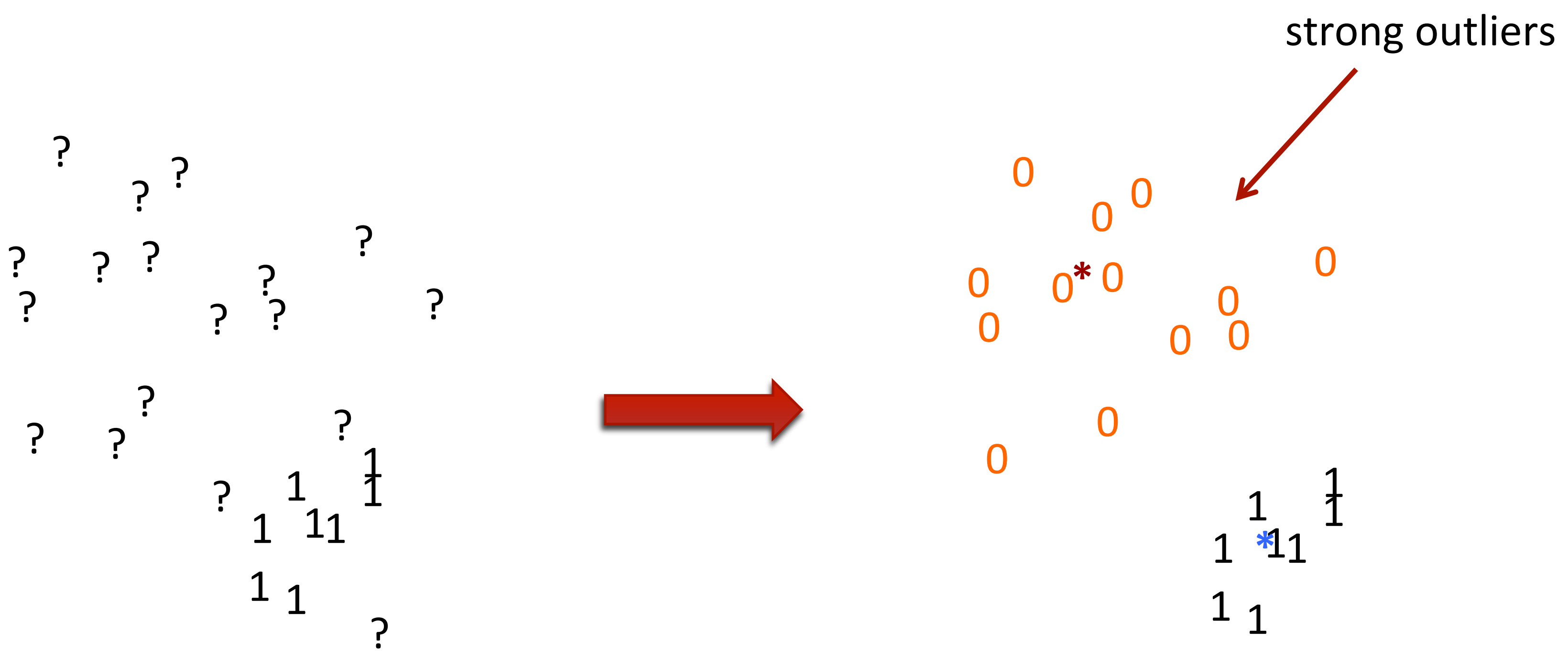
# Nearest Centroid – SVM combination



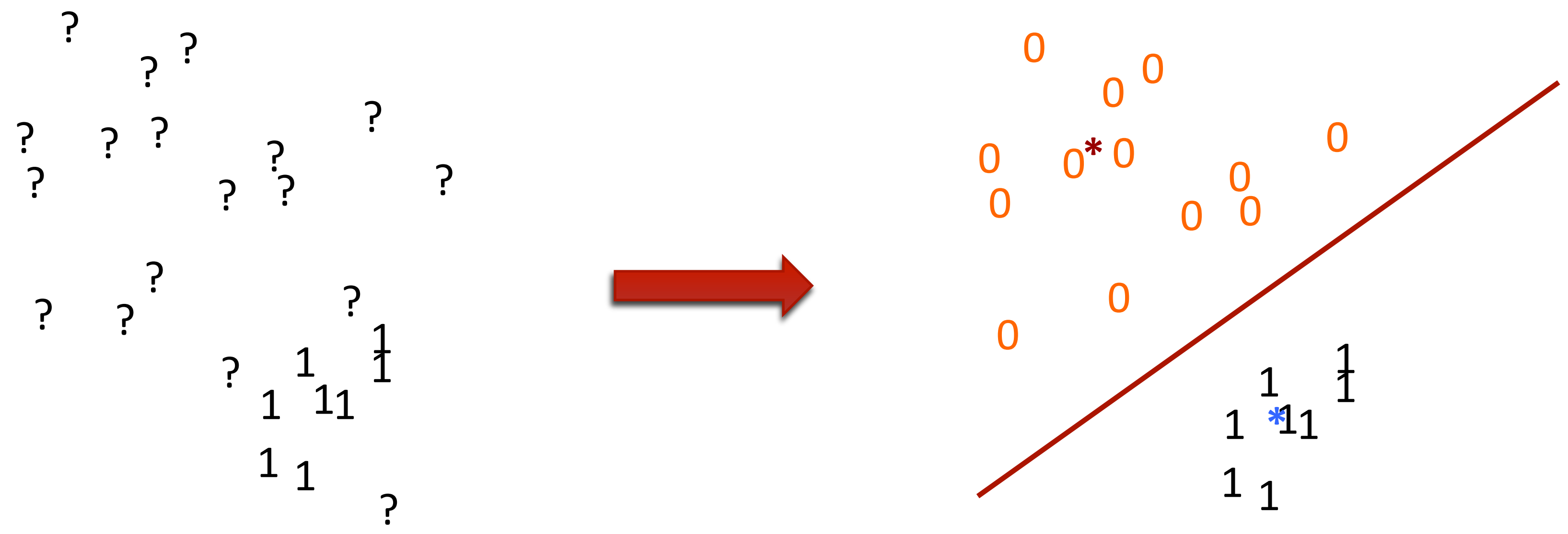
# Nearest Centroid – SVM combination



# Nearest Centroid – SVM combination



## Nearest Centroid – SVM combination



## Experiment

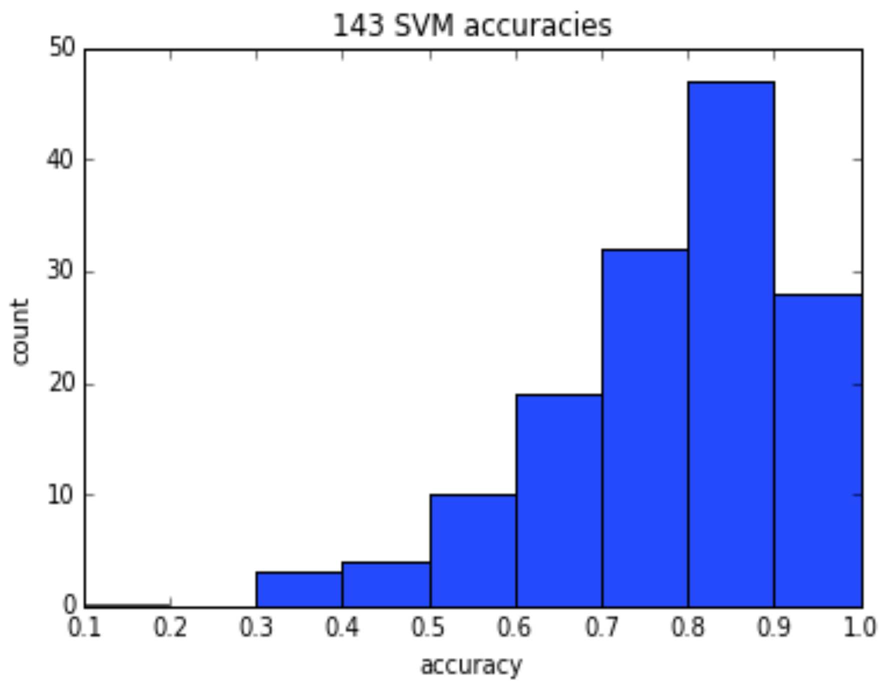
- Data set
  - Yelp dataset (JSON) from Yelp Dataset Challenge
    - Business info.
    - Reviews
    - Users info.
- Task
  - Predict whether a particular user will visit a particular restaurant
    - Check-in history not given / leaving a review implies a visit
    - We can only observe restaurants this user has chosen to visit

## Experiment

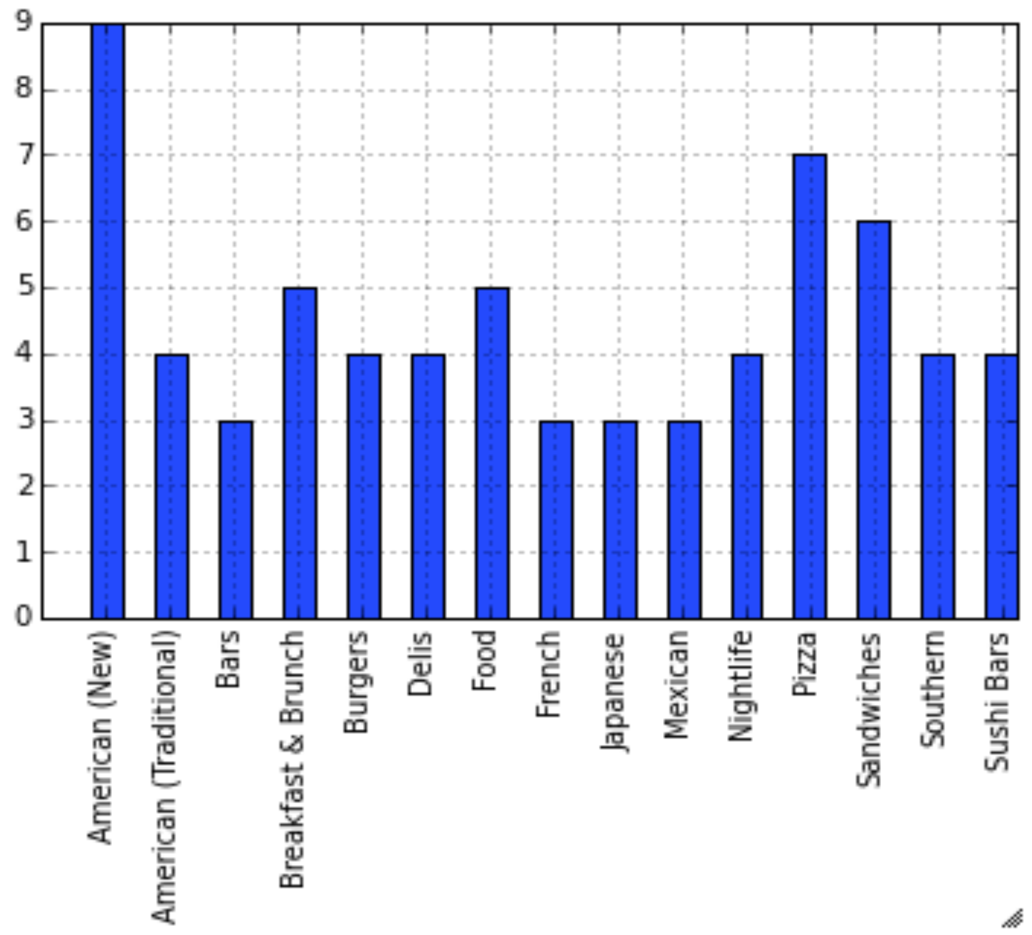
- Preprocessing
  - Restaurants from Charlotte, NC area
  - Customers who has left at least 50 reviews (active users), 142 in total
  - Binarize categorical data, standardize continuous data
  - x: restaurant info: {food types, location, stars, # of reviews, etc.}
- Model
  - For each user i
    - Extract restaurants i has left a review
    - Perform Nearest Centroid until convergence
    - Cross validate γ and C using sklearn.grid_search.GridSearch
    - Train SVM with best γ and C
    - Test on an independent testing set

# Results

- Baseline
  - 50% (random guess)
- 90%+ of SVM better than baseline
- 22.5% test errors on average



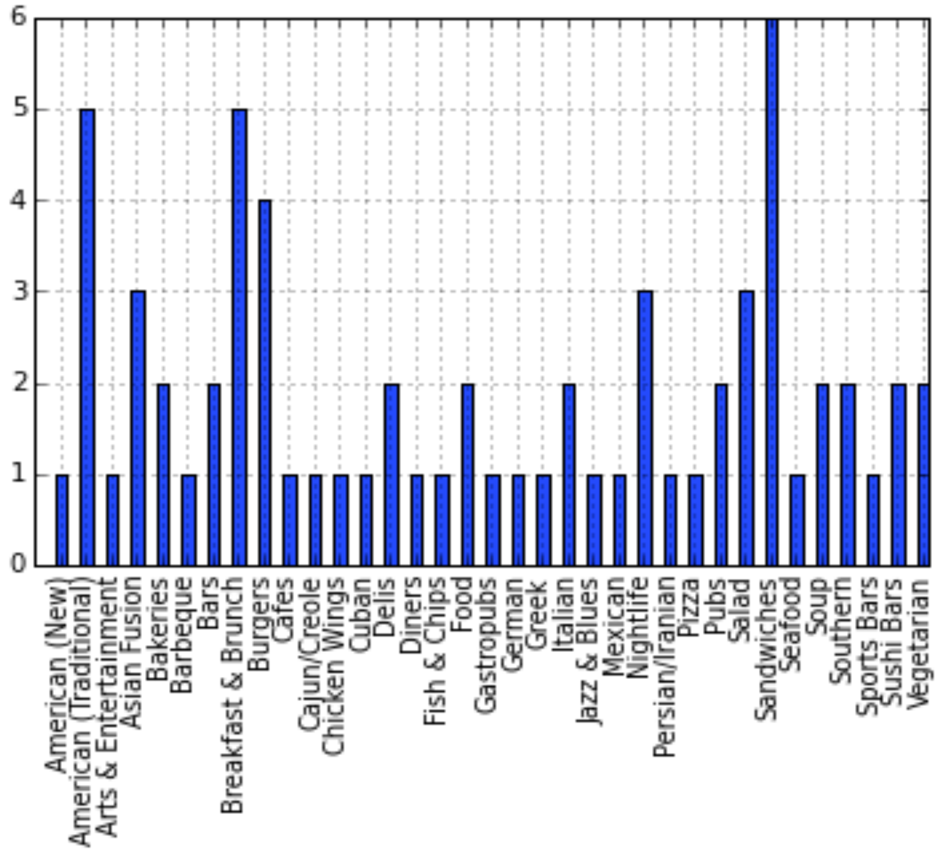
# A Closer Look



| | |
|---|---|
| American(new): | 9 |
| American(traditional): | 4 |
| Bars: | 3 |
| Breakfast & brunch : | 5 |
| Burgers: | 4 |
| Delis: | 4 |
| French: | 3 |
| Japanese: | 3 |
| Nightlife: | 4 |
| Pizza: | 7 |
| Sandwiches: | 6 |
| Southern: | 4 |
| Sushi Bars: | 4 |
| Frequency < 2: | omitted |

# A Closer Look

True Positive



False Negative

American (traditional): 1
Bars: 1
Delis: 1
Greek: 1
Italian: 1
Latin American: 1
Mexican: 1
Nightlife: 1
Pizza: 1
Seafood: 1
Steakhouses: 1
Wine Bars: 1