EE4077 Fundamentals of Machine Learning

Multi-class Classification

Fall 2021

EE–Marmara University

Outline

1. Non-linear Decision Boundary

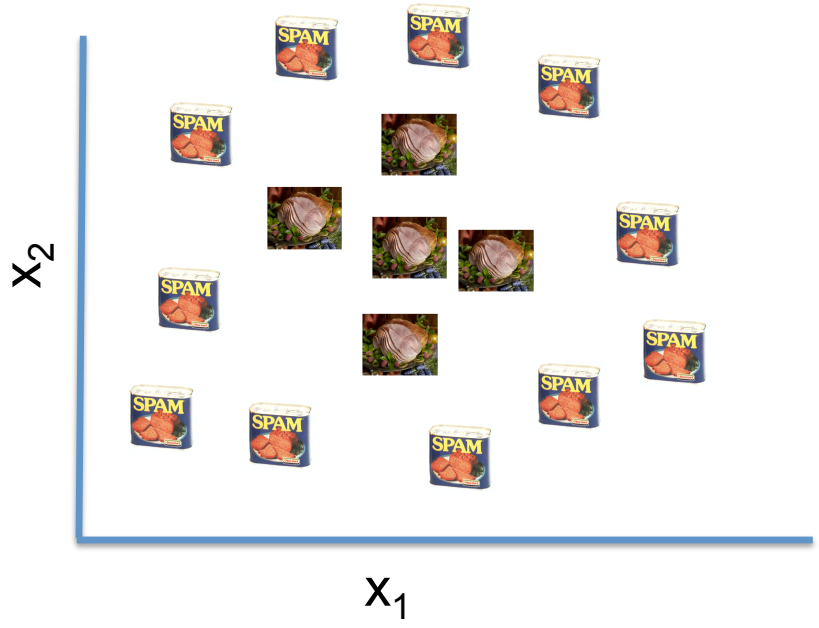
2. Multi-class Classification

    Multi-class Naive Bayes

    Multi-class Logistic Regression

Non-linear Decision Boundary

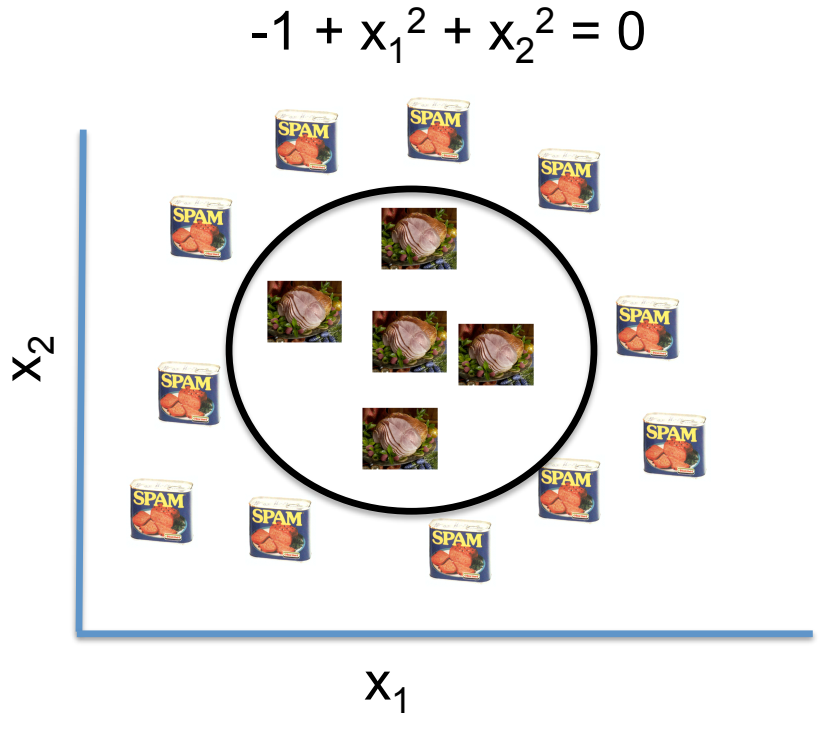
How to handle more complex decision boundaries?



- This data is not linear separable
- Use non-linear basis functions to add more features

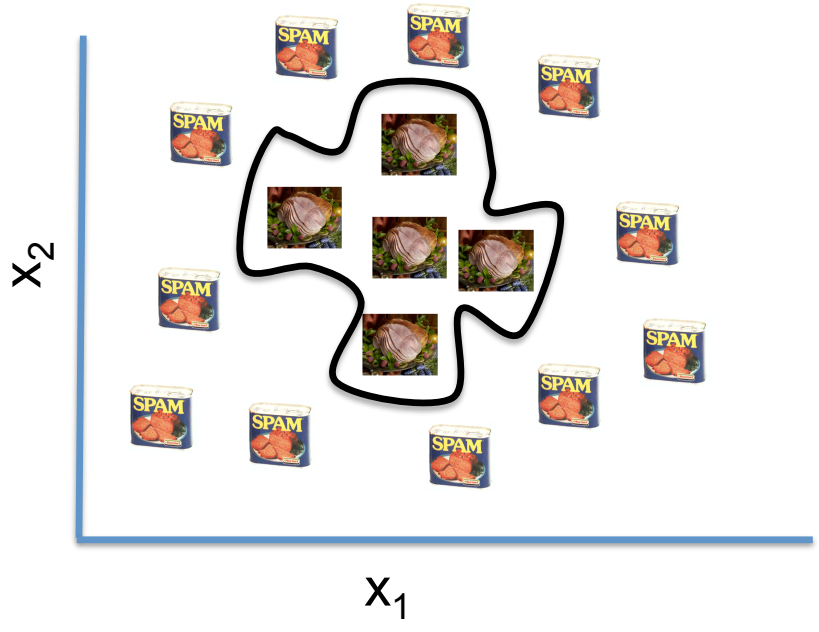
Adding polynomial features

- New feature vector is $\mathbf{x} = [1, x_1, x_2, x_1^2, x_2^2]$
- $\Pr(y = 1|\mathbf{x}) = \sigma(w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2)$
- If $\mathbf{w} = [-1, 0, 0, 1, 1]$, the boundary is $-1 + x_1^2 + x_2^2 = 0$
  - If $-1 + x_1^2 + x_2^2 \geq 0$ declare spam
  - If $-1 + x_1^2 + x_2^2 < 0$ declare ham



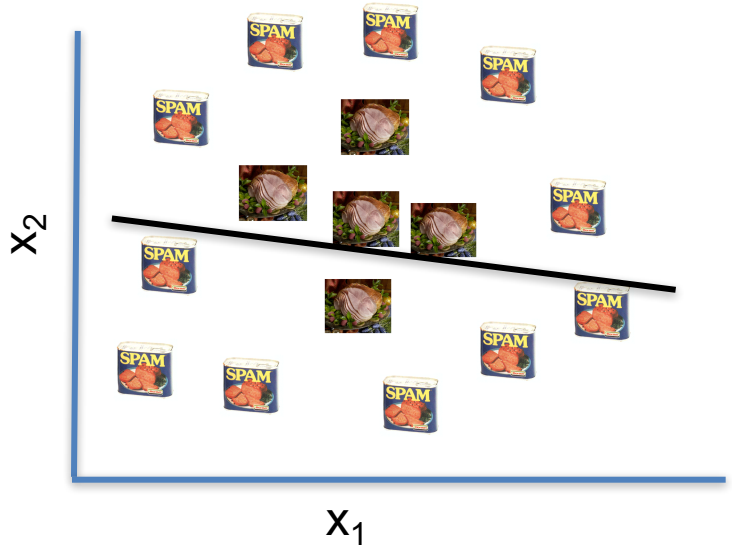
Adding polynomial features

- What if we add many more features and define $\mathbf{x} = [1, x_1, x_2, x_1^2, x_2^2, x_1^3, x_2^3, \ldots]$?
- We get a complex decision boundary

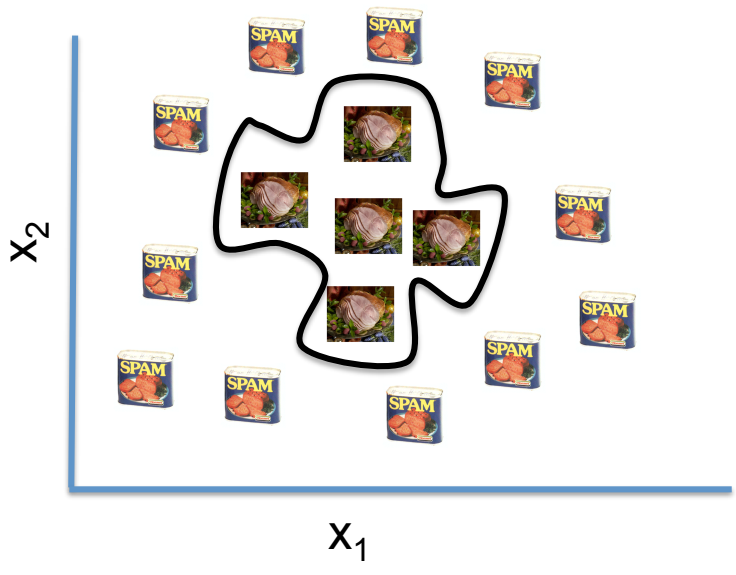


Can result in overfitting and bad generalization to new data points

Concept-check: Bias-Variance Trade-off



high bias



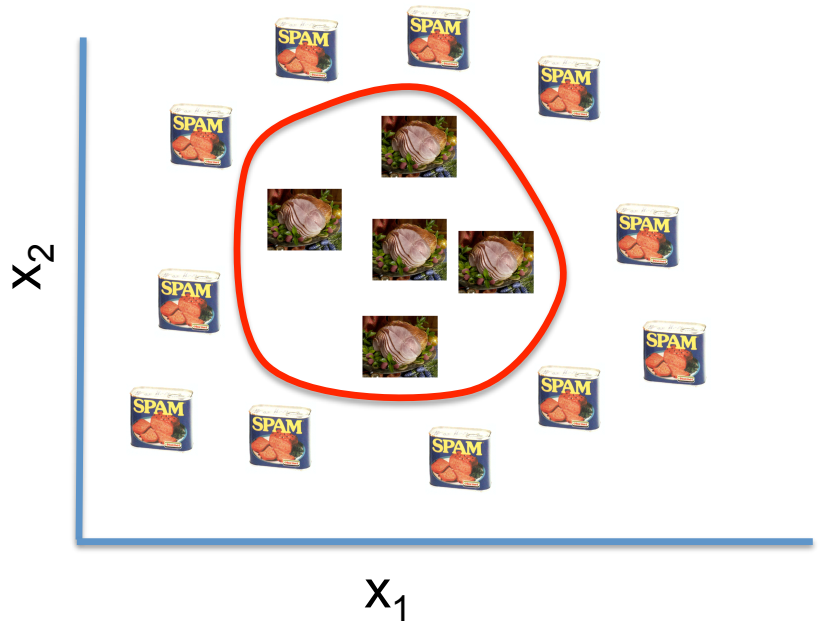
high variance

Solution to Overfitting: Regularization

- Add regularization term to be cross entropy loss function

$$\mathcal{E}(\boldsymbol{w}) = -\sum_n \{y_n \log \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) + (1-y_n)\log[1-\sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]\} + \underbrace{\frac{1}{2}\lambda \|w\|_2^2}_{\text{regularization}}$$

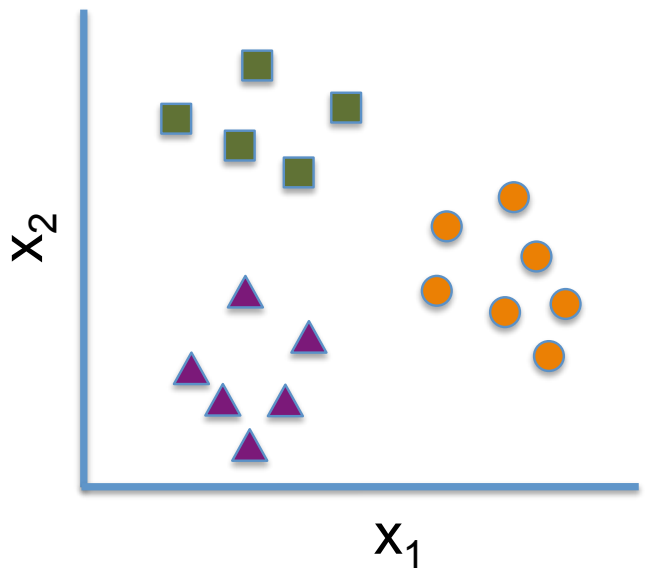
- Perform gradient descent on this regularized function
- Often, we do NOT regularize the bias term w_0 (you will see this in the homework)



Multi-class Classification

What if there are more than 2 classes?

- Dog vs. cat. vs crocodile
- Movie genres (action, horror, comedy, . . .)
- Part of speech tagging (verb, noun, adjective, . . .)
- . . .



Setup

Predict multiple classes/outcomes $C_1, C_2, \ldots, C_K$:

- Weather prediction: sunny, cloudy, raining, etc
- Optical character recognition: 10 digits + 26 characters (lower and upper cases) + special characters, etc.

K =number of classes

Methods we've studied for binary classification:

- Naive Bayes
- Logistic regression

Do they generalize to multi-class classification?

Naive Bayes is already multi-class

Formal Definition

Given a random vector $\mathbf{X} \in \mathbb{R}^{\mathrm{K}}$ and a dependent variable $Y \in [\mathrm{C}]$, the Naive Bayes model defines the joint distribution

$$P(\mathbf{X} = \mathbf{x}, Y = c) = P(Y = c)P(\mathbf{X} = \mathbf{x}|Y = c) \tag{1}$$

$$= P(Y = c) \prod_{k=1}^{\mathrm{K}} P(\mathrm{word}_k|Y = c)^{x_k} \tag{2}$$

$$= \pi_c \prod_{k=1}^{\mathrm{K}} \theta_{ck}^{x_k} \tag{3}$$

where x_k is the number of occurrences of the kth word, π_c is the prior probability of class c (which allows multiple classes!), and θ_{ck} is the weight of the kth word for the cth class.

Learning problem

Training data

$$\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{\mathrm{N}} \rightarrow \mathcal{D} = \{(\{x_{nk}\}_{k=1}^{\mathrm{K}}, y_n)\}_{n=1}^{\mathrm{N}}$$

Goal

Learn $\pi_c, c = 1, 2, \cdots, \mathrm{C}$, and $\theta_{ck}, \forall c \in [\mathrm{C}], k \in [\mathrm{K}]$ under the constraints:

$$\sum_c \pi_c = 1$$

and

$$\sum_k \theta_{ck} = \sum_k P(\mathrm{word}_k | Y = c) = 1$$

as well as π_c, $\theta_{ck} \geq 0$.

Our hammer: maximum likelihood estimation

Log-Likelihood of the training data

$$\begin{aligned}
\mathcal{L} &= \log P(\mathcal{D}) = \log \prod_{n=1}^{N} \pi_{y_n} P(\mathbf{x}_n | y_n) \\
&= \log \prod_{n=1}^{N} \left(\pi_{y_n} \prod_k \theta_{y_n k}^{x_{nk}} \right) \\
&= \sum_n \left(\log \pi_{y_n} + \sum_k x_{nk} \log \theta_{y_n k} \right) \\
&= \sum_n \log \pi_{y_n} + \sum_{n,k} x_{nk} \log \theta_{y_n k}
\end{aligned}$$

Optimize it!

$$(\pi_c^*, \theta_{ck}^*) = \arg\max \sum_n \log \pi_{y_n} + \sum_{n,k} x_{nk} \log \theta_{y_n k}$$

Our hammer: maximum likelihood estimation

Optimization Problem

$$(\pi_c^*, \theta_{ck}^*) = \arg\max \sum_n \log \pi_{y_n} + \sum_{n,k} x_{nk} \log \theta_{y_n k}$$

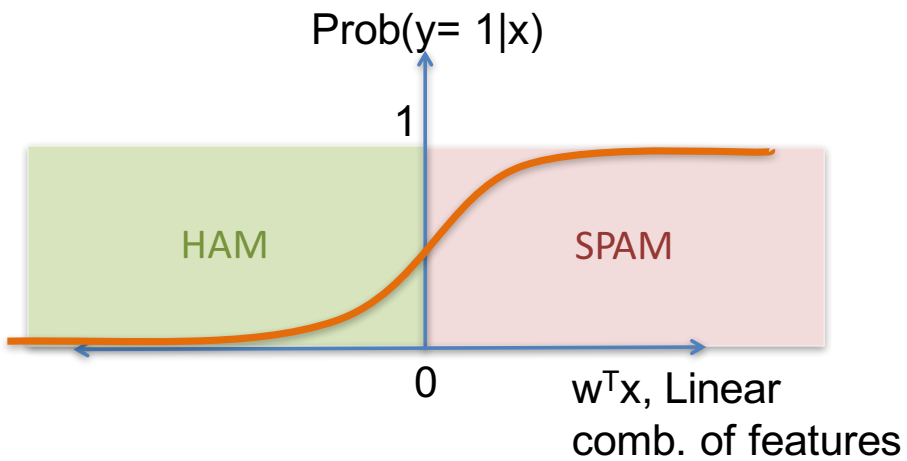
Solution

$$\theta_{ck}^* = \frac{\#\text{of times word } k \text{ shows up in data points labeled as } c}{\#\text{total trials for data points labeled as } c}$$

$$\pi_c^* = \frac{\#\text{of data points labeled as c}}{\text{N}}$$

Logistic regression for predicting multiple classes?

- The linear decision boundary that we optimized was specific to binary classification.
  - If $\sigma(\mathbf{w}^\top \mathbf{x}) \geq 0.5$ declare $y = 1$ (spam)
  - If $\sigma(\mathbf{w}^\top \mathbf{x}) < 0.5$ declare $y = 0$ (ham)
- How to extend it to multi-class classification?

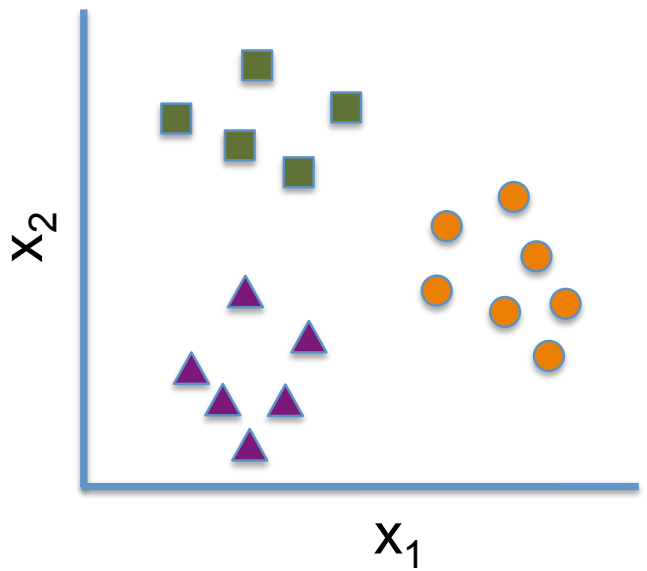


$y = 1$ for spam, $y = 0$ for ham

Idea: Express as multiple binary classification problems

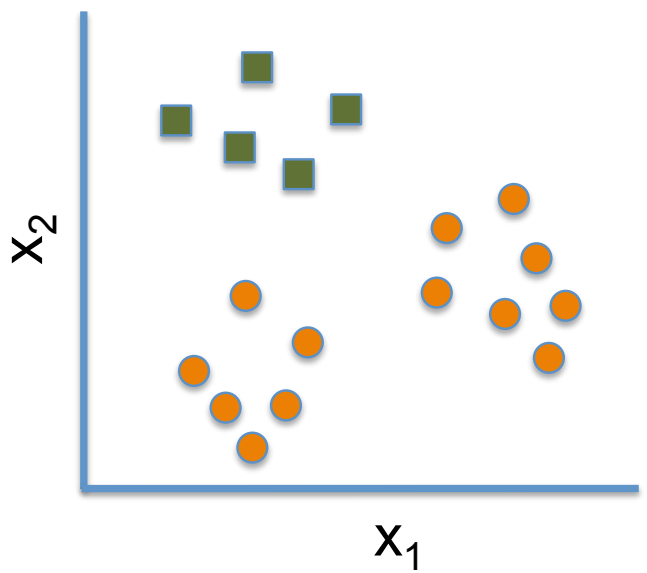
The One-versus-Rest or One-Versus-All Approach

- For each class C_k, change the problem into binary classification
  1. Relabel training data with label C_k, into POSITIVE (or '1')
  2. Relabel all the rest data into NEGATIVE (or '0')
- Repeat this multiple times: Train K binary classifiers, using logistic regression to differentiate the two classes each time



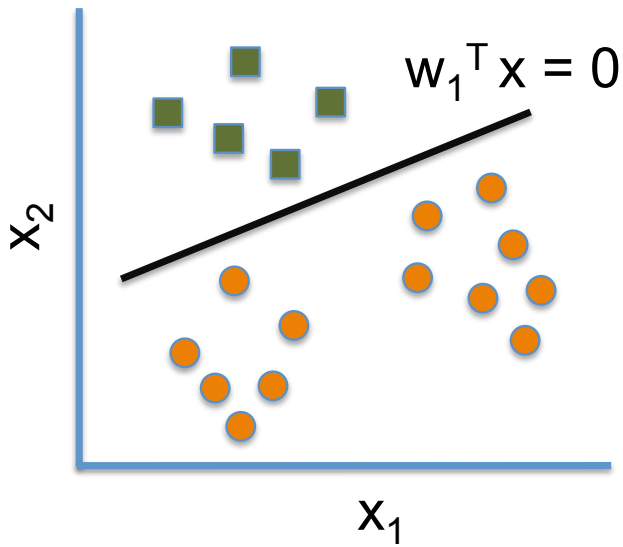
The One-versus-Rest or One-Versus-All Approach

- For each class C_k, change the problem into binary classification
  1. Relabel training data with label C_k, into POSITIVE (or '1')
  2. Relabel all the rest data into NEGATIVE (or '0')
- Repeat this multiple times: Train K binary classifiers, using logistic regression to differentiate the two classes each time



The One-versus-Rest or One-Versus-All Approach

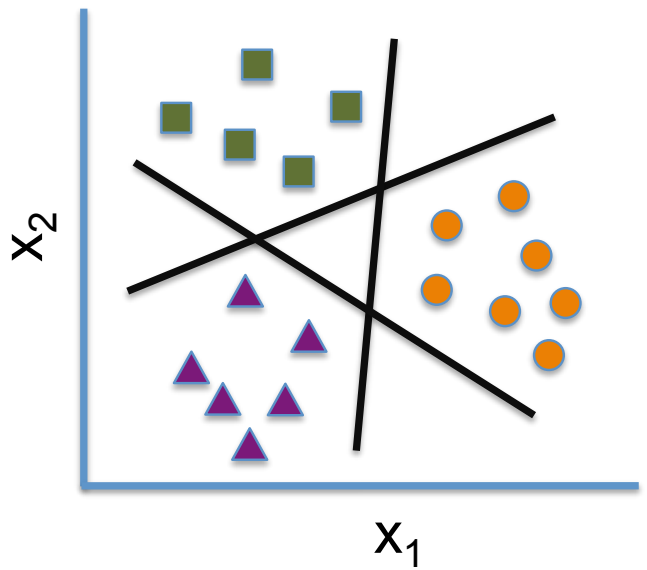
- For each class C_k, change the problem into binary classification
  1. Relabel training data with label C_k, into POSITIVE (or '1')
  2. Relabel all the rest data into NEGATIVE (or '0')
- Repeat this multiple times: Train K binary classifiers, using logistic regression to differentiate the two classes each time



The One-versus-Rest or One-Versus-All Approach

How to combine these linear decision boundaries?

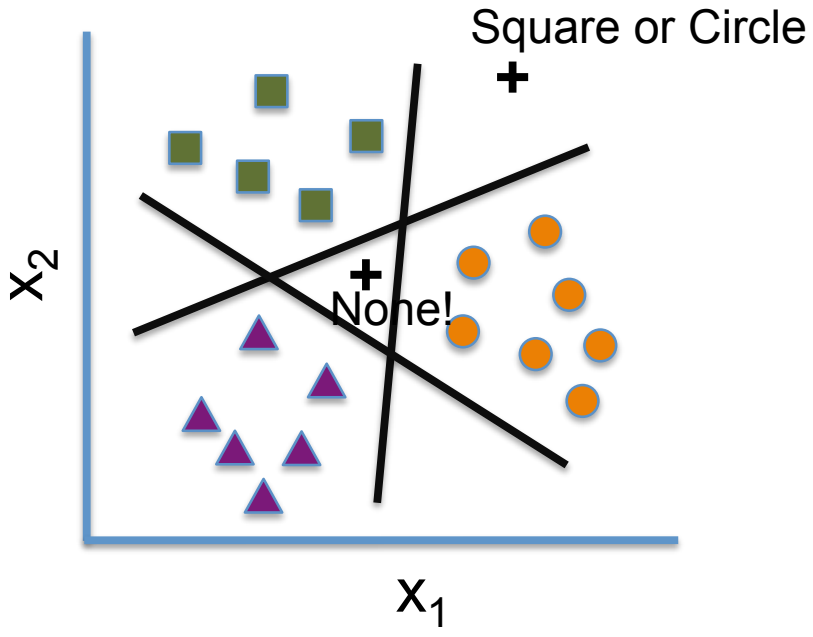
- There is ambiguity in some of the regions (the 4 triangular areas)



The One-versus-Rest or One-Versus-All Approach

How to combine these linear decision boundaries?

- There is ambiguity in some of the regions (the 4 triangular areas)
- How do we resolve this?

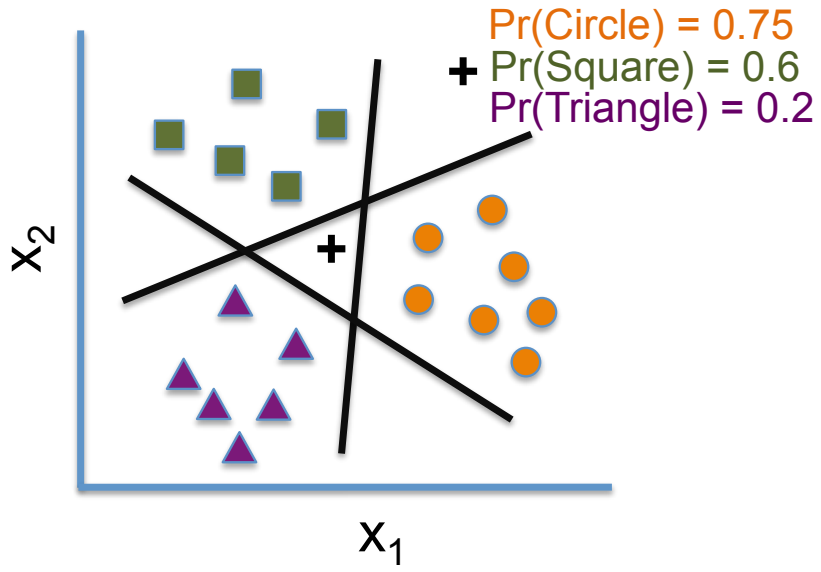


The One-versus-Rest or One-Versus-All Approach

How to combine these linear decision boundaries?

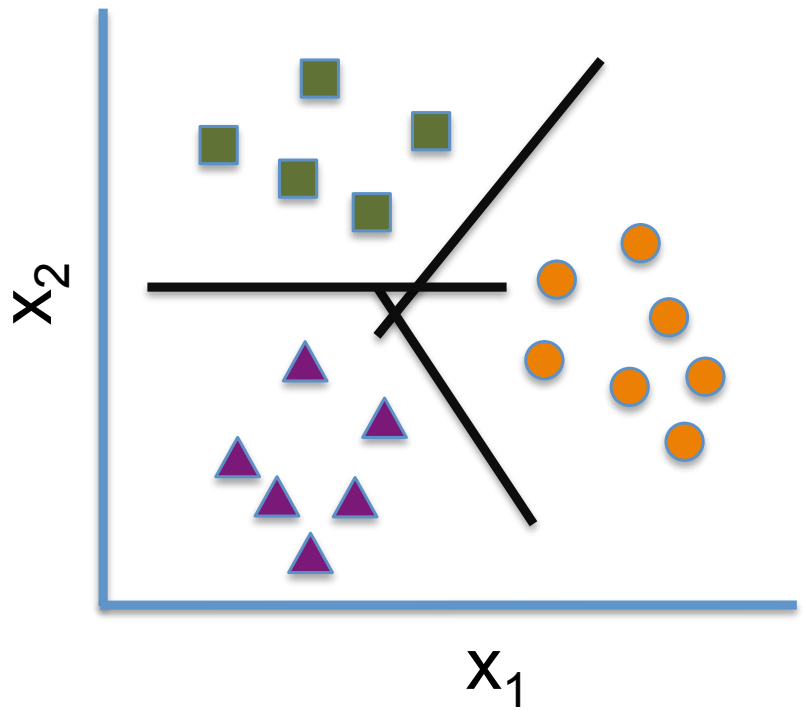
- Use the confidence estimates $\Pr(y = C_1|\mathbf{x}) = \sigma(\mathbf{w}_1^\top \mathbf{x})$, $\ldots \Pr(y = C_K|\mathbf{x}) = \sigma(\mathbf{w}_K^\top \mathbf{x})$
- Declare class C_k^* that maximizes

$$k^* = \arg \max_{k=1,\ldots,K} \Pr(y = C_k|\mathbf{x}) = \sigma(\mathbf{w}_k^\top \mathbf{x})$$



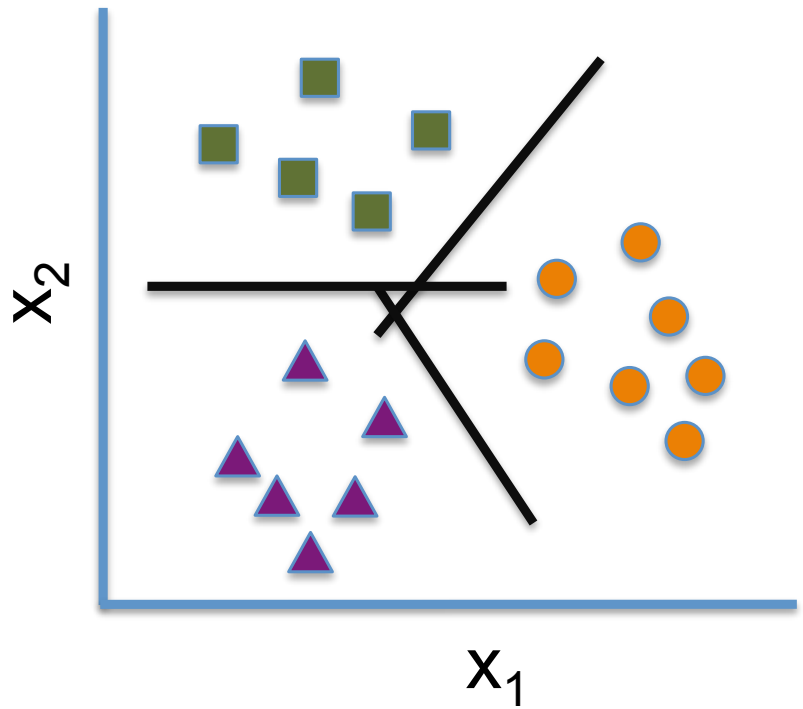
The One-Versus-One Approach

- For each **pair** of classes C_k and $C_{k'}$, change the problem into binary classification
  1. Relabel training data with label C_k, into POSITIVE (or '1')
  2. Relabel training data with label $C_{k'}$ into NEGATIVE (or '0')
  3. **Disregard** all other data



The One-Versus-One Approach

- How many binary classifiers for K classes? $K(K-1)/2$
- How to combine their outputs?
- Given $\boldsymbol{x}$, count the $K(K-1)/2$ votes from outputs of all binary classifiers and declare the winner as the predicted class.
- Use confidence scores to resolve ties



Contrast these approaches

Number of Binary Classifiers to be trained

- **One-Versus-All:** K classifiers.
- **One-Versus-One:** $K(K-1)/2$ classifiers – bad if K is large

Effect of Relabeling and Splitting Training Data

- **One-Versus-All:** imbalance in the number of positive and negative samples can cause bias in each trained classifier
- **One-Versus-One:** each classifier trained on a small subset of data (only those labeled with those two classes would be involved), which can result in high variance

Any other ideas?

- Hierarchical classification – we will see this in decision trees
- Multinomial Logistic Regression – directly output probabilities of y being in each of the K classes, instead of reducing to a binary classification problem.

Multinomial logistic regression

Intuition:

from the decision rule of our naive Bayes classifier

$$y^* = \arg\max_k p(y = C_k|\boldsymbol{x}) = \arg\max_k \log p(\boldsymbol{x}|y = C_k)p(y = C_k)$$
$$= \arg\max_k \log \pi_k + \sum_i x_i \log \theta_{ki} = \arg\max_k \boldsymbol{w}_k^\top \boldsymbol{x}$$

Essentially, we are comparing

$$\boldsymbol{w}_1^\top \boldsymbol{x}, \boldsymbol{w}_2^\top \boldsymbol{x}, \cdots, \boldsymbol{w}_\mathrm{K}^\top \boldsymbol{x}$$

with **one** for each category.

First try

So, can we define the following conditional model?

$$p(y = C_k|\boldsymbol{x}) = \sigma[\boldsymbol{w}_k^\top \boldsymbol{x}].$$

This would **not** work because:

$$\sum_k p(y = C_k|\boldsymbol{x}) = \sum_k \sigma[\boldsymbol{w}_k^\top \boldsymbol{x}] \neq 1.$$

each summand can be any number (independently) between 0 and 1.
But we are close!

Learn the K linear models jointly to ensure this property holds!

Multinomial logistic regression

- Model: For each class C_k, we have a parameter vector $\boldsymbol{w}_k$ and model the posterior probability as:

$$p(C_k|\boldsymbol{x}) = \frac{e^{\boldsymbol{w}_k^\top \boldsymbol{x}}}{\sum_{k'} e^{\boldsymbol{w}_{k'}^\top \boldsymbol{x}}} \quad \leftarrow \quad \textit{This is called softmax function}$$

- Decision boundary: Assign $\boldsymbol{x}$ with the label that is the maximum of posterior:

$$\arg\max_k P(C_k|\boldsymbol{x}) \rightarrow \arg\max_k \boldsymbol{w}_k^\top \boldsymbol{x}.$$

Sanity check

Multinomial model reduce to binary logistic regression when $K = 2$

$$p(C_1|\boldsymbol{x}) = \frac{e^{\boldsymbol{w}_1^\top \boldsymbol{x}}}{e^{\boldsymbol{w}_1^\top \boldsymbol{x}} + e^{\boldsymbol{w}_2^\top \boldsymbol{x}}} = \frac{1}{1 + e^{-(\boldsymbol{w}_1 - \boldsymbol{w}_2)^\top \boldsymbol{x}}}$$
$$= \frac{1}{1 + e^{-\boldsymbol{w}^\top \boldsymbol{x}}}$$

Multinomial thus generalizes the (binary) logistic regression to deal with multiple classes.

Parameter estimation

Discriminative approach: maximize conditional likelihood

$$\log P(\mathcal{D}) = \sum_n \log P(y_n|\boldsymbol{x}_n)$$

We will change y_n to $\boldsymbol{y}_n = [y_{n1}\ y_{n2}\ \cdots\ y_{nK}]^\top$, a K-dimensional vector using 1-of-K encoding.

$$y_{nk} = \begin{cases} 1 & \text{if } y_n = k \\ 0 & \text{otherwise} \end{cases}$$

Ex: if $y_n = 2$, then, $\boldsymbol{y}_n = [0\ 1\ 0\ 0\ \cdots\ 0]^\top$.

$$\Rightarrow \sum_n \log P(y_n|\boldsymbol{x}_n) = \sum_n \log \prod_{k=1}^{K} P(C_k|\boldsymbol{x}_n)^{y_{nk}} = \sum_n \sum_k y_{nk} \log P(C_k|\boldsymbol{x}_n)$$

Cross-entropy error function

Definition: negative log likelihood

$$\mathcal{E}(\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_K) = -\sum_n \sum_k y_{nk} \log P(C_k|\boldsymbol{x}_n)$$

$$= -\sum_n \sum_k y_{nk} \log \left(\frac{e^{\boldsymbol{w}_k^\top \boldsymbol{x}_n}}{\sum_{k'} e^{\boldsymbol{w}_{k'}^\top \boldsymbol{x}_n}} \right)$$

Properties

- Convex, therefore unique global optimum
- Optimization requires numerical procedures, analogous to those used for binary logistic regression

Summary

You should know

- What is logistic regression and solving for $\mathbf{w}$ using gradient descent on the cross entropy loss function
- Difference between Naive Bayes and Logistic Regression
- How to solve for the model parameters using gradient descent
- How to handle multiclass classification: one-versus-all, one-versus-one, multinomial regression