18-661 Introduction to Machine Learning

Multi-class Classification

Spring 2020

ECE – Carnegie Mellon University

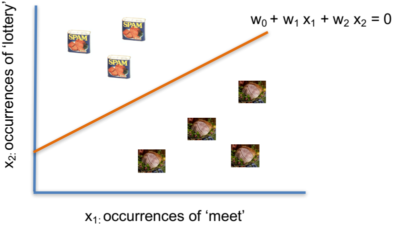
Outline

1. Review of Logistic regression
2. Non-linear Decision Boundary
3. Multi-class Classification
    - Multi-class Naive Bayes
    - Multi-class Logistic Regression

Review of Logistic regression

Intuition: Logistic Regression

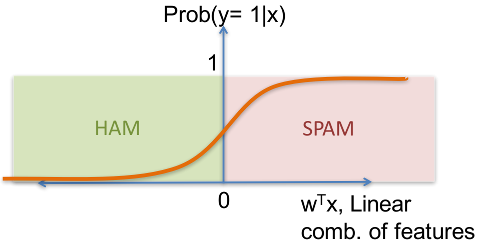
- $x_1 = \#$ of times 'meet' appears in an email
- $x_2 = \#$ of times 'lottery' appears in an email
- Define feature vector $\mathbf{x} = [1, x_1, x_2]$
- Learn the decision boundary $w_0 + w_1 x_1 + w_2 x_2 = 0$ such that
  - If $\mathbf{w}^\top \mathbf{x} \geq 0$ declare $y = 1$ (spam)
  - If $\mathbf{w}^\top \mathbf{x} < 0$ declare $y = 0$ (ham)



Key Idea: map features into points in a high-dimensional space, and use hyperplanes to separate them

Intuition: Logistic Regression

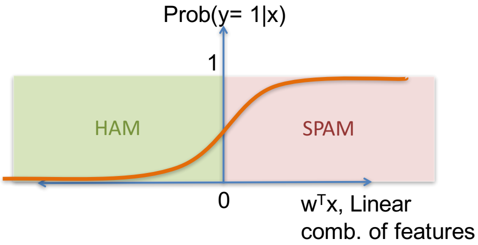
- Suppose we want to output the probability of an email being spam/ham instead of just 0 or 1
- This gives information about the confidence in the decision
- Use a function $\sigma(\mathbf{w}^\top \mathbf{x})$ that maps $\mathbf{w}^\top \mathbf{x}$ to a value between 0 and 1



Probability that predicted label is 1 (spam)

Intuition: Logistic Regression

- Suppose we want to output the probability of an email being spam/ham instead of just 0 or 1
- This gives information about the confidence in the decision
- Use a function $\sigma(\mathbf{w}^\top \mathbf{x})$ that maps $\mathbf{w}^\top \mathbf{x}$ to a value between 0 and 1



Probability that predicted label is 1 (spam)

Key Problem: Finding optimal weights $\mathbf{w}$ that accurately predict this probability for a new email

Formal Setup: Binary Logistic Classification

- Input: $\boldsymbol{x} = [1, x_1, x_2, \ldots x_D] \in \mathbb{R}^{D+1}$

Formal Setup: Binary Logistic Classification

- Input: $\boldsymbol{x} = [1, x_1, x_2, \ldots x_D] \in \mathbb{R}^{D+1}$
- Output: $y \in \{0, 1\}$

Formal Setup: Binary Logistic Classification

- Input: $\boldsymbol{x} = [1, x_1, x_2, \ldots x_D] \in \mathbb{R}^{D+1}$
- Output: $y \in \{0, 1\}$
- Training data: $\mathcal{D} = \{(\boldsymbol{x}_n, y_n), n = 1, 2, \ldots, N\}$

Formal Setup: Binary Logistic Classification

- Input: $\boldsymbol{x} = [1, x_1, x_2, \dots x_D] \in \mathbb{R}^{D+1}$
- Output: $y \in \{0, 1\}$
- Training data: $\mathcal{D} = \{(\boldsymbol{x}_n, y_n), n = 1, 2, \dots, N\}$

Formal Setup: Binary Logistic Classification

- Input: $\boldsymbol{x} = [1, x_1, x_2, \ldots x_D] \in \mathbb{R}^{D+1}$
- Output: $y \in \{0, 1\}$
- Training data: $\mathcal{D} = \{(\boldsymbol{x}_n, y_n), n = 1, 2, \ldots, N\}$
- Model:

$$p(y = 1|\boldsymbol{x}; \boldsymbol{w}) = \sigma(\boldsymbol{w}^\top \boldsymbol{x})$$

Formal Setup: Binary Logistic Classification

- Input: $\boldsymbol{x} = [1, x_1, x_2, \ldots x_D] \in \mathbb{R}^{D+1}$
- Output: $y \in \{0, 1\}$
- Training data: $\mathcal{D} = \{(\boldsymbol{x}_n, y_n), n = 1, 2, \ldots, N\}$
- Model:

$$p(y = 1 | \boldsymbol{x}; \boldsymbol{w}) = \sigma(\boldsymbol{w}^\top \boldsymbol{x})$$

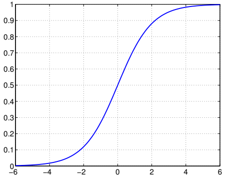
Formal Setup: Binary Logistic Classification

- Input: $\boldsymbol{x} = [1, x_1, x_2, \ldots x_D] \in \mathbb{R}^{D+1}$
- Output: $y \in \{0, 1\}$
- Training data: $\mathcal{D} = \{(\boldsymbol{x}_n, y_n), n = 1, 2, \ldots, N\}$
- Model:

$$p(y = 1|\boldsymbol{x}; \boldsymbol{w}) = \sigma(\boldsymbol{w}^\top \boldsymbol{x})$$

and $\sigma[\cdot]$ stands for the *sigmoid* function

$$\sigma(a) = \frac{1}{1 + e^{-a}}$$



How to optimize w?

Probability of a single training sample $(\boldsymbol{x}_n, y_n)$

$$p(y_n|\boldsymbol{x}_n; \boldsymbol{w}) = \begin{cases} \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) & \text{if} \quad y_n = 1 \\ 1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) & \text{otherwise} \end{cases}$$

How to optimize w?

Probability of a single training sample $(\boldsymbol{x}_n, y_n)$

$$p(y_n|\boldsymbol{x}_n; \boldsymbol{w}) = \begin{cases} \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) & \text{if} \quad y_n = 1 \\ 1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) & \text{otherwise} \end{cases}$$

Compact expression, exploring that y_n is either 1 or 0

$$p(y_n|\boldsymbol{x}_n; \boldsymbol{w}) = \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)^{y_n}[1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]^{1-y_n}$$

Minimize the negative log-likelihood of the whole training data $\mathcal{D}$, i.e. *cross-entropy error function*

$$\mathcal{E}(\boldsymbol{w}) = -\sum_n \{y_n \log \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) + (1 - y_n) \log[1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]\}$$

Gradient Descent Update for Logistic Regression

Cross-entropy Error Function

$$\mathcal{E}(\boldsymbol{w}) = -\sum_n \{y_n \log \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) + (1 - y_n) \log[1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]\}$$

Gradient Descent Update for Logistic Regression

Cross-entropy Error Function

$$\mathcal{E}(\boldsymbol{w}) = -\sum_n \{y_n \log \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) + (1 - y_n) \log[1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]\}$$

Simple fact: derivatives of $\sigma(a)$

$$\frac{d}{d\,a}\sigma(a) = \sigma(a)[1 - \sigma(a)]$$

Gradient Descent Update for Logistic Regression

Cross-entropy Error Function

$$\mathcal{E}(\boldsymbol{w}) = -\sum_n \{y_n \log \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) + (1 - y_n) \log[1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]\}$$

Simple fact: derivatives of $\sigma(a)$

$$\frac{d}{d\,a}\sigma(a) = \sigma(a)[1 - \sigma(a)]$$

Gradient of cross-entropy loss

$$\frac{\partial \mathcal{E}(\boldsymbol{w})}{\partial \boldsymbol{w}} = \sum_n \left\{\sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) - y_n\right\} \boldsymbol{x}_n$$

Gradient Descent Update for Logistic Regression

Cross-entropy Error Function

$$\mathcal{E}(\boldsymbol{w}) = -\sum_n \{y_n \log \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) + (1 - y_n) \log[1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]\}$$

Simple fact: derivatives of $\sigma(a)$

$$\frac{d}{d\,a}\sigma(a) = \sigma(a)[1 - \sigma(a)]$$

Gradient of cross-entropy loss

$$\frac{\partial \mathcal{E}(\boldsymbol{w})}{\partial \boldsymbol{w}} = \sum_n \left\{\sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) - y_n\right\} \boldsymbol{x}_n$$

Remark

- $e_n = \left\{\sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) - y_n\right\}$ is called *error* for the nth training sample.

Numerical optimization

Gradient descent for logistic regression

- Choose a proper step size $\eta > 0$
- Iteratively update the parameters following the negative gradient to minimize the error function

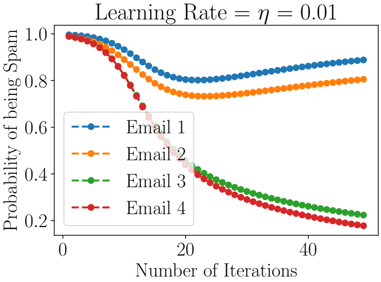
$$\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} - \eta \sum_n \left\{ \sigma(\boldsymbol{x}_n^\top \boldsymbol{w}^{(t)}) - y_n \right\} \boldsymbol{x}_n$$

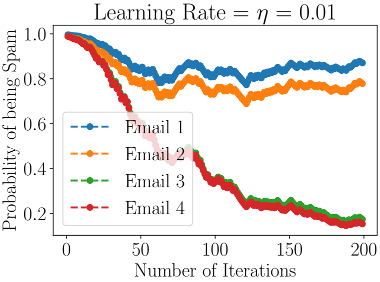
- Can also perform stochastic gradient descent (with a possibly different learning rate η)

$$\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} - \eta \left\{ \sigma(\boldsymbol{x}_{i_t}^\top \boldsymbol{w}^{(t)}) - y_{i_t} \right\} \boldsymbol{x}_{i_t}$$

where i_t is drawn uniformly at randomly from the training data $\{1, 2, \cdots\}$

Batch gradient descent vs SGD





Batch GD

fewer iterations,

every iteration uses all samples

SGD

more iterations,

every iteration uses one sample

Logistic regression vs linear regression

	logistic regression	linear regression
Training data	$(\boldsymbol{x}_n, y_n), y_n \in \{0,1\}$	$(\boldsymbol{x}_n, y_n), y_n \in \mathbb{R}$
loss function	cross-entropy	RSS
prob. interpretation	$y_n \vert \boldsymbol{x}_n, \boldsymbol{w} \sim \mathrm{Ber}(\sigma(\boldsymbol{w}^\top \boldsymbol{x}_n))$	$y_n \vert \boldsymbol{x}_n, \boldsymbol{w} \sim \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{x}_n, \sigma^2)$
gradient	$\sum_n \left(\sigma(\boldsymbol{x}_n^\top \boldsymbol{w}) - y_n\right) \boldsymbol{x}_n$	$\sum_n \left(\boldsymbol{x}_n^\top \boldsymbol{w} - y_n\right) \boldsymbol{x}_n$

Cross-entropy loss function (logistic regression):

$$\mathcal{E}(\boldsymbol{w}) = -\sum_n \{y_n \log \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) + (1 - y_n) \log[1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]\}$$

RSS loss function (linear regression):

$$RSS(\boldsymbol{w}) = \frac{1}{2} \sum_n (y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2$$

Outline

1. Review of Logistic regression

2. Non-linear Decision Boundary

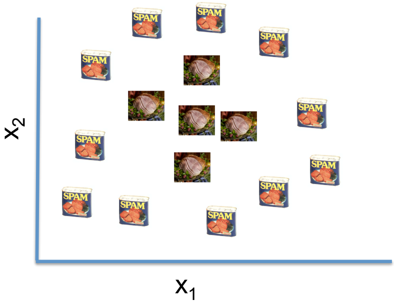
3. Multi-class Classification

   Multi-class Naive Bayes

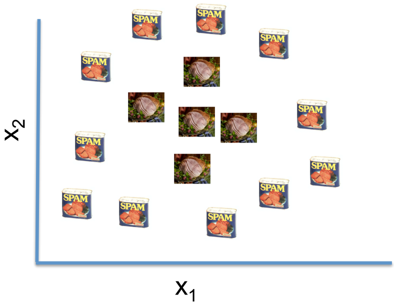
   Multi-class Logistic Regression

Non-linear Decision Boundary

How to handle more complex decision boundaries?

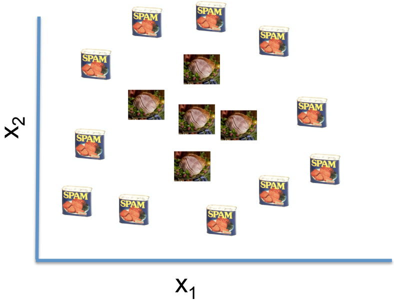


How to handle more complex decision boundaries?



- This data is not linear separable

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 $\mathbf{x} = [1, x_1, x_2, x_1^2, x_2^2]$

Adding polynomial features

- New feature vector is $\mathbf{x} = [1, x_1, x_2, x_1^2, x_2^2]$
- $\Pr(y = 1|\mathbf{x}) = \sigma(w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2)$

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 $-1 + x_1^2 + x_2^2 = 0$

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

Adding polynomial features

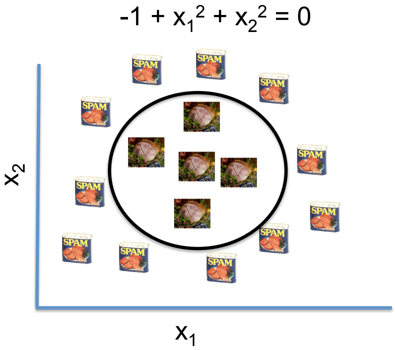
- New feature vector is $\mathbf{x} = [1, x_1, x_2, x_1^2, x_2^2]$
- $\Pr(y = 1|\mathbf{x}) = \sigma(w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2)$
- If $\mathbf{w} = [-1, 0, 0, 1, 1]$, the boundary is $-1 + x_1^2 + x_2^2 = 0$
  - If $-1 + x_1^2 + x_2^2 \geq 0$ declare spam
  - If $-1 + x_1^2 + x_2^2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

Adding polynomial features

- 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

- 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 $\mathbf{x} = [1, x_1, x_2, x_1^2, x_2^2, x_1^3, x_2^3, \ldots]$?

Adding polynomial features

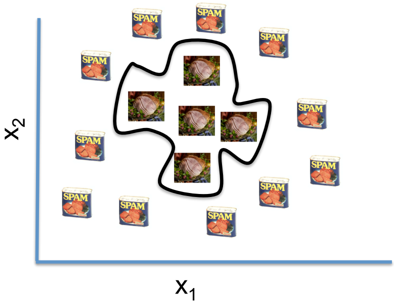
- What if we add many more features and define $\mathbf{x} = [1, x_1, x_2, x_1^2, x_2^2, x_1^3, x_2^3, \ldots]$?
- We get a complex decision boundary

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

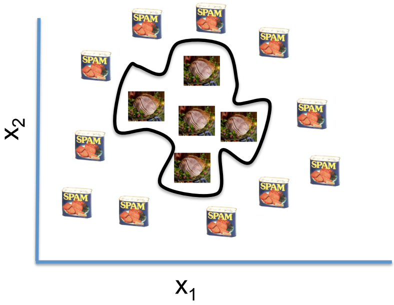
Adding polynomial features

- 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



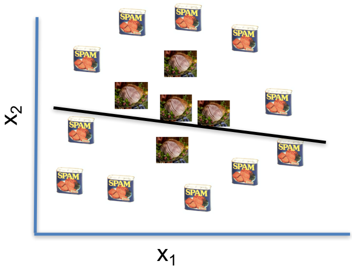
Adding polynomial features

- 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

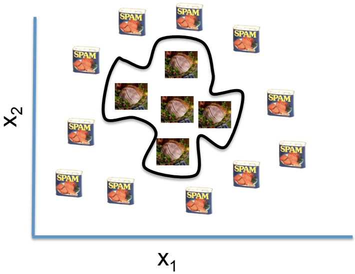


Can result in overfitting and bad generalization to new data points

Concept-check: Bias-Variance Trade-off



high bias



high variance

Solution to Overfitting: Regularization

- Add regularization term to be cross entropy loss function

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

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

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)

Solution to Overfitting: Regularization

- Add regularization term to be cross entropy loss function

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

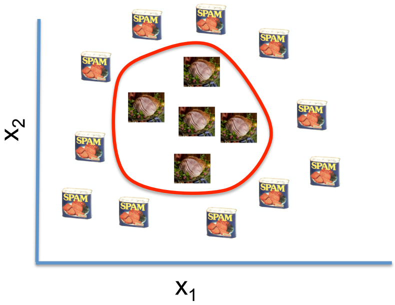
- Perform gradient descent on this regularized function
- Often, we do NOT regularize the bias term w_0 (you will see this in the homework)

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)



Outline

1. Review of Logistic regression

2. Non-linear Decision Boundary

3. Multi-class Classification

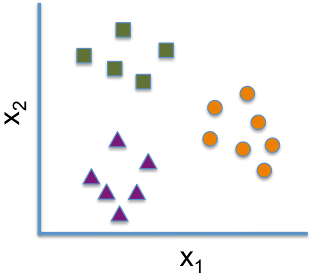
   Multi-class Naive Bayes

   Multi-class Logistic Regression

Multi-class Classification

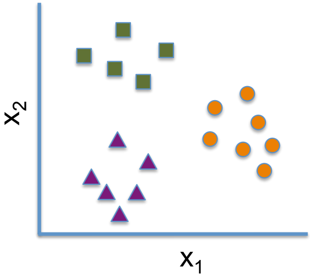
What if there are more than 2 classes?

- Dog vs. cat. vs crocodile



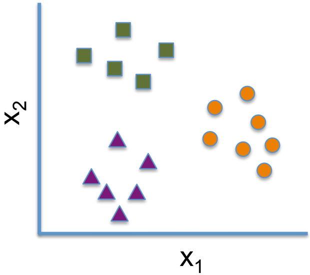
What if there are more than 2 classes?

- Dog vs. cat. vs crocodile
- Movie genres (action, horror, comedy, . . .)



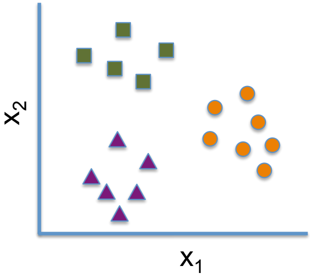
What if there are more than 2 classes?

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



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

$$
\begin{aligned}
P(\mathbf{X} = \mathbf{x}, Y = c) &= P(Y = c) P(\mathbf{X} = \mathbf{x} | Y = c) && (1) \\
&= P(Y = c) \prod_{k=1}^{\mathsf{K}} P(\mathrm{word}_k | Y = c)^{x_k} && (2) \\
&= \pi_c \prod_{k=1}^{\mathsf{K}} \theta_{ck}^{x_k} && (3)
\end{aligned}
$$

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

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

Goal

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

$$\sum_c \pi_c = 1$$

and

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

as well as $\pi_c, \theta_{ck} \geq 0$.

Our hammer: maximum likelihood estimation

Log-Likelihood of the training data

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

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

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

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

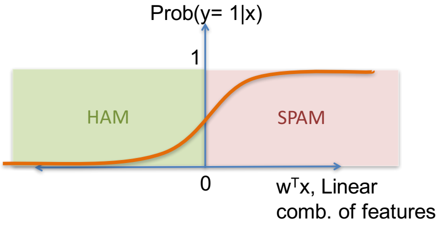
$$\theta_{ck}^* = \frac{\#\text{of times word } k \text{ shows up in data points labeled as } c}{\#\text{total trials for data points labeled as } c}$$
$$\pi_c^* = \frac{\#\text{of data points labeled as c}}{\text{N}}$$

Outline

1. Review of Logistic regression
2. Non-linear Decision Boundary
3. Multi-class Classification
   - Multi-class Naive Bayes
   - Multi-class Logistic Regression

Logistic regression for predicting multiple classes?

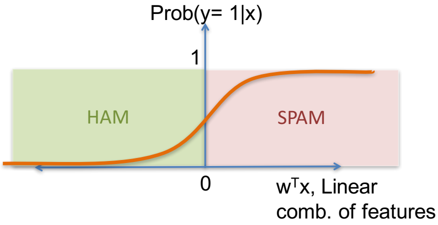
- The linear decision boundary that we optimized was specific to binary classification.



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

Logistic regression for predicting multiple classes?

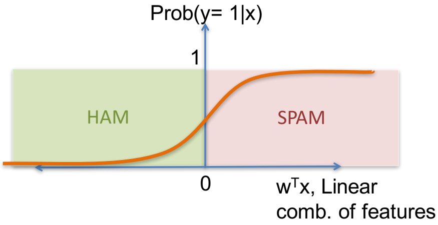
- The linear decision boundary that we optimized was specific to binary classification.
  - If $\sigma(\mathbf{w}^\top \mathbf{x}) \geq 0.5$ declare $y = 1$ (spam)



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

Logistic regression for predicting multiple classes?

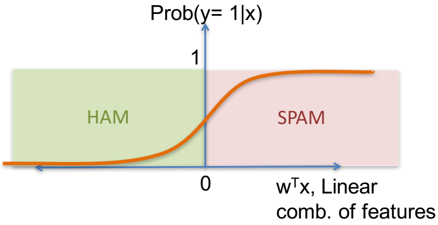
- The linear decision boundary that we optimized was specific to binary classification.
  - If $\sigma(\mathbf{w}^\top \mathbf{x}) \geq 0.5$ declare $y = 1$ (spam)
  - If $\sigma(\mathbf{w}^\top \mathbf{x}) < 0.5$ declare $y = 0$ (ham)



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

Logistic regression for predicting multiple classes?

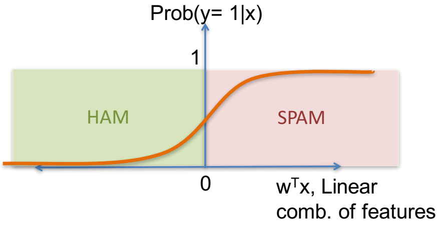
- The linear decision boundary that we optimized was specific to binary classification.
  - If $\sigma(\mathbf{w}^\top \mathbf{x}) \geq 0.5$ declare $y = 1$ (spam)
  - If $\sigma(\mathbf{w}^\top \mathbf{x}) < 0.5$ declare $y = 0$ (ham)
- How to extend it to multi-class classification?



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

Logistic regression for predicting multiple classes?

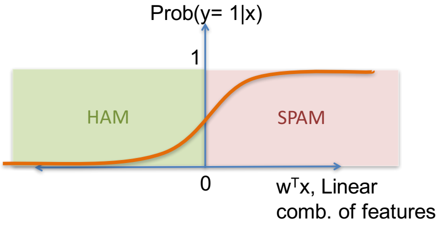
- The linear decision boundary that we optimized was specific to binary classification.
  - If $\sigma(\mathbf{w}^\top \mathbf{x}) \geq 0.5$ declare $y = 1$ (spam)
  - If $\sigma(\mathbf{w}^\top \mathbf{x}) < 0.5$ declare $y = 0$ (ham)
- How to extend it to multi-class classification?



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

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?

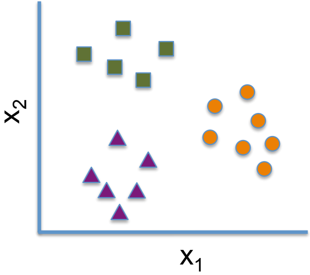


$y = 1$ for spam, $y = 0$ for ham

Idea: Express as multiple binary classification problems

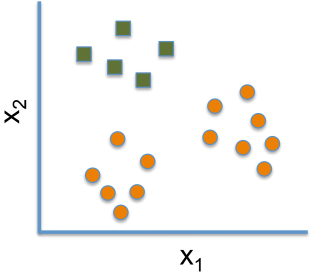
The One-versus-Rest or One-Versus-All Approach

- 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



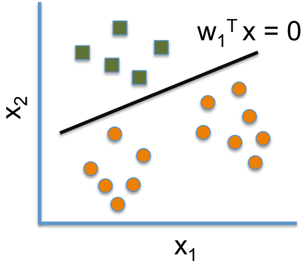
The One-versus-Rest or One-Versus-All Approach

- 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

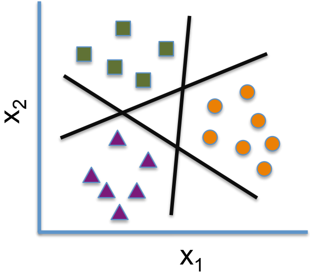
- For each class C_k, change the problem into binary classification
  1. Relabel training data with label C_k, into POSITIVE (or '1')
  2. Relabel all the rest data into NEGATIVE (or '0')
- Repeat this multiple times: Train K binary classifiers, using logistic regression to differentiate the two classes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



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

How to combine these linear decision boundaries?

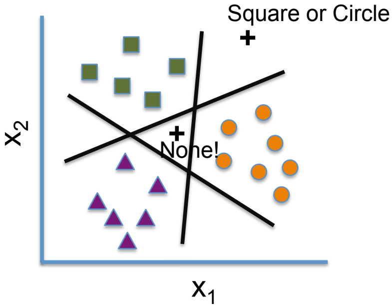
- There is ambiguity in some of the regions (the 4 triangular areas)



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?

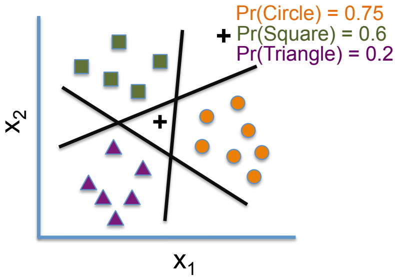


The One-versus-Rest or One-Versus-All Approach

How to combine these linear decision boundaries?

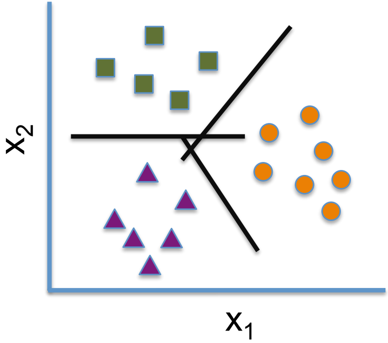
- Use the confidence estimates $\Pr(y = C_1|\mathbf{x}) = \sigma(\mathbf{w}_1^\top \mathbf{x})$, $\ldots \Pr(y = C_K|\mathbf{x}) = \sigma(\mathbf{w}_K^\top \mathbf{x})$
- Declare class C_k^* that maximizes

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



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

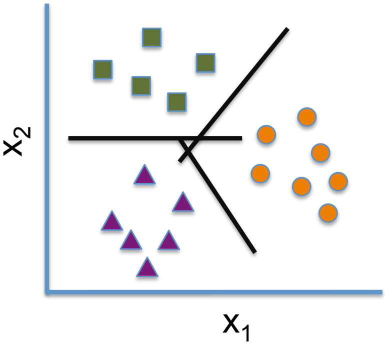
- How many binary classifiers for K classes?

The One-Versus-One Approach

- How many binary classifiers for K classes?

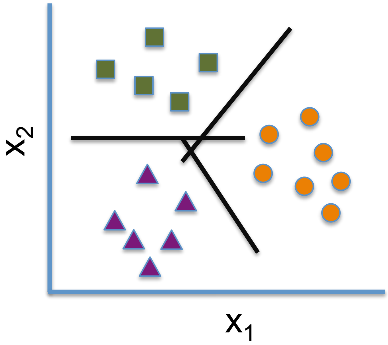
The One-Versus-One Approach

- How many binary classifiers for K classes? $K(K-1)/2$
- How to combine their outputs?



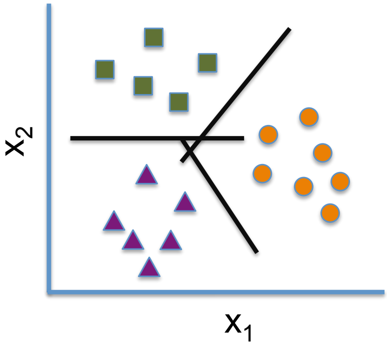
The One-Versus-One Approach

- 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.



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

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

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?

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

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

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

Multinomial logistic regression

Intuition:

from the decision rule of our naive Bayes classifier

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

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?

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

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!

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:

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

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

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

How does the softmax function behave?

Suppose we have

$$\boldsymbol{w}_1^\top \boldsymbol{x} = 100, \quad \boldsymbol{w}_2^\top \boldsymbol{x} = 50, \quad \boldsymbol{w}_3^\top \boldsymbol{x} = -20.$$

How does the softmax function behave?

Suppose we have

$$\boldsymbol{w}_1^\top \boldsymbol{x} = 100, \quad \boldsymbol{w}_2^\top \boldsymbol{x} = 50, \quad \boldsymbol{w}_3^\top \boldsymbol{x} = -20.$$

We would pick the **winning** class label 1.

Softmax translates these scores into well-formed conditional probabilities

$$p(y = 1|\boldsymbol{x}) = \frac{e^{100}}{e^{100} + e^{50} + e^{-20}} < 1$$

- preserves relative ordering of scores
- maps scores to values between 0 and 1 that also sum to 1

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

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

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, $\boldsymbol{y}_n = [0\ 1\ 0\ 0\ \cdots\ 0]^\top$.

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

$$\mathcal{E}(\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_K) = -\sum_n \sum_k y_{nk} \log P(C_k|\boldsymbol{x}_n)$$
$$= -\sum_n \sum_k y_{nk} \log \left(\frac{e^{\boldsymbol{w}_k^\top \boldsymbol{x}_n}}{\sum_{k'} e^{\boldsymbol{w}_{k'}^\top \boldsymbol{x}_n}} \right)$$

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