# Gaussian and Linear Discriminant Analysis; Multiclass Classification

Professor Ameet Talwalkar

Slide Credit: Professor Fei Sha

# Outline

1. Administration
2. Review of last lecture
3. Generative versus discriminative
4. Multiclass classification

# Announcements

- Homework 2: due on Thursday

# Outline

# Logistic classification

## Setup for two classes

- Input: $\boldsymbol{x} \in \mathbb{R}^D$
- Output: $y \in \{0, 1\}$
- Training data: $\mathcal{D} = \{(\boldsymbol{x}_n, y_n), n = 1, 2, \ldots, N\}$
- Model of *conditional distribution*

$$p(y = 1 | \boldsymbol{x}; b, \boldsymbol{w}) = \sigma[g(\boldsymbol{x})]$$

where

$$g(\boldsymbol{x}) = b + \sum_d w_d x_d = b + \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}$$

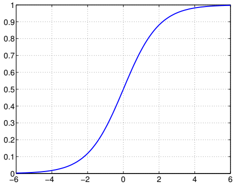
# Why the sigmoid function?

## What does it look like?

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

where

$$a = b + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$$




## Properties

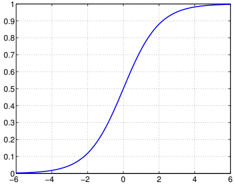
# Why the sigmoid function?

## What does it look like?

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

where

$$a = b + \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}$$




## Properties

- Bounded between 0 and 1 $\leftarrow$ thus, interpretable as probability
- Monotonically increasing thus, usable to derive classification rules
  - $\sigma(a) > 0.5$, positive (classify as '1')
  - $\sigma(a) < 0.5$, negative (classify as '0')
  - $\sigma(a) = 0.5$, undecidable
- Nice computational properties Derivative is in a simple form

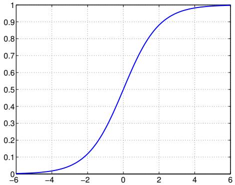
# Why the sigmoid function?

## What does it look like?

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

where

$$a = b + \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}$$




## Properties

- Bounded between 0 and 1 $\leftarrow$ thus, interpretable as probability
- Monotonically increasing thus, usable to derive classification rules
  - $\sigma(a) > 0.5$, positive (classify as '1')
  - $\sigma(a) < 0.5$, negative (classify as '0')
  - $\sigma(a) = 0.5$, undecidable
- Nice computational properties Derivative is in a simple form

## Linear or nonlinear classifier?

# Linear or nonlinear?

$\sigma(a)$ **is nonlinear**, however, the decision boundary is determined by

$$\sigma(a) = 0.5 \Rightarrow a = 0 \Rightarrow g(\boldsymbol{x}) = b + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} = 0$$

which is a *linear* function in $\boldsymbol{x}$

We often call $b$ the offset term.

# Likelihood function

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

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

# Likelihood function

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

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

**Compact expression, exploring that** $y_n$ **is either 1 or 0**

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

# Maximum likelihood estimation

**Cross-entropy error (negative log-likelihood)**

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

**Numerical optimization**

- Gradient descent: simple, scalable to large-scale problems
- Newton method: fast but not scalable

# Numerical optimization

## Gradient descent

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

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

# Numerical optimization

## Gradient descent

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

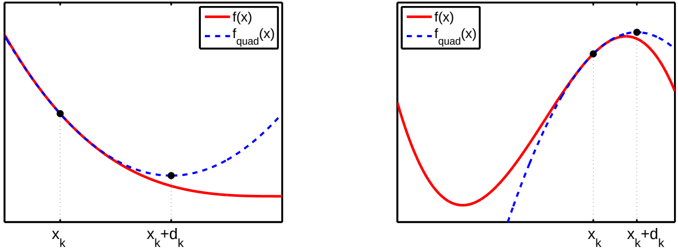
$$\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} - \eta \sum_n \left\{ \sigma(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n) - y_n \right\} \boldsymbol{x}_n$$

## Remarks

- Gradient is direction of steepest ascent.
- The step size needs to be chosen carefully to ensure convergence.
- The step size can be adaptive (i.e. varying from iteration to iteration).
- Variant called *stochastic* gradient descent (later this quarter).

# Intuition for Newton's method

**Approximate the true function with an easy-to-solve optimization problem**




In particular, we can approximate the cross-entropy error function around $\boldsymbol{w}^{(t)}$ by a quadratic function (its second order Taylor expansion), and then minimize this quadratic function

# Update Rules

**Gradient descent**

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

**Newton method**

$$\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} - {\boldsymbol{H}^{(t)}}^{-1} \nabla \mathcal{E}(\boldsymbol{w}^{(t)})$$

# Contrast gradient descent and Newton's method

# Contrast gradient descent and Newton's method

**Similar**

- Both are iterative procedures.

**Different**

- Newton's method requires second-order derivatives (less scalable, but faster convergence)
- Newton's method does not have the magic $\eta$ to be set

# Outline

# Naive Bayes and logistic regression: two different modelling paradigms

Consider spam classification problem

- First Strategy:
  - Use training set to find a decision boundary in the feature space that separates spam and non-spam emails
  - Given a test point, predict its label based on which side of the boundary it is on.

# Naive Bayes and logistic regression: two different modelling paradigms

Consider spam classification problem

- First Strategy:
  - Use training set to find a decision boundary in the feature space that separates spam and non-spam emails
  - Given a test point, predict its label based on which side of the boundary it is on.
- Second Strategy:
  - Look at spam emails and build a model of what they look like. Similarly, build a model of what non-spam emails look like.
  - To classify a new email, match it against both the spam and non-spam models to see which is the better fit.

# Naive Bayes and logistic regression: two different modelling paradigms

Consider spam classification problem

- First Strategy:
  - Use training set to find a decision boundary in the feature space that separates spam and non-spam emails
  - Given a test point, predict its label based on which side of the boundary it is on.
- Second Strategy:
  - Look at spam emails and build a model of what they look like. Similarly, build a model of what non-spam emails look like.
  - To classify a new email, match it against both the spam and non-spam models to see which is the better fit.

First strategy is discriminative (e.g., logistic regression)
Second strategy is generative (e.g., naive bayes)

# Generative vs Discriminative

## Discriminative

- Requires only specifying a model for the conditional distribution $p(y|x)$, and thus, maximizes the *conditional* likelihood $\sum_n \log p(y_n|\boldsymbol{x}_n)$.
- Models that try to learn mappings directly from feature space to the labels are also discriminative, e.g., perceptron, SVMs (covered later)

# Generative vs Discriminative

## Discriminative

- Requires only specifying a model for the conditional distribution $p(y|x)$, and thus, maximizes the *conditional* likelihood $\sum_n \log p(y_n|\boldsymbol{x}_n)$.
- Models that try to learn mappings directly from feature space to the labels are also discriminative, e.g., perceptron, SVMs (covered later)

## Generative

- Aims to model the joint probability $p(x, y)$ and thus maximize the *joint* likelihood $\sum_n \log p(\boldsymbol{x}_n, y_n)$.
- The generative models we'll cover do so by modeling $p(x|y)$ and $p(y)$

# Generative vs Discriminative

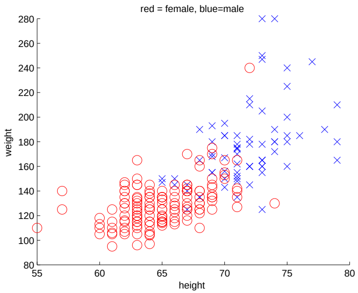
## Discriminative

- Requires only specifying a model for the conditional distribution $p(y|x)$, and thus, maximizes the *conditional* likelihood $\sum_n \log p(y_n|\boldsymbol{x}_n)$.
- Models that try to learn mappings directly from feature space to the labels are also discriminative, e.g., perceptron, SVMs (covered later)

## Generative

- Aims to model the joint probability $p(x, y)$ and thus maximize the *joint* likelihood $\sum_n \log p(\boldsymbol{x}_n, y_n)$.
- The generative models we'll cover do so by modeling $p(x|y)$ and $p(y)$
- Let's look at two more examples: Gaussian (or Quadratic) Discriminative Analysis and Linear Discriminative Analysis

# Determining sex (man or woman) based on measurements

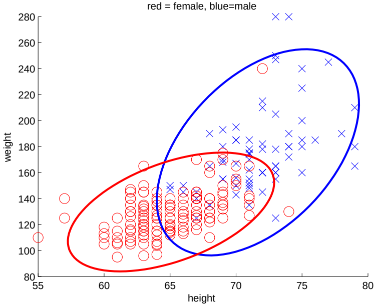


# Generative approach

**Model joint distribution of ($x =$ (height, weight), $y =$sex)**

*our data*

| Sex | Height | Weight |
|---|---|---|
| 1 | $6'$ | $175$ |
| 2 | $5'2"$ | $120$ |
| 1 | $5'6"$ | $140$ |
| 1 | $6'2"$ | $240$ |
| 2 | $5.7"$ | $130$ |
| $\cdots$ | $\cdots$ | $\cdots$ |




Intuition: we will model how heights vary (according to a Gaussian) in each sub-population (male and female).

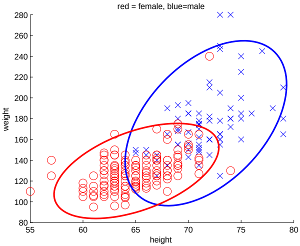
*Note*: This is similar to Naive Bayes (in particular problem 1 of HW2)

# Model of the joint distribution (1D)

$$p(x, y) = p(y)p(x|y)$$

$$= \begin{cases} p_1 \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}} & \text{if } y = 1 \\ p_2 \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{(x-\mu_2)^2}{2\sigma_2^2}} & \text{if } y = 2 \end{cases}$$

$p_1 + p_2 = 1$ are *prior* probabilities, and $p(x|y)$ is a *class conditional distribution*



# Parameter estimation

**Log Likelihood of training data** $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$ with $y_n \in \{1, 2\}$

$$\log P(\mathcal{D}) = \sum_n \log p(x_n, y_n)$$
$$= \sum_{n:y_n=1} \log \left( p_1 \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(x_n-\mu_1)^2}{2\sigma_1^2}} \right)$$
$$+ \sum_{n:y_n=2} \log \left( p_2 \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{(x_n-\mu_2)^2}{2\sigma_2^2}} \right)$$

# Parameter estimation

**Log Likelihood of training data** $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^{N}$ with $y_n \in \{1, 2\}$

$$\begin{aligned}
\log P(\mathcal{D}) &= \sum_n \log p(x_n, y_n) \\
&= \sum_{n:y_n=1} \log \left( p_1 \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(x_n-\mu_1)^2}{2\sigma_1^2}} \right) \\
&+ \sum_{n:y_n=2} \log \left( p_2 \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{(x_n-\mu_2)^2}{2\sigma_2^2}} \right)
\end{aligned}$$

**Max log likelihood** $(p_1^*, p_2^*, \mu_1^*, \mu_2^*, \sigma_1^*, \sigma_2^*) = \arg\max \log P(\mathcal{D})$

# Parameter estimation

**Log Likelihood of training data** $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^{N}$ with $y_n \in \{1, 2\}$

$$\log P(\mathcal{D}) = \sum_n \log p(x_n, y_n)$$

$$= \sum_{n:y_n=1} \log \left( p_1 \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(x_n-\mu_1)^2}{2\sigma_1^2}} \right)$$

$$+ \sum_{n:y_n=2} \log \left( p_2 \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{(x_n-\mu_2)^2}{2\sigma_2^2}} \right)$$

**Max log likelihood** $(p_1^*, p_2^*, \mu_1^*, \mu_2^*, \sigma_1^*, \sigma_2^*) = \arg\max \log P(\mathcal{D})$

- In HW2 Problem 1 we look at variant of Naive Bayes where $\sigma_1^* = \sigma_2^*$

# Parameter estimation

**Log Likelihood of training data** $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^{N}$ with $y_n \in \{1, 2\}$

$$\log P(\mathcal{D}) = \sum_n \log p(x_n, y_n)$$
$$= \sum_{n: y_n = 1} \log \left( p_1 \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(x_n - \mu_1)^2}{2\sigma_1^2}} \right)$$
$$+ \sum_{n: y_n = 2} \log \left( p_2 \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{(x_n - \mu_2)^2}{2\sigma_2^2}} \right)$$

**Max log likelihood** $(p_1^*, p_2^*, \mu_1^*, \mu_2^*, \sigma_1^*, \sigma_2^*) = \arg\max \log P(\mathcal{D})$

- In HW2 Problem 1 we look at variant of Naive Bayes where $\sigma_1^* = \sigma_2^*$

**Max likelihood ($D > 1$)** $(p_1^*, p_2^*, \boldsymbol{\mu}_1^*, \boldsymbol{\mu}_2^*, \boldsymbol{\Sigma}_1^*, \boldsymbol{\Sigma}_2^*) = \arg\max \log P(\mathcal{D})$

# Parameter estimation

**Log Likelihood of training data** $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$ with $y_n \in \{1, 2\}$

$$\log P(\mathcal{D}) = \sum_n \log p(x_n, y_n)$$
$$= \sum_{n:y_n=1} \log \left( p_1 \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(x_n-\mu_1)^2}{2\sigma_1^2}} \right)$$
$$+ \sum_{n:y_n=2} \log \left( p_2 \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{(x_n-\mu_2)^2}{2\sigma_2^2}} \right)$$

**Max log likelihood** $(p_1^*, p_2^*, \mu_1^*, \mu_2^*, \sigma_1^*, \sigma_2^*) = \arg\max \log P(\mathcal{D})$

- In HW2 Problem 1 we look at variant of Naive Bayes where $\sigma_1^* = \sigma_2^*$

**Max likelihood ($D > 1$)** $(p_1^*, p_2^*, \boldsymbol{\mu}_1^*, \boldsymbol{\mu}_2^*, \boldsymbol{\Sigma}_1^*, \boldsymbol{\Sigma}_2^*) = \arg\max \log P(\mathcal{D})$

- For Naive Bayes we assume $\boldsymbol{\Sigma}_i^*$ is diagonal

# Decision boundary

**As before, the Bayes optimal one under the assumed joint distribution depends on**

$$p(y = 1|x) \geq p(y = 2|x)$$

which is equivalent to

$$p(x|y = 1)p(y = 1) \geq p(x|y = 2)p(y = 2)$$

# Decision boundary

**As before, the Bayes optimal one under the assumed joint distribution depends on**

$$p(y = 1|x) \geq p(y = 2|x)$$

which is equivalent to

$$p(x|y = 1)p(y = 1) \geq p(x|y = 2)p(y = 2)$$

Namely,

$$-\frac{(x - \mu_1)^2}{2\sigma_1^2} - \log\sqrt{2\pi}\sigma_1 + \log p_1 \geq -\frac{(x - \mu_2)^2}{2\sigma_2^2} - \log\sqrt{2\pi}\sigma_2 + \log p_2$$

# Decision boundary

**As before, the Bayes optimal one under the assumed joint distribution depends on**

$$p(y = 1|x) \geq p(y = 2|x)$$

which is equivalent to

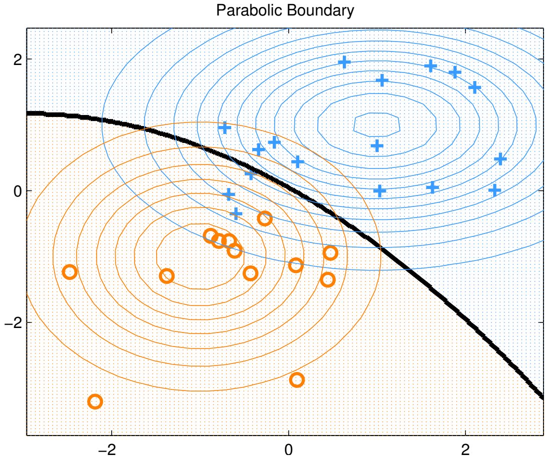
$$p(x|y = 1)p(y = 1) \geq p(x|y = 2)p(y = 2)$$

Namely,

$$-\frac{(x-\mu_1)^2}{2\sigma_1^2} - \log\sqrt{2\pi}\sigma_1 + \log p_1 \geq -\frac{(x-\mu_2)^2}{2\sigma_2^2} - \log\sqrt{2\pi}\sigma_2 + \log p_2$$

$\Rightarrow ax^2 + bx + c \geq 0$ $\leftarrow$ the decision boundary not *linear*!

# Example of nonlinear decision boundary




*Note*: the boundary is characterized by a quadratic function, giving rise to the shape of a parabolic curve.

# A special case: what if we assume the two Gaussians have the same variance?

$$-\frac{(x-\mu_1)^2}{2\sigma_1^2} - \log \sqrt{2\pi}\sigma_1 + \log p_1 \geq -\frac{(x-\mu_2)^2}{2\sigma_2^2} - \log \sqrt{2\pi}\sigma_2 + \log p_2$$

with $\sigma_1 = \sigma_2$

# A special case: what if we assume the two Gaussians have the same variance?

$$-\frac{(x-\mu_1)^2}{2\sigma_1^2} - \log\sqrt{2\pi}\sigma_1 + \log p_1 \geq -\frac{(x-\mu_2)^2}{2\sigma_2^2} - \log\sqrt{2\pi}\sigma_2 + \log p_2$$

with $\sigma_1 = \sigma_2$

We get a linear decision boundary: $bx + c \geq 0$

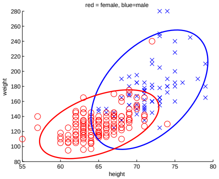
# A special case: what if we assume the two Gaussians have the same variance?

$$-\frac{(x-\mu_1)^2}{2\sigma_1^2} - \log\sqrt{2\pi}\sigma_1 + \log p_1 \geq -\frac{(x-\mu_2)^2}{2\sigma_2^2} - \log\sqrt{2\pi}\sigma_2 + \log p_2$$

with $\sigma_1 = \sigma_2$

We get a linear decision boundary: $bx + c \geq 0$

*Note*: equal variances across two different categories could be a very strong assumption.




For example, from the plot, it does seem that the *male* population has slightly bigger variance (i.e., bigger ellipse) than the *female* population. So the assumption might not be applicable.

# Mini-summary

## Gaussian discriminant analysis

- A generative approach, assuming the data modeled by

$$p(x, y) = p(y)p(x|y)$$

  where $p(x|y)$ is a Gaussian distribution.

- Parameters (of those Gaussian distributions) are estimated by maximizing the likelihood
  - Computationally, estimating those parameters are very easy — it amounts to computing sample mean vectors and covariance matrices
- Decision boundary
  - In general, nonlinear functions of $x$ — in this case, we call the approach *quadratic discriminant analysis*
  - In the special case we assume equal variance of the Gaussian distributions, we get a linear decision boundary — we call the approach *linear discriminant analysis*

# So what is the discriminative counterpart?

**Intuition**

The decision boundary in Gaussian discriminant analysis is

$$ax^2 + bx + c = 0$$

**Let us model the conditional distribution analogously**

$$p(y|x) = \sigma[ax^2 + bx + c] = \frac{1}{1 + e^{-(ax^2+bx+c)}}$$

Or, even simpler, going after the decision boundary of linear discriminant analysis

$$p(y|x) = \sigma[bx + c]$$

Both look very similar to logistic regression — i.e. we focus on writing down the *conditional* probability, *not* the joint probability.

# Does this change how we estimate the parameters?

**First change: a smaller number of parameters to estimate**

Our models are only parameterized by $a, b$ and $c$. There is no prior probabilities ($p_1$, $p_2$) or Gaussian distribution parameters ($\mu_1$, $\mu_2$, $\sigma_1$ and $\sigma_2$).

# Does this change how we estimate the parameters?

**First change: a smaller number of parameters to estimate**

Our models are only parameterized by $a, b$ and $c$. There is no prior probabilities ($p_1$, $p_2$) or Gaussian distribution parameters ($\mu_1$, $\mu_2$, $\sigma_1$ and $\sigma_2$).

**Second change: we need to maximize the conditional likelihood $p(y|x)$**

$$(a^*, b^*, c^*) = \arg\min - \sum_n \{y_n \log \sigma(ax_n^2 + bx_n + c) \quad (1)$$

$$+ (1 - y_n) \log[1 - \sigma(ax_n^2 + bx_n + c)]\} \quad (2)$$

*Computationally harder!*

# How easy for our Gaussian discriminant analysis?

**Example**

$$p_1 = \frac{\#\text{ of training samples in class 1}}{\#\text{ of training samples}} \tag{3}$$

$$\mu_1 = \frac{\sum_{n:y_n=1} x_n}{\#\text{ of training samples in class 1}} \tag{4}$$

$$\sigma_1^2 = \frac{\sum_{n:y_n=1}(x_n - \mu_1)^2}{\#\text{ of training samples in class 1}} \tag{5}$$

*Note*: detailed derivation is in the books. They can be generalized rather easily to multi-variate distributions as well as multiple classes.

# Generative versus discriminative: which one to use?

**There is no fixed rule**

- Selecting which type of method to use is dataset/task specific
- It depends on how well your modeling assumption fits the data
- For instance, as we show in HW2, when data follows a specific variant of the Gaussian Naive Bayes assumption, $p(y|x)$ necessarily follows a logistic function. However, the converse is not true.
  - Gaussian Naive Bayes makes a stronger assumption than logistic regression
  - When data follows this assumption, Gaussian Naive Bayes will likely yield a model that better fits the data
  - But logistic regression is more robust and less sensitive to incorrect modelling assumption

# Outline

# Setup

**Suppose we need to 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

**Studied methods**

- Nearest neighbor classifier
- Naive Bayes
- Gaussian discriminant analysis
- Logistic regression

# Logistic regression for predicting multiple classes? Easy

**The approach of "one versus the rest"**

- 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')

  This step is often called *1-of-K* encoding. That is, only one is nonzero and everything else is zero.

  Example: for class $C_2$, data go through the following change

$$(\boldsymbol{x}_1, C_1) \rightarrow (\boldsymbol{x}_1, 0), (\boldsymbol{x}_2, C_3) \rightarrow (\boldsymbol{x}_2, 0), \ldots, (\boldsymbol{x}_n, C_2) \rightarrow (\boldsymbol{x}_n, 1), \ldots,$$

# Logistic regression for predicting multiple classes? Easy

**The approach of "one versus the rest"**

- 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')

  This step is often called *1-of-K* encoding. That is, only one is nonzero and everything else is zero.
  Example: for class $C_2$, data go through the following change

$$(\boldsymbol{x}_1, C_1) \rightarrow (\boldsymbol{x}_1, 0), (\boldsymbol{x}_2, C_3) \rightarrow (\boldsymbol{x}_2, 0), \ldots, (\boldsymbol{x}_n, C_2) \rightarrow (\boldsymbol{x}_n, 1), \ldots,$$

- Train $K$ binary classifiers using logistic regression to differentiate the two classes

# Logistic regression for predicting multiple classes? Easy

**The approach of "one versus the rest"**

- 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')

  This step is often called *1-of-K* encoding. That is, only one is nonzero and everything else is zero.
  Example: for class $C_2$, data go through the following change

$$(\boldsymbol{x}_1, C_1) \rightarrow (\boldsymbol{x}_1, 0), (\boldsymbol{x}_2, C_3) \rightarrow (\boldsymbol{x}_2, 0), \ldots, (\boldsymbol{x}_n, C_2) \rightarrow (\boldsymbol{x}_n, 1), \ldots,$$

- Train $K$ binary classifiers using logistic regression to differentiate the two classes
- When predicting on $\boldsymbol{x}$, combine the outputs of all binary classifiers
  1. What if all the classifiers say NEGATIVE?
  2. What if multiple classifiers say POSITIVE?

# Yet, another easy approach

**The approach of "one versus one"**

- 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

Ex: for class $C_1$ and $C_2$,

$$(\boldsymbol{x}_1, C_1), (\boldsymbol{x}_2, C_3), (\boldsymbol{x}_3, C_2), \ldots \rightarrow (\boldsymbol{x}_1, 1), (\boldsymbol{x}_3, 0), \ldots$$

# Yet, another easy approach

**The approach of "one versus one"**

- 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

  Ex: for class $C_1$ and $C_2$,

$$(\boldsymbol{x}_1, C_1), (\boldsymbol{x}_2, C_3), (\boldsymbol{x}_3, C_2), \ldots \rightarrow (\boldsymbol{x}_1, 1), (\boldsymbol{x}_3, 0), \ldots$$

- Train $K(K-1)/2$ binary classifiers using logistic regression to differentiate the two classes

# Yet, another easy approach

## The approach of "one versus one"

- 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

  Ex: for class $C_1$ and $C_2$,

$$(\boldsymbol{x}_1, C_1), (\boldsymbol{x}_2, C_3), (\boldsymbol{x}_3, C_2), \ldots \rightarrow (\boldsymbol{x}_1, 1), (\boldsymbol{x}_3, 0), \ldots$$

- Train $K(K-1)/2$ binary classifiers using logistic regression to differentiate the two classes
- When predicting on $\boldsymbol{x}$, combine the outputs of all binary classifiers
  There are $K(K-1)/2$ votes!

# Contrast these two approaches

**Pros and cons of each approach**

- *one versus the rest*: only needs to train $K$ classifiers.

# Contrast these two approaches

## Pros and cons of each approach

- *one versus the rest*: only needs to train $K$ classifiers.
  - Makes a *big* difference if you have a lot of *classes* to go through.
  - Can you think of a good application example where there are a lot of classes?

# Contrast these two approaches

**Pros and cons of each approach**

- *one versus the rest*: only needs to train $K$ classifiers.
  - Makes a *big* difference if you have a lot of *classes* to go through.
  - Can you think of a good application example where there are a lot of classes?
- *one versus one*: only needs to train a smaller subset of data (only those labeled with those two classes would be involved).

# Contrast these two approaches

## Pros and cons of each approach

- *one versus the rest*: only needs to train $K$ classifiers.
  - Makes a *big* difference if you have a lot of *classes* to go through.
  - Can you think of a good application example where there are a lot of classes?
- *one versus one*: only needs to train a smaller subset of data (only those labeled with those two classes would be involved).
  - Makes a *big* difference if you have a lot of *data* to go through.

# Contrast these two approaches

**Pros and cons of each approach**

- *one versus the rest*: only needs to train $K$ classifiers.
  - Makes a *big* difference if you have a lot of *classes* to go through.
  - Can you think of a good application example where there are a lot of classes?
- *one versus one*: only needs to train a smaller subset of data (only those labeled with those two classes would be involved).
  - Makes a *big* difference if you have a lot of *data* to go through.

**Bad about both of them**

*Combining classifiers' outputs seem to be a bit tricky*.

Any other good methods?

# Multinomial logistic regression

**Intuition: from the decision rule of our naive Bayes classifier**

$$y^* = \arg\max_c p(y = c|\boldsymbol{x}) = \arg\max_c \log p(\boldsymbol{x}|y = c)p(y = c) \quad (6)$$

$$= \arg\max_c \log \pi_c + \sum_k z_k \log \theta_{ck} = \arg\max_c \boldsymbol{w}_c^{\mathrm{T}} \boldsymbol{x} \quad (7)$$

# Multinomial logistic regression

**Intuition: from the decision rule of our naive Bayes classifier**

$$y^* = \arg\max_c p(y = c|\boldsymbol{x}) = \arg\max_c \log p(\boldsymbol{x}|y = c)p(y = c) \quad (6)$$

$$= \arg\max_c \log \pi_c + \sum_k z_k \log \theta_{ck} = \arg\max_c \boldsymbol{w}_c^{\mathrm{T}} \boldsymbol{x} \quad (7)$$

**Essentially, we are comparing**

$$\boldsymbol{w}_1^{\mathrm{T}} \boldsymbol{x}, \boldsymbol{w}_2^{\mathrm{T}} \boldsymbol{x}, \cdots, \boldsymbol{w}_{\mathsf{C}}^{\mathrm{T}} \boldsymbol{x} \quad (8)$$

with *one* for each category.

# First try

**So, can we define the following conditional model?**

$$p(y = c|\boldsymbol{x}) = \sigma[\boldsymbol{w}_c^{\mathrm{T}}\boldsymbol{x}]$$

# First try

**So, can we define the following conditional model?**

$$p(y = c|\boldsymbol{x}) = \sigma[\boldsymbol{w}_c^{\mathrm{T}}\boldsymbol{x}]$$

This would *not* work at least for the reason

$$\sum_c p(y = c|\boldsymbol{x}) = \sum_c \sigma[\boldsymbol{w}_c^{\mathrm{T}}\boldsymbol{x}] \neq 1$$

as 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?**

$$p(y = c|\boldsymbol{x}) = \sigma[\boldsymbol{w}_c^{\mathrm{T}}\boldsymbol{x}]$$

This would *not* work at least for the reason

$$\sum_c p(y = c|\boldsymbol{x}) = \sum_c \sigma[\boldsymbol{w}_c^{\mathrm{T}}\boldsymbol{x}] \neq 1$$

as each summand can be any number (independently) between 0 and 1.
*But we are close!*
We can learn the $k$ linear models jointly to ensure this property holds!

# Definition of multinomial logistic regression

## Model

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

$$p(C_k|\boldsymbol{x}) = \frac{e^{\boldsymbol{w}_k^{\mathrm{T}}\boldsymbol{x}}}{\sum_{k'} e^{\boldsymbol{w}_{k'}^{\mathrm{T}}\boldsymbol{x}}} \qquad \leftarrow \qquad \text{This is called } \textit{softmax} \text{ function}$$

# Definition of multinomial logistic regression

**Model**

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

$$p(C_k|\boldsymbol{x}) = \frac{e^{\boldsymbol{w}_k^{\mathrm{T}}\boldsymbol{x}}}{\sum_{k'} e^{\boldsymbol{w}_{k'}^{\mathrm{T}}\boldsymbol{x}}} \qquad \leftarrow \qquad \text{This is called } \textit{softmax} \text{ function}$$

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

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

# How does the softmax function behave?

**Suppose we have**

$$\boldsymbol{w}_1^{\mathrm{T}}\boldsymbol{x} = 100, \boldsymbol{w}_2^{\mathrm{T}}\boldsymbol{x} = 50, \boldsymbol{w}_3^{\mathrm{T}}\boldsymbol{x} = -20$$

# How does the softmax function behave?

**Suppose we have**

$$\boldsymbol{w}_1^{\mathrm{T}}\boldsymbol{x} = 100, \boldsymbol{w}_2^{\mathrm{T}}\boldsymbol{x} = 50, \boldsymbol{w}_3^{\mathrm{T}}\boldsymbol{x} = -20$$

We could have picked the *winning* class label $1$ with certainty according to our classification rule.

**Softmax translates these scores into well-formed conditional probababilities**

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

- 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 $K = 2$

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

*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 $\boldsymbol{y}_n = [y_{n1}\ y_{n2}\ \cdots\ y_{nK}]^{\mathrm{T}}$, 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, $\boldsymbol{y}_n = [0\ 1\ 0\ 0\ \cdots\ 0]^{\mathrm{T}}$.

# 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 $\boldsymbol{y}_n = [y_{n1} \; y_{n2} \; \cdots \; y_{nK}]^{\mathrm{T}}$, 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, $\boldsymbol{y}_n = [0 \; 1 \; 0 \; 0 \; \cdots \; 0]^{\mathrm{T}}$.

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

# Cross-entropy error function

**Definition**: negative log likelihood

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

# Cross-entropy error function

**Definition**: negative log likelihood

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

**Properties**

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