EECS 545: Machine Learning
Lecture 5. Classification 2

Honglak Lee

1/26/2026

UNIVERSITY OF MICHIGAN

Logistics/Announcements

- HW 2 will be released on 1/27, due on Feb 10 th
- Project Proposal due Feb 3rd
- HW 1 will be due tomorrow on 1/27 at 11:55 pm
  - No point deduction for HW1 through Jan 30th, Friday
    - When you still have tokens, each day consumes 1 late day token
    - When you run out of late day tokens, -10 points per late day
    - No submission will be accepted after 1/30

Outline

- Probabilistic Discriminative models
  - Objective: maximize **conditional likelihood** over training data

$$\prod_i P(y^{(i)}|\mathbf{x}^{(i)}, \mathbf{w})$$

  - Logistic Regression (covered in previous lecture)
  - Softmax Regression: Multiclass extension of logistic regression (covered in previous lecture)
- Probabilistic Generative models
  - Objective: maximize joint likelihood over training data

$$\prod_i P(\mathbf{x}^{(i)}, y^{(i)}|\mathbf{w})$$

  - Gaussian Discriminant Analysis
  - Naive Bayes
- Discriminant functions (non-probabilistic)
  - Perceptron learning algorithm

Probabilistic Generative Models

Learning the Classifier

- For classification, we want to compute $p(C_k \mid \mathbf{x})$
  (a) **Discriminative** models: Directly model $p(C_k \mid \mathbf{x})$ and learn parameters from the training set.
    - Logistic regression
    - Softmax regression
  (b) **Generative** models: Learn joint densities $p(\mathbf{x}, C_k)$ by learning $p(\mathbf{x} \mid C_k)$ and $p(C_k)$, and then use Bayes rule for predicting the class C_k given $\mathbf{x}$:
    - Gaussian Discriminant Analysis
    - Naive Bayes

Probabilistic Generative Models

- Bayes' theorem reduces the classification problem $p(C_k \mid \mathbf{x})$ to estimating the distribution of the data:

$$p(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)p(C_k)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|C_k)p(C_k)}{\sum_{k'} p(\mathbf{x}|C_{k'})p(C_{k'})}$$

- Density estimation can be decomposed into learning distributions from training data.
  - $p(C_k)$
  - $p(\mathbf{x} \mid C_k)$
- Maximum likelihood estimation for $p(\mathbf{x}, C_k)$

Probabilistic Generative Models

- For two classes, Bayes' theorem says:

$$p(C_1|\mathbf{x}) = \frac{p(\mathbf{x}|C_1)p(C_1)}{p(\mathbf{x}|C_1)p(C_1) + p(\mathbf{x}|C_2)p(C_2)}$$

Probabilistic Generative Models

- For two classes, Bayes' theorem says:

$$p(C_1|\mathbf{x}) = \frac{p(\mathbf{x}|C_1)p(C_1)}{p(\mathbf{x}|C_1)p(C_1) + p(\mathbf{x}|C_2)p(C_2)}$$

- Use *log odds* (i.e., logit "score"):

$$a = \log \frac{p(C_1|\mathbf{x})}{p(C_2|\mathbf{x})} = \log \frac{p(\mathbf{x}|C_1)p(C_1)}{p(\mathbf{x}|C_2)p(C_2)}$$

Probabilistic Generative Models

- For two classes, Bayes' theorem says:

$$p(C_1|\mathbf{x}) = \frac{p(\mathbf{x}|C_1)p(C_1)}{p(\mathbf{x}|C_1)p(C_1) + p(\mathbf{x}|C_2)p(C_2)}$$

- Use *log odds* (i.e., logit "score"):

$$a = \log \frac{p(C_1|\mathbf{x})}{p(C_2|\mathbf{x})} = \log \frac{p(\mathbf{x}|C_1)p(C_1)}{p(\mathbf{x}|C_2)p(C_2)}$$

- Then we can define the posterior via the *sigmoid*:

$$p(C_1|\mathbf{x}) = \frac{1}{1 + \exp(-a)} = \sigma(a)$$

Gaussian Discriminant Analysis

Gaussian Discriminant Analysis

- Probability of class label
  - $p(C_k)$: Constant (e.g., Bernoulli)
- Conditional probability of data given a class
  - $p(\mathbf{x} \mid C_k)$: Gaussian distribution

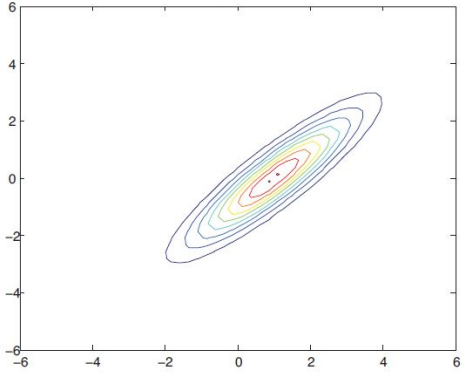
$$p(\mathbf{x} \mid C_k) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\mathbf{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x} - \mu_k)^\top \mathbf{\Sigma}^{-1}(\mathbf{x} - \mu_k)\right\}$$

- Classification: use Bayes rule (previous slide)

Examples of Gaussian Distributions

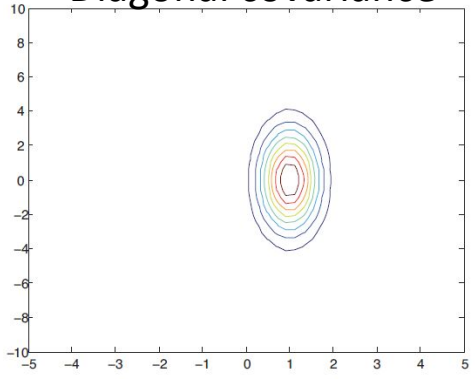
- Probability density p(x) for 2 dimensional case

Full covariance



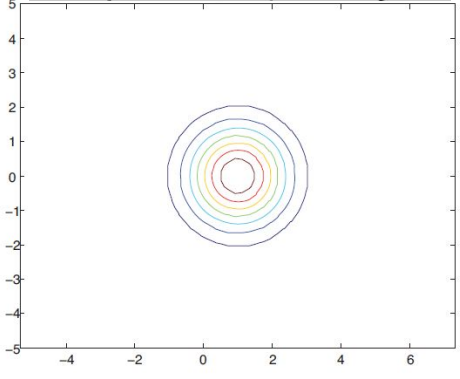
(a)

Diagonal covariance

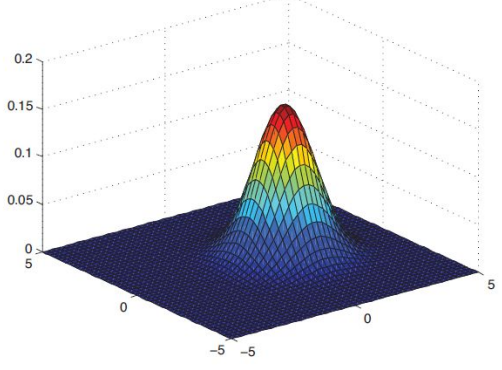


(b

Spherical ($\Sigma \propto I$)

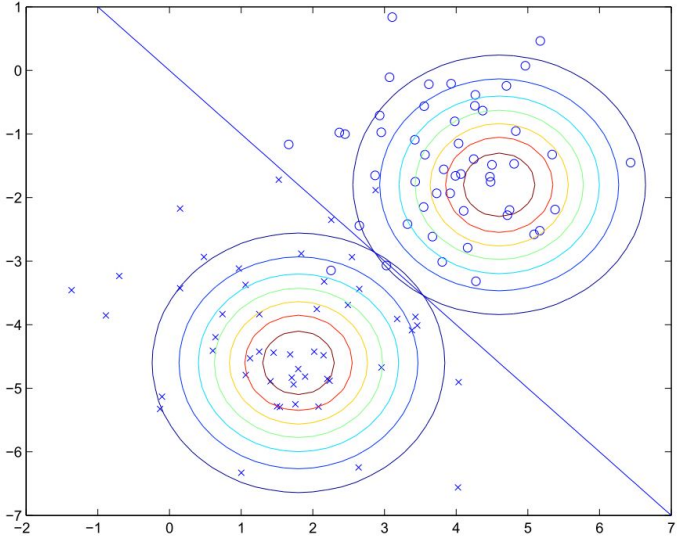


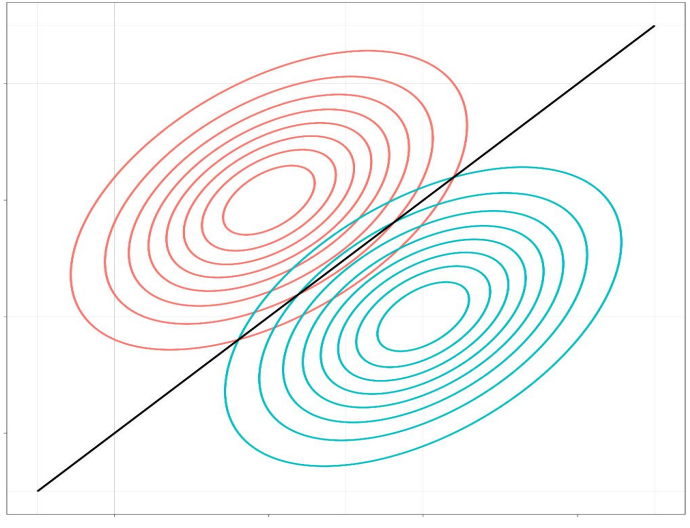
Spherical (3d plot)



Gaussian Discriminant Analysis

- Basic GDA assumes the same covariance for all classes
  - The figure below shows class-specific density and decision boundary. Note the linear decision boundary for any types of covariance matrices!





Prediction: Class-Conditional Densities

- Suppose we model $p(\mathbf{x} \mid C_k)$ as Gaussians with the same covariance matrix.

$$p(\mathbf{x} \mid C_k) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\mathbf{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mu_k)^\top \mathbf{\Sigma}^{-1}(\mathbf{x}-\mu_k)\right\}$$

- This gives us $p(C_1 \mid \mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + w_0)$
  - where $\mathbf{w} = \mathbf{\Sigma}^{-1}(\mu_1 - \mu_2)$

  and $w_0 = -\frac{1}{2}\mu_1^\top \mathbf{\Sigma}^{-1}\mu_1 + \frac{1}{2}\mu_2^\top \mathbf{\Sigma}^{-1}\mu_2 + \log\frac{p(C_1)}{p(C_2)}$

Derivation

$$
\begin{aligned}
P(x, C_1) &= P(x \mid C_1) P(C_1) \\
&= \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu_1)^\top \Sigma^{-1}(x-\mu_1)\right\} P(C_1) \\
P(x, C_2) &= P(x \mid C_2) P(C_2) \\
&= \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu_2)^\top \Sigma^{-1}(x-\mu_2)\right\} P(C_2)
\end{aligned}
$$

Derivation

$$
\begin{aligned}
P(x, C_1) &= P(x \mid C_1)\, P(C_1) \\
&= \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu_1)^\top \Sigma^{-1}(x-\mu_1)\right\} P(C_1) \\
P(x, C_2) &= P(x \mid C_2)\, P(C_2) \\
&= \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu_2)^\top \Sigma^{-1}(x-\mu_2)\right\} P(C_2)
\end{aligned}
$$

$$
\log \frac{P(C_1 \mid \mathbf{x})}{P(C_2 \mid \mathbf{x})} = \log \frac{P(C_1 \mid \mathbf{x})}{1 - P(C_1 \mid \mathbf{x})}
$$

"Log-odds"

Derivation

$$
\begin{aligned}
P(x, C_1) &= P(x \mid C_1)\, P(C_1) \\
&= \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu_1)^\top \Sigma^{-1}(x-\mu_1)\right\} P(C_1) \\
P(x, C_2) &= P(x \mid C_2)\, P(C_2) \\
&= \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu_2)^\top \Sigma^{-1}(x-\mu_2)\right\} P(C_2)
\end{aligned}
$$

$$
\log \frac{P(C_1 \mid \mathbf{x})}{P(C_2 \mid \mathbf{x})} = \log \frac{P(C_1 \mid \mathbf{x})}{1 - P(C_1 \mid \mathbf{x})}
$$

"Log-odds"

$$
= \log \frac{\exp\left\{-\frac{1}{2}(\mathbf{x}-\mu_1)^\top \Sigma^{-1}(\mathbf{x}-\mu_1)\right\}}{\exp\left\{-\frac{1}{2}(\mathbf{x}-\mu_2)^\top \Sigma^{-1}(\mathbf{x}-\mu_2)\right\}} + \log \frac{P(C_1)}{P(C_2)}
$$

Derivation

$$
\begin{aligned}
P(x, C_1) &= P(x \mid C_1) P(C_1) \\
&= \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu_1)^\top \Sigma^{-1}(x-\mu_1)\right\} P(C_1) \\
P(x, C_2) &= P(x \mid C_2) P(C_2) \\
&= \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu_2)^\top \Sigma^{-1}(x-\mu_2)\right\} P(C_2)
\end{aligned}
$$

$$
\log \frac{P(C_1 \mid \mathbf{x})}{P(C_2 \mid \mathbf{x})} = \log \frac{P(C_1 \mid \mathbf{x})}{1 - P(C_1 \mid \mathbf{x})}
$$

"Log-odds"

$$
\begin{aligned}
&= \log \frac{\exp\left\{-\frac{1}{2}(\mathbf{x}-\mu_1)^\top \Sigma^{-1}(\mathbf{x}-\mu_1)\right\}}{\exp\left\{-\frac{1}{2}(\mathbf{x}-\mu_2)^\top \Sigma^{-1}(\mathbf{x}-\mu_2)\right\}} + \log \frac{P(C_1)}{P(C_2)} \\
&= \left\{-\frac{1}{2}(\mathbf{x}-\mu_1)^\top \Sigma^{-1}(\mathbf{x}-\mu_1)\right\} - \left\{-\frac{1}{2}(\mathbf{x}-\mu_2)^\top \Sigma^{-1}(\mathbf{x}-\mu_2)\right\} + \log \frac{P(C_1)}{P(C_2)}
\end{aligned}
$$

Derivation

$$
\begin{aligned}
P(x, C_1) &= P(x \mid C_1) P(C_1) \\
&= \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu_1)^\top \Sigma^{-1}(x-\mu_1)\right\} P(C_1) \\
P(x, C_2) &= P(x \mid C_2) P(C_2) \\
&= \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu_2)^\top \Sigma^{-1}(x-\mu_2)\right\} P(C_2)
\end{aligned}
$$

$$
\log \frac{P(C_1 \mid \mathbf{x})}{P(C_2 \mid \mathbf{x})} = \log \frac{P(C_1 \mid \mathbf{x})}{1 - P(C_1 \mid \mathbf{x})}
$$

"Log-odds"

$$
\begin{aligned}
&= \log \frac{\exp\left\{-\frac{1}{2}(\mathbf{x}-\mu_1)^\top \Sigma^{-1}(\mathbf{x}-\mu_1)\right\}}{\exp\left\{-\frac{1}{2}(\mathbf{x}-\mu_2)^\top \Sigma^{-1}(\mathbf{x}-\mu_2)\right\}} + \log \frac{P(C_1)}{P(C_2)} \\
&= \left\{-\frac{1}{2}(\mathbf{x}-\mu_1)^\top \Sigma^{-1}(\mathbf{x}-\mu_1)\right\} - \left\{-\frac{1}{2}(\mathbf{x}-\mu_2)^\top \Sigma^{-1}(\mathbf{x}-\mu_2)\right\} + \log \frac{P(C_1)}{P(C_2)} \\
&= (\mu_1-\mu_2)^\top \Sigma^{-1}\mathbf{x} - \frac{1}{2}\mu_1^\top \Sigma^{-1}\mu_1 + \frac{1}{2}\mu_2^\top \Sigma^{-1}\mu_2 + \log \frac{P(C_1)}{P(C_2)}
\end{aligned}
$$

Derivation

$$
\begin{aligned}
P(x, C_1) &= P(x \mid C_1)\, P(C_1) \\
&= \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu_1)^\top \Sigma^{-1}(x-\mu_1)\right\} P(C_1) \\
P(x, C_2) &= P(x \mid C_2)\, P(C_2) \\
&= \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu_2)^\top \Sigma^{-1}(x-\mu_2)\right\} P(C_2)
\end{aligned}
$$

$$
\log \frac{P(C_1 \mid \mathbf{x})}{P(C_2 \mid \mathbf{x})} = \log \frac{P(C_1 \mid \mathbf{x})}{1 - P(C_1 \mid \mathbf{x})}
$$

"Log-odds"

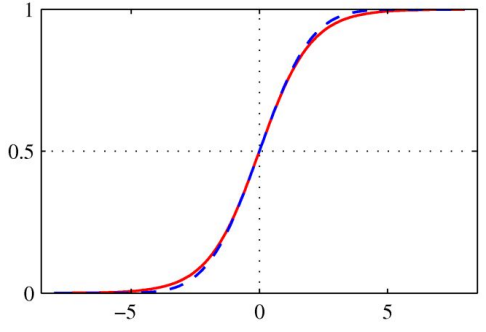
$$
\begin{aligned}
&= \log \frac{\exp\left\{-\frac{1}{2}(\mathbf{x}-\mu_1)^\top \Sigma^{-1}(\mathbf{x}-\mu_1)\right\}}{\exp\left\{-\frac{1}{2}(\mathbf{x}-\mu_2)^\top \Sigma^{-1}(\mathbf{x}-\mu_2)\right\}} + \log \frac{P(C_1)}{P(C_2)} \\
&= \left\{-\frac{1}{2}(\mathbf{x}-\mu_1)^\top \Sigma^{-1}(\mathbf{x}-\mu_1)\right\} - \left\{-\frac{1}{2}(\mathbf{x}-\mu_2)^\top \Sigma^{-1}(\mathbf{x}-\mu_2)\right\} + \log \frac{P(C_1)}{P(C_2)} \\
&= (\mu_1-\mu_2)^\top \Sigma^{-1}\mathbf{x} - \frac{1}{2}\mu_1^\top \Sigma^{-1}\mu_1 + \frac{1}{2}\mu_2^\top \Sigma^{-1}\mu_2 + \log \frac{P(C_1)}{P(C_2)} \\
&= \left(\Sigma^{-1}(\mu_1-\mu_2)\right)^\top \mathbf{x} + w_0
\end{aligned}
$$

$$
\text{where } w_0 = -\frac{1}{2}\mu_1^\top \boldsymbol{\Sigma}^{-1}\mu_1 + \frac{1}{2}\mu_2^\top \boldsymbol{\Sigma}^{-1}\mu_2 + \log \frac{p(C_1)}{p(C_2)}
$$

Prediction: Class-Conditional Densities for shared covariances

- $p(C_k \mid \mathbf{x})$ is a sigmoid function:

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



  - with log-odds (*logit* function):

$$a = \log\left(\frac{\sigma}{1-\sigma}\right) = \left(\boldsymbol{\Sigma}^{-1}(\mu_1 - \mu_2)\right)^{\top}\mathbf{x} + w_0$$

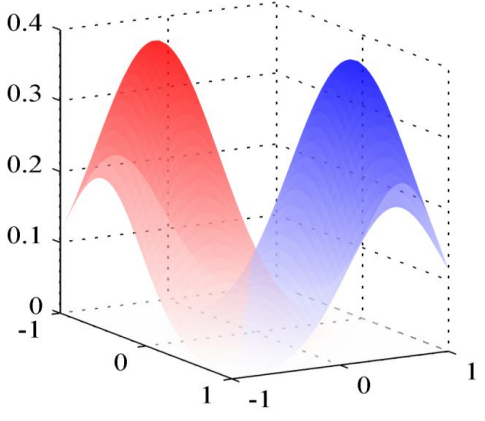
$$\text{where } w_0 = -\frac{1}{2}\mu_1^{\top}\boldsymbol{\Sigma}^{-1}\mu_1 + \frac{1}{2}\mu_2^{\top}\boldsymbol{\Sigma}^{-1}\mu_2 + \log\frac{p(C_1)}{p(C_2)}$$

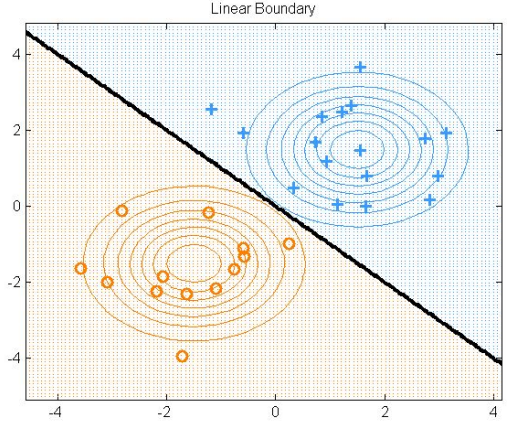
- Generalizes to *normalized exponential*, or *softmax* :

$$p_i = \frac{\exp(q_i)}{\sum_j \exp(q_j)}$$

Prediction: Linear Decision Boundaries

- At decision boundary, we have $p(C_1|\,\mathrm{x}) = p(C_2|\,\mathrm{x})$
- With the same covariance matrices, the boundary $p(C_1|\,\mathrm{x}) = p(C_2|\,\mathrm{x})$ is linear.
  - Different class $p(C_1)$, $p(C_2)$ just shift it around.





Likelihood function of generative models

- The likelihood of Data $\{(\mathbf{x}^{(n)}, y^{(n)})\}$

$$P(D|\mathbf{w}) = \prod_{i=1}^{N} P(\mathbf{x}^{(i)}, y^{(i)}|\mathbf{w}) \longrightarrow P(\mathbf{X}, \mathbf{y}|\mathbf{w})$$

Compact notation:
This is called joint likelihood.

Decomposition of the joint probability

$$= \prod_{i=1}^{N} P(\mathbf{x}^{(i)}|y^{(i)}, \mathbf{w}) P(y^{(i)}|\mathbf{w})$$

Learning parameters via maximum likelihood

- Given training data $\{(\mathbf{x}^{(1)}, y^{(1)}), \cdots, (\mathbf{x}^{(N)}, y^{(N)})\}$ and a generative model ("shared covariance")

$$p(y) = \phi^y (1-\phi)^{1-y}$$

$$p(\mathbf{x}|y=0) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu_0)^\top \Sigma^{-1}(\mathbf{x}-\mu_0)\right)$$

$$p(\mathbf{x}|y=1) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu_1)^\top \Sigma^{-1}(\mathbf{x}-\mu_1)\right)$$

Learning via maximum likelihood

- Maximum likelihood estimation (HW2):

$$\phi = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\{y^{(i)} = 1\}$$

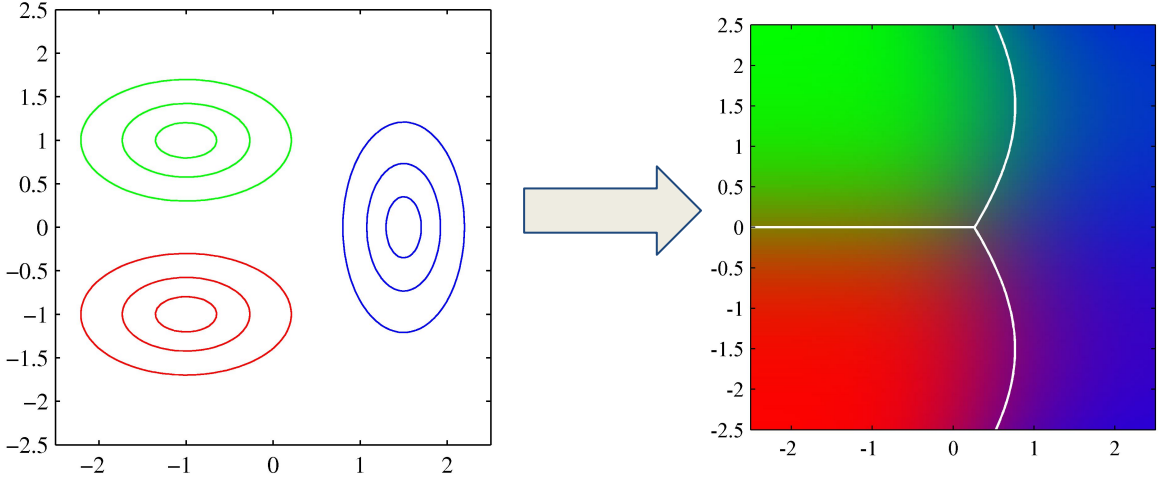
$$\mu_0 = \frac{\sum_{i=1}^{N} \mathbb{I}\{y^{(i)} = 0\}\mathbf{x}^{(i)}}{\sum_{i=1}^{N} \mathbb{I}\{y^{(i)} = 0\}}$$

$$\mu_1 = \frac{\sum_{i=1}^{N} \mathbb{I}\{y^{(i)} = 1\}\mathbf{x}^{(i)}}{\sum_{i=1}^{N} \mathbb{I}\{y^{(i)} = 1\}}$$

$$\Sigma = \frac{1}{N}\sum_{i=1}^{N} (\mathbf{x}^{(i)} - \mu_{y^{(i)}})(\mathbf{x}^{(i)} - \mu_{y_{(i)}})^\top$$

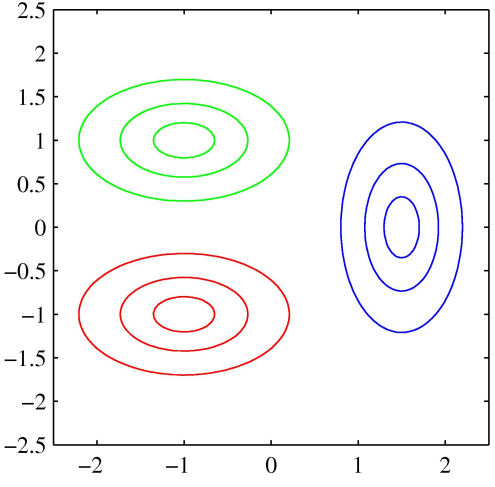
Different Covariance

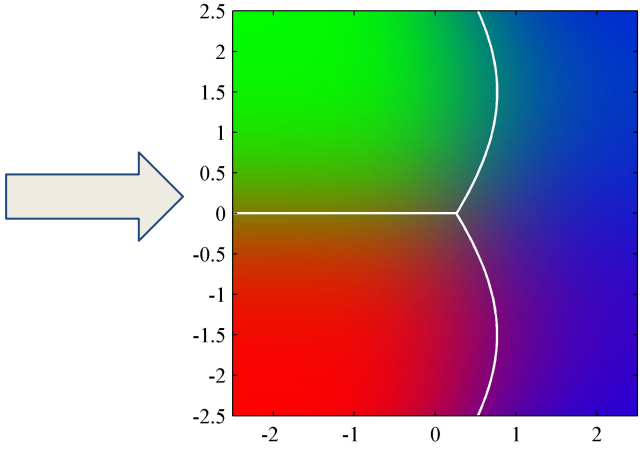
- Decision boundaries between some classes can be quadratic when they have **different** covariances.

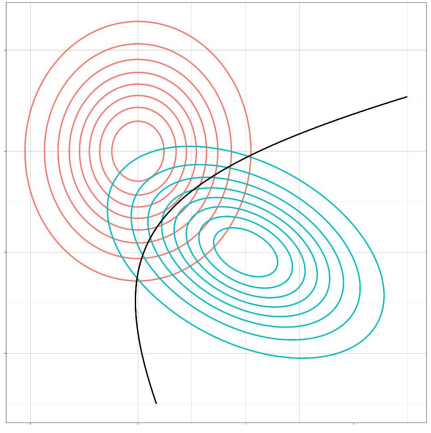


Different Covariance

- Decision boundaries between some classes can be quadratic when they have **different** covariances.







Comparison between GDA and Logistic regression (or softmax regression)

- Logistic regression:
  - For an M-dimensional feature space, this model has M parameters to fit.
- Gaussian Discriminative Analysis (*assuming x is M-dim.*)
  - $2M$ parameters for the means of $p(\mathbf{x} \mid C_1)$ and $p(\mathbf{x} \mid C_2)$
  - $M(M+1)/2$ parameters for the shared covariance matrix
- Logistic regression has less parameters and is more flexible about data distribution.
- GDA has a stronger modeling assumption, and works well when the distribution follows the assumption.

Naive Bayes Classifier

Naive Bayes classifier

- Probability of class label:
  - $p(C_k)$: Constant (e.g., Bernoulli)
- Conditional probability of data given the class
  - Naive Bayes assumption: $p(\mathbf{x} \mid C_k)$ is factorized (Each coordinate of $\mathbf{x}$ is conditionally independent of other coordinates given the class label)

$$P(x_1, ..., x_M|C_k) = P(x_1|C_k) \cdots P(x_M|C_k) = \prod_{j=1}^{M} P(x_j|C_k)$$

- Classification: use Bayes rule

(binary) $$P(C_1|\mathbf{x}) = \frac{P(C_1, \mathbf{x})}{P(\mathbf{x})} = \frac{P(C_1, \mathbf{x})}{P(C_1, \mathbf{x}) + P(C_2, \mathbf{x})}$$

Naive Bayes classifier

- When classifying, we can simply find the class C_k that maximizes $P(C_k|\mathbf{x})$ using the Bayes rule:

$$\arg\max_k P(C_k|\mathbf{x}) = \arg\max_k P(C_k, \mathbf{x})$$

Naive Bayes classifier

- When classifying, we can simply find the class C_k that maximizes $P(C_k|\mathbf{x})$ using the Bayes rule:

$$\begin{aligned}\arg\max_k P(C_k|\mathbf{x}) &= \arg\max_k P(C_k, \mathbf{x}) \\ &= \arg\max_k P(C_k)P(\mathbf{x}|C_k)\end{aligned}$$

Naive Bayes classifier

- When classifying, we can simply find the class C_k that maximizes $P(C_k|\mathbf{x})$ using the Bayes rule:

$$\begin{aligned}
\arg\max_k P(C_k|\mathbf{x}) &= \arg\max_k P(C_k, \mathbf{x}) \\
&= \arg\max_k P(C_k) P(\mathbf{x}|C_k) \\
&= \arg\max_k P(C_k) \prod_{j=1}^{M} P(x_j|C_k)
\end{aligned}$$

Naive Bayes assumption

Example 1: Naive Bayes for real-valued inputs

- Probability of class label:
  - $p(C_k)$: Constant (e.g., Bernoulli)
- Conditional probability of data given the class
  - Naive Bayes assumption: $P(\mathbf{x}|C_k)$ is factorized (e.g., 1D Gaussian)

$$
\begin{aligned}
P(x_1, \ldots, x_M|C_k) &= P(x_1|C_k) \cdots P(x_M|C_k) \\
&= \prod_{j=1}^{M} P(x_j|C_k) \\
&= \prod_{j=1}^{M} \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}\right)
\end{aligned}
$$

  - Note: this is equivalent to GDA with diagonal covariance!!

Example 2: Spam mail classification

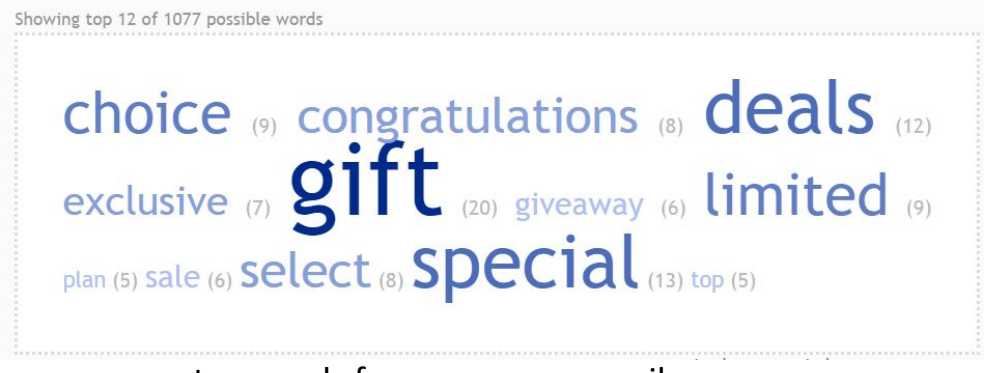
- Label: y=1 (spam), y=0 (non-spam)
- Features:
  - x_j: *j*-th word in the mail, where M is the vocabulary size.
  - Multinomial variable (M-dimensional binary vector with only one coordinate with 1)
- Naive Bayes Assumption:
  - Given a class label y, each word in a mail is a independent multinomial variable.

Naive Bayes Spam classifier

- Model

$$P(\text{spam}) = Bernoulli(\phi)$$
$$P(\text{word}|\text{spam}) = Multinomial(\mu_1^s, \ldots, \mu_M^s)$$
$$P(\text{word}|\text{nonspam}) = Multinomial(\mu_1^{ns}, \ldots, \mu_M^{ns})$$

top words from my **non-spam** emails



top words from my **spam** emails

Naive Bayes Spam classifier

- Model

$$P(\text{spam}) = Bernoulli(\phi)$$
$$P(\text{word}|\text{spam}) = Multinomial(\mu_1^s, \ldots, \mu_M^s)$$
$$P(\text{word}|\text{nonspam}) = Multinomial(\mu_1^{ns}, \ldots, \mu_M^{ns})$$

- Goal

  Find ϕ, μ^s, μ^{ns} that best fits the data $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})\}$

Naive Bayes Spam classifier

- Model

$$P(\text{spam}) = Bernoulli(\phi)$$
$$P(\text{word}|\text{spam}) = Multinomial(\mu_1^s, \ldots, \mu_M^s)$$
$$P(\text{word}|\text{nonspam}) = Multinomial(\mu_1^{ns}, \ldots, \mu_M^{ns})$$

- Goal

  Find ϕ, μ^s, μ^{ns} that best fits the data $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})\}$ by maximizing the joint likelihood:

$$\prod_{i=1}^{N} P(\mathbf{x}^{(i)}, y^{(i)})$$

- Joint Likelihood (joint probability of inputs/labels)
  - Note that the joint likelihood is conditioned on parameters ϕ, μ^s, μ^{ns}

Naive Bayes Spam classifier

- Model

$$P(\text{spam}) = Bernoulli(\phi)$$
$$P(\text{word}|\text{spam}) = Multinomial(\mu_1^s, \ldots, \mu_M^s)$$
$$P(\text{word}|\text{nonspam}) = Multinomial(\mu_1^{ns}, \ldots, \mu_M^{ns})$$

- Goal

  Find ϕ, μ^s, μ^{ns} that best fits the data $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})\}$

- Likelihood - conditioned on parameters ϕ, μ^s, μ^{ns}

$$\prod_{i=1}^{N} P(\mathbf{x}^{(i)}, y^{(i)})$$
$$= \prod_{i=1}^{N} P(\mathbf{x}^{(i)}|y^{(i)})P(y^{(i)})$$
$$= \underbrace{\left(\prod_{i:y^{(i)}=1} P(\mathbf{x}^{(i)}|y^{(i)})P(y^{(i)})\right)}_{\text{Spam}} \underbrace{\left(\prod_{i:y^{(i)}=0} P(\mathbf{x}^{(i)}|y^{(i)})P(y^{(i)})\right)}_{\text{Non-spam}}$$

Naive Bayes Spam classifier

- Likelihood - spam

$$\left(\prod_{i:y^{(i)}=1} P(\mathbf{x}^{(i)}|y^{(i)})P(y^{(i)}) \right)$$

$x_k^{(i)}$ — i-th mail; k-th word

- Naive Bayes assumption:

$$P(\text{spam}) = Bernoulli(\phi)$$
$$P(\text{word}|\text{spam}) = Multinomial(\mu_1^s, \ldots, \mu_M^s)$$

$$P(\mathbf{x}^{(i)}|y^{(i)}=1) = \prod_{k=1}^{\text{len}(\mathbf{x}^{(i)})} P(x_k^{(i)}|y^{(i)}=1)$$

$$= \prod_{k=1}^{\text{len}(\mathbf{x}^{(i)})} \prod_{j=1}^{M} (\mu_j^s)^{\mathbb{I}(x_k^{(i)}=\text{“}j\text{”th word})}$$

$$P(y^{(i)}=1) = \phi$$

Naive Bayes Spam classifier

- Likelihood - spam (cont')

$$\left(\prod_{i:y^{(i)}=1} P\left(\mathbf{x}^{(i)} \mid y^{(i)}\right) P\left(y^{(i)}\right) \right)$$

$$= \left(\prod_{i:y^{(i)}=1}^{N} \prod_{k=1}^{\text{len}\left(x^{(i)}\right)} \prod_{j=1}^{M} \left(\mu_j^s\right)^{\mathbb{I}\left(x_k^{(i)}=\text{“}j\text{”th word}\right)} \phi \right)$$

Naive Bayes Spam classifier

- Likelihood - spam (cont')

$$\left(\prod_{i:y^{(i)}=1} P\left(\mathbf{x}^{(i)} \mid y^{(i)}\right) P\left(y^{(i)}\right) \right)$$

$$= \left(\prod_{i:y^{(i)}=1}^{N} \prod_{k=1}^{\text{len}\left(x^{(i)}\right)} \prod_{j=1}^{M} \left(\mu_j^s\right)^{\mathbb{I}\left(x_k^{(i)}=\text{“}j\text{”th word}\right)} \phi \right)$$

$$= \left(\prod_{i:y^{(i)}=1}^{N} \prod_{k=1}^{\text{len}\left(x^{(i)}\right)} \prod_{j=1}^{M} \left(\mu_j^s\right)^{\mathbb{I}\left(x_k^{(i)}=\text{“}j\text{”th word}\right)} \right) \left(\prod_{i:y^{(i)}=1}^{N} \phi \right)$$

Naive Bayes Spam classifier

- Likelihood - spam (cont')

$$\left(\prod_{i:y^{(i)}=1} P\left(\mathbf{x}^{(i)} \mid y^{(i)}\right) P\left(y^{(i)}\right) \right)$$

$$= \left(\prod_{i:y^{(i)}=1}^{N} \prod_{k=1}^{\text{len}\left(x^{(i)}\right)} \prod_{j=1}^{M} \left(\mu_j^s\right)^{\mathbb{I}\left(x_k^{(i)}=\text{“}j\text{”th word}\right)} \phi \right)$$

$$= \left(\prod_{i:y^{(i)}=1}^{N} \prod_{k=1}^{\text{len}\left(x^{(i)}\right)} \prod_{j=1}^{M} \left(\mu_j^s\right)^{\mathbb{I}\left(x_k^{(i)}=\text{“}j\text{”th word}\right)} \right) \left(\prod_{i:y^{(i)}=1}^{N} \phi \right)$$

$$= \left(\prod_{j=1}^{M} \left(\mu_j^s\right)^{\sum_{i:y^{(i)}=1}^{N} \sum_{k=1}^{\text{len}\left(x^{(i)}\right)} \mathbb{I}\left(x_k^{(i)}=\text{“}j\text{”th word}\right)} \right) \left(\prod_{i:y^{(i)}=1}^{N} \phi \right)$$

Naive Bayes Spam classifier

- Likelihood - spam (cont')

$$\left(\prod_{i:y^{(i)}=1} P\left(\mathbf{x}^{(i)} \mid y^{(i)}\right) P\left(y^{(i)}\right) \right)$$

$$= \left(\prod_{i:y^{(i)}=1}^{N} \prod_{k=1}^{\text{len}\left(x^{(i)}\right)} \prod_{j=1}^{M} \left(\mu_j^s\right)^{\mathbb{I}\left(x_k^{(i)}=\text{“}j\text{”th word}\right)} \phi \right)$$

$$= \left(\prod_{i:y^{(i)}=1}^{N} \prod_{k=1}^{\text{len}\left(x^{(i)}\right)} \prod_{j=1}^{M} \left(\mu_j^s\right)^{\mathbb{I}\left(x_k^{(i)}=\text{“}j\text{”th word}\right)} \right) \left(\prod_{i:y^{(i)}=1}^{N} \phi \right)$$

$$= \left(\prod_{j=1}^{M} \left(\mu_j^s\right)^{\sum_{i:y^{(i)}=1}^{N} \sum_{k=1}^{\text{len}\left(x^{(i)}\right)} \mathbb{I}\left(x_k^{(i)}=\text{“}j\text{”th word}\right)} \right) \left(\prod_{i:y^{(i)}=1}^{N} \phi \right)$$

$$= \left(\prod_{j=1}^{M} \left(\mu_j^s\right)^{N_j^{\text{spam}}} \right) \phi^{N^{\text{spam}}}$$

Definition:

N^{spam}: total # examples for spam

N_j^{spam}: total # of word j from the all spam emails

Naive Bayes Spam classifier

- Likelihood - non-spam

$$\left(\prod_{i:y^{(i)}=1} P(\mathbf{x}^{(i)}|y^{(i)})P(y^{(i)})\right)\left(\prod_{i:y^{(i)}=0} P(\mathbf{x}^{(i)}|y^{(i)})P(y^{(i)})\right)$$

- Similarly for non-spam mails,

$$P(\text{spam}) = Bernoulli(\phi)$$
$$P(\text{word}|\text{nonspam}) = Multinomial(\mu_1^{ns}, \ldots, \mu_M^{ns})$$

$$P(\mathbf{x}^{(i)}|y^{(i)}=0) = \prod_{k=1}^{\text{len}(\mathbf{x}^{(i)})} P(x_k^{(i)}|y^{(i)}=0)$$

$$= \prod_{k=1}^{\text{len}(\mathbf{x}^{(i)})} \prod_{j=1}^{M} (\mu_j^{ns})^{I(x_k^{(i)}=\text{“}j\text{”th word})}$$

$$P(y^{(i)}=0) = 1-\phi$$

Maximum likelihood estimation

- Putting together:

$$\prod_{i=1}^{N} P(\mathbf{x}^{(i)}, y^{(i)})$$

$$= \left(\prod_{i:y^{(i)}=1} P(\mathbf{x}^{(i)}|y^{(i)})P(y^{(i)}) \right) \left(\prod_{i:y^{(i)}=0} P(\mathbf{x}^{(i)}|y^{(i)})P(y^{(i)}) \right)$$

Maximum likelihood estimation

- Putting together:

$$\prod_{i=1}^{N} P(\mathbf{x}^{(i)}, y^{(i)})$$

$$= \left(\prod_{i:y^{(i)}=1} P(\mathbf{x}^{(i)}|y^{(i)})P(y^{(i)}) \right) \left(\prod_{i:y^{(i)}=0} P(\mathbf{x}^{(i)}|y^{(i)})P(y^{(i)}) \right)$$

$$= \left(\phi^{N^{spam}} \prod_{word\, j} \left(\mu_j^s\right)^{N_j^{spam}} \right) \left((1-\phi)^{N^{nonspam}} \prod_{word\, j} \left(\mu_j^{ns}\right)^{N_j^{nonspam}} \right)$$

Recall:

N^{spam}: total # examples for spam

$N^{nonspam}$: total # examples for non-spam

N_j^{spam}: total # word j from the entire spam emails

$N_j^{nonspam}$: total # word j from the entire nonspam emails

Maximum likelihood estimation

- Putting together:

$$\prod_{i=1}^{N} P(\mathbf{x}^{(i)}, y^{(i)})$$

$$= \left(\prod_{i:y^{(i)}=1} P(\mathbf{x}^{(i)}|y^{(i)})P(y^{(i)}) \right) \left(\prod_{i:y^{(i)}=0} P(\mathbf{x}^{(i)}|y^{(i)})P(y^{(i)}) \right)$$

$$= \left(\phi^{N^{spam}} \prod_{word\, j} \left(\mu_j^s\right)^{N_j^{spam}} \right) \left((1-\phi)^{N^{nonspam}} \prod_{word\, j} \left(\mu_j^{ns}\right)^{N_j^{nonspam}} \right)$$

- Joint Log-likelihood

$$\log P(\mathcal{D})$$

$$= \log \prod_{i=1}^{N} P(x^{(i)}, y^{(i)})$$

$$= N^{spam} \log \phi + \sum_{word\, j} N_j^{spam} \log \mu_j^s + N^{nonspam} \log(1-\phi) + \sum_{word\, j} N_j^{nonspam} \log \mu_j^{ns}$$

Maximum likelihood estimation

- Joint Log-likelihood

$$\begin{aligned}
& \log P(\mathcal{D}) \\
= \quad & \log \prod_{i=1}^{N} P(x^{(i)}, y^{(i)}) \\
= \quad & N^{spam} \log \phi + \sum_{word\, j} N_j^{spam} \log \mu_j^s + N^{nonspam} \log(1-\phi) + \sum_{word\, j} N_j^{nonspam} \log \mu_j^{ns}
\end{aligned}$$

- Maximum-likelihood
  - Take the derivative of log-likelihood w.r.t. the parameters, and set it to zero.

Maximum likelihood estimation

- From $\frac{\partial l}{\partial \phi} = \frac{1}{\phi} N^{spam} - \frac{1}{1-\phi} N^{nonspam} = 0$

  We get $\phi = \frac{N^{spam}}{N^{spam} + N^{nonspam}}$

- Removing dependent variables:

$$\sum_{word\, j=1}^{M} N_j^{spam} \log \mu_j^s = \sum_{word\, j=1}^{M-1} N_j^{spam} \log \mu_j^s + N_M^{spam} \log(1 - \sum_{j=1}^{M-1} \mu_j^s)$$

$$\frac{\partial}{\partial \mu_j^s} \left(\sum_{word\, j=1}^{M} N_j^{spam} \log \mu_j^s \right) = \frac{N_j^{spam}}{\mu_j^s} - \frac{N_M^{spam}}{1 - \sum_{j'=1}^{M-1} \mu_{j'}^s} = 0$$

$$\text{s.t.} \quad \sum_j \mu_j^s = 1$$

$$\sum_j \mu_j^{ns} = 1$$

Maximum likelihood estimation

- From $\frac{\partial l}{\partial \phi} = \frac{1}{\phi} N^{spam} - \frac{1}{1-\phi} N^{nonspam} = 0$

  We get $\phi = \frac{N^{spam}}{N^{spam} + N^{nonspam}}$

- Removing dependent variables:

$$\sum_{word\, j=1}^{M} N_j^{spam} \log \mu_j^s = \sum_{word\, j=1}^{M-1} N_j^{spam} \log \mu_j^s + N_M^{spam} \log(1 - \sum_{j=1}^{M-1} \mu_j^s)$$

$$\frac{\partial}{\partial \mu_j^s} \left(\sum_{word\, j=1}^{M} N_j^{spam} \log \mu_j^s \right) = \frac{N_j^{spam}}{\mu_j^s} - \frac{N_M^{spam}}{1 - \sum_{j'=1}^{M-1} \mu_{j'}^s} = 0$$

$$\frac{N_j^{spam}}{\mu_j^s} = constant, \forall j$$

$$\mu_j^s = \frac{N_j^{spam}}{\sum_{j'} N_{j'}^{spam}}$$

Maximum likelihood estimation

- Summary:

$$P(spam) = \phi = \frac{N^{spam}}{N^{spam} + N^{nonspam}}$$

$$P(word = j|spam) = \mu_j^s = \frac{N_j^{spam}}{\sum_{j'} N_{j'}^{spam}}$$

$$P(word = j|non - spam) = \mu_j^{ns} = \frac{N_j^{nonspam}}{\sum_{j'} N_{j'}^{nonspam}}$$

Recall:
N^{spam}: total # examples for spam
$N^{nonspam}$: total # examples for non-spam

N_j^{spam}: total # word j from the entire spam emails
$N_j^{nonspam}$: total # word j from the entire nonspam emails

Laplace Smoothing

- Maximum likelihood is problematic when a specific word count is 0
  - Leads to probability of 0!
- Solution: Put "imaginary" counts for each word
  - prevent zero probability estimates (overfitting)!
  - E.g.: Adding "1" as imaginary count for each word

$$P(spam) = \phi = \frac{N^{spam}}{N^{spam} + N^{nonspam}}$$

$$P(word = j|spam) = \mu_j^s = \frac{N_j^{spam} + 1}{\sum_{j'} N_{j'}^{spam} + M}$$

$$P(word = j|non - spam) = \mu_j^{ns} = \frac{N_j^{nonspam} + 1}{\sum_{j'} N_{j'}^{nonspam} + M}$$

Comparison: Discriminative vs. Generative

- The *generative* approach is typically model-based, and it can generate synthetic data from $p(\mathbf{x} | C_k)$.
  - By comparing the synthetic data and real data, we get a sense of how good the generative model is.

- The *discriminative* approach will typically have fewer parameters to estimate and have less assumptions about data distribution.
  - Linear (e.g. logistic regression) v/s quadratic (e.g., Gaussian discriminant analysis) in the dimension of the input.
  - Less generative assumptions about the data (however, constructing the features may require domain knowledge)

Discriminant Functions

The Perceptron

- A "generalized linear function"

$$h(\mathbf{x}) = f(\mathbf{w}^\top \phi(\mathbf{x}))$$

where

$$f(a) = \begin{cases} +1, & a \geq 0 \\ -1, & a < 0 \end{cases}$$

- Uses target code: y=+1 for C_1, y=-1 for C_2.
- This means that we always want:

$$\mathbf{w}^\top \phi(\mathbf{x}^{(n)}) y^{(n)} > 0$$

The Perceptron Criterion

- Only count errors from misclassified points:

$$E_P(\mathbf{w}) = -\sum_{\mathbf{x}^{(n)} \in \mathcal{M}} \mathbf{w}^\top \phi(\mathbf{x}^{(n)}) y^{(n)}$$

  - where $\mathcal{M}$ is the set of **misclassified** points.
- Stochastic gradient descent:
  - Update the weight vector according to the each misclassified sample (i.e., take gradient per sample):

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_P(\mathbf{w}) = \mathbf{w}^{(\tau)} + \eta \phi(\mathbf{x}^{(n)}) y^{(n)}$$

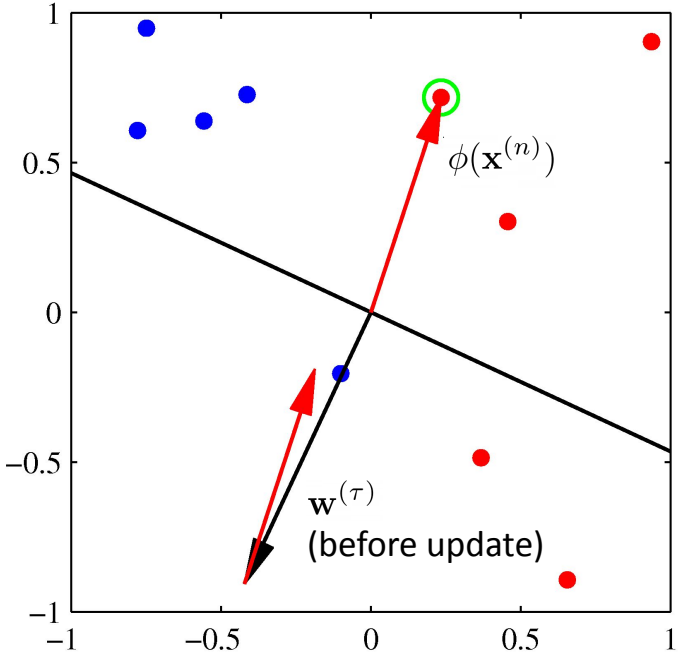
Note: update only for misclassified examples

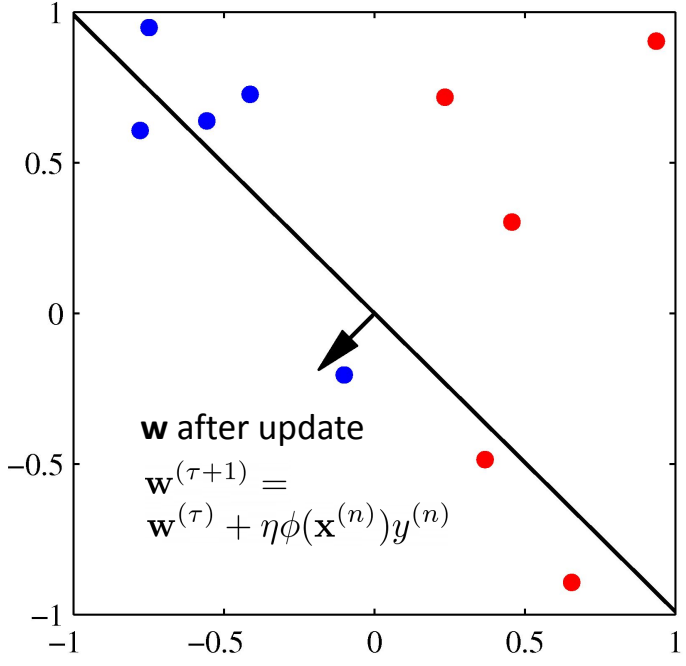
Perceptron Learning (1)

red: positive (y= +1)
blue: negative (y= -1)

- If $\mathbf{x}^{(n)}$ is misclassified, add $\phi(\mathbf{x}^{(n)})$ into $\mathbf{w}$.

green circle: misclassified sample



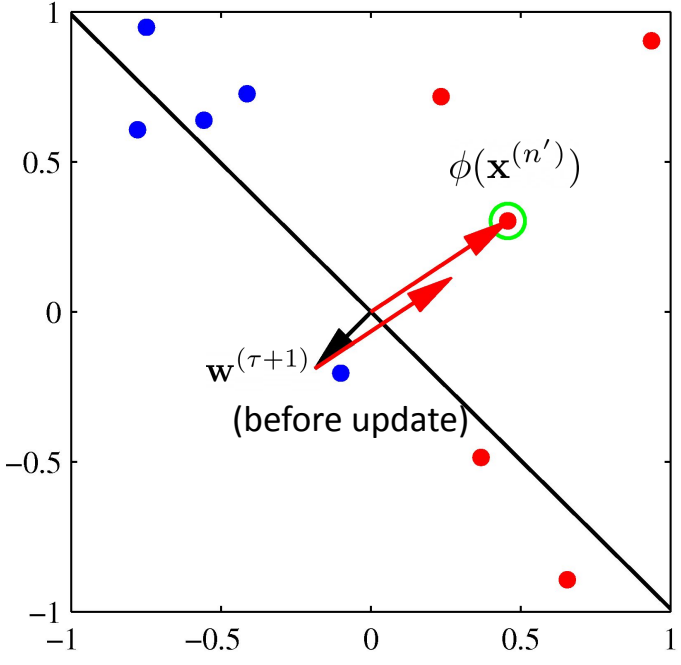


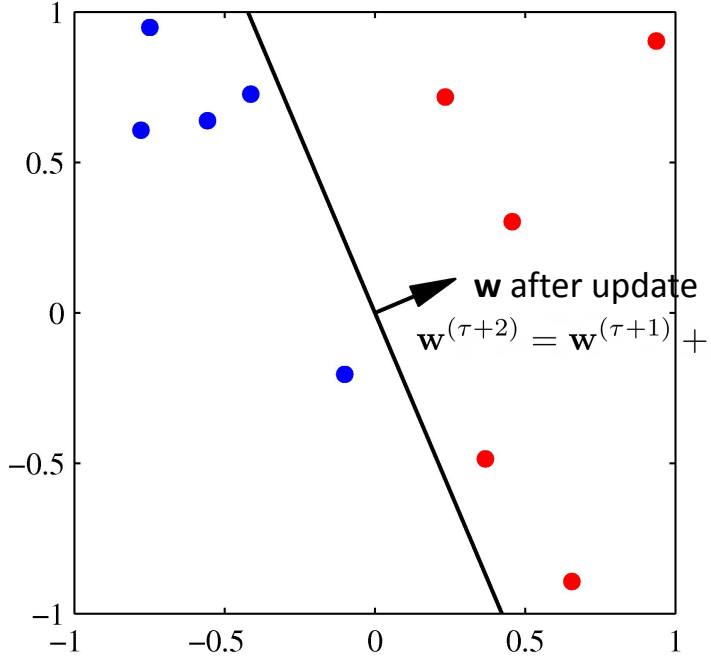
Perceptron Learning (2)

red: positive (y= +1)
blue: negative (y= -1)

- If $\mathbf{x}^{(n)}$ is misclassified, add $\phi(\mathbf{x}^{(n)})$ into $\mathbf{w}$.

green circle: misclassified sample





Perceptron Learning

- Perceptron Convergence Theorem (Block, 1962, and Novikoff, 1962):
  - If there exists an exact solution (i.e., if the training data is linearly separable)
  - then the learning algorithm will find it in a finite number of steps.
- Limitations of perceptron learning:
  - The convergence can be very slow.
  - If dataset is not linearly separable, it won't converge.
  - Does not generalize well to K>2 classes.

Any feedback (about lecture, slide, homework, project, etc.)?

(via **anonymous** google form: https://forms.gle/fpYmiBtG9Me5qbP37)

Change Log of lecture slides:
https://docs.google.com/document/d/e/2PACX-1vS6WIDIvW16-DZF8Ja7SpbMYK732Cy62xgNfr5kS-dt34fhMr8RD-6xvApS41Vmqi2y5wTolk_WI3SB/pub