Chapter 4: Linear Methods for Classification

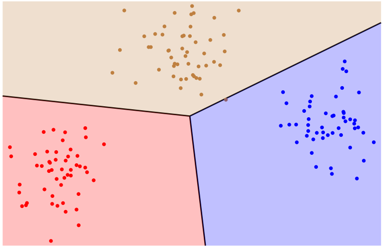
DD3364

March 23, 2012

Introduction

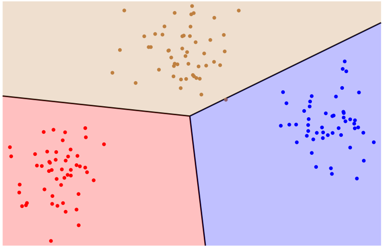
Focus on linear classification

- Want to learn a predictor $G : \mathbb{R}^p \rightarrow \mathcal{G} = \{1, \ldots, K\}$
- G divides input space into regions labelled according to their classification.
- The boundaries between these regions are termed the **decision boundaries**.
- When these **decision boundaries** are linear we term the classification method as linear.



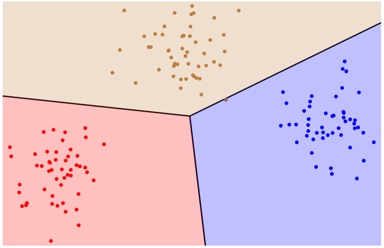
Focus on linear classification

- Want to learn a predictor $G : \mathbb{R}^p \to \mathcal{G} = \{1, \ldots, K\}$
- G divides input space into regions labelled according to their classification.
- The boundaries between these regions are termed the **decision boundaries**.
- When these **decision boundaries** are linear we term the classification method as linear.



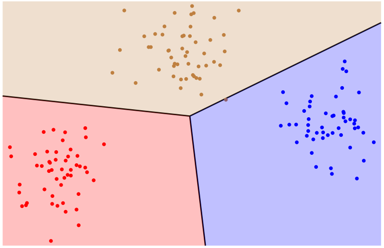
Focus on linear classification

- Want to learn a predictor $G : \mathbb{R}^p \rightarrow \mathcal{G} = \{1, \ldots, K\}$
- G divides input space into regions labelled according to their classification.
- The boundaries between these regions are termed the **decision boundaries**.
- When these **decision boundaries** are linear we term the classification method as linear.



Focus on linear classification

- Want to learn a predictor $G : \mathbb{R}^p \to \mathcal{G} = \{1, \ldots, K\}$
- G divides input space into regions labelled according to their classification.
- The boundaries between these regions are termed the **decision boundaries**.
- When these **decision boundaries** are linear we term the classification method as linear.



An example when a linear decision boundaries arises

- Learn a **discriminant function** $\delta_k(x)$ for each class k and set

$$\boxed{G(x) = \arg\max_k \delta_k(x)}$$

- This generates a linear decision boundary when $\exists$ some monotone transformation g of $\delta_k(x)$ which is linear.

- That is g is a monotone function s.t.

$$g(\delta_k(x)) = \gamma_{k0} + \gamma_k^t x$$

An example when a linear decision boundaries arises

- Learn a **discriminant function** $\delta_k(x)$ for each class k and set

$$\boxed{G(x) = \arg\max_k \delta_k(x)}$$

- This generates a linear decision boundary when $\exists$ some monotone transformation g of $\delta_k(x)$ which is linear.

- That is g is a monotone function s.t.

$$g(\delta_k(x)) = \gamma_{k0} + \gamma_k^t x$$

An example when a linear decision boundaries arises

- Learn a **discriminant function** $\delta_k(x)$ for each class k and set

$$\boxed{G(x) = \arg\max_k \delta_k(x)}$$

- This generates a linear decision boundary when $\exists$ some monotone transformation g of $\delta_k(x)$ which is linear.

- That is g is a monotone function s.t.

$$g(\delta_k(x)) = \gamma_{k0} + \gamma_k^t x$$

Examples of discriminant functions

- **Example 1**: Fit a linear regression model to the class indicator variables. Then the discriminant functions are

$$\delta_k(x) = \hat{\beta}_{k0} + \hat{\beta}_k^t x$$

- **Example 2**: Use the posterior probabilities $P(G = k \mid X = x)$ as the discriminant functions $\delta_k(x)$
  - A popular model when there are two classes is:

$$P(G = 1|X = x) = \frac{\exp(\beta_0 + \beta^t x)}{1 + \exp(\beta_0 + \beta^t x)}$$
$$P(G = 2|X = x) = \frac{1}{1 + \exp(\beta_0 + \beta^t x)}$$

  - $g(p) = \log(p/(1-p))$ can be applied as a monotonic function to $\delta_k(x) = P(G = 1|X = x)$ to make it linear.

Examples of discriminant functions

- **Example 1**: Fit a linear regression model to the class indicator variables. Then the discriminant functions are

$$\delta_k(x) = \hat{\beta}_{k0} + \hat{\beta}_k^t x$$

- **Example 2**: Use the posterior probabilities $P(G = k \mid X = x)$ as the discriminant functions $\delta_k(x)$
  - A popular model when there are two classes is:

$$P(G = 1|X = x) = \frac{\exp(\beta_0 + \beta^t x)}{1 + \exp(\beta_0 + \beta^t x)}$$
$$P(G = 2|X = x) = \frac{1}{1 + \exp(\beta_0 + \beta^t x)}$$

  - $g(p) = \log(p/(1-p))$ can be applied as a monotonic function to $\delta_k(x) = P(G = 1|X = x)$ to make it linear.

Examples of discriminant functions

- **Example 1**: Fit a linear regression model to the class indicator variables. Then the discriminant functions are

$$\delta_k(x) = \hat{\beta}_{k0} + \hat{\beta}_k^t x$$

- **Example 2**: Use the posterior probabilities $P(G = k \mid X = x)$ as the discriminant functions $\delta_k(x)$
  - A popular model when there are two classes is:

$$P(G = 1|X = x) = \frac{\exp(\beta_0 + \beta^t x)}{1 + \exp(\beta_0 + \beta^t x)}$$
$$P(G = 2|X = x) = \frac{1}{1 + \exp(\beta_0 + \beta^t x)}$$

  - $g(p) = \log(p/(1-p))$ can be applied as a monotonic function to $\delta_k(x) = P(G = 1|X = x)$ to make it linear.

Can directly learn the linear decision boundary

- For a two class problem with p-dimensional inputs this $\Longrightarrow$ modelling the decision boundary as a hyperplane.

- This chapter looks at two methods which explicitly look for the **separating hyperplane**. These are
    - **Perceptron** model and algorithm of *Rosenblatt*,
    - **SVM** model and algorithm of *Vapnik*
    - In the forms quoted both these algorithms find separating hyperplanes if they exist and fail of the points are not linearly separable.
    - There are fixes for the non-separable case but we will not consider these today.

Can directly learn the linear decision boundary

- For a two class problem with p-dimensional inputs this $\Longrightarrow$ modelling the decision boundary as a hyperplane.
- This chapter looks at two methods which explicitly look for the **separating hyperplane**. These are
  - **Perceptron** model and algorithm of *Rosenblatt*,
  - **SVM** model and algorithm of *Vapnik*
  - In the forms quoted both these algorithms find separating hyperplanes if they exist and fail of the points are not linearly separable.
  - There are fixes for the non-separable case but we will not consider these today.

Can directly learn the linear decision boundary

- For a two class problem with p-dimensional inputs this $\Longrightarrow$ modelling the decision boundary as a hyperplane.
- This chapter looks at two methods which explicitly look for the **separating hyperplane**. These are
  - **Perceptron** model and algorithm of *Rosenblatt*,
  - **SVM** model and algorithm of *Vapnik*
  - In the forms quoted both these algorithms find separating hyperplanes if they exist and fail of the points are not linearly separable.
  - There are fixes for the non-separable case but we will not consider these today.

Can directly learn the linear decision boundary

- For a two class problem with p-dimensional inputs this $\implies$ modelling the decision boundary as a hyperplane.
- This chapter looks at two methods which explicitly look for the **separating hyperplane**. These are
  - **Perceptron** model and algorithm of *Rosenblatt*,
  - **SVM** model and algorithm of *Vapnik*
  - In the forms quoted both these algorithms find separating hyperplanes if they exist and fail of the points are not linearly separable.
  - There are fixes for the non-separable case but we will not consider these today.

Can directly learn the linear decision boundary

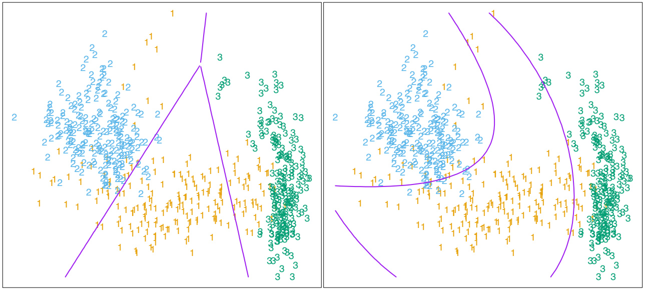
- For a two class problem with p-dimensional inputs this $\Longrightarrow$ modelling the decision boundary as a hyperplane.
- This chapter looks at two methods which explicitly look for the **separating hyperplane**. These are
  - **Perceptron** model and algorithm of *Rosenblatt*,
  - **SVM** model and algorithm of *Vapnik*
  - In the forms quoted both these algorithms find separating hyperplanes if they exist and fail of the points are not linearly separable.
  - There are fixes for the non-separable case but we will not consider these today.

Can directly learn the linear decision boundary

- For a two class problem with p-dimensional inputs this $\Longrightarrow$ modelling the decision boundary as a hyperplane.
- This chapter looks at two methods which explicitly look for the **separating hyperplane**. These are
  - **Perceptron** model and algorithm of *Rosenblatt*,
  - **SVM** model and algorithm of *Vapnik*
  - In the forms quoted both these algorithms find separating hyperplanes if they exist and fail of the points are not linearly separable.
  - There are fixes for the non-separable case but we will not consider these today.

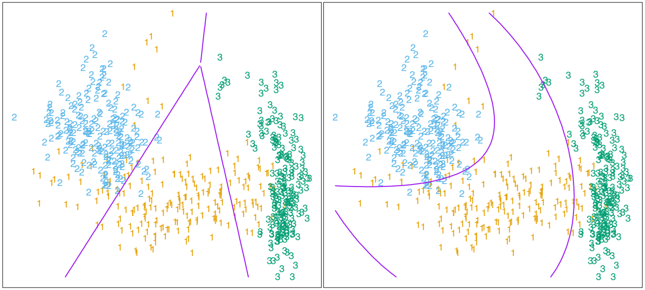
Linear decision boundaries can be made non-linear

- Can expand the variable set $X_1, X_2, \ldots, X_p$ by including their squares and cross-products $X_1^2, X_2^2, \ldots, X_p^2, X_1X_2, X_1X_2, \ldots$
- This adds $p(p+1)/2$ additional variables.
- Linear decision boundaries in the augmented space corresponds to quadratic decision boundaries in the original space.



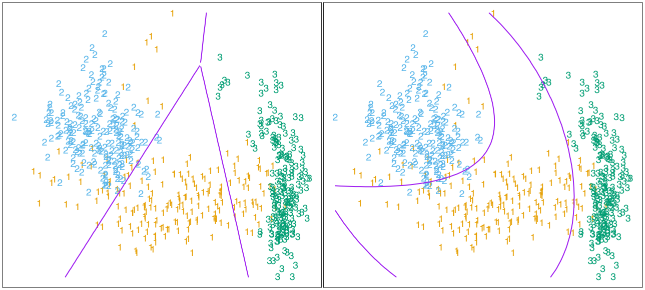
Linear decision boundaries can be made non-linear

- Can expand the variable set $X_1, X_2, \ldots, X_p$ by including their squares and cross-products $X_1^2, X_2^2, \ldots, X_p^2, X_1X_2, X_1X_2, \ldots$
- This adds $p(p+1)/2$ additional variables.
- Linear decision boundaries in the augmented space corresponds to quadratic decision boundaries in the original space.



Linear decision boundaries can be made non-linear

- Can expand the variable set $X_1, X_2, \ldots, X_p$ by including their squares and cross-products $X_1^2, X_2^2, \ldots, X_p^2, X_1X_2, X_1X_2, \ldots$
- This adds $p(p+1)/2$ additional variables.
- Linear decision boundaries in the augmented space corresponds to quadratic decision boundaries in the original space.



Linear Regression of an Indicator Matrix

Use linear regression to find discriminant functions

- Have training data $\{(x_i, g_i)\}_{i=1}^n$ where each $x_i \in \mathbb{R}^p$ and $g_i \in \{1, \ldots, K\}$.

- For each k construct a linear discriminant $\delta_k(x)$ via:
  1. For $i = 1, \ldots, n$ set
  $$y_i = \begin{cases} 0 & \text{if } g_i \neq k \\ 1 & \text{if } g_i = k \end{cases}$$
  2. Compute $(\hat{\beta}_{0k}, \hat{\beta}_k) = \arg\min_{\beta_0, \beta_k} \sum_{i=1}^n (y_i - \beta_0 - \beta_k^t x_i)^2$
  3. Define
  $$\boxed{\delta_k(x) = \hat{\beta}_{0k} + \hat{\beta}_k^t x}$$

- Classify a new point x with
$$\boxed{G(x) = \arg\max_k \delta_k(x)}$$

Use linear regression to find discriminant functions

- Have training data $\{(x_i, g_i)\}_{i=1}^n$ where each $x_i \in \mathbb{R}^p$ and $g_i \in \{1, \ldots, K\}$.
- For each k construct a linear discriminant $\delta_k(x)$ via:
  1. For $i = 1, \ldots, n$ set
     $$y_i = \begin{cases} 0 & \text{if } g_i \neq k \\ 1 & \text{if } g_i = k \end{cases}$$
  2. Compute $(\hat\beta_{0k}, \hat\beta_k) = \arg\min_{\beta_0, \beta_k} \sum_{i=1}^n (y_i - \beta_0 - \beta_k^t x_i)^2$
  3. Define
     $$\boxed{\delta_k(x) = \hat\beta_{0k} + \hat\beta_k^t x}$$
- Classify a new point x with
  $$\boxed{G(x) = \arg\max_k \delta_k(x)}$$

Use linear regression to find discriminant functions

- Have training data $\{(x_i, g_i)\}_{i=1}^n$ where each $x_i \in \mathbb{R}^p$ and $g_i \in \{1, \ldots, K\}$.

- For each k construct a linear discriminant $\delta_k(x)$ via:
  1. For $i = 1, \ldots, n$ set
  $$y_i = \begin{cases} 0 & \text{if } g_i \neq k \\ 1 & \text{if } g_i = k \end{cases}$$
  2. Compute $(\hat{\beta}_{0k}, \hat{\beta}_k) = \arg \min\limits_{\beta_0, \beta_k} \sum_{i=1}^n (y_i - \beta_0 - \beta_k^t x_i)^2$
  3. Define
  $$\boxed{\delta_k(x) = \hat{\beta}_{0k} + \hat{\beta}_k^t x}$$

- Classify a new point x with
$$\boxed{G(x) = \arg \max_k \delta_k(x)}$$

Use linear regression to find discriminant functions

- Have training data $\{(x_i, g_i)\}_{i=1}^n$ where each $x_i \in \mathbb{R}^p$ and $g_i \in \{1, \ldots, K\}$.

- For each k construct a linear discriminant $\delta_k(x)$ via:
  1. For $i = 1, \ldots, n$ set
  $$y_i = \begin{cases} 0 & \text{if } g_i \neq k \\ 1 & \text{if } g_i = k \end{cases}$$
  2. Compute $(\hat\beta_{0k}, \hat\beta_k) = \arg\min\limits_{\beta_0, \beta_k} \sum_{i=1}^n (y_i - \beta_0 - \beta_k^t x_i)^2$
  3. Define
  $$\boxed{\delta_k(x) = \hat\beta_{0k} + \hat\beta_k^t x}$$

- Classify a new point x with
$$\boxed{G(x) = \arg\max_k \, \delta_k(x)}$$

Use linear regression to find discriminant functions

- Have training data $\{(x_i, g_i)\}_{i=1}^n$ where each $x_i \in \mathbb{R}^p$ and $g_i \in \{1, \ldots, K\}$.

- For each k construct a linear discriminant $\delta_k(x)$ via:
  1. For $i = 1, \ldots, n$ set
  $$y_i = \begin{cases} 0 & \text{if } g_i \neq k \\ 1 & \text{if } g_i = k \end{cases}$$
  2. Compute $(\hat{\beta}_{0k}, \hat{\beta}_k) = \arg\min\limits_{\beta_0, \beta_k} \sum_{i=1}^n (y_i - \beta_0 - \beta_k^t x_i)^2$
  3. Define
  $$\boxed{\delta_k(x) = \hat{\beta}_{0k} + \hat{\beta}_k^t x}$$

- Classify a new point x with
$$\boxed{G(x) = \arg\max_k \delta_k(x)}$$

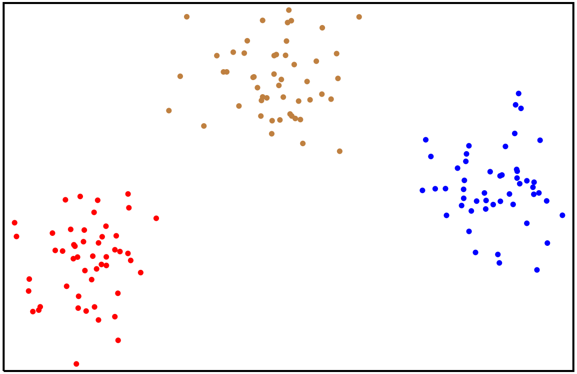
Use linear regression to find discriminant functions

- Have training data $\{(x_i, g_i)\}_{i=1}^n$ where each $x_i \in \mathbb{R}^p$ and $g_i \in \{1, \ldots, K\}$.

- For each k construct a linear discriminant $\delta_k(x)$ via:
  1. For $i = 1, \ldots, n$ set
  $$y_i = \begin{cases} 0 & \text{if } g_i \neq k \\ 1 & \text{if } g_i = k \end{cases}$$
  2. Compute $(\hat\beta_{0k}, \hat\beta_k) = \arg\min_{\beta_0, \beta_k} \sum_{i=1}^n (y_i - \beta_0 - \beta_k^t x_i)^2$
  3. Define
  $$\boxed{\delta_k(x) = \hat\beta_{0k} + \hat\beta_k^t x}$$

- Classify a new point x with
$$\boxed{G(x) = \arg\max_k \delta_k(x)}$$

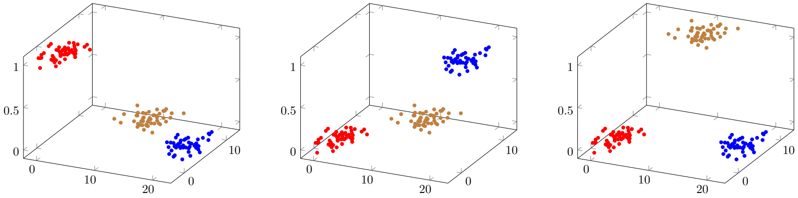
3 class example



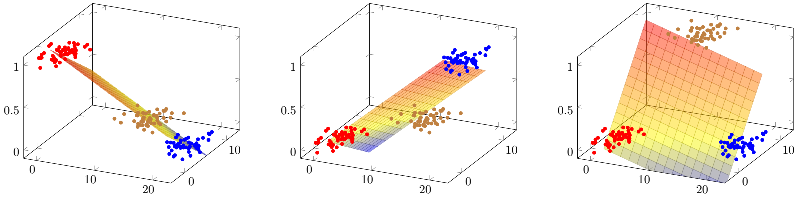
Use linear regression of an indicator matrix to find the discriminant functions for the above 3-classes.

Construct K linear regression problems

For each k construct the response vectors from the class labels

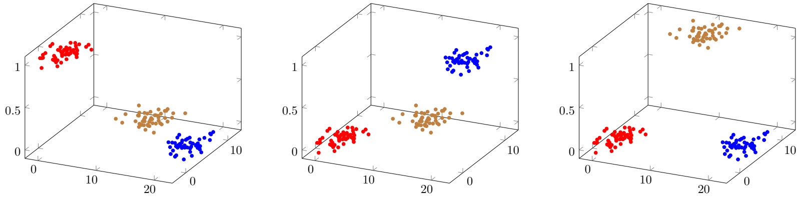


For each k fit a hyperplane that minimizes the RSS

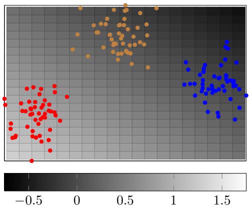


Construct K discriminant functions

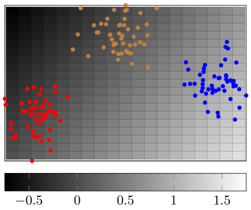
For each k construct the response vectors from the class labels



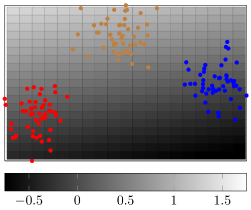
The k discriminant fns defined by the least square hyperplanes



$\delta_1(x)$

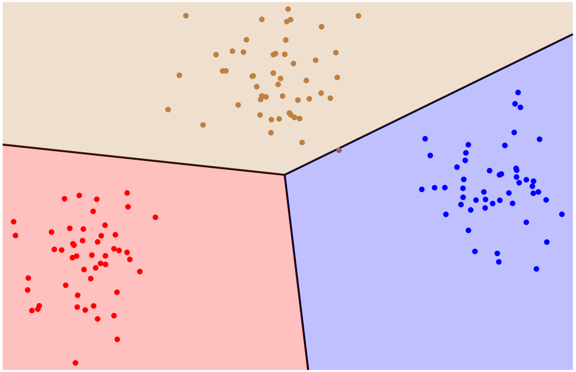


$\delta_2(x)$



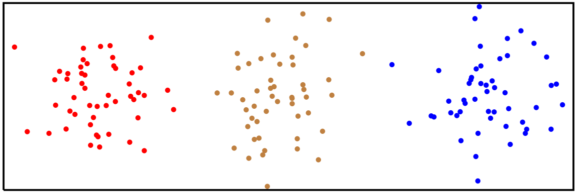
$\delta_3(x)$

The decision boundary defined by these discriminant fns

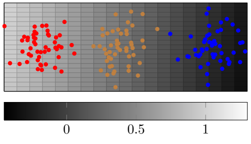


This approach will fail in this case

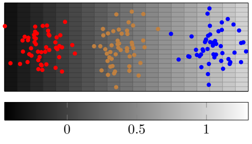
The training data from 3 classes



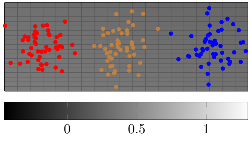
The discriminant functions learnt via regression



$\delta_1(x)$

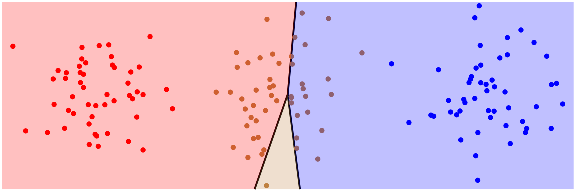


$\delta_2(x)$

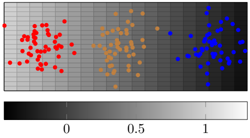


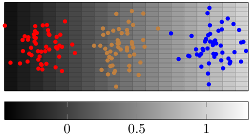
$\delta_3(x)$

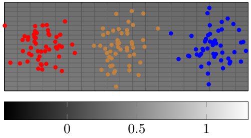
The resulting decision boundary



The discriminant functions learnt via regression


$\delta_1(x)$


$\delta_2(x)$


$\delta_3(x)$

The resulting decision boundary

- In this last example masking has occurred.
- This occurs because of the rigid nature of the linear discriminant functions.
- This example is extreme but for large K and small p such maskings occur naturally.
- The other methods in this chapter are based on linear decision functions of x, but they are learnt in a smarter why...

The resulting decision boundary

- In this last example masking has occurred.
- This occurs because of the rigid nature of the linear discriminant functions.
- This example is extreme but for large K and small p such maskings occur naturally.
- The other methods in this chapter are based on linear decision functions of x, but they are learnt in a smarter why...

The resulting decision boundary

- In this last example masking has occurred.
- This occurs because of the rigid nature of the linear discriminant functions.
- This example is extreme but for large K and small p such maskings occur naturally.
- The other methods in this chapter are based on linear decision functions of x, but they are learnt in a smarter why...

The resulting decision boundary

- In this last example masking has occurred.
- This occurs because of the rigid nature of the linear discriminant functions.
- This example is extreme but for large K and small p such maskings occur naturally.
- The other methods in this chapter are based on linear decision functions of x, but they are learnt in a smarter why...

Linear Discriminant Analysis

Optimal classification requires the posterior

- To perform optimal classification need to know $P(G \mid X)$. Let
  - $f_k(x)$ represent the class-conditional $P(X \mid G = k)$ **and**
  - π_k be the prior probability of class k with $\sum_{k=1}^{K} \pi_k = 1$
- A simple application of **Bayes Rule** gives

$$P(G = k \mid X = x) = \frac{f_k(x)\pi_k}{\sum_{l=1}^{K} f_l(x)\pi_l}$$

- Therefore for classification having $f_k(x)$ is almost equivalent to having $P(G = k \mid X = x)$.

Optimal classification requires the posterior

- To perform optimal classification need to know $P(G \mid X)$. Let
  - $f_k(x)$ represent the class-conditional $P(X \mid G = k)$ **and**
  - π_k be the prior probability of class k with $\sum_{k=1}^{K} \pi_k = 1$
- A simple application of **Bayes Rule** gives

$$\boxed{P(G = k \mid X = x) = \frac{f_k(x)\pi_k}{\sum_{l=1}^{K} f_l(x)\pi_l}}$$

- Therefore for classification having $f_k(x)$ is almost equivalent to having $P(G = k \mid X = x)$.

Optimal classification requires the posterior

- To perform optimal classification need to know $P(G \,|\, X)$. Let
  - $f_k(x)$ represent the class-conditional $P(X \,|\, G = k)$ **and**
  - π_k be the prior probability of class k with $\sum_{k=1}^{K} \pi_k = 1$
- A simple application of **Bayes Rule** gives

$$\boxed{P(G = k \,|\, X = x) = \frac{f_k(x)\pi_k}{\sum_{l=1}^{K} f_l(x)\pi_l}}$$

- Therefore for classification having $f_k(x)$ is almost equivalent to having $P(G = k \,|\, X = x)$.

Optimal classification requires the posterior

- To perform optimal classification need to know $P(G \,|\, X)$. Let
  - $f_k(x)$ represent the class-conditional $P(X \,|\, G = k)$ **and**
  - π_k be the prior probability of class k with $\sum_{k=1}^{K} \pi_k = 1$
- A simple application of **Bayes Rule** gives

$$\boxed{P(G = k \,|\, X = x) = \frac{f_k(x)\pi_k}{\sum_{l=1}^{K} f_l(x)\pi_l}}$$

- Therefore for classification having $f_k(x)$ is almost equivalent to having $P(G = k \,|\, X = x)$.

How to model the class densities

- Many methods are based on specific models of $f_k(x)$
  - linear and quadratic discriminant functions use Gaussian distributions,
  - mixture of Gaussian distributions produce non-linear decision boundaries,
  - non-parametric density estimates which allow the most flexibility,
  - Naive Bayes where $f_k(X) = \prod_{j=1}^{p} f_{kj}(X_j)$.

How to model the class densities

- Many methods are based on specific models of $f_k(x)$
  - linear and quadratic discriminant functions use Gaussian distributions,
  - mixture of Gaussian distributions produce non-linear decision boundaries,
  - non-parametric density estimates which allow the most flexibility,
  - Naive Bayes where $f_k(X) = \prod_{j=1}^{p} f_{kj}(X_j)$.

How to model the class densities

- Many methods are based on specific models of $f_k(x)$
  - linear and quadratic discriminant functions use Gaussian distributions,
  - mixture of Gaussian distributions produce non-linear decision boundaries,
  - non-parametric density estimates which allow the most flexibility,
  - Naive Bayes where $f_k(X) = \prod_{j=1}^{p} f_{kj}(X_j)$.

How to model the class densities

- Many methods are based on specific models of $f_k(x)$
  - linear and quadratic discriminant functions use Gaussian distributions,
  - mixture of Gaussian distributions produce non-linear decision boundaries,
  - non-parametric density estimates which allow the most flexibility,
  - Naive Bayes where $f_k(X) = \prod_{j=1}^{p} f_{kj}(X_j)$.

How to model the class densities

- Many methods are based on specific models of $f_k(x)$
  - linear and quadratic discriminant functions use Gaussian distributions,
  - mixture of Gaussian distributions produce non-linear decision boundaries,
  - non-parametric density estimates which allow the most flexibility,
  - Naive Bayes where $f_k(X) = \prod_{j=1}^{p} f_{kj}(X_j)$.

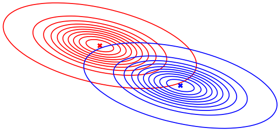
Multivariate Gaussian class densities

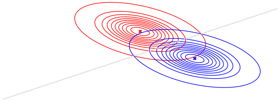
- Model each $f_k(x)$ as a multivariate Gaussian

$$\boxed{f_k(x) = \frac{1}{\sqrt[p]{2\pi}\sqrt{|\Sigma_k|}} \exp\left\{-.5(x-\mu_k)^t\,\Sigma_k^{-1}\,(x-\mu_k)\right\}}$$

- **Linear Discriminant Analysis** (LDA) arises in the special case when

$$\boxed{\Sigma_k = \Sigma \text{ for all } k}$$


class distributions


decision boundary

One gets linear decision boundaries.

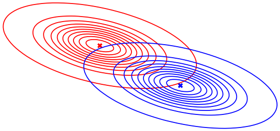
Multivariate Gaussian class densities

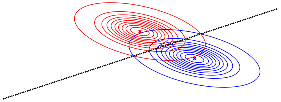
- Model each $f_k(x)$ as a multivariate Gaussian

$$\boxed{f_k(x) = \frac{1}{\sqrt[p]{2\pi}\sqrt{|\Sigma_k|}} \exp\left\{-.5(x-\mu_k)^t\,\Sigma_k^{-1}\,(x-\mu_k)\right\}}$$

- **Linear Discriminant Analysis** (LDA) arises in the special case when

$$\boxed{\Sigma_k = \Sigma \text{ for all } k}$$


class distributions


decision boundary

One gets linear decision boundaries.

LDA

- Can see this as

$$\begin{aligned}\log \frac{P(G=k\,|\,X=x)}{P(G=l\,|\,X=x)} &= \log \frac{f_k(x)}{f_l(x)} + \log \frac{\pi_k}{\pi_l} \\ &= \log \frac{\pi_k}{\pi_l} - .5\,\mu_k^t\,\Sigma^{-1}\mu_k + .5\,\mu_l^t\,\Sigma^{-1}\mu_l \\ &\qquad + x^t\Sigma^{-1}(\mu_k - \mu_l) \\ &= x^t a + b \qquad \leftarrow \text{a linear function}\end{aligned}$$

- The equal covariance matrices allow the $x^t\Sigma_k^{-1}x$ and $x^t\Sigma_l^{-1}x$ terms to cancel out.

- From the log-odds function we see that the linear discriminant functions

$$\boxed{\delta_k(x) = x^t\,\Sigma^{-1}\mu_k - .5\,\mu_k^t\,\Sigma^{-1}\mu_k + \log \pi_k}$$

are an equivalent description of the decision rule with

$$G(x) = \arg\max_k \delta_k(x)$$

LDA

- Can see this as

$$\begin{aligned}\log \frac{P(G=k\,|\,X=x)}{P(G=l\,|\,X=x)} &= \log \frac{f_k(x)}{f_l(x)} + \log \frac{\pi_k}{\pi_l} \\ &= \log \frac{\pi_k}{\pi_l} - .5\,\mu_k^t\,\Sigma^{-1}\mu_k + .5\,\mu_l^t\,\Sigma^{-1}\mu_l \\ &\quad + x^t\Sigma^{-1}(\mu_k - \mu_l) \\ &= x^t a + b \qquad \leftarrow \text{a linear function}\end{aligned}$$

- The equal covariance matrices allow the $x^t\Sigma_k^{-1}x$ and $x^t\Sigma_l^{-1}x$ terms to cancel out.

- From the log-odds function we see that the linear discriminant functions

$$\boxed{\delta_k(x) = x^t\,\Sigma^{-1}\mu_k - .5\,\mu_k^t\,\Sigma^{-1}\mu_k + \log \pi_k}$$

are an equivalent description of the decision rule with

$$G(x) = \arg\max_k \delta_k(x)$$

LDA

- Can see this as

$$\begin{aligned}\log \frac{P(G=k \mid X=x)}{P(G=l \mid X=x)} &= \log \frac{f_k(x)}{f_l(x)} + \log \frac{\pi_k}{\pi_l} \\ &= \log \frac{\pi_k}{\pi_l} - .5\,\mu_k^t\,\Sigma^{-1}\mu_k + .5\,\mu_l^t\,\Sigma^{-1}\mu_l \\ &\qquad + x^t\Sigma^{-1}(\mu_k - \mu_l) \\ &= x^t a + b \qquad \leftarrow \text{a linear function}\end{aligned}$$

- The equal covariance matrices allow the $x^t\Sigma_k^{-1}x$ and $x^t\Sigma_l^{-1}x$ terms to cancel out.

- From the log-odds function we see that the linear discriminant functions

$$\boxed{\delta_k(x) = x^t\,\Sigma^{-1}\mu_k - .5\,\mu_k^t\,\Sigma^{-1}\mu_k + \log \pi_k}$$

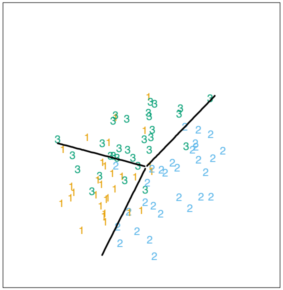
are an equivalent description of the decision rule with

$$G(x) = \arg\max_k \delta_k(x)$$

LDA: Some practicalities

In practice don't know the parameters of the Gaussian distributions and estimate these from the training data.
Let n_k be the number of class k observations then

- $\hat{\pi}_k = n_k/n$
- $\hat{\mu}_k = \sum_{g_i=k} x_i/n_k$
- $\hat{\Sigma}_k = \sum_{k=1}^{K} \sum_{g_i=k}(x_i - \hat{\mu_k})(x_i - \hat{\mu_k})^t/(n-K)$

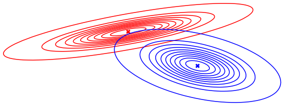


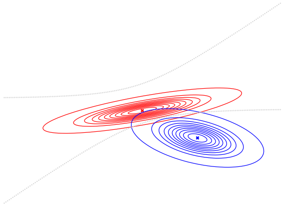
When Σ_k's are not all equal

- If the Σ_k are not assumed to be equal then the quadratic terms remain and we get quadratic discriminant functions (QDA)

$$\boxed{\delta_k(x) = -.5 \log|\Sigma_k| - .5\,(x-\mu_k)^t\,\Sigma_k^{-1}(x-\mu_k) + \log \pi_k}$$

- In this case the decision boundary between classes are described by a quadratic equation $\{x : \delta_k(x) = \delta_l(x)\}$.


class distributions

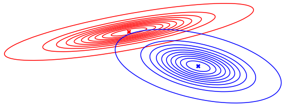

decision boundaries

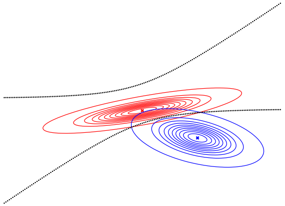
When Σ_k's are not all equal

- If the Σ_k are not assumed to be equal then the quadratic terms remain and we get quadratic discriminant functions (QDA)

$$\boxed{\delta_k(x) = -.5 \log |\Sigma_k| - .5\,(x-\mu_k)^t\, \Sigma_k^{-1}(x-\mu_k) + \log \pi_k}$$

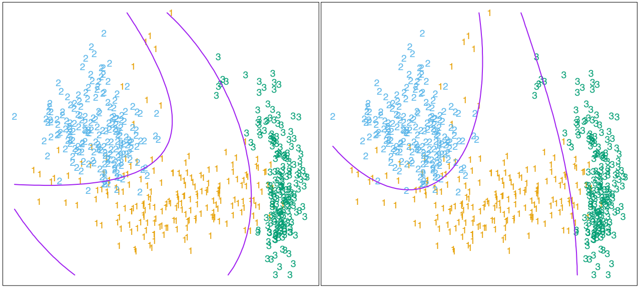
- In this case the decision boundary between classes are described by a quadratic equation $\{x : \delta_k(x) = \delta_l(x)\}$.


class distributions


decision boundaries

Best way to compute a quadratic discriminant function?

Left plot shows the quadratic decision boundaries found using LDA in the five dimensional space $X_1, X_2, X_1^2, X_2^2, X_1X_2$.



Right plot shows the quadratic decision boundaries found by QDA.

LDA and QDA summary

- These methods can be surprisingly effective.
- Can explain this

Reduced-Rank Linear Discriminant Analysis

Affine subspace defined by centroids of the classes

- Have K centroids in a p-dimensional input space: $\mu_1, \ldots, \mu_K$
- These centroids define an $K-1$ dimensional affine subspace H_{K-1} where if $u \in H_{K-1}$ then

$$\begin{aligned} u &= \mu_1 + \alpha_1(\mu_2 - \mu_1) + \alpha_2(\mu_3 - \mu_1) + \cdots + \alpha_{K-1}(\mu_K - \mu_1) \\ &= \mu_1 + \alpha_1\, d_1 + \alpha_2\, d_2 + \cdots + \alpha_{K-1}\, d_{K-1} \end{aligned}$$

- If $x \in \mathbb{R}^p$ then it can be written as

$$x = \mu_1 + \gamma_1\, d_1 + \gamma_2\, d_2 + \cdots + \gamma_{K-1}\, d_{K-1} + x^\perp, \quad \text{where } x^\perp \in H_{K-1}^\perp.$$

- If x has been whitened with respect to the common covariance matrix then the Mahalhobnis distance to centroid μ_j

$$\begin{aligned} \|x - \mu_j\| &= \|\mu_1 + \gamma_1\, d_1 + \gamma_2\, d_2 + \cdots + \gamma_{K-1}\, d_{K-1} + x^\perp - \mu_j\| \\ &= \|2\mu_1 + \gamma_1\, d_1 + \cdots + (\gamma_{j-1} - 1)\, d_{j-1} + \cdots + \gamma_{K-1}\, d_{K-1} + x^\perp\| \end{aligned}$$

- $x^\perp$ does not change with μ_j, therefore to locate the closest centroid can ignore it.

Affine subspace defined by centroids of the classes

- Have K centroids in a p-dimensional input space: $\mu_1, \ldots, \mu_K$
- These centroids define an $K-1$ dimensional affine subspace H_{K-1} where if $u \in H_{K-1}$ then

$$\begin{aligned} u &= \mu_1 + \alpha_1(\mu_2 - \mu_1) + \alpha_2(\mu_3 - \mu_1) + \cdots + \alpha_{K-1}(\mu_K - \mu_1) \\ &= \mu_1 + \alpha_1\, d_1 + \alpha_2\, d_2 + \cdots + \alpha_{K-1}\, d_{K-1} \end{aligned}$$

- If $x \in \mathbb{R}^p$ then it can be written as

$$x = \mu_1 + \gamma_1\, d_1 + \gamma_2\, d_2 + \cdots + \gamma_{K-1}\, d_{K-1} + x^\perp, \quad \text{where } x^\perp \in H_{K-1}^\perp.$$

- If x has been whitened with respect to the common covariance matrix then the Mahalhobnis distance to centroid μ_j

$$\begin{aligned} \|x - \mu_j\| &= \|\mu_1 + \gamma_1\, d_1 + \gamma_2\, d_2 + \cdots + \gamma_{K-1}\, d_{K-1} + x^\perp - \mu_j\| \\ &= \|2\mu_1 + \gamma_1\, d_1 + \cdots + (\gamma_{j-1} - 1)\, d_{j-1} + \cdots + \gamma_{K-1}\, d_{K-1} + x^\perp\| \end{aligned}$$

- $x^\perp$ does not change with μ_j, therefore to locate the closest centroid can ignore it.

Affine subspace defined by centroids of the classes

- Have K centroids in a p-dimensional input space: $\mu_1, \ldots, \mu_K$
- These centroids define an $K-1$ dimensional affine subspace H_{K-1} where if $u \in H_{K-1}$ then

$$\begin{aligned} u &= \mu_1 + \alpha_1(\mu_2 - \mu_1) + \alpha_2(\mu_3 - \mu_1) + \cdots + \alpha_{K-1}(\mu_K - \mu_1) \\ &= \mu_1 + \alpha_1\, d_1 + \alpha_2\, d_2 + \cdots + \alpha_{K-1}\, d_{K-1} \end{aligned}$$

- If $x \in \mathbb{R}^p$ then it can be written as

$$x = \mu_1 + \gamma_1\, d_1 + \gamma_2\, d_2 + \cdots + \gamma_{K-1}\, d_{K-1} + x^\perp, \quad \text{where } x^\perp \in H_{K-1}^\perp.$$

- If x has been whitened with respect to the common covariance matrix then the Mahalhobnis distance to centroid μ_j

$$\begin{aligned} \|x - \mu_j\| &= \|\mu_1 + \gamma_1\, d_1 + \gamma_2\, d_2 + \cdots + \gamma_{K-1}\, d_{K-1} + x^\perp - \mu_j\| \\ &= \|2\mu_1 + \gamma_1\, d_1 + \cdots + (\gamma_{j-1} - 1)\, d_{j-1} + \cdots + \gamma_{K-1}\, d_{K-1} + x^\perp\| \end{aligned}$$

- $x^\perp$ does not change with μ_j, therefore to locate the closest centroid can ignore it.

Affine subspace defined by centroids of the classes

- Have K centroids in a p-dimensional input space: $\mu_1, \ldots, \mu_K$
- These centroids define an $K-1$ dimensional affine subspace H_{K-1} where if $u \in H_{K-1}$ then

$$\begin{aligned} u &= \mu_1 + \alpha_1(\mu_2 - \mu_1) + \alpha_2(\mu_3 - \mu_1) + \cdots + \alpha_{K-1}(\mu_K - \mu_1) \\ &= \mu_1 + \alpha_1\, d_1 + \alpha_2\, d_2 + \cdots + \alpha_{K-1}\, d_{K-1} \end{aligned}$$

- If $x \in \mathbb{R}^p$ then it can be written as

$$x = \mu_1 + \gamma_1\, d_1 + \gamma_2\, d_2 + \cdots + \gamma_{K-1}\, d_{K-1} + x^\perp, \quad \text{where } x^\perp \in H_{K-1}^\perp.$$

- If x has been whitened with respect to the common covariance matrix then the Mahalhobnis distance to centroid μ_j

$$\begin{aligned} \|x - \mu_j\| &= \|\mu_1 + \gamma_1\, d_1 + \gamma_2\, d_2 + \cdots + \gamma_{K-1}\, d_{K-1} + x^\perp - \mu_j\| \\ &= \|2\mu_1 + \gamma_1\, d_1 + \cdots + (\gamma_{j-1} - 1)\, d_{j-1} + \cdots + \gamma_{K-1}\, d_{K-1} + x^\perp\| \end{aligned}$$

- $x^\perp$ does not change with μ_j, therefore to locate the closest centroid can ignore it.

Affine subspace defined by centroids of the classes

- Have K centroids in a p-dimensional input space: $\mu_1, \ldots, \mu_K$
- These centroids define an $K-1$ dimensional affine subspace H_{K-1} where if $u \in H_{K-1}$ then

$$\begin{aligned} u &= \mu_1 + \alpha_1(\mu_2 - \mu_1) + \alpha_2(\mu_3 - \mu_1) + \cdots + \alpha_{K-1}(\mu_K - \mu_1) \\ &= \mu_1 + \alpha_1\, d_1 + \alpha_2\, d_2 + \cdots + \alpha_{K-1}\, d_{K-1} \end{aligned}$$

- If $x \in \mathbb{R}^p$ then it can be written as

$$x = \mu_1 + \gamma_1\, d_1 + \gamma_2\, d_2 + \cdots + \gamma_{K-1}\, d_{K-1} + x^\perp, \quad \text{where } x^\perp \in H_{K-1}^\perp.$$

- If x has been whitened with respect to the common covariance matrix then the Mahalhobnis distance to centroid μ_j

$$\begin{aligned} \|x - \mu_j\| &= \|\mu_1 + \gamma_1\, d_1 + \gamma_2\, d_2 + \cdots + \gamma_{K-1}\, d_{K-1} + x^\perp - \mu_j\| \\ &= \|2\mu_1 + \gamma_1\, d_1 + \cdots + (\gamma_{j-1} - 1)\, d_{j-1} + \cdots + \gamma_{K-1}\, d_{K-1} + {\color{red}x^\perp}\| \end{aligned}$$

- ${\color{red}x^\perp}$ does not change with μ_j, therefore to locate the closest centroid can ignore it.

To summarize

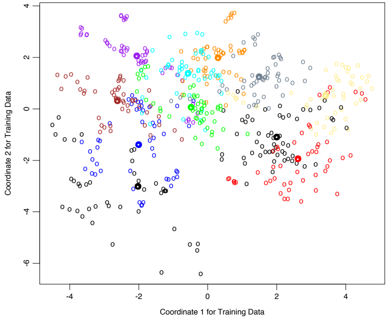
- K centroids in p-dimensional input space lie in an affine subspace of dimension $\leq K - 1$.
- If $p \gg K$ this is a big drop in dimension.
- To locate the closest centroid can ignore the directions orthogonal to this subspace if the data has been sphered.
- Therefore can just project X^* onto this centroid-spanning subspace H_{K-1} and make comparisons there.
- **LDA** thus performs dimensionality reduction and one need only consider the data in a subspace of dimension at most $K - 1$.

What about a subspace of dimension $L < K - 1$?

- If $K > 3$ can ask the question:
  Which subspace of dimensional $L < K - 1$ should we project onto for optimality w.r.t. LDA?

- **Fisher** defined optimal as the projected centroids are spread out as much as possible in terms of variance.

- Find the principal component subspace of the centroids.



- In this example have 11 classes with 10 dimensional input vectors.

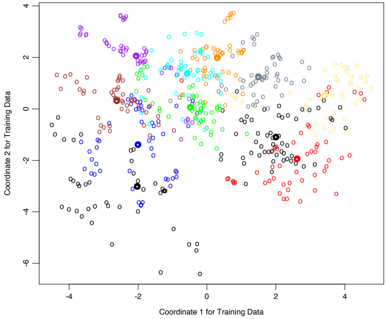
- The bold dots correspond to the centroids projected onto the top 2 principal directions.

What about a subspace of dimension $L < K - 1$?

- If $K > 3$ can ask the question:
  Which subspace of dimensional $L < K - 1$ should we project onto for optimality w.r.t. LDA?

- **Fisher** defined optimal as the projected centroids are spread out as much as possible in terms of variance.

- Find the principal component subspace of the centroids.

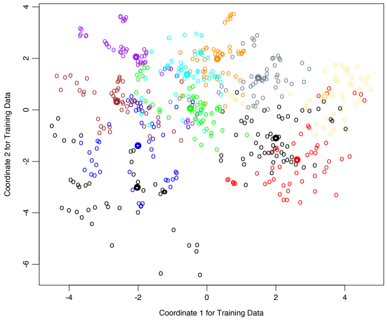


- In this example have 11 classes with 10 dimensional input vectors.

- The bold dots correspond to the centroids projected onto the top 2 principal directions.

What about a subspace of dimension $L < K - 1$?

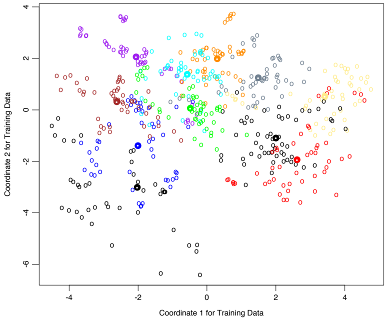
- If $K > 3$ can ask the question:
  Which subspace of dimensional $L < K - 1$ should we project onto for optimality w.r.t. LDA?
- **Fisher** defined optimal as the projected centroids are spread out as much as possible in terms of variance.
- Find the principal component subspace of the centroids.



- In this example have 11 classes with 10 dimensional input vectors.
- The bold dots correspond to the centroids projected onto the top 2 principal directions.

What about a subspace of dimension $L < K - 1$?

- If $K > 3$ can ask the question:
  Which subspace of dimensional $L < K - 1$ should we project onto for optimality w.r.t. LDA?
- **Fisher** defined optimal as the projected centroids are spread out as much as possible in terms of variance.
- Find the principal component subspace of the centroids.



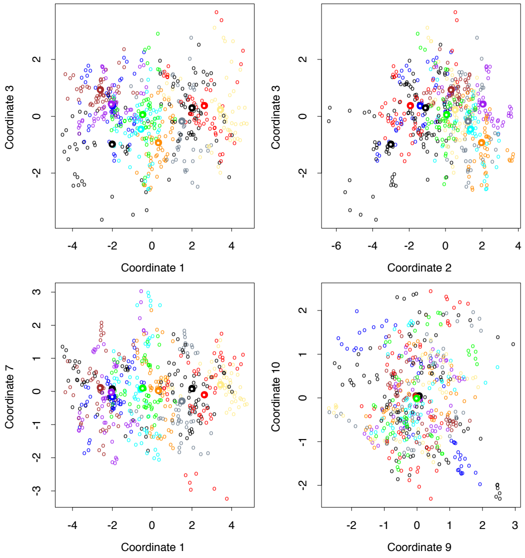
- In this example have 11 classes with 10 dimensional input vectors.
- The bold dots correspond to the centroids projected onto the top 2 principal directions.

The optimal sequence of subspaces

- To find the sequences of optimal subspaces for LDA:
    1. Compute the $K \times p$ matrix of class centroids M and the common covariance matrix W - the within-class variance.
    2. Compute $M^* = MW^{-\frac{1}{2}}$ using the eigen-decomposition of W
    3. Compute B^* the covariance matrix of M^* - the between-class variance.
    4. B^*'s eigen-decomposition is $B^* = V^* D_B V$. The columns of v_l^* of V^* define basis of the optimal subspace.
- The lth discriminant variable is given by $Z_l = v_l^* W^{-\frac{1}{2}} X$

The optimal sequence of subspaces

- To find the sequences of optimal subspaces for LDA:
  1. Compute the $K \times p$ matrix of class centroids M and the common covariance matrix W - the within-class variance.
  2. Compute $M^* = MW^{-\frac{1}{2}}$ using the eigen-decomposition of W
  3. Compute B^* the covariance matrix of M^* - the between-class variance.
  4. B^*'s eigen-decomposition is $B^* = V^* D_B V$. The columns of v_l^* of V^* define basis of the optimal subspace.
- The lth discriminant variable is given by $Z_l = v_l^* W^{-\frac{1}{2}} X$

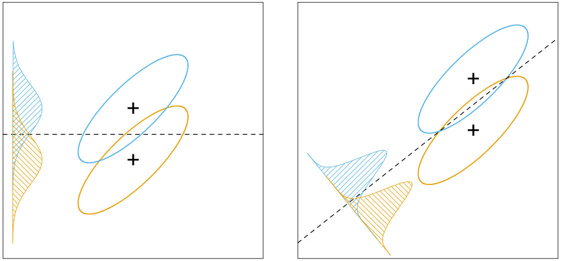


Note as the rank of the canonical variates increase the projected centroids become less spread out.

LDA via the Fisher criterion

Fisher arrived at this decomposition via a different route. He posed the problem

> *Find the linear combination $Z = aX$ such that the between-class variance is maximized relative to the within-class variance.*

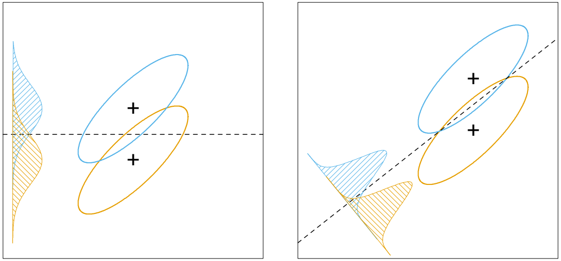


Why this criterion makes sense

LDA via the Fisher criterion

Fisher arrived at this decomposition via a different route. He posed the problem

> *Find the linear combination $Z = aX$ such that the between-class variance is maximized relative to the within-class variance.*



Why this criterion makes sense

The Fisher criterion

- W is the common covariance matrix of the original data X.
- B is the covariance matrix of the centroid matrix M
- Then for the projected data Z
  1. The between-class variance of Z is $a^t B a$
  2. The within-class variance of Z is $a^t W a$
- **Fisher**'s problem amounts to maximizing the *Raleigh quotient*

$$\boxed{\max_a \frac{a^t\, B\, a}{a^t\, W\, a}}$$

or equivalently

$$\boxed{\max_a a^t\, B\, a \;\; \textbf{subject to} \;\; a^t\, W\, a = 1}$$

The Fisher criterion

- W is the common covariance matrix of the original data X.
- B is the covariance matrix of the centroid matrix M
- Then for the projected data Z
  1. The between-class variance of Z is $a^t B a$
  2. The within-class variance of Z is $a^t W a$
- **Fisher**'s problem amounts to maximizing the *Raleigh quotient*

$$\boxed{\max_a \frac{a^t\, B\, a}{a^t\, W\, a}}$$

or equivalently

$$\boxed{\max_a a^t\, B\, a \;\; \textbf{subject to} \;\; a^t\, W\, a = 1}$$

The Fisher criterion

- W is the common covariance matrix of the original data X.
- B is the covariance matrix of the centroid matrix M
- Then for the projected data Z
  1. The between-class variance of Z is $a^t Ba$
  2. The within-class variance of Z is $a^t W a$
- **Fisher**'s problem amounts to maximizing the *Raleigh quotient*

$$\boxed{\max_a \frac{a^t\, B\, a}{a^t\, W\, a}}$$

or equivalently

$$\boxed{\max_a a^t\, B\, a \;\; \textbf{subject to} \;\; a^t\, W\, a = 1}$$

The Fisher criterion

- **Fisher**'s problem amounts to maximizing the *Raleigh quotient*

$$\boxed{a_1 = \arg\max_a a^t\, B\, a \ \ \textbf{subject to} \ \ a^t\, W\, a = 1}$$

- This is a generalized eigenvalue problem with a given by the largest eigenvalue of $W^{-1}B$.
- Can be shown that a_1 is equal to $W^{-\frac{1}{2}} v_1^*$ defined earlier.
- Can find the next direction a_2

$$\boxed{a_2 = \arg\max_a \frac{a^t\, B\, a}{a^t\, W\, a} \ \ \textbf{subject to} \ \ a^t\, W\, a_1 = 0}$$

  Once again $a_2 = W^{-\frac{1}{2}} v_2^*$.
- In a similar fashion can find $a_3, a_4, \ldots$

The Fisher criterion

- **Fisher**'s problem amounts to maximizing the *Raleigh quotient*

$$\boxed{a_1 = \arg\max_a a^t\, B\, a \;\; \textbf{subject to} \;\; a^t\, W\, a = 1}$$

- This is a generalized eigenvalue problem with a given by the largest eigenvalue of $W^{-1}B$.

- Can be shown that a_1 is equal to $W^{-\frac{1}{2}}v_1^*$ defined earlier.

- Can find the next direction a_2

$$\boxed{a_2 = \arg\max_a \frac{a^t\, B\, a}{a^t\, W\, a} \;\; \textbf{subject to} \;\; a^t\, W\, a_1 = 0}$$

  Once again $a_2 = W^{-\frac{1}{2}}v_2^*$.

- In a similar fashion can find $a_3, a_4, \ldots$

The Fisher criterion

- **Fisher**'s problem amounts to maximizing the *Raleigh quotient*

$$\boxed{a_1 = \arg\max_a a^t\, B\, a \ \ \textbf{subject to} \ \ a^t\, W\, a = 1}$$

- This is a generalized eigenvalue problem with a given by the largest eigenvalue of $W^{-1}B$.

- Can be shown that a_1 is equal to $W^{-\frac{1}{2}}v_1^*$ defined earlier.

- Can find the next direction a_2

$$\boxed{a_2 = \arg\max_a \frac{a^t\, B\, a}{a^t\, W\, a} \ \ \textbf{subject to} \ \ a^t\, W\, a_1 = 0}$$

Once again $a_2 = W^{-\frac{1}{2}}v_2^*$.

- In a similar fashion can find $a_3, a_4, \ldots$

The Fisher criterion

- **Fisher**'s problem amounts to maximizing the *Raleigh quotient*

$$\boxed{a_1 = \arg\max_a a^t\, B\, a \ \ \textbf{subject to} \ \ a^t\, W\, a = 1}$$

- This is a generalized eigenvalue problem with a given by the largest eigenvalue of $W^{-1}B$.

- Can be shown that a_1 is equal to $W^{-\frac{1}{2}}v_1^*$ defined earlier.

- Can find the next direction a_2

$$\boxed{a_2 = \arg\max_a \frac{a^t\, B\, a}{a^t\, W\, a} \ \ \textbf{subject to} \ \ a^t\, W\, a_1 = 0}$$

  Once again $a_2 = W^{-\frac{1}{2}}v_2^*$.

- In a similar fashion can find $a_3, a_4, \ldots$

The Fisher criterion

- **Fisher**'s problem amounts to maximizing the *Raleigh quotient*

$$\boxed{a_1 = \arg\max_a a^t\,B\,a \ \ \textbf{subject to} \ \ a^t\,W\,a = 1}$$

- This is a generalized eigenvalue problem with a given by the largest eigenvalue of $W^{-1}B$.
- Can be shown that a_1 is equal to $W^{-\frac{1}{2}}v_1^*$ defined earlier.
- Can find the next direction a_2

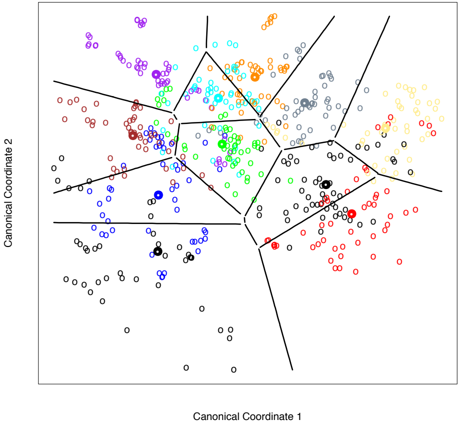
$$\boxed{a_2 = \arg\max_a \frac{a^t\,B\,a}{a^t\,W\,a} \ \ \textbf{subject to} \ \ a^t\,W\,a_1 = 0}$$

  Once again $a_2 = W^{-\frac{1}{2}}v_2^*$.

- In a similar fashion can find $a_3, a_4, \ldots$

Classification in the reduced subspace

- The a_l's are referred to as discriminant coordinates or canonical variates.



- In this example have 11 classes with 10 dimensional input vectors.

- The decision boundaries based on using basic linear discrimination in the low dimensional space given by the first 2 canonical variates.

Logistic Regression

Logistic regression

- Arises from trying to model the posterior probabilities of the K classes using linear functions in x while ensuring they sum to one.

- The simple model used is for $k = 1, \ldots, K-1$

$$\boxed{P(G = k|X = x) = \frac{\exp(\beta_{k0} + \beta_k^t x)}{1 + \sum_{l=1}^{K-1} \exp(\beta_{l0} + \beta_l^t x)}}$$

  and $k = K$

$$\boxed{P(G = K|X = x) = \frac{1}{1 + \sum_{l=1}^{K-1} \exp(\beta_{l0} + \beta_l^t x)}}$$

- These posterior probabilities clearly sum to one.

Logistic regression

- Arises from trying to model the posterior probabilities of the K classes using linear functions in x while ensuring they sum to one.

- The simple model used is for $k = 1, \ldots, K-1$

$$\boxed{P(G = k|X = x) = \frac{\exp(\beta_{k0} + \beta_k^t x)}{1 + \sum_{l=1}^{K-1} \exp(\beta_{l0} + \beta_l^t x)}}$$

  and $k = K$

$$\boxed{P(G = K|X = x) = \frac{1}{1 + \sum_{l=1}^{K-1} \exp(\beta_{l0} + \beta_l^t x)}}$$

- These posterior probabilities clearly sum to one.

Logistic regression

- This model: $k = 1, \dots, K-1$

$$\boxed{P(G = k|X = x) = \frac{\exp(\beta_{k0} + \beta_k^t x)}{1 + \sum_{l=1}^{K-1} \exp(\beta_{l0} + \beta_l^t x)}}$$

and $k = K$

$$\boxed{P(G = K|X = x) = \frac{1}{1 + \sum_{l=1}^{K-1} \exp(\beta_{l0} + \beta_l^t x)}}$$

induces linear decision boundaries between classes as

$$\{x : P(G = k|X = x) = P(G = l|X = x)\}$$

is the same as

$$\{x : (\beta_{k0} - \beta_{l0}) + (\beta_k - \beta_l)^t x = 0\}$$

for $1 \leq k < K$ and $1 \leq l < K$.

Fitting Logistic regression models

- To simplify notation let
  1. $\theta = \{\beta_{10}, \beta_1^t, \beta_{20}, \beta_2^t, \ldots\}$ **and**
  2. $P(G = k | X = x) = p_k(x; \theta)$
- Given training data $\{(x_i, g_i)\}_{i=1}^n$ one usually fits the logistic regression model by maximum likelihood.
- The log-likelihood for the n observations is

$$\ell(\theta) = \log\left(\prod_{i=1}^n p_{g_i}(x_i; \theta)\right) = \sum_{i=1}^n \log(p_{g_i}(x_i; \theta))$$

in my opinion this is an abuse of terminology as the posterior probabilities are being used...

Fitting Logistic regression models: The two class case

$$\boxed{p_1(x;\beta) = \frac{\exp(\beta^t x)}{1+\exp(\beta^t x)} \quad \textbf{and} \quad p_2(x;\beta) = 1 - p_1(x;\beta)}$$

Let $\beta = \theta = (\beta_{10}, \beta_1^t)$ and assume x_i's include the constant term 1.

A convenient way to write the likelihood for one sample (x_i, g_i) is:

- Code the two-class g_i as a $\{0,1\}$ response y_i where

$$y_i = \begin{cases} 1 & \text{if } g_i = 1 \\ 0 & \text{if } g_i = 2 \end{cases}$$

- Then one can write

$$\boxed{p_{g_i}(x_i;\beta) = y_i\, p_1(x_i;\beta) + (1-y_i)(1-p_1(x_i;\beta))}$$

Fitting Logistic regression models: The two class case

Similarly

$$\boxed{\log p_{g_i}(x_i;\beta) = y_i\, \log p_1(x_i;\beta) + (1-y_i)\log(1-p_1(x_i;\beta))}$$

The log-likelihood of the data becomes

$$\begin{aligned}
\ell(\beta) &= \sum_{i=1}^{n} \left[y_i\, \log p_1(x_i;\beta) + (1-y_i)\log(1-p_1(x_i;\beta))\right] \\
&= \sum_{i=1}^{n} \left[y_i \beta^t x_i - y_i\, \log(1+e^{\beta^t x_i}) - (1-y_i)\log(1+e^{\beta^t x_i})\right] \\
&= \sum_{i=1}^{n} \left[y_i \beta^t x_i - \log(1+e^{\beta^t x_i})\right]
\end{aligned}$$

Fitting Logistic regression models: The two class case

$$\boxed{\ell(\beta) = \sum_{i=1}^{n} \left[y_i \beta^t x_i - \log(1 + e^{\beta^t x_i}) \right]}$$

- To maximize the log-likelihood set its derivatives to zero to get

$$\begin{aligned}
\frac{\partial \ell(\beta)}{\partial \beta} &= \sum_{i=1}^{n} \left[x_i y_i - x_i \frac{\exp(\beta^t x_i)}{1 + \exp(\beta^t x_i)} \right] \\
&= \sum_{i=1}^{n} x_i \left(y_i - \frac{\exp(\beta^t x_i)}{1 + \exp(\beta^t x_i)} \right) \\
&= \sum_{i=1}^{n} x_i (y_i - p_1(x_i; \beta)) = 0
\end{aligned}$$

- These are $(p + 1)$ equations **non-linear** equations in β.
- Must solve iteratively and in the book they use the **Newton-Raphson** algorithm.

The two class case: Iterative optimization

Newton-Raphson requires both the gradient

$$\boxed{\frac{\partial \ell(\beta)}{\partial \beta} = \sum_{i=1}^{n} x_i (y_i - p_1(x_i;\beta))}$$

and Hessian matrix

$$\boxed{\frac{\partial \ell(\beta)}{\partial \beta \partial \beta^t} = -\sum_{i=1}^{n} x_i x_i^t \, p_1(x_i;\beta)(1 - p_1(x_i;\beta))}$$

Starting with β^{old}, a single Newton update step is

$$\boxed{\beta_{\text{new}} = \beta^{\text{old}} - \left(\frac{\partial \ell(\beta)}{\partial \beta \partial \beta^t} \right)^{-1} \frac{\partial \ell(\beta)}{\partial \beta}}$$

where the derivatives are calculated at β^{old}.

Iterative optimization in matrix notation

Write the Hessian and gradient in matrix notation. Let

- $\mathbf{X}$ be the $N \times (p+1)$ matrix with $(1, x_i^t)$ on each row,
- $p = (p_1(x_1; \beta^{\text{old}}), p_1(x_2; \beta^{\text{old}}), \ldots, p_1(x_n; \beta^{\text{old}}))^t$
- $\mathbf{W}$ is $n \times n$ diagonal matrix with ith diagonal element $p_1(x_1; \beta^{\text{old}})(1 - p_1(x_1; \beta^{\text{old}}))$.

Then

$$\boxed{\frac{\partial \ell(\beta)}{\partial \beta} = \mathbf{X}^t (y - p)}$$

and

$$\boxed{\frac{\partial \ell(\beta)}{\partial \beta \partial \beta^t} = -\mathbf{X}^t \mathbf{W} \mathbf{X}}$$

Iterative optimization as iterative weighted ls

The Newton step is then

$$\begin{aligned}\beta^{\text{new}} &= \beta^{\text{old}} + (\mathbf{X}^t\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^t(y-p)\\ &= (\mathbf{X}^t\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^t\mathbf{W}\left(\mathbf{X}\beta^{\text{old}} + \mathbf{W}^{-1}(y-p)\right)\\ &= (\mathbf{X}^t\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^t\mathbf{W}z\end{aligned}$$

Have re-expressed the Newton step as a weighted least squares step

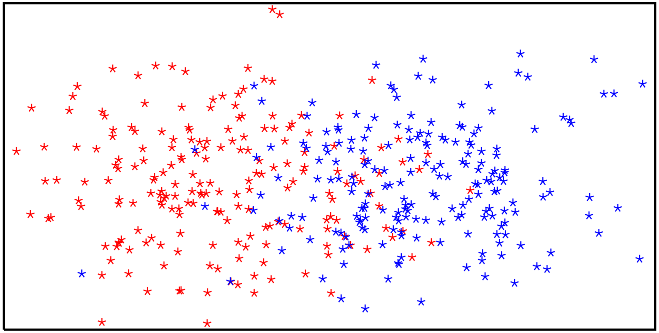
$$\boxed{\beta^{\text{new}} = \arg\min_{\beta}\,(z-\mathbf{X}\beta)^t\mathbf{W}(z-\mathbf{X}\beta)}$$

with response

$$\boxed{z = \mathbf{X}\beta^{\text{old}} + \mathbf{W}^{-1}(y-p)}$$

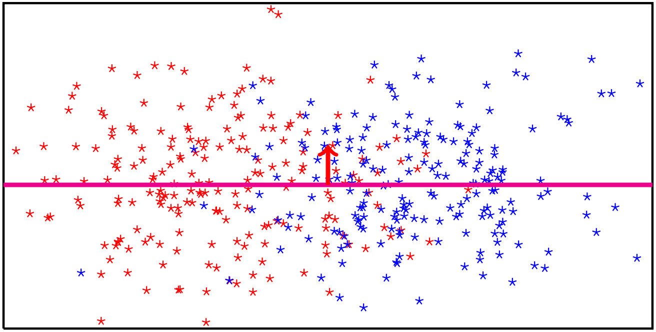
known as the adjusted response. Note at iteration each $\mathbf{W}, p$ and z change.

An toy example



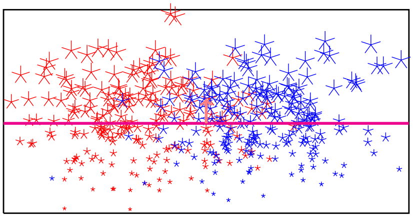
- Two class problem with 2 dimensional input vectors.
- Use **Logistic Regression** to find a decision boundary

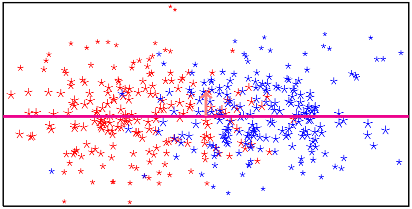
Illustration of the optimization process

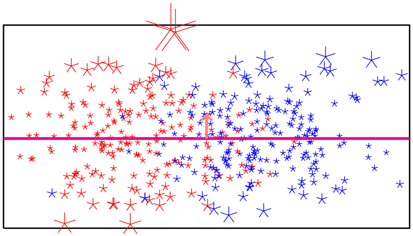


- The current estimate $\hat{\beta}^{\text{cur}}$

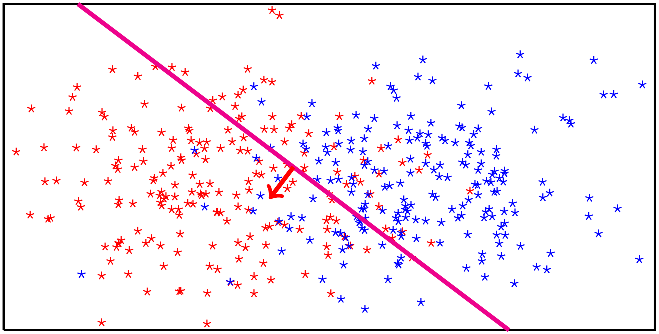
Quantities involved in the weighted least sqs


Size $\propto p_1(x_i; \hat{\beta}^{\text{cur}})$


Size $\propto p_1(x_i; \hat{\beta}^{\text{cur}})(1 - p_1(x_i; \hat{\beta}^{\text{cur}})) = \mathbf{W}_{ii}$

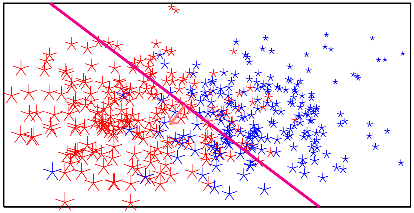

Size $\propto 1/\mathbf{W}_{ii}$

Update the estimate of $\hat{\beta}^{\text{cur}}$

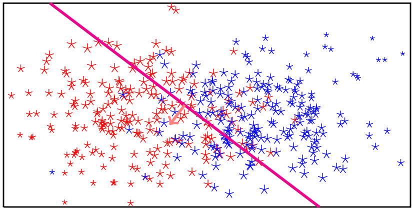


- The current estimate $\hat{\beta}^{\text{cur}}$

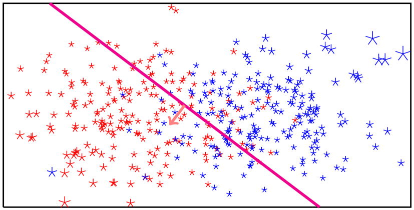
Quantities involved in the weighted least sqs



Size $\propto p_1(x_i; \hat{\beta}^{\textsf{cur}})$

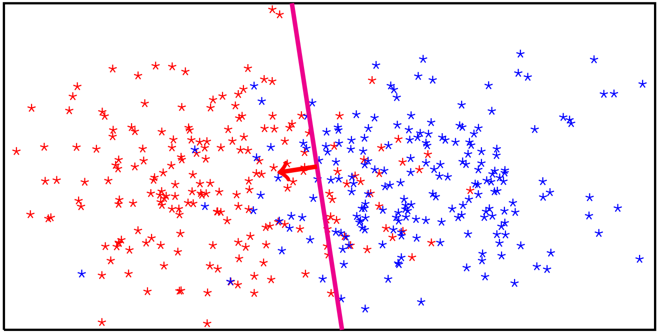


Size $\propto p_1(x_i; \hat{\beta}^{\textsf{cur}})(1 - p_1(x_i; \hat{\beta}^{\textsf{cur}})) = \mathbf{W}_{ii}$



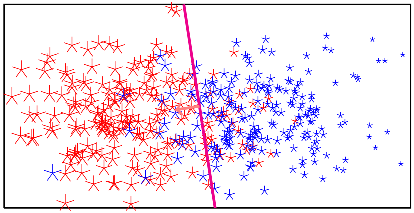
Size $\propto 1/\mathbf{W}_{ii}$

Update the estimate of $\hat{\beta}^{\text{cur}}$

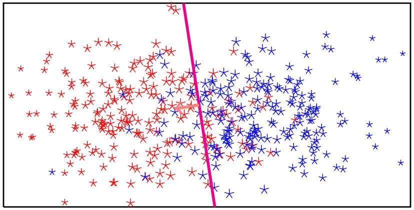


- The current estimate $\hat{\beta}^{\text{cur}}$

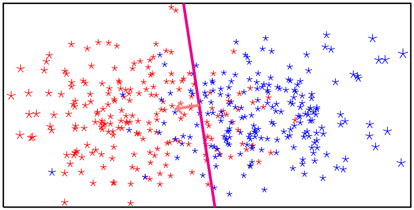
Quantities involved in the weighted least sqs



Size $\propto p_1(x_i; \hat{\beta}^{\textbf{cur}})$

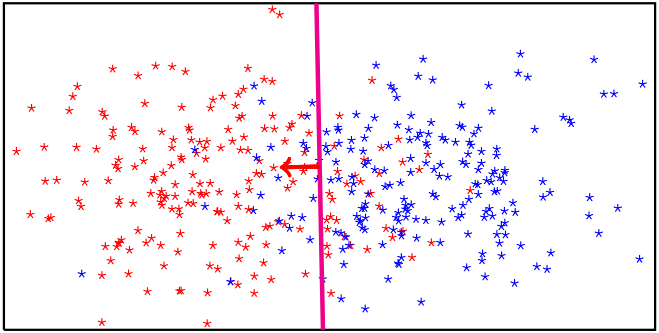


Size $\propto p_1(x_i; \hat{\beta}^{\textbf{cur}})(1 - p_1(x_i; \hat{\beta}^{\textbf{cur}})) = \mathbf{W}_{ii}$



Size $\propto 1/\mathbf{W}_{ii}$

Update the estimate of $\hat{\beta}^{\text{cur}}$



- The current estimate $\hat{\beta}^{\text{cur}}$
- Logistic regression converges to this decision boundary.

L_1 **regularized logistic regression**

L_1 regularized logistic regression

The L_1 penalty can be used for variable selection in logistic regression by maximizing a penalized version of the log-likelihood

$$\boxed{\max_{\beta_0,\beta_1}\left\{\sum_{i=1}^{n}\left[y_i(\beta_0+\beta^t x_i)-\log(1+e^{\beta_0+\beta^t x_i})\right]-\lambda\sum_{j=1}^{p}|\beta_j|\right\}}$$

Note:

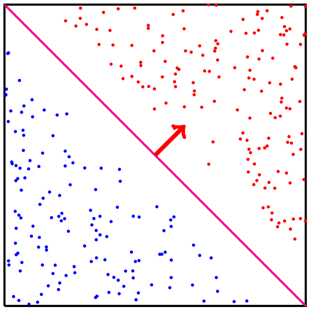
- the intercept, β_0, is not included in the penalty term,
- the predictors should be standardized to ensure the penalty is meaningful,
- the above cost function is concave and a solution can be found using non-linear programming methods.

Separating Hyperplanes

Directly estimating separating hyperplanes

- In this section describe separating hyperplane classifiers - will only consider separable training data.
- Construct linear decision boundaries that explicitly try to separate the data into different classes as well as possible.
- A hyperplane is defined as

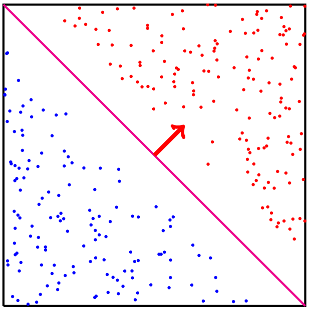
$$\{x : \hat{\beta}_0 + \hat{\beta}^t x = 0\}$$



Directly estimating separating hyperplanes

- In this section describe separating hyperplane classifiers - will only consider separable training data.
- Construct linear decision boundaries that explicitly try to separate the data into different classes as well as possible.
- A hyperplane is defined as

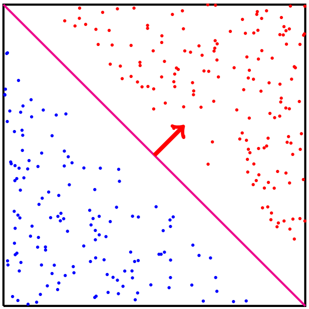
$$\{x : \hat{\beta}_0 + \hat{\beta}^t x = 0\}$$



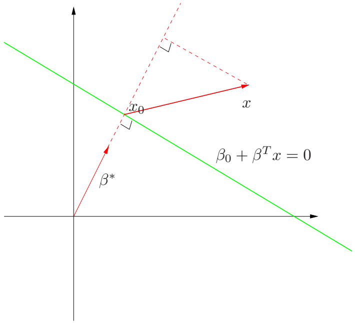
Directly estimating separating hyperplanes

- In this section describe separating hyperplane classifiers - will only consider separable training data.
- Construct linear decision boundaries that explicitly try to separate the data into different classes as well as possible.
- A hyperplane is defined as

$$\{x : \hat{\beta}_0 + \hat{\beta}^t x = 0\}$$

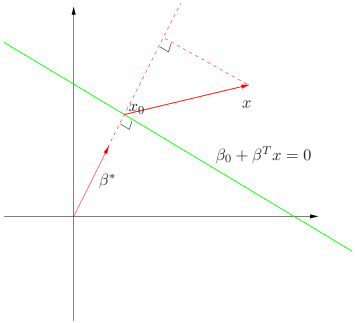


Review of some vector algebra



- Above is shown a hyperplane L defined by
$$f(x) = \beta_0 + \beta^t x = 0$$
- If $x_1, x_2 \in L$ then $\beta^t(x_1 - x_2) = 0 \implies \beta^* = \beta/\|\beta\|$ is normal to L
- If $x_0 \in L$ then $\beta^t x_0 = -\beta_0$.
- The signed distance of point x to L is
$$\beta^{*t}(x - x_0) = \frac{1}{\|\beta\|}(\beta^t x + \beta_0) = \frac{1}{\|f'(x)\|} f(x) \propto f(x)$$

Review of some vector algebra



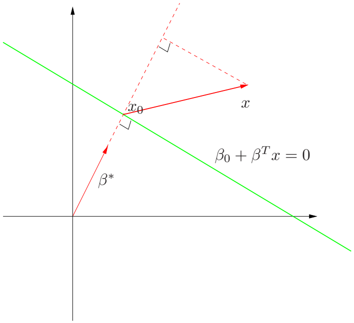
- Above is shown a hyperplane L defined by

$$f(x) = \beta_0 + \beta^t x = 0$$

- If $x_1, x_2 \in L$ then $\beta^t(x_1 - x_2) = 0 \implies \beta^* = \beta / \|\beta\|$ is normal to L
- If $x_0 \in L$ then $\beta^t x_0 = -\beta_0$.
- The signed distance of point x to L is

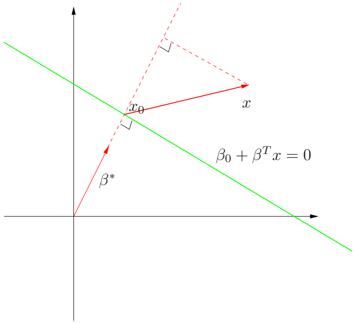
$$\beta^{*t}(x - x_0) = \frac{1}{\|\beta\|}(\beta^t x + \beta_0) = \frac{1}{\|f'(x)\|} f(x) \propto f(x)$$

Review of some vector algebra



- Above is shown a hyperplane L defined by
$$f(x) = \beta_0 + \beta^t x = 0$$
- If $x_1, x_2 \in L$ then $\beta^t(x_1 - x_2) = 0 \implies \beta^* = \beta / \|\beta\|$ is normal to L
- If $x_0 \in L$ then $\beta^t x_0 = -\beta_0$.
- The signed distance of point x to L is
$$\beta^{*t}(x - x_0) = \frac{1}{\|\beta\|}(\beta^t x + \beta_0) = \frac{1}{\|f'(x)\|} f(x) \propto f(x)$$

Review of some vector algebra



- Above is shown a hyperplane L defined by
$$f(x) = \beta_0 + \beta^t x = 0$$
- If $x_1, x_2 \in L$ then $\beta^t(x_1 - x_2) = 0 \implies \beta^* = \beta/\|\beta\|$ is normal to L
- If $x_0 \in L$ then $\beta^t x_0 = -\beta_0$.
- The signed distance of point x to L is
$$\beta^{*t}(x - x_0) = \frac{1}{\|\beta\|}(\beta^t x + \beta_0) = \frac{1}{\|f'(x)\|} f(x) \propto f(x)$$

Perceptron Learning

Rosenblatt's Perceptron Learning Algorithm

Perceptron learning algorithm tries to find a separating hyperplane by minimizing the distance of misclassified points to the decision boundary.

The Objective Function

- Have labelled training data $\{(x_i, y_i)\}$ with $x_i \in \mathbb{R}^p$ and $y_i \in \{-1, 1\}$.
- A point x_i is misclassified if $\operatorname{sign}(\beta_0 + \beta^t x_i) \neq y_i$
- This can be re-stated as: a point x_i is misclassified if
$$y_i(\beta_0 + \beta^t x_i) < 0$$
- The goal is to find β_0 and β which minimize
$$\boxed{D(\beta, \beta_0) = -\sum_{i \in \mathcal{M}} y_i(x_i^t \beta + \beta_0)}$$
where $\mathcal{M}$ is the index of the misclassified points.

Rosenblatt's Perceptron Learning Algorithm

Perceptron learning algorithm tries to find a separating hyperplane by minimizing the distance of misclassified points to the decision boundary.

The Objective Function

- Have labelled training data $\{(x_i, y_i)\}$ with $x_i \in \mathbb{R}^p$ and $y_i \in \{-1, 1\}$.
- A point x_i is misclassified if $\text{sign}(\beta_0 + \beta^t x_i) \neq y_i$
- This can be re-stated as: a point x_i is misclassified if

$$y_i(\beta_0 + \beta^t x_i) < 0$$

- The goal is to find β_0 and β which minimize

$$\boxed{D(\beta, \beta_0) = -\sum_{i \in \mathcal{M}} y_i(x_i^t \beta + \beta_0)}$$

where $\mathcal{M}$ is the index of the misclassified points.

Rosenblatt's Perceptron Learning Algorithm

Perceptron learning algorithm tries to find a separating hyperplane by minimizing the distance of misclassified points to the decision boundary.

The Objective Function

- Have labelled training data $\{(x_i, y_i)\}$ with $x_i \in \mathbb{R}^p$ and $y_i \in \{-1, 1\}$.
- A point x_i is misclassified if $\text{sign}(\beta_0 + \beta^t x_i) \neq y_i$
- This can be re-stated as: a point x_i is misclassified if
$$y_i(\beta_0 + \beta^t x_i) < 0$$
- The goal is to find β_0 and β which minimize
$$\boxed{D(\beta, \beta_0) = -\sum_{i \in \mathcal{M}} y_i(x_i^t \beta + \beta_0)}$$
where $\mathcal{M}$ is the index of the misclassified points.

Rosenblatt's Perceptron Learning Algorithm

Perceptron learning algorithm tries to find a separating hyperplane by minimizing the distance of misclassified points to the decision boundary.

The Objective Function

- Have labelled training data $\{(x_i, y_i)\}$ with $x_i \in \mathbb{R}^p$ and $y_i \in \{-1, 1\}$.
- A point x_i is misclassified if $\text{sign}(\beta_0 + \beta^t x_i) \neq y_i$
- This can be re-stated as: a point x_i is misclassified if

$$y_i(\beta_0 + \beta^t x_i) < 0$$

- The goal is to find β_0 and β which minimize

$$\boxed{D(\beta, \beta_0) = -\sum_{i \in \mathcal{M}} y_i(x_i^t \beta + \beta_0)}$$

where $\mathcal{M}$ is the index of the misclassified points.

Rosenblatt's Perceptron Learning Algorithm

Perceptron learning algorithm tries to find a separating hyperplane by minimizing the distance of misclassified points to the decision boundary.

The Objective Function

- Have labelled training data $\{(x_i, y_i)\}$ with $x_i \in \mathbb{R}^p$ and $y_i \in \{-1, 1\}$.
- A point x_i is misclassified if $\operatorname{sign}(\beta_0 + \beta^t x_i) \neq y_i$
- This can be re-stated as: a point x_i is misclassified if
$$y_i(\beta_0 + \beta^t x_i) < 0$$
- The goal is to find β_0 and β which minimize
$$\boxed{D(\beta, \beta_0) = -\sum_{i \in \mathcal{M}} y_i(x_i^t \beta + \beta_0)}$$
where $\mathcal{M}$ is the index of the misclassified points.

Perceptron Learning: The Objective Function

Want to find β_0 and β which minimize

$$D(\beta, \beta_0) = -\sum_{i \in \mathcal{M}} y_i (x_i^t \beta + \beta_0) = -\sum_{i \in \mathcal{M}} y_i \, f_{\beta,\beta_0}(x_i)$$

- $D(\beta, \beta_0)$ is non-negative.
- $D(\beta, \beta_0)$ is proportional to the distance of the misclassified points to the decision boundary defined by $\beta_0 + \beta^t x = 0$.

Questions:

- Is there a unique β, β_0 which minimizes $D(\beta, \beta_0)$ (disregarding re-scaling of β and β_0) ?
- Can we say anything about the form of $D(\beta, \beta_0)$?

Perceptron Learning: The Objective Function

Want to find β_0 and β which minimize

$$D(\beta, \beta_0) = -\sum_{i \in \mathcal{M}} y_i (x_i^t \beta + \beta_0) = -\sum_{i \in \mathcal{M}} y_i \, f_{\beta,\beta_0}(x_i)$$

- $D(\beta, \beta_0)$ is non-negative.
- $D(\beta, \beta_0)$ is proportional to the distance of the misclassified points to the decision boundary defined by $\beta_0 + \beta^t x = 0$.

Questions:

- Is there a unique β, β_0 which minimizes $D(\beta, \beta_0)$ (disregarding re-scaling of β and β_0) ?
- Can we say anything about the form of $D(\beta, \beta_0)$?

Perceptron Learning: The Objective Function

Want to find β_0 and β which minimize

$$D(\beta, \beta_0) = -\sum_{i \in \mathcal{M}} y_i (x_i^t \beta + \beta_0) = -\sum_{i \in \mathcal{M}} y_i \, f_{\beta,\beta_0}(x_i)$$

- $D(\beta, \beta_0)$ is non-negative.
- $D(\beta, \beta_0)$ is proportional to the distance of the misclassified points to the decision boundary defined by $\beta_0 + \beta^t x = 0$.

Questions:

- Is there a unique β, β_0 which minimizes $D(\beta, \beta_0)$ (disregarding re-scaling of β and β_0) ?
- Can we say anything about the form of $D(\beta, \beta_0)$?

Perceptron Learning: The Objective Function

Want to find β_0 and β which minimize

$$D(\beta, \beta_0) = -\sum_{i \in \mathcal{M}} y_i (x_i^t \beta + \beta_0) = -\sum_{i \in \mathcal{M}} y_i \, f_{\beta, \beta_0}(x_i)$$

- $D(\beta, \beta_0)$ is non-negative.
- $D(\beta, \beta_0)$ is proportional to the distance of the misclassified points to the decision boundary defined by $\beta_0 + \beta^t x = 0$.

Questions:

- Is there a unique β, β_0 which minimizes $D(\beta, \beta_0)$ (disregarding re-scaling of β and β_0) ?
- Can we say anything about the form of $D(\beta, \beta_0)$?

Perceptron Learning: The Objective Function

Want to find β_0 and β which minimize

$$D(\beta, \beta_0) = -\sum_{i \in \mathcal{M}} y_i(x_i^t \beta + \beta_0) = -\sum_{i \in \mathcal{M}} y_i \, f_{\beta,\beta_0}(x_i)$$

- $D(\beta, \beta_0)$ is non-negative.
- $D(\beta, \beta_0)$ is proportional to the distance of the misclassified points to the decision boundary defined by $\beta_0 + \beta^t x = 0$.

Questions:

- Is there a unique β, β_0 which minimizes $D(\beta, \beta_0)$ (disregarding re-scaling of β and β_0) ?
- Can we say anything about the form of $D(\beta, \beta_0)$?

Perceptron Learning: Optimizing the Objective Function

- The gradient, assuming a fixed $\mathcal{M}$, is given by

$$\frac{\partial D(\beta, \beta_0)}{\partial \beta} = -\sum_{i \in \mathcal{M}} y_i \, x_i, \qquad \frac{\partial D(\beta, \beta_0)}{\partial \beta_0} = -\sum_{i \in \mathcal{M}} y_i$$

- **Stochastic gradient descent** is used to minimize $D(\beta, \beta_0)$ so an update step is made after each observation is visited.

- Identify a misclassified example wrt the current estimate of β and β_0 and make the update

$$\boxed{\beta \leftarrow \beta + \rho y_i x_i \qquad \textbf{and} \qquad \beta_0 \leftarrow \beta_0 + \rho y_i}$$

  where ρ is the learning rate.

- Repeat this step until no points are misclassified.

Perceptron Learning: Optimizing the Objective Function

- The gradient, assuming a fixed $\mathcal{M}$, is given by

$$\frac{\partial D(\beta, \beta_0)}{\partial \beta} = -\sum_{i \in \mathcal{M}} y_i \, x_i, \qquad \frac{\partial D(\beta, \beta_0)}{\partial \beta_0} = -\sum_{i \in \mathcal{M}} y_i$$

- **Stochastic gradient descent** is used to minimize $D(\beta, \beta_0)$ so an update step is made after each observation is visited.

- Identify a misclassified example wrt the current estimate of β and β_0 and make the update

$$\boxed{\beta \leftarrow \beta + \rho y_i x_i \qquad \textbf{and} \qquad \beta_0 \leftarrow \beta_0 + \rho y_i}$$

  where ρ is the learning rate.

- Repeat this step until no points are misclassified.

Perceptron Learning: Optimizing the Objective Function

- The gradient, assuming a fixed $\mathcal{M}$, is given by

$$\frac{\partial D(\beta, \beta_0)}{\partial \beta} = -\sum_{i \in \mathcal{M}} y_i \, x_i, \qquad \frac{\partial D(\beta, \beta_0)}{\partial \beta_0} = -\sum_{i \in \mathcal{M}} y_i$$

- **Stochastic gradient descent** is used to minimize $D(\beta, \beta_0)$ so an update step is made after each observation is visited.

- Identify a misclassified example wrt the current estimate of β and β_0 and make the update

$$\boxed{\beta \leftarrow \beta + \rho y_i x_i \qquad \textbf{and} \qquad \beta_0 \leftarrow \beta_0 + \rho y_i}$$

  where ρ is the learning rate.

- Repeat this step until no points are misclassified.

Perceptron Learning: Optimizing the Objective Function

- The gradient, assuming a fixed $\mathcal{M}$, is given by

$$\frac{\partial D(\beta, \beta_0)}{\partial \beta} = -\sum_{i \in \mathcal{M}} y_i\, x_i, \qquad \frac{\partial D(\beta, \beta_0)}{\partial \beta_0} = -\sum_{i \in \mathcal{M}} y_i$$

- **Stochastic gradient descent** is used to minimize $D(\beta, \beta_0)$ so an update step is made after each observation is visited.

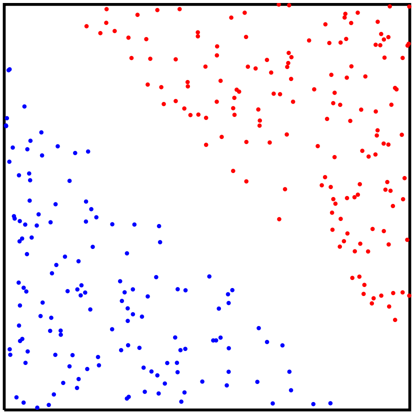
- Identify a misclassified example wrt the current estimate of β and β_0 and make the update

$$\boxed{\beta \leftarrow \beta + \rho y_i x_i \qquad \textbf{and} \qquad \beta_0 \leftarrow \beta_0 + \rho y_i}$$

  where ρ is the learning rate.

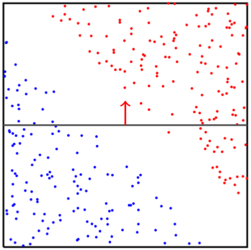
- Repeat this step until no points are misclassified.

Perceptron Learning: An Example

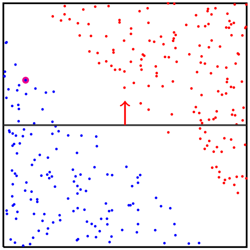


Want to find a separating hyperplane between the red and blue points.

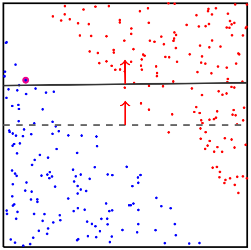
Perceptron Learning: One Iteration



Current estimate
$\beta^{(0)}$

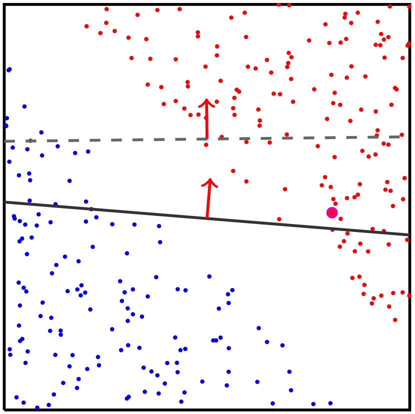


Point misclassified
by $\beta^{(0)}$



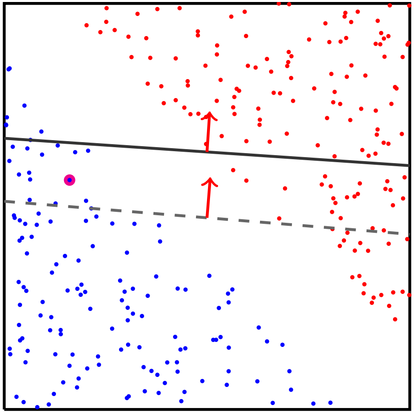
Use gradient at point
to get $\beta^{(1)}$

Perceptron Learning: Sequence of iterations



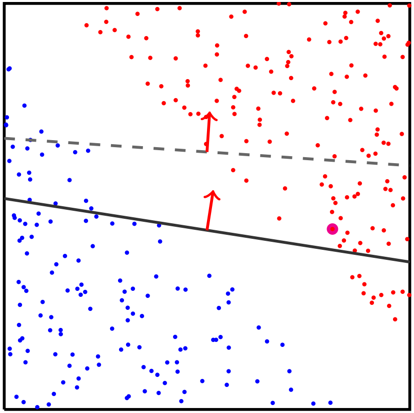
$\beta^{(2)}$

Perceptron Learning: Sequence of iterations



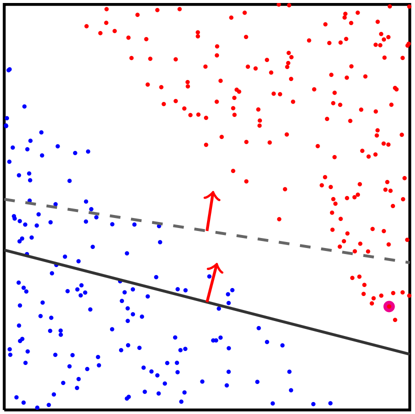
$\beta^{(3)}$

Perceptron Learning: Sequence of iterations



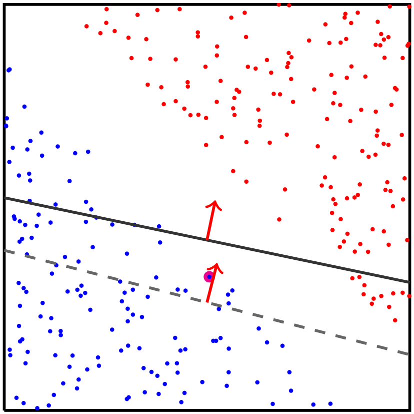
$\beta^{(4)}$

Perceptron Learning: Sequence of iterations



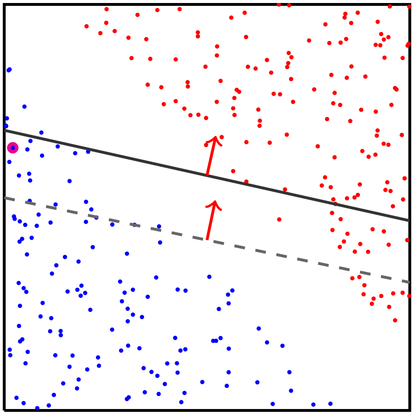
$\beta^{(5)}$

Perceptron Learning: Sequence of iterations



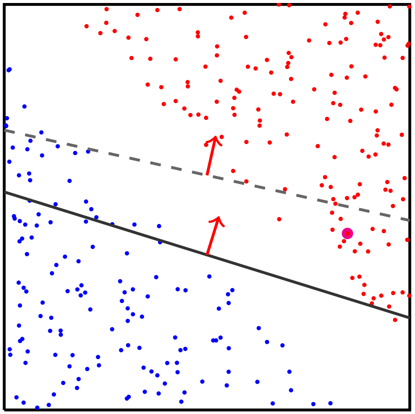
$\beta^{(6)}$

Perceptron Learning: Sequence of iterations



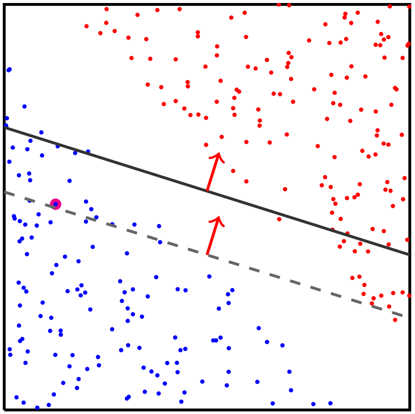
$\beta^{(7)}$

Perceptron Learning: Sequence of iterations



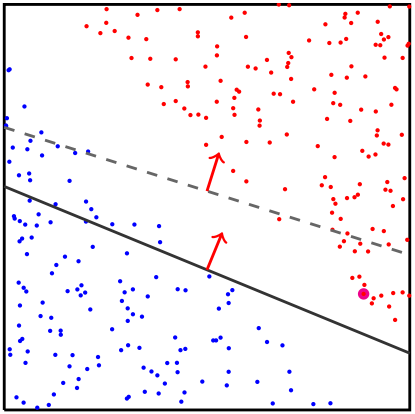
$\beta^{(8)}$

Perceptron Learning: Sequence of iterations



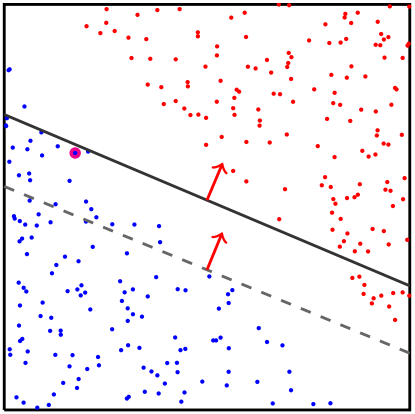
$\beta^{(9)}$

Perceptron Learning: Sequence of iterations



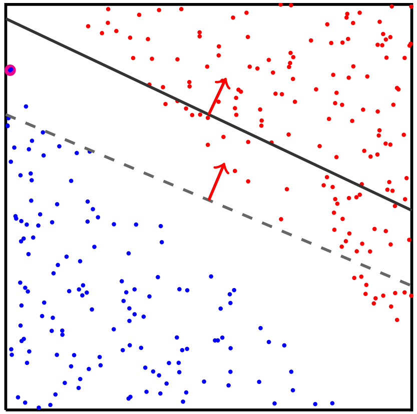
$\beta^{(10)}$

Perceptron Learning: Sequence of iterations



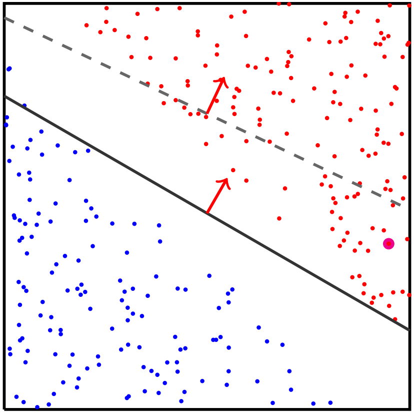
$\beta^{(11)}$

Perceptron Learning: Sequence of iterations



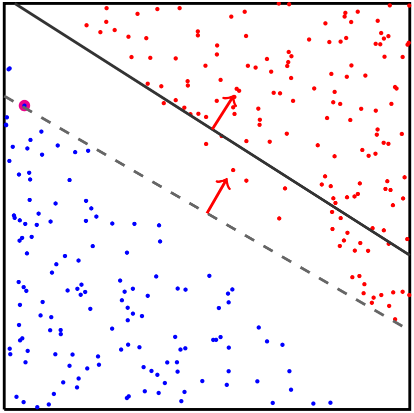
$\beta^{(12)}$

Perceptron Learning: Sequence of iterations



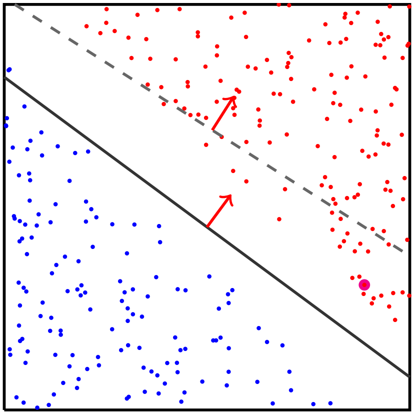
$\beta^{(13)}$

Perceptron Learning: Sequence of iterations



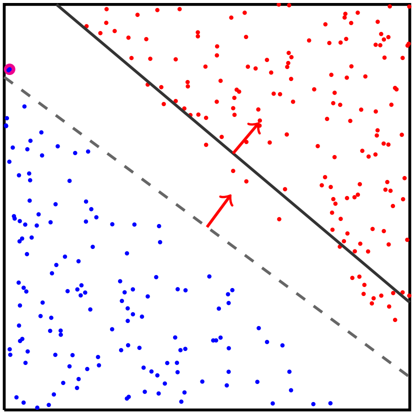
$\beta^{(14)}$

Perceptron Learning: Sequence of iterations



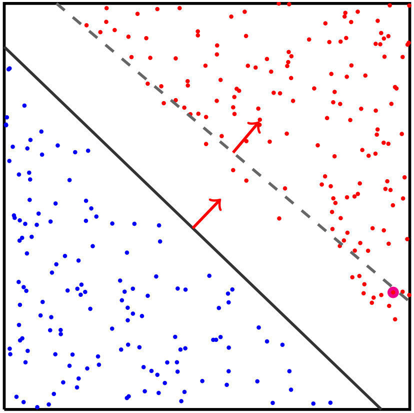
$\beta^{(15)}$

Perceptron Learning: Sequence of iterations



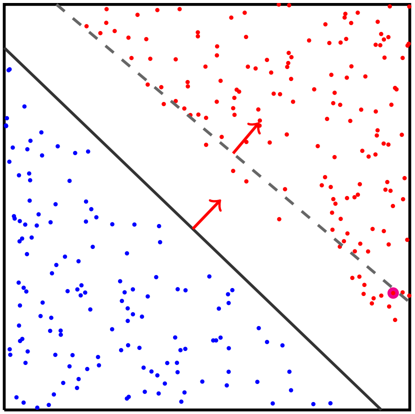
$\beta^{(16)}$

Perceptron Learning: Sequence of iterations



$\beta^{(17)}$

Perceptron Learning: Sequence of iterations



$\beta^{(17)}$

Is this the best separating hyperplane we could have found?

Perceptron Learning Algorithm: Properties

Pros

- If the classes are linearly separable, the algorithm converges to a separating hyperplane in a finite number of steps.

Cons

- All separating hyperplanes are considered equally valid.
- One found depends on the initial guess for β and β_0.
- The **finite** number of steps can be very large.
- If the data is non-separable, the algorithm will not converge.

Perceptron Learning Algorithm: Properties

Pros

- If the classes are linearly separable, the algorithm converges to a separating hyperplane in a finite number of steps.

Cons

- All separating hyperplanes are considered equally valid.
- One found depends on the initial guess for β and β_0.
- The **finite** number of steps can be very large.
- If the data is non-separable, the algorithm will not converge.

Perceptron Learning Algorithm: Properties

Pros

- If the classes are linearly separable, the algorithm converges to a separating hyperplane in a finite number of steps.

Cons

- All separating hyperplanes are considered equally valid.
- One found depends on the initial guess for β and β_0.
- The **finite** number of steps can be very large.
- If the data is non-separable, the algorithm will not converge.

Perceptron Learning Algorithm: Properties

Pros

- If the classes are linearly separable, the algorithm converges to a separating hyperplane in a finite number of steps.

Cons

- All separating hyperplanes are considered equally valid.
- One found depends on the initial guess for β and β_0.
- The **finite** number of steps can be very large.
- If the data is non-separable, the algorithm will not converge.

Perceptron Learning Algorithm: Properties

Pros

- If the classes are linearly separable, the algorithm converges to a separating hyperplane in a finite number of steps.

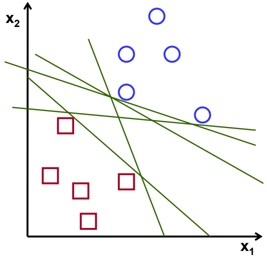
Cons

- All separating hyperplanes are considered equally valid.
- One found depends on the initial guess for β and β_0.
- The **finite** number of steps can be very large.
- If the data is non-separable, the algorithm will not converge.

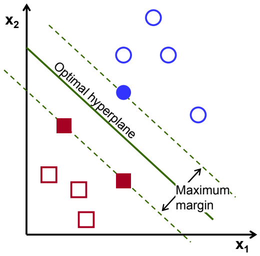
Optimal Separating Hyperplanes

Optimal Separating Hyperplane

- The **optimal separating hyperplane** separates the two classes and maximizes the distance to the closes point from either class [Vapnik 1996].
- This provides
  - a unique definition of the separating hyperplane



Which separating hyperplane?

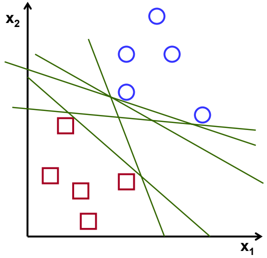


One which maximizes margin

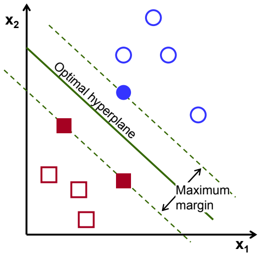
  - a decision boundary that generalizes well.

Optimal Separating Hyperplane

- The **optimal separating hyperplane** separates the two classes and maximizes the distance to the closes point from either class [Vapnik 1996].
- This provides
  - a unique definition of the separating hyperplane



Which separating hyperplane?



One which maximizes margin

  - a decision boundary that generalizes well.

Stating the optimization problem

- A first attempt

$$\boxed{\max_{\beta,\,\beta_0,\,\|\beta\|\,=\,1} M \ \ \textbf{subject to} \ \ y_i(\beta^t x_i + \beta_0) \geq M\|\beta\|, \ \ i = 1, \ldots, n}$$

- The conditions ensure all the training points are a signed distance M from the decision boundary defined by β and β_0.
- Want to find the largest such M and its associated β and β_0.

Stating the optimization problem

- Remove the constraint $\|\beta\| = 1$ by adjusting the constraints on the training data as follows:

$$\boxed{\max_{\beta,\,\beta_0} M \ \ \textbf{subject to}\ \ y_i(\beta^t x_i + \beta_0) \geq M\|\beta\|, \ \ i = 1, \ldots, n}$$

- For any β and β_0 fulfilling the above constraints then $\alpha\beta$ and $\alpha\beta_0$ with $\alpha > 0$ also fulfills the constraints.

- Therefore can arbitrarily set $\|\beta\| = 1/M$.

- Then the above optimization problem is equivalent to

$$\boxed{\min_{\beta,\,\beta_0} \frac{1}{2}\|\beta\|^2 \ \ \textbf{subject to}\ \ y_i(\beta^t x_i + \beta_0) \geq 1, \ \ i = 1, \ldots, n}$$

Stating the optimization problem

- Remove the constraint $\|\beta\| = 1$ by adjusting the constraints on the training data as follows:

$$\max_{\beta,\,\beta_0} M \ \ \textbf{subject to} \ \ y_i(\beta^t x_i + \beta_0) \geq M\|\beta\|, \ \ i = 1, \ldots, n$$

- For any β and β_0 fulfilling the above constraints then $\alpha\beta$ and $\alpha\beta_0$ with $\alpha > 0$ also fulfills the constraints.

- Therefore can arbitrarily set $\|\beta\| = 1/M$.

- Then the above optimization problem is equivalent to

$$\min_{\beta,\,\beta_0} \frac{1}{2}\|\beta\|^2 \ \ \textbf{subject to} \ \ y_i(\beta^t x_i + \beta_0) \geq 1, \ \ i = 1, \ldots, n$$

Stating the optimization problem

- Remove the constraint $\|\beta\| = 1$ by adjusting the constraints on the training data as follows:

$$\boxed{\max_{\beta,\,\beta_0} M \ \ \textbf{subject to} \ \ y_i(\beta^t x_i + \beta_0) \geq M\|\beta\|, \ \ i = 1, \ldots, n}$$

- For any β and β_0 fulfilling the above constraints then $\alpha\beta$ and $\alpha\beta_0$ with $\alpha > 0$ also fulfills the constraints.

- Therefore can arbitrarily set $\|\beta\| = 1/M$.

- Then the above optimization problem is equivalent to

$$\boxed{\min_{\beta,\,\beta_0} \frac{1}{2}\|\beta\|^2 \ \ \textbf{subject to} \ \ y_i(\beta^t x_i + \beta_0) \geq 1, \ \ i = 1, \ldots, n}$$

Stating the optimization problem

- Remove the constraint $\|\beta\| = 1$ by adjusting the constraints on the training data as follows:

$$\boxed{\max_{\beta,\,\beta_0} M \;\; \textbf{subject to} \;\; y_i(\beta^t x_i + \beta_0) \geq M\|\beta\|, \;\; i = 1, \ldots, n}$$

- For any β and β_0 fulfilling the above constraints then $\alpha\beta$ and $\alpha\beta_0$ with $\alpha > 0$ also fulfills the constraints.

- Therefore can arbitrarily set $\|\beta\| = 1/M$.

- Then the above optimization problem is equivalent to

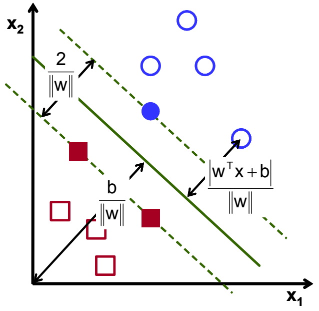
$$\boxed{\min_{\beta,\,\beta_0} \frac{1}{2}\|\beta\|^2 \;\; \textbf{subject to} \;\; y_i(\beta^t x_i + \beta_0) \geq 1, \;\; i = 1, \ldots, n}$$

Stating the optimization problem

- With this formulation of the problem

$$\boxed{\min_{\beta,\,\beta_0} \frac{1}{2}\|\beta\|^2 \ \ \textbf{subject to} \ \ y_i(\beta^t x_i + \beta_0) \geq 1, \ \ i = 1, \ldots, n}$$

- The margin has thickness $1/\|\beta\|$ as shown in figure (notation slightly different).



The solution to this constrained optimization problem

$$\boxed{\min_{\beta,\,\beta_0} \frac{1}{2}\|\beta\|^2 \ \textbf{subject to}\ \ y_i(\beta^t x_i + \beta_0) \geq 1, \ \ i = 1, \ldots, n}$$

- This is a **convex optimization problem** - quadratic objective function with linear inequality constraints.
- Its associated primal *Lagrangian* function is

$$\mathcal{L}_p(\beta, \beta_0, \alpha) = \frac{1}{2}\|\beta\|^2 + \sum_{i=1}^{n} \alpha_i y_i (1 - \beta^t x_i - \beta_0)$$

- β^* and β_0^* is a minimum point of the cost function stated at the top if...

The solution to this constrained optimization problem

$$\boxed{\min_{\beta,\,\beta_0} \frac{1}{2}\|\beta\|^2 \ \textbf{subject to} \ y_i(\beta^t x_i + \beta_0) \geq 1, \ \ i = 1, \ldots, n}$$

- This is a **convex optimization problem** - quadratic objective function with linear inequality constraints.
- Its associated primal *Lagrangian* function is

$$\mathcal{L}_p(\beta, \beta_0, \alpha) = \frac{1}{2}\|\beta\|^2 + \sum_{i=1}^{n} \alpha_i y_i (1 - \beta^t x_i - \beta_0)$$

- β^* and β_0^* is a minimum point of the cost function stated at the top if...

The solution to this constrained optimization problem

$$\boxed{\min_{\beta,\,\beta_0} \frac{1}{2}\|\beta\|^2 \ \ \textbf{subject to}\ \ y_i(\beta^t x_i + \beta_0) \geq 1, \ \ i = 1, \ldots, n}$$

- This is a **convex optimization problem** - quadratic objective function with linear inequality constraints.
- Its associated primal *Lagrangian* function is

$$\mathcal{L}_p(\beta, \beta_0, \alpha) = \frac{1}{2}\|\beta\|^2 + \sum_{i=1}^{n} \alpha_i y_i (1 - \beta^t x_i - \beta_0)$$

- β^* and β_0^* is a minimum point of the cost function stated at the top if...

The solution to this constrained optimization problem

$$\boxed{\min_{\beta,\,\beta_0} \frac{1}{2}\|\beta\|^2 \;\; \textbf{subject to} \;\; y_i(\beta^t x_i + \beta_0) \geq 1, \;\; i = 1, \ldots, n}$$

The **Karush-Kuhn-Tucker** conditions state that $\beta_1^* = (\beta_0^*, \beta^*)$ is a minimum of this cost function if $\exists$ a unique α^* s.t.

1. $\nabla_{\beta_1} \mathcal{L}_p(\beta_1^*, \alpha^*) = 0$
2. $\alpha_j^* \geq 0$ for $j = 1, \ldots, n$
3. $\alpha_j^*(1 - y_j(\beta_0^* + x_j^t \beta^*)) = 0$ for $j = 1, \ldots, n$
4. $(1 - y_j(\beta_0^* + x_j^t \beta^*)) \leq 0$ for $j = 1, \ldots, n$
5. Plus positive definite constraints on $\nabla_{\beta_1 \beta_1} \mathcal{L}_p(\beta_1^*, \alpha^*)$

Let's check what the KKT conditions imply

Active constraints and Inactive constraints:

Let $\mathcal{A}$ be the set of indices with $\alpha_j^* > 0$ then

$$\boxed{\mathcal{L}_p(\beta_1^*, \alpha^*) = \frac{1}{2}\|\beta^*\|^2 + \sum_{j\in\mathcal{A}} \alpha_j^* \left(1 - y_j(\beta_0^* + x_j^t \beta^*)\right).}$$

- Condition KKT 1, $\nabla_{\beta_1} \mathcal{L}_p(\beta_1^*, \alpha^*) = 0$, implies

$$\beta^* = \sum_{j\in\mathcal{A}} \alpha_j^* y_j x_j \quad \text{and} \quad 0 = \sum_{j\in\mathcal{A}} \alpha_j^* y_j$$

- Condition KKT 3, $\alpha_j^*(1 - y_j(\beta_0^* + x_j^t \beta^*)) = 0$, implies
  1. $y_j(\beta_0^* + x_j^t \beta^*) = 1$ for all $j \in \mathcal{A}$,
  2. if $y_i(\beta_0^* + x_i^t \beta^*) > 1$ then $\alpha_i = 0$ and $i \notin \mathcal{A}$
  3. $\mathcal{L}_p(\beta_1^*, \alpha^*) = .5\|\beta^*\|^2$.

Let's check what the KKT conditions imply

Active constraints and Inactive constraints:

Let $\mathcal{A}$ be the set of indices with $\alpha_j^* > 0$ then

$$\boxed{\mathcal{L}_p(\beta_1^*, \alpha^*) = \frac{1}{2}\|\beta^*\|^2 + \sum_{j \in \mathcal{A}} \alpha_j^* \left(1 - y_j(\beta_0^* + x_j^t \beta^*)\right).}$$

- Condition KKT 1, $\nabla_{\beta_1} \mathcal{L}_p(\beta_1^*, \alpha^*) = 0$, implies

$$\beta^* = \sum_{j \in \mathcal{A}} \alpha_j^* y_j x_j \quad \text{and} \quad 0 = \sum_{j \in \mathcal{A}} \alpha_j^* y_j$$

- Condition KKT 3, $\alpha_j^*(1 - y_j(\beta_0^* + x_j^t \beta^*)) = 0$, implies
  1. $y_j(\beta_0^* + x_j^t \beta^*) = 1$ for all $j \in \mathcal{A}$,
  2. if $y_i(\beta_0^* + x_i^t \beta^*) > 1$ then $\alpha_i = 0$ and $i \notin \mathcal{A}$
  3. $\mathcal{L}_p(\beta_1^*, \alpha^*) = .5\|\beta^*\|^2$.

To summarize

- As we have a convex optimization problem it has one local minimum.
- At this minimum β_1^* there exist a unique α^* s.t. β_1^* and α^* fulfill the KKT conditions.
- Let $\mathcal{A}$ be the set of indices with $\alpha_j^* > 0$ then

1. if $i \in \mathcal{A}$ then $y_i(\beta_0^* + x_i^t \beta^*) = 1$ and therefore x_i lies on the boundary of the margin.

   x_i is called a support vector.

2. And if $i \notin \mathcal{A}$ then $y_i(\beta_0^* + x_i^t \beta^*) > 1$ and x_i lies outside of the margin.

3. β^* is a linear combination of the support vectors

$$\boxed{\beta^* = \sum_{j \in \mathcal{A}} \alpha_j^* y_j x_j}$$

To summarize

- As we have a convex optimization problem it has one local minimum.
- At this minimum β_1^* there exist a unique α^* s.t. β_1^* and α^* fulfill the KKT conditions.
- Let $\mathcal{A}$ be the set of indices with $\alpha_j^* > 0$ then
  1. if $i \in \mathcal{A}$ then $y_i(\beta_0^* + x_i^t\beta^*) = 1$ and therefore x_i lies on the boundary of the margin.

     x_i is called a support vector.
  2. And if $i \notin \mathcal{A}$ then $y_i(\beta_0^* + x_i^t\beta^*) > 1$ and x_i lies outside of the margin.
  3. β^* is a linear combination of the support vectors

$$\boxed{\beta^* = \sum_{j \in \mathcal{A}} \alpha_j^* y_j x_j}$$

To summarize

- As we have a convex optimization problem it has one local minimum.
- At this minimum β_1^* there exist a unique α^* s.t. β_1^* and α^* fulfill the KKT conditions.
- Let $\mathcal{A}$ be the set of indices with $\alpha_j^* > 0$ then
  1. if $i \in \mathcal{A}$ then $y_i(\beta_0^* + x_i^t \beta^*) = 1$ and therefore x_i lies on the boundary of the margin.

     x_i is called a support vector.
  2. And if $i \notin \mathcal{A}$ then $y_i(\beta_0^* + x_i^t \beta^*) > 1$ and x_i lies outside of the margin.
  3. β^* is a linear combination of the support vectors

$$\boxed{\beta^* = \sum_{j \in \mathcal{A}} \alpha_j^* y_j x_j}$$

To summarize

- As we have a convex optimization problem it has one local minimum.
- At this minimum β_1^* there exist a unique α^* s.t. β_1^* and α^* fulfill the KKT conditions.
- Let $\mathcal{A}$ be the set of indices with $\alpha_j^* > 0$ then

  1. if $i \in \mathcal{A}$ then $y_i(\beta_0^* + x_i^t \beta^*) = 1$ and therefore x_i lies on the boundary of the margin.

     x_i is called a support vector.

  2. And if $i \notin \mathcal{A}$ then $y_i(\beta_0^* + x_i^t \beta^*) > 1$ and x_i lies outside of the margin.

  3. β^* is a linear combination of the support vectors

$$\boxed{\beta^* = \sum_{j \in \mathcal{A}} \alpha_j^* y_j x_j}$$

To summarize

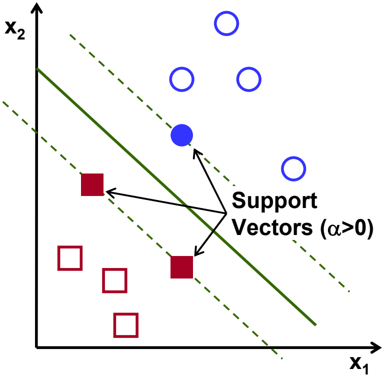
- As we have a convex optimization problem it has one local minimum.
- At this minimum β_1^* there exist a unique α^* s.t. β_1^* and α^* fulfill the KKT conditions.
- Let $\mathcal{A}$ be the set of indices with $\alpha_j^* > 0$ then

  1. if $i \in \mathcal{A}$ then $y_i(\beta_0^* + x_i^t\beta^*) = 1$ and therefore x_i lies on the boundary of the margin.

     x_i is called a support vector.
  2. And if $i \notin \mathcal{A}$ then $y_i(\beta_0^* + x_i^t\beta^*) > 1$ and x_i lies outside of the margin.
  3. β^* is a linear combination of the support vectors

$$\boxed{\beta^* = \sum_{j \in \mathcal{A}} \alpha_j^* y_j x_j}$$

To summarize



$$\boxed{\beta^* = \sum_{j \in \mathcal{A}} \alpha_j^* y_j x_j}$$

How do I calculate α^*?

- You have seen that the optimal solution is a weighted sum of the support vectors.
- But how can we calculate these weights?
- Most common approach is to solve the Dual Lagrange problem as opposed to the Primal Lagrange problem. (The solutions to these problems are the same because of the original quadratic cost function and linear inequality constraints.)
- This Dual problem is an easier constrained optimization and is also convex. It has the form

$$\max_{\alpha} \left\{ \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{k=1}^{n} \alpha_i \alpha_k y_i y_k x_i^t x_k \right\} \quad \textbf{subject to} \quad \alpha_i \geq 0 \ \forall i$$

How do I calculate α^*?

- You have seen that the optimal solution is a weighted sum of the support vectors.
- But how can we calculate these weights?
- Most common approach is to solve the Dual Lagrange problem as opposed to the Primal Lagrange problem. (The solutions to these problems are the same because of the original quadratic cost function and linear inequality constraints.)
- This Dual problem is an easier constrained optimization and is also convex. It has the form

$$\max_{\alpha} \left\{ \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{k=1}^{n} \alpha_i \alpha_k y_i y_k x_i^t x_k \right\} \quad \textbf{subject to} \quad \alpha_i \geq 0 \ \forall i$$

How do I calculate α^*?

- You have seen that the optimal solution is a weighted sum of the support vectors.
- But how can we calculate these weights?
- Most common approach is to solve the Dual Lagrange problem as opposed to the Primal Lagrange problem. (The solutions to these problems are the same because of the original quadratic cost function and linear inequality constraints.)
- This Dual problem is an easier constrained optimization and is also convex. It has the form

$$\max_{\alpha} \left\{ \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{k=1}^{n} \alpha_i \alpha_k y_i y_k x_i^t x_k \right\} \quad \textbf{subject to} \quad \alpha_i \geq 0 \ \ \forall i$$

How do I calculate α^*?

- You have seen that the optimal solution is a weighted sum of the support vectors.
- But how can we calculate these weights?
- Most common approach is to solve the Dual Lagrange problem as opposed to the Primal Lagrange problem. (The solutions to these problems are the same because of the original quadratic cost function and linear inequality constraints.)
- This Dual problem is an easier constrained optimization and is also convex. It has the form

$$\boxed{\max_{\alpha} \left\{ \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{k=1}^{n} \alpha_i \alpha_k y_i y_k x_i^t x_k \right\} \ \textbf{subject to} \ \ \alpha_i \geq 0 \ \ \forall i}$$