Linear and Non-linear Classifiers

Adway Mitra
MLFA AI42001
Center for Artificial Intelligence
Indian Institute of Technology Kharagpur

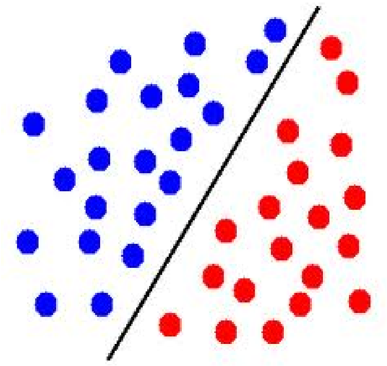
August 30, 2019

Linear Classifiers

- Linear structures $y = w^T x + b$
  1. 2D space: line
  2. 3D space: plane
  3. higher dimensions: hyperplane!
- Any hyperplane w divides the space into two *half-spaces*
- Positive halfspace: $\{x : w^T x + b > 0\}$, Negative halfspace: $\{x : w^T x + b < 0\}$
- Hyperplane classifier: $y = sign(w^T x + b)$

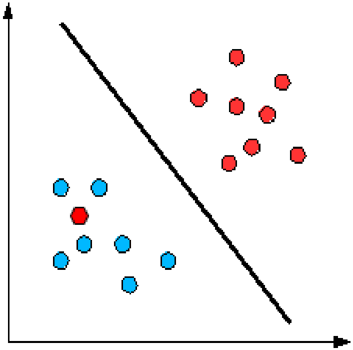
Linear Separability

- Does there exist any line/hyperplane that separate the classes?
- If so, the data is linearly separable!



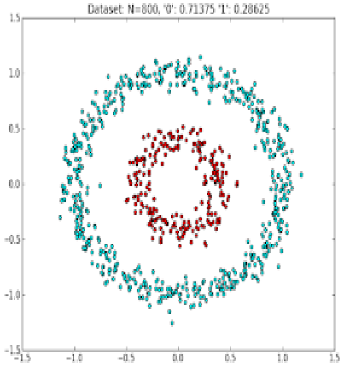
Linear Separability

- Does there exist a line/hyperplane that separate the classes?
- If so, with some exceptional points, the data is *almost* linearly separable!



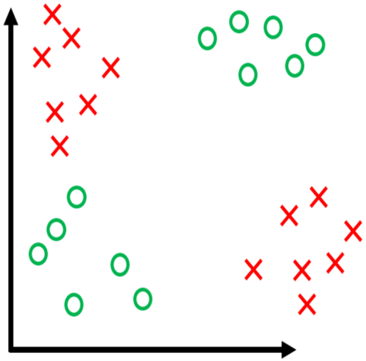
Non-Linear Sepaeability

- Does there exist a non-linear structure that separate the classes?
- If yes, the data is non- linearly separable!



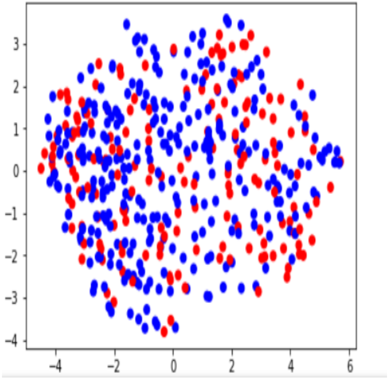
Multi-layer Separability

- Do there exist multiple linear or non-linear structures that separate the classes?
- If yes, the data is multi-layer separable!



Inseparability

- Does there exist any linear or non-linear structure that separate the classes?
- If no, the data is inseparable!



Classification Strategies

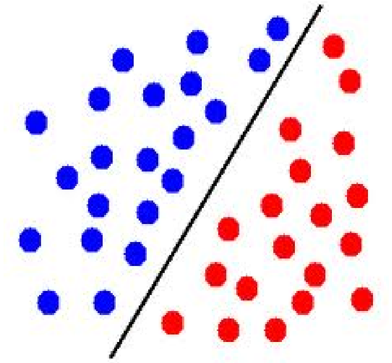
- **If data is linearly separable**:
  - Find *any* separating hyperplane(s)
  - Find *the best* separating hyperplane(s)
- **If data is almost linearly separable**: same as above
- **If data is non-linearly separable**:
  - Convert the data to linearly separable form (!!) and then use linear classifier
  - Use non-linear classifier
- **If data is multi-layer separable**: multi-layer version of above
- **If data is inseparable**: need to find local structures (KNN, Bayesian Classifier etc)

Classification Strategies

- **If data is linearly separable**:
  - Find *any* separating hyperplane(s) - Perceptron
  - Find *the best* separating hyperplane(s)
- **If data is almost linearly separable**: same as above
- **If data is non-linearly separable**:
  - Convert the data to linearly separable form (!!) and then use linear classifier
  - Use non-linear classifier
- **If data is multi-layer separable**: multi-layer version of above
- **If data is inseparable**: need to find local structures (KNN, Bayesian Classifier etc)

Perceptron

- Aim: to find *any* separating hyperplane for binary classifiction
- Labels: +1 or -1
- Prediction: $y = sign(w^T x + b)$



Perceptron

- Aim: to find *any* separating hyperplane for binary classifiction
- Prediction: $y = sign(w^T x + b)$
- Perceptron Algorithm input: $\{X_i, Y_i\}_{i=1}^N$ (training set)
- Perceptron Algorithm output: (w, b)

Perceptron Algorithm

Initialize $w = w_0$, $b = 0$

Repeat till stopping criteria satisfied

- For $i \ in\{1, N\}$ (each training sample)
  - if $y_i(w^T x_i + b) < 0$ (misclassification)
    - $w = w + x_i y_i$ (update w)
    - $b = b + y_i$ (update b)

Possible choices for stopping criteria:

1. All examples correctly classified
2. A fixed number of iterations completed
3. w does not change much on updation

Why Perceptron Algorithm works?

- When any example x_i is misclassified: $y_i(w_{old}^T x_i + b_{old}) < 0$
- Update: $w_{new} = w_{old} + x_i y_i$, $b_{new} = b_{old} + y_i$
- $y_i(w_{new}^T x_i + b_{new}) = y_i(w_{old}^T x_i + b_{old}) + x^T x + 1$
- Negative quantity $y_i(w_{old}^T x_i + b_{old})$ boosted by positive quantity $x^T x + 1$!
- $y_i(w_{new}^T x_i + b_{new})$ either positive or closer to positive!
- So, we make some improvement at every misclassification!

Hyperplanes, Margins, and Perceptron

- Orthogonal distance of any point x from a hyperplane w: $\gamma(w, b, x) = \frac{|w^T x + b|}{||w||_2}$
- **Margin** of a dataset $D = \{x_i\}_{i=1}^N$ from w: minimum orthogonal distance of its points from w
- $\gamma(w, b, D) = min_{i=1}^N \gamma(w, b, x_i)$
- **Block and Novikoff Theorem**: if dataset is linearly separable with margin γ, then perceptron converges after $\frac{R^2}{\gamma^2}$ updates where $R = max_{i=1}^N ||x_i||_2$

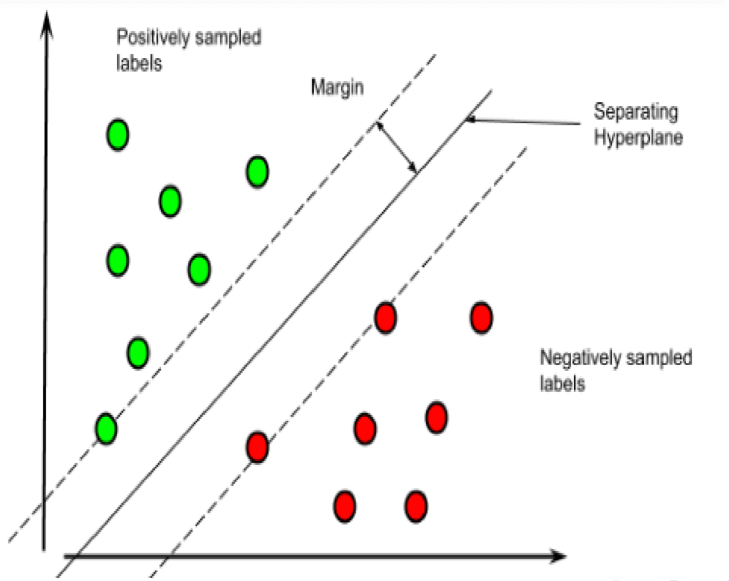
Classification Strategies

- **If data is linearly separable**:
  - Find *any* separating hyperplane(s) - Perceptron
  - Find *the best* separating hyperplane(s) - Max-Margin Classifier
- **If data is almost linearly separable**: same as above
- **If data is non-linearly separable**:
  - Convert the data to linearly separable form (!!) and then use linear classifier
  - Use non-linear classifier
- **If data is multi-layer separable**: multi-layer version of above
- **If data is inseparable**: need to find local structures (KNN, Bayesian Classifier etc)

Marginal Classifier

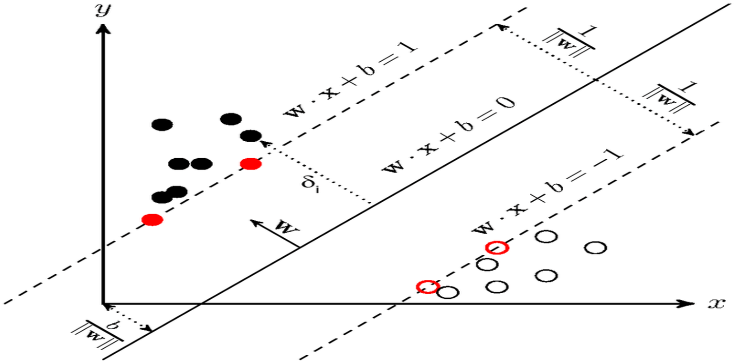
- Multiple hyperplanes can separate linearly separable data (by definition)
- The classes have margins $\gamma(w, b, D^{+1})$ and $\gamma(w, b, D^{-1})$ from any hyperplane (w, b)
- Total margin of a hyperplane $\gamma(w, b, D^{+1}) + \gamma(w, b, D^{-1})$
- *Marginal classifiers* have margin 0 from at least one of the classes
- Marginal classifiers usually closer to one class, leaves little room for error

Marginal Classifier



Max-margin Classifier

- Let (w, b^{+1}) and (w, b^{-1}) be two marginal classifiers, parallel to each other
- By linear transformation of data, they become $(w, b+1)$ and $(w, b-1)$
- Consider the central hyperplane (w, b): total margin $= \frac{2}{||w||_2}$



y
x
$\mathbf{w} \cdot \mathbf{x} + b = 1$
$\mathbf{w} \cdot \mathbf{x} + b = 0$
$\mathbf{w} \cdot \mathbf{x} + b = -1$
$\frac{1}{\|\mathbf{w}\|}$
$\frac{1}{\|\mathbf{w}\|}$
δ_i
$\mathbf{w}$
$\frac{b}{\|\mathbf{w}\|}$

Max-margin Classifier

- Central hyperplane: most robust classifier, enough room for error
- Max-margin: choose (w, b) such that the margin $\frac{2}{||w||_2}$ is **maximized**
- Constraints imposed on w by the classification

$$\hat{w}, \hat{b} = argmin_{w,b} \frac{1}{2} ||w||_2^2$$
$$s.t. y_i(w^T x_i + b) \geq 1; i \in \{1, N\} \quad (1)$$

Max-margin Classsifier

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2}||w||_2^2 + \sum_{i=1}^{N} \alpha_i(1 - y_i(w^T x_i + b)) \tag{2}$$

- Optimization problem with additional variables $\{\alpha_i\}_{i=1}^N$ (Lagrange Multipliers)
- Differentiate the objective function $\mathcal{L}$ w.r.t all variables and equate to 0

Max-margin Classsifier - Alternate View

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2}||w||_2^2 + \sum_{i=1}^{N} \alpha_i(1 - y_i(w^T x_i + b)) \tag{3}$$

- $\sum_{i=1}^{N} \alpha_i(1 - y_i(w^T x_i + b))$: *empirical risk*, fitting the data
- $\frac{1}{2}||w||_2^2$: *structural risk*, regularizer
- Similar to ridge regression?

Max-margin Classifier

- $w = \sum_{i=1}^{N} \alpha_i y_i x_i$, $\sum_{i=1}^{N} \alpha_i y_i = 0$
- *Dual problem*: substitute w and b in $\mathcal{L}$ with α
- This problem can be solved by *Quadratic Programming* approach

$$\mathcal{L}(\alpha) = \sum_{i=1}^{N} \alpha_i - \sum_{i=1,j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j$$
$$\textit{such that} \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{4}$$

Support Vector Machine

- $\hat{w} = \sum_{i=1}^{N} \alpha_i y_i x_i$,
- $\hat{b} = -\frac{1}{2}(min_{i:y_i=+1}\hat{w}^T x_i + max_{i:y_i=-1}\hat{w}^T x_i)$
- For most points, $\alpha_i = 0$
- w is determined by the remaining points called *Support Vectors*
- Classification model is called *Support Vector Machine*
- Prediction on test points: $\hat{y_{test}} = sign(\sum_{i=1}^{N} \alpha_i y_i x_i^T x_{test} + \hat{b})$
- note: dot product of x_{test} with all support vectors

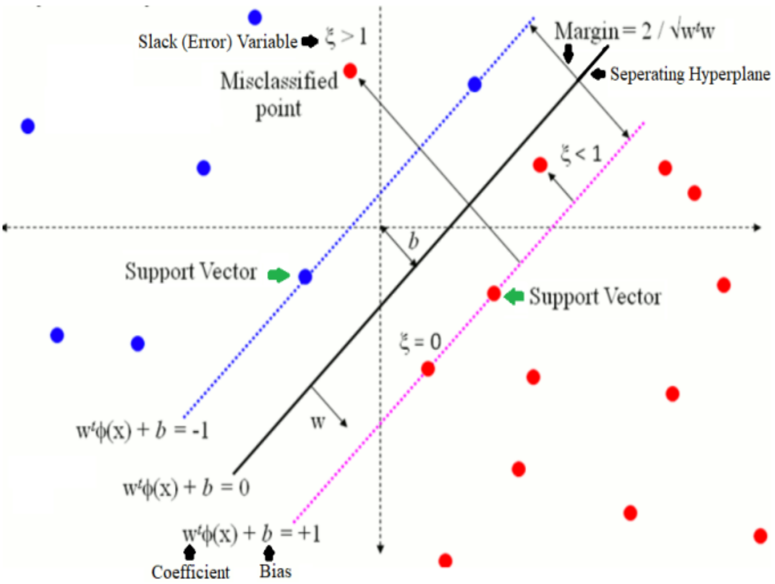
Classification Strategies

- **If data is linearly separable**:
  - Find *any* separating hyperplane(s) - Perceptron
  - Find *the best* separating hyperplane(s) - Max-Margin Classifier
- **If data is almost linearly separable**: Soft-margin Support Vector Machine (SVM)
- **If data is non-linearly separable**:
  - Convert the data to linearly separable form (!!) and then use linear classifier
  - Use non-linear classifier
- **If data is multi-layer separable**: multi-layer version of above
- **If data is inseparable**: need to find local structures (KNN, Bayesian Classifier etc)

Soft-margin Support Vector Machine (SVM)

- Linearly separable data: $y_i(w^T x_i + b) \geq 1; i \in \{1, N\}$
- Now we have a few points which do not satisfy the above
- $y_i(w^T x_i + b) \geq 1 - \xi_i; i \in \{1, N\}$
- ξ are *slack variables*,
- $\xi_i = 0$ for those i beyond respective marginal classifiers (i.e. $y_i(w^T x_i + b) \geq 1$)
- For points between the marginal classifier and optimal classifier: $y_i(w^T x_i + b) \geq 0$, i.e. $0 < \xi_i < 1$
- For points beyond optimal classifier: $y_i(w^T x_i + b) < 0$, i.e. $\xi_i > 1$

Support Vector Machine (SVM)



Soft-margin Support Vector Machine (SVM)

$$(\hat{w}, \hat{b}, \xi) = argmin_{w,b} \frac{1}{2}||w||_2^2 + C \sum_{i=1}^{N} xi_i \; such \; that$$
$$y_i(w^T x_i + b) \geq 1 - \xi_i; i \in \{1, N\}$$
$$\xi_i \geq 0; i \in \{1, N\} \qquad (5)$$

The new objective function:

$$\mathcal{L}(w, b, \xi, \alpha, \beta) = \frac{1}{2}||w||_2^2 + \sum_{i=1}^{N} \alpha_i(1 - \xi_i - y_i(w^T x_i + b)) + \sum_{i=1}^{N}(C - \beta_i)$$

Approach: Once again solve $\frac{\partial \mathcal{L}}{\partial w} = 0$, $\frac{\partial \mathcal{L}}{\partial b} = 0$, $\frac{\partial \mathcal{L}}{\partial xi_i} = 0$

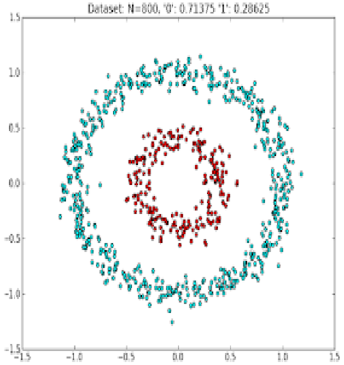
Soft-margin Support Vector Machine (SVM)

- Once again, $w = \sum_{i=1}^{N} \alpha_i y_i x_i$
- α obtained by solving dual problem by QP, α sparse
- Three types of support vectors
  1. Lying on the margin classifiers ($\xi_i = 0$)
  2. Lying between the margin classifier and optimal classifier ($0 < \xi_i < 1$)
  3. Lying beyond the optimal classifier $\xi_i > 1$

Classification Strategies

- **If data is linearly separable**:
  - Find *any* separating hyperplane(s) - Perceptron
  - Find *the best* separating hyperplane(s) - Max-Margin Classifier
- **If data is almost linearly separable**: Support Vector Machine (SVM)
- **If data is non-linearly separable**:
  - Convert the data to linearly separable form (!!) and then use linear classifier: Kernelized SVM
  - Use non-linear classifier
- **If data is multi-layer separable**: multi-layer version of above
- **If data is inseparable**: need to find local structures (KNN, Bayesian Classifier etc)

Transforming Data to L-S space



Separating structure:

$$\begin{aligned} y &= sign((x - x_0)^T(x - x_0) - R) \\ &= sign(x^T x - 2x_0^T x + x_0^T x_0 - R) \qquad (7) \end{aligned}$$

Transforming Data to L-S space

- Define $\Phi(x) = [x^T x; -2x; 1]$
- Define $w = [1; x_0; x_0^T x_0 - R]$
- Separating structure changes to $y = sign(w^T \Phi(x))$
- Clearly in the space of $\Phi(x)$, this is a linear classifier!
- We can now apply SVM on the transformed D+2-dim space!
- Training data is now $\{\Phi(x_i), y_i\}_{i=1}^{N}$

Kernel Trick

- SVM Dual formulation on transformed space:

$$\mathcal{L}(\alpha) = \sum_{i=1}^{N} \alpha_i - \sum_{i=1,j=1}^{N} \alpha_i \alpha_j y_i y_j \Phi(x_i)^T \Phi(x_j)$$
$$\text{such that} \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{8}$$

  Prediction: $\hat{y}_{test} = sign(\sum_{i=1}^{N} \alpha_i y_i \Phi(x_i)^T \Phi(x_{test}) + b)$
- Identifying such a Φ not always easy!
- Kernel Trick But we need not find Φ!

Kernel Trick

$$\mathcal{L}(\alpha) = \sum_{i=1}^{N} \alpha_i - \sum_{i=1,j=1}^{N} \alpha_i \alpha_j y_i y_j \Phi(x_i)^T \Phi(x_j)$$
$$\textit{such that } \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{9}$$

Prediction: $\hat{y}_{test} = sign(\sum_{i=1}^{N} \alpha_i y_i \Phi(x_i)^T \Phi(x_{test}) + b)$

- SVM Dual formulation on transformed space:
- Observe: we only need $\Phi(x_i)^T \Phi(x_j)$ (dot products)
- Kernel function: $K(x_i, x_j) = \Phi(x_i)^T \Phi(x_j)$
- Kernel trick: define K instead of defining Φ!

Kernel Functions

$$\mathcal{L}(\alpha) = \sum_{i=1}^{N} \alpha_i - \sum_{i=1,j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$
$$\textit{such that} \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{10}$$

Prediction: $\hat{y}_{test} = sign(\sum_{i=1}^{N} \alpha_i y_i K(x_i, x_{test}) + b)$

- Some functions can be used as K, i.e. at least one Φ exist for them
- Mercer's Condition for Kernel Functions

Kernel Functions

Mercer's Condition: For every function f such that $\int f(x)^2 dx < \infty$, the function K should satisfy $\int\int K(x,y)f(x)f(y)dxdy \geq 0$
Note: if $K1$ and $K2$ are valid kernel functions, then $\alpha_1 K_1 + \alpha_2 K_2$ is also a valid kernel function if $K_1, K_2 \geq 0$
Some common Kernel functions:

- Linear Kernel (trivial): $K(x,y) = x^T y$
- Polynomial Kernel: $K(x,y) = (1 + x^T y)^d$
- Radial Basis or Gaussian Kernel: $K(x,y) = exp(-\gamma ||x - z||_2^2)$

Classification Strategies

- **If data is linearly separable**:
  - Find *any* separating hyperplane(s) - Perceptron
  - Find *the best* separating hyperplane(s) - Max-Margin Classifier
- **If data is almost linearly separable**: Support Vector Machine (SVM)
- **If data is non-linearly separable**:
  - Convert the data to linearly separable form (!!) and then use linear classifier: Kernelized SVM
  - Use non-linear classifier Neural Network
- **If data is multi-layer separable**: multi-layer version of above
- **If data is inseparable**: need to find local structures (KNN, Bayesian Classifier etc)

Classification Strategies

- **If data is linearly separable**:
  - Find *any* separating hyperplane(s) - Perceptron
  - Find *the best* separating hyperplane(s) - Max-Margin Classifier
- **If data is almost linearly separable**: Support Vector Machine (SVM)
- **If data is non-linearly separable**:
  - Convert the data to linearly separable form (!!) and then use linear classifier: Kernelized SVM
  - Use non-linear classifier Neural Network
- **If data is multi-layer separable**: Deep Neural Networks
- **If data is inseparable**: need to find local structures (KNN, Bayesian Classifier etc)