CS-453 Machine Learning

Nazar Khan

Department of Computer Science
University of the Punjab

Linear Classification

Classification

- In the previous topic, regression, the goal was to predict *continuous* target variable(s) t given input variables vector x.
- In *classification*, the goal is to predict *discrete* target variable(s) t given input variables vector x.
- Input space is divided into *decision regions*.
- Boundaries between regions are called *decision boundaries/surfaces*.
- Training corresponds to finding optimal decision boundaries given training data $\{(\mathrm{x}_1, t_1), \ldots, (\mathrm{x}_N, t_N)\}$.

Classification

- Assign $\mathbf{x}$ to 1-of-K discrete classes C_k.
- Most commonly, the classes are distinct. That is, $\mathbf{x}$ is assigned to one and only one class.
- Convenient coding schemes for targets t are
  - $0/1$ coding for binary classification.
  - 1-of-K coding for multi-class classification. Example, for $\mathbf{x}$ belonging to class 3, the $K \times 1$ target vector will be coded as $\mathbf{t} = (0, 0, 1, 0, \dots, 0)^T$.

Linear Classification

- Like regression, the simplest classification model is *linear classification*.
  - This means that the decision surfaces are linear functions of x, for example $y(\mathrm{x}, \mathrm{w}) = \mathrm{w}^T\mathrm{x} + w_0 = 0$.
  - That is, a linear decision surface is a $D-1$ dimensional hyperplane in D-dimensional space.
- Data in which classes can be *separated exactly* by *linear decision surfaces* is called *linearly separable*.

Linear Classification

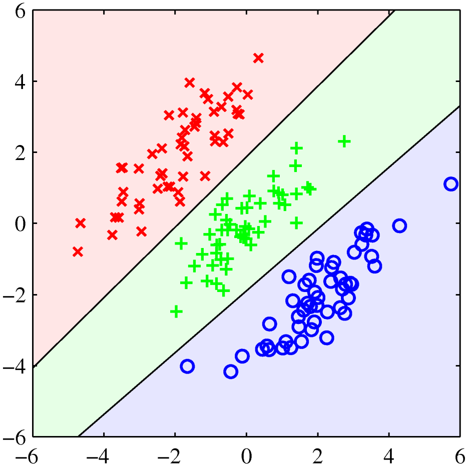


Figure: Linearly separable data and corresponding linear decision boundaries.

Linear Classification
Generalized Linear Model

- The simplest linear regression model computes continuous outputs $y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$.
- By passing these continuous outputs through a non-linear function $f(\cdot)$, we can obtain discrete class labels.

$$y(\mathbf{x}) = f(\mathbf{w}^T\mathbf{x} + w_0)$$

- This is known as a *generalised linear model* and $f(\cdot)$ is known as the *activation function*.

Linear Classification
Generalized Linear Model

- Decision surfaces correspond to all inputs x where $y(\mathrm{x}) = \mathrm{const}$. This is equivalent to the condition $\mathrm{w}^T\mathrm{x} + w_0 = \mathrm{const}$.
- Therefore, decision surfaces are linear functions of the input x, even if $f(\cdot)$ is non-linear.
- As before, we can replace x by a non-linear transformation $\phi(\mathrm{x})$ and learn non-linear boundaries in x-space by learning linear boundaries in ϕ-space.

Linear Discriminant Functions

Two class case

- The simplest linear discriminant function is given by $y(\mathrm{x}) = \mathrm{w}^T\mathrm{x} + w_0$ where w is called the *weight vector* and w_0 is called the *bias*.
- Classification is performed via the non-linear step

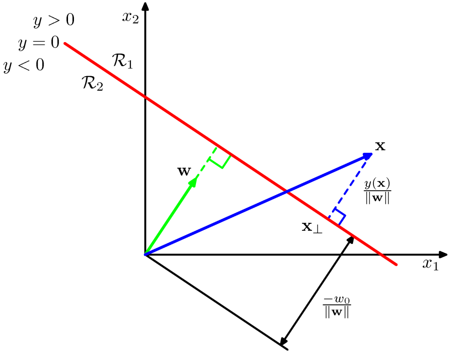
$$\mathrm{class}(\mathrm{x}) = \begin{cases} \mathcal{C}_1 & \text{if } y(\mathrm{x}) \geq 0 \\ \mathcal{C}_2 & \text{if } y(\mathrm{x}) < 0 \end{cases}$$

- We can view $-w_0$ as a *threshold*.

Linear Discriminant Functions

Two class case

- Weight vector w is always orthogonal to the decision surface.



- Proof: For *any* two points x_A and x_B on the surface, $y(\mathrm{x}_A) = y(\mathrm{x}_B) = 0 \Rightarrow \mathrm{w}^T(\mathrm{x}_A - \mathrm{x}_B) = 0$. Since vector $\mathrm{x}_A - \mathrm{x}_B$ is along the surface, w must be orthogonal.

Linear Discriminant Functions
Two class case

- Normal distance of any point x from decision boundary can be computed as $d = \frac{y(\mathrm{x})}{||\mathrm{w}||}$.
  - Proof:

$$
\begin{aligned}
\mathrm{x} &= \mathrm{x}_\perp + d\frac{\mathrm{w}}{||\mathrm{w}||} \\
\Rightarrow \underbrace{\mathrm{w}^T\mathrm{x} + w_0}_{y(\mathrm{x})} &= \underbrace{\mathrm{w}^T\mathrm{x}_\perp + w_0}_{y(\mathrm{x}_\perp)=0} + d\underbrace{\mathrm{w}^T\frac{\mathrm{w}}{||\mathrm{w}||}}_{||\mathrm{w}||} \\
\Rightarrow d &= \frac{y(\mathrm{x})}{||\mathrm{w}||}
\end{aligned}
$$

- Normal distance to boundary from origin $(\mathrm{x} = 0)$ is $\frac{w_0}{||\mathrm{w}||}$.

Linear Discriminant Functions

- For notational convenience, bias can be included as a component of the weight vector via

$$\tilde{\mathbf{w}} = (w_0, \mathbf{w})$$
$$\tilde{\mathbf{x}} = (1, \mathbf{x})$$
$$y(\mathbf{x}) = \tilde{\mathbf{w}}^T \tilde{\mathbf{x}}$$

Linear Discriminant Functions
Multiclass case

- For K class classification with $K > 2$, we have 3 options
  1. Learn $K - 1$ *one-vs-rest* binary classifiers.
  2. Learn $K(K-1)/2$ *one-vs-one* binary classifiers for every possible pair of classes. Each point can be classified based on majority vote among the discriminant functions.
  3. Learn K discriminant functions $y_1, \ldots, y_K$ and then $\text{class}(\mathrm{x}) = \arg\max_k y_k(\mathrm{x})$.
- Options 1 and 2 lead to ambiguous classification regions.

Linear Discriminant Functions

Multiclass Ambiguity

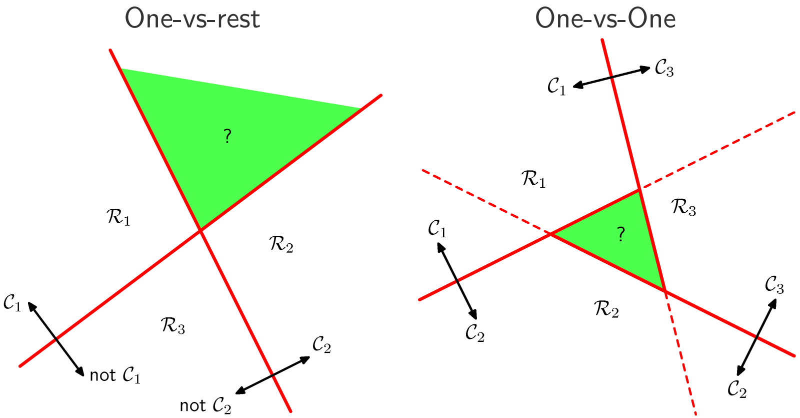


Figure: Ambiguity of multiclass classification using two-class linear discriminant functions.

Linear Discriminant Functions
Multiclass case

- We can write the K-class discriminant function as

$$y(x) = \tilde{W}^T \tilde{x}$$

- For learning, we can write the error function as

$$\begin{aligned} E(\tilde{W}) &= \frac{1}{2} \sum_{n=1}^{N} ||y(x_n) - t_n||^2 \\ &= \frac{1}{2} \sum_{n=1}^{N} (\tilde{W}^T \tilde{x}_n - t_n)^T (\tilde{W}^T \tilde{x}_n - t_n) \end{aligned}$$

Linear Discriminant Functions
Least Squares Solution

- Optimal discriminant function parameters $\tilde{\mathrm{W}}^*$ that minimize the SSE $E(\tilde{\mathrm{W}})$ are known as the *least-squares-solution*.
- Can be computed as
$$\tilde{\mathrm{W}}^* = \tilde{\mathrm{X}}^\dagger \mathrm{T}$$
where $\tilde{\mathrm{X}}^\dagger$ is the pseudo-inverse of the design matrix $\tilde{\mathrm{X}}$ and T is the matrix of target vectors.
- As before, we can also work in ϕ-space where we will use the corresponding matrix $\tilde{\mathbf{\Phi}}$ as the design matrix.

Linear Discriminant Functions
Least Squares Solution

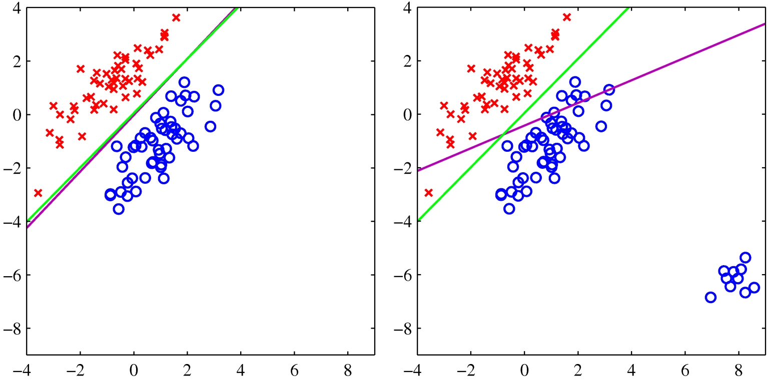


Figure: Least squares solution is sensitive to outliers.

Fisher's Linear Discriminant

Two class case

- Project all data onto a single vector w.
- Classify by thresholding projected coefficents.
- Optimal vector is one which
  - maximises between-class distance, and
  - minimises within-class distance.

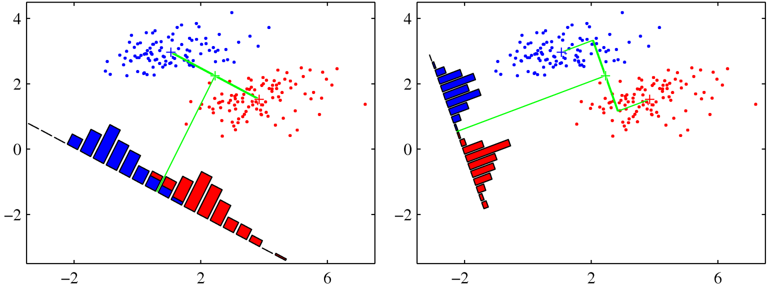


Figure: Fisher's linear discriminant. Classify by thresholding projections onto a vector w that maximises inter-class distance and minimises intra-class distances.

Fisher's Linear Discriminant
Two class case

- Let $\mathrm{m}_k = \frac{\sum_{n\in\mathcal{C}_k} \mathrm{x}_n}{N_k}$ be the mean vector of points belonging to class $\mathcal{C}_k$.
- Projection of this mean is then $m_k = \mathrm{w}^T \mathrm{m}_k$.
- Variance around projected mean can be written as $s_k^2 = \sum_{n\in\mathcal{C}_k} (\mathrm{w}^T\mathrm{x}_n - \mathrm{w}^T\mathrm{m}_k)^2$.
- Suitability of any projection direction w can then be written as

$$\begin{aligned} J(\mathrm{w}) &= \frac{\text{Inter-class variance}}{\text{Intra-class variance}} \\ &= \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} \\ &= \frac{(\mathrm{w}^T\mathrm{m}_2 - \mathrm{w}^T\mathrm{m}_1)^2}{\sum_{n\in\mathcal{C}_1}(\mathrm{w}^T\mathrm{x}_n - \mathrm{w}^T\mathrm{m}_1)^2 + \sum_{n\in\mathcal{C}_2}(\mathrm{w}^T\mathrm{x}_n - \mathrm{w}^T\mathrm{m}_2)^2} \end{aligned}$$

Fisher's Linear Discriminant

Two class case

$$
\begin{aligned}
J(\mathrm{w}) &= \frac{(\mathrm{w}^T(\mathrm{m}_2 - \mathrm{m}_1))(\mathrm{w}^T(\mathrm{m}_2 - \mathrm{m}_1))^T}{\sum_{k=1}^{2} \sum_{n \in \mathcal{C}_k} (\mathrm{w}^T(\mathrm{x}_n - \mathrm{m}_k))^2} \\
&= \frac{\mathrm{w}^T (\mathrm{m}_2 - \mathrm{m}_1)(\mathrm{m}_2 - \mathrm{m}_1)^T \mathrm{w}}{\mathrm{w}^T \left(\sum_{k=1}^{2} \sum_{n \in \mathcal{C}_k} (\mathrm{x}_n - \mathrm{m}_k)(\mathrm{x}_n - \mathrm{m}_k)^T \right) \mathrm{w}} \\
&= \frac{\mathrm{w}^T \mathrm{S}_B \mathrm{w}}{\mathrm{w}^T \mathrm{S}_W \mathrm{w}} \qquad (\mathrm{S}_B \text{ and } \mathrm{S}_W \text{ are symmetric due to outer-products})
\end{aligned}
$$

$$
\begin{aligned}
\nabla_{\mathrm{w}} E(\mathrm{w}) &= \frac{\mathrm{w}^T \mathrm{S}_W \mathrm{w} \nabla_{\mathrm{w}} (\mathrm{w}^T \mathrm{S}_B \mathrm{w}) - \mathrm{w}^T \mathrm{S}_B \mathrm{w} \nabla_{\mathrm{w}} (\mathrm{w}^T \mathrm{S}_W \mathrm{w})}{(\mathrm{w}^T \mathrm{S}_W \mathrm{w})^2} \quad (\because \text{ quotient rule}) \\
&= \frac{\mathrm{w}^T \mathrm{S}_B \mathrm{w} (2 \mathrm{S}_W \mathrm{w}) - \mathrm{w}^T \mathrm{S}_W \mathrm{w} (2 \mathrm{S}_B \mathrm{w})}{(\mathrm{w}^T \mathrm{S}_W \mathrm{w})^2} \quad \left(\because \nabla_{\mathbf{v}} (\mathrm{v}^T \mathrm{M} \mathrm{v}) = (\mathrm{M} + \mathrm{M}^T)\mathrm{v}\right)
\end{aligned}
$$

Fisher's Linear Discriminant

Two class case

- Objective J can be maximized by equating gradient to the 0 vector

$$\mathbf{w}^T\mathbf{S}_B\mathbf{w}(\mathbf{S}_W\mathbf{w}) = \mathbf{w}^T\mathbf{S}_W\mathbf{w}(\mathbf{S}_B\mathbf{w})$$

- Since we only care about the direction of projection, we can drop the scalar factors to get

$$\begin{aligned}
\mathbf{S}_W\mathbf{w} &= \mathbf{S}_B\mathbf{w} \\
\mathbf{S}_W\mathbf{w} &= (\mathbf{m}_2 - \mathbf{m}_1)\underbrace{(\mathbf{m}_2 - \mathbf{m}_1)^T\mathbf{w}}_{\text{scalar}} \\
\mathbf{S}_W\mathbf{w} &\propto (\mathbf{m}_2 - \mathbf{m}_1) \\
\mathbf{w} &\propto \mathbf{S}_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1)
\end{aligned}$$

Perceptron Algorithm
Two-class Classification

- Target t_n is taken to be either $+1$ or -1.
- A perceptron classifies its input via the non-linear step function

$$y(\phi) = \begin{cases} 1 & \text{if } \mathrm{w}^T\phi_n \geq 0 \\ -1 & \text{if } \mathrm{w}^T\phi_n < 0 \end{cases}$$

- Extremely simplified model of biological neuron.
- *Perceptron criterion*: $\mathrm{w}^T\phi_n t_n > 0$ for correctly classified point.
- Error can be defined on the set $\mathcal{M}(\mathrm{w})$ of misclassified points.

$$E(\mathrm{w}) = \sum_{n \in \mathcal{M}(w)} -\mathrm{w}^T\phi_n t_n$$

- Optimal w can be learned via gradient descent.
- For linearly separable data, perceptron learning is guaranteed to find the decision boundary in finite iterations.

Gradient Descent

- Gradient is the direction (in input space) of maximum rate of increase of a function.
- To minimize, move in negative gradient direction.

$$\mathrm{w}^{\text{new}} = \mathrm{w}^{\text{old}} - \eta \nabla_{\mathrm{w} E(\mathrm{w})}$$

- Also known as *gradient descent*.
- Local versus global minima.
- Learning rate η should be decayed to *avoid osscillation* and to *converge* to a local minimum.
- Different types of gradient descent:
  - Batch ($\mathrm{w}^{\text{new}} = \mathrm{w}^{\text{old}} - \eta \nabla_{\mathrm{w}} E$)
  - Sequential ($\mathrm{w}^{\text{new}} = \mathrm{w}^{\text{old}} - \eta \nabla_{\mathrm{w}} E_n$)
  - Stochastic (same as sequential but n is chosen randomly).
  - Mini-batches ($\mathrm{w}^{\text{new}} = \mathrm{w}^{\text{old}} - \eta \nabla_{\mathrm{w}} E_{\mathcal{B}}$)
- Most common variations are stochastic gradient descent (SGD) and SGD using mini-batches.