# Deep Learning - Foundations and Concepts

## Chapter 5. Single-layer Networks: Classification

nonlineark@github

February 13, 2025

# Outline

1. Discriminant Functions
2. Decision Theory

# Discriminant functions

- The goal in classification is to take an input vector $x \in \mathbb{R}^D$ and assign it to one of $K$ discrete classes $\mathcal{C}_k$.
- A discriminant is a function that takes an input vector $x$ and assigns it to one of $K$ classes, denoted $\mathcal{C}_k$.
- We will restrict attention to linear discriminants, for which the decision surfaces are hyperplaines.

# Two classes

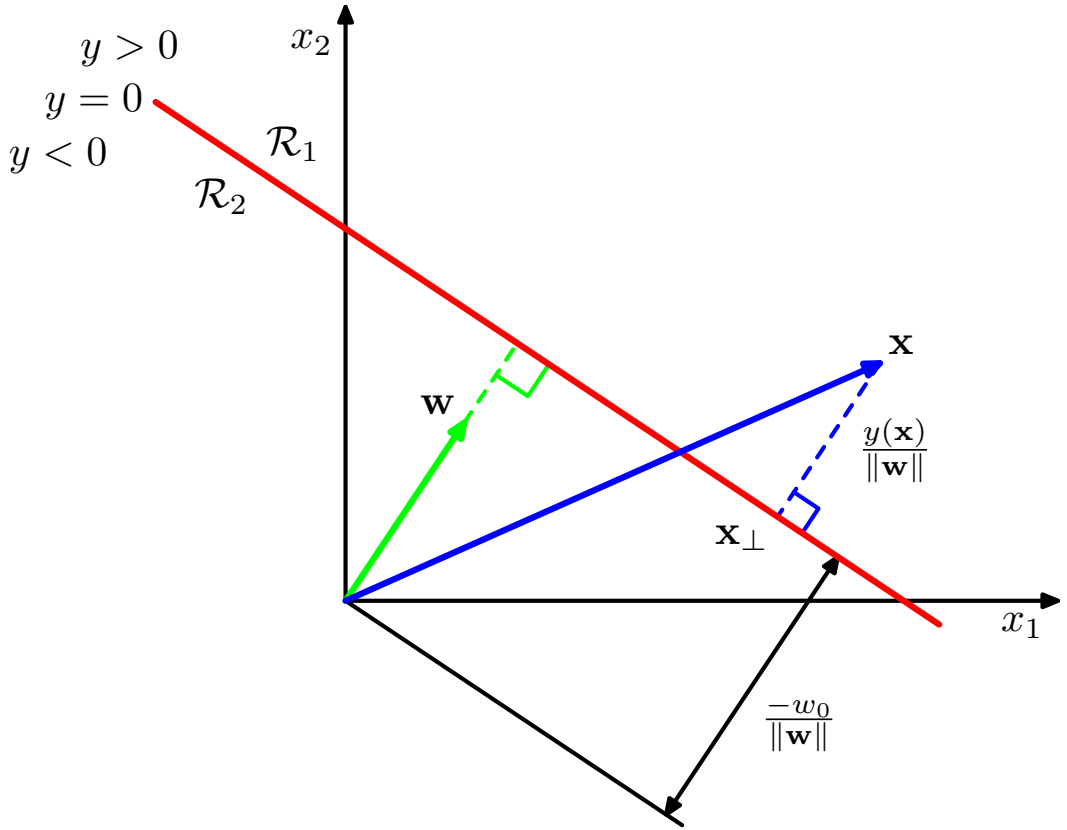
Taking a linear function of the input vector:

$$y(x) = w^T x + w_0$$

- An input vector is assigned to class $\mathcal{C}_1$ if $y(x) \geq 0$ and to class $\mathcal{C}_2$ otherwise.
- The decision boundary is a $(D-1)$-dimensional hyperplane.

# Two classes

Figure: The geometry of a linear discriminant function in two dimensions



# Two classes

It's easy to see that:

- $w$ is orthogonal to the decision surface.
- $w$ points to the direction of the increase of $y$.

Also the value of $y(x)$ gives a signed measure of the perpendicular distance $r$ of the point $x$ from the decision surface:

$$x = x_\perp + r\frac{w}{||w||}$$

$$y(x) = w^T x + w_0 = w^T x_\perp + w_0 + r||w|| = r||w||$$

$$r = \frac{y(x)}{||w||}$$

In particular, the signed distance of the origin from the decision surface is given by $\frac{w_0}{||w||}$.

# Multiple classes

Building a $K$-class discriminant by combining a number of two-class discriminant functions usually doens't work:

- One-versus-the-rest.
- One-versus-one.

# Multiple classes

Consider a single $K$-class discriminant comprising $K$ linear functions of the form:

$$y_k(x) = w_k^T x + w_{k0}$$

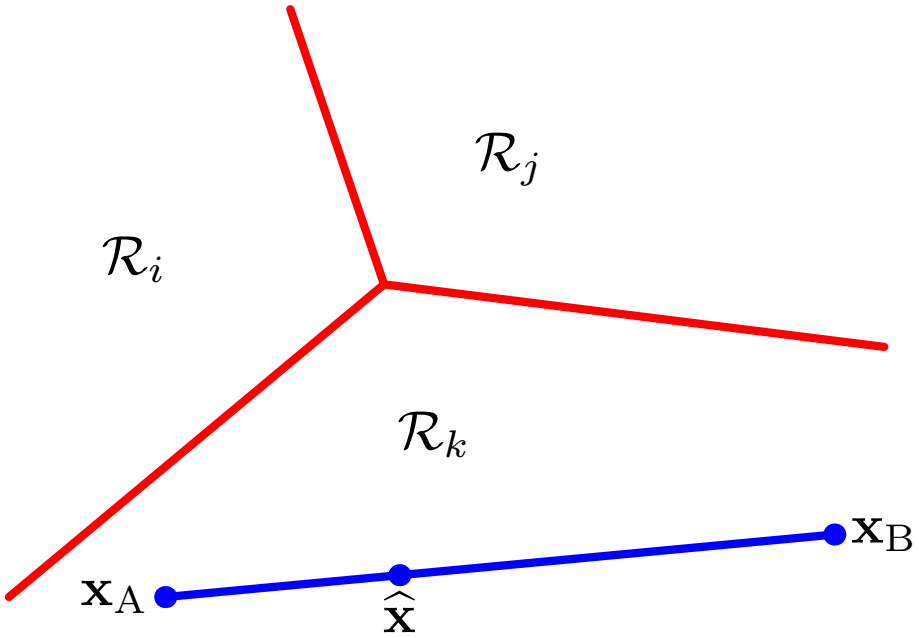
Assign a point $x$ to class $\mathcal{C}_k$ if $y_k(x) > y_j(x)$ for all $j \neq k$. The decision boundary between class $\mathcal{C}_k$ and $\mathcal{C}_j$ is given by $y_k(x) = y_j(x)$ and corresponds to a $(D-1)$-dimensional hyperplane:

$$(w_k - w_j)^T x + (w_{k0} - w_{j0}) = 0$$

The decision regions of such a discriminant are always singly connected and convex.

# Multiple classes

Figure: The decision regions for a multi-class linear discriminant



# Linear squares for classification

Consider a general classification problem with $K$ classes:

- There are $N$ input data: $x^1, \ldots, x^N$, where $x^n \in \mathbb{R}^D$.
- There are $N$ target data: $t^1, \ldots, t^N$ using a 1-of-$K$ binary coding scheme, thus $t^n \in \mathbb{R}^K$.
  - Let $T = \begin{pmatrix} t^1 & t^2 & \ldots & t^N \end{pmatrix}^T \in \mathbb{R}^{N \times K}$.
- Each class $\mathcal{C}_k$ is described by its own linear model so that $y_k(x) = w_k^T x + w_{k0}$.
  - Let $\tilde{w}_k = \begin{pmatrix} w_{k0} \\ w_k \end{pmatrix}$ and $\tilde{W} = \begin{pmatrix} \tilde{w}_1 & \tilde{w}_2 & \ldots & \tilde{w}_K \end{pmatrix} \in \mathbb{R}^{(D+1) \times K}$.
  - Let $\tilde{x} = \begin{pmatrix} 1 \\ x \end{pmatrix}$ and $\tilde{X} = \begin{pmatrix} \tilde{x}^1 & \tilde{x}^2 & \ldots & \tilde{x}^N \end{pmatrix}^T \in \mathbb{R}^{N \times (D+1)}$.
  - Then $y_k(x) = \tilde{w}_k^T \tilde{x}$ and $y(x) = \tilde{W}^T \tilde{x}$.

# Linear squares for classification

Let's determine the parameter matrix $\tilde{W}$ by minimizing a sum-of-squares error function:

$$
\begin{aligned}
E_D(\tilde{W}) &= \frac{1}{2}\sum_{i,j}(\tilde{X}\tilde{W} - T)^2_{ij} \\
&= \frac{1}{2}\mathrm{tr}((\tilde{X}\tilde{W} - T)^T(\tilde{X}\tilde{W} - T)) \\
DE_D(\tilde{W})H &= \mathrm{tr}((\tilde{X}\tilde{W} - T)^T\tilde{X}H) \\
\tilde{W}_* &= (\tilde{X}^T\tilde{X})^{-1}\tilde{X}^T T
\end{aligned}
$$

# Linear squares for classification

What property does $y(x) = \tilde{W}_*^T \tilde{x}$ has? Because $t^n$ is using a 1-of-$K$ binary coding scheme, we know:

$$\begin{pmatrix} 1 & 1 & \dots & 1 \end{pmatrix} t^n = 1$$

Thus we have:

$$\begin{aligned} \begin{pmatrix} 1 & 1 & \dots & 1 \end{pmatrix} y(x) &= \begin{pmatrix} 1 & 1 & \dots & 1 \end{pmatrix} \tilde{W}_*^T \tilde{x} \\ &= \begin{pmatrix} 1 & 1 & \dots & 1 \end{pmatrix} T^T \tilde{X} (\tilde{X}^T \tilde{X})^{-1} \tilde{x} \\ &= \begin{pmatrix} 1 & 1 & \dots & 1 \end{pmatrix} \tilde{X} (\tilde{X}^T \tilde{X})^{-1} \tilde{x} \\ &= \mathrm{e}_1^T \tilde{X}^T \tilde{X} (\tilde{X}^T \tilde{X})^{-1} \tilde{x} = \mathrm{e}_1^T \tilde{x} = 1 \end{aligned}$$

That is, the predictions made by the model will have the property that the elements of $y(x)$ will sum to $1$ for any value of $x$.

# Linear squares for classification

- The model outputs cannot be interpreted as probabilities because they are not contrained to lie within the interval $(0, 1)$.
- If the true distribution of the data is markedly different from being Gaussian, the least squares can give poor results.
- Least squares is very sensitive to the presence of outliers (a.k.a., lack robustness).

# Misclassification rate

To minimize the chance of assigning $x$ to the wrong class, intuitively we would choose the class having the higher posterior probability.

- Divide the input space into regions $\mathcal{R}_k$ called decision regions.
- All points in $\mathcal{R}_k$ are assigned to class $\mathcal{C}_k$.

We want to maximize the probability of being correct:

$$p(\text{correct}) = \sum_{k=1}^{K} p(x \in \mathcal{R}_k, \mathcal{C}_k) = \sum_{k=1}^{K} \int_{\mathcal{R}_k} p(\mathcal{C}_k|x)p(x)\mathrm{d}x$$

It's easy to see that this is maximized when the regions $\mathcal{R}_k$ are chosen such that each $x$ is assigned to the class for which $p(\mathcal{C}_k|x)$ is largest. So the intuition is indeed correct.

# Expected loss

- Sometimes, our objective will be more complex than minimizing the number of misclassifications.
- We can introduce a loss function which measure loss incurred in taking any of the available decisions or actions and minimize the total loss.

If the true class for $x$ is $\mathcal{C}_k$ and we assign $x$ to $\mathcal{C}_j$, we incur some level of loss denoted by $L_{kj}$. Because we do not know the true class, instead of minimizing the loss function, we minimize its average:

$$E(L) = \sum_k \sum_j \int_{\mathcal{R}_j} L_{kj} p(x, \mathcal{C}_k) \mathrm{d}x = \sum_j \int_{\mathcal{R}_j} \sum_k L_{kj} p(\mathcal{C}_k|x) p(x) \mathrm{d}x$$

The decision rule that minimizes the expected loss assigns $x$ to the class $j$ for which $\sum_k L_{kj} p(\mathcal{C}_k|x)$ is a minimum.

# The reject option

- Classification errors arise when the largest of the posterior probabilities is significantly less than $1$.
- Reject option: Avoid making decisions on such cases to obtain a lower error rate.
- Introduce a threshold $\theta$ and reject inputs $x$ when the largest of the posterior probabilities is less than or equal to $\theta$:
  - $\theta = 1$: All examples are rejected.
  - $\theta < \frac{1}{K}$: No examples are rejected.

# Inference and decision

There are three distinct approaches to solving decision problems:

- Generative models:
  - Solve the inference problem of determining the class-conditional densities $p(x|\mathcal{C}_k)$.
  - Infer the prior class probabilities $p(\mathcal{C}_k)$.
  - Find the posterior class probabilities $p(\mathcal{C}_k|x) = \frac{p(x|\mathcal{C}_k)p(\mathcal{C}_k)}{p(x)}$.
  - Use decision theory to determine the class membership for each new input $x$.
- Discriminative models:
  - Solve the inference problem of determining the posterior class probabilities $p(\mathcal{C}_k|x)$.
  - Use decision theory to assign each new $x$ to one of the classes.
- Discriminant functions:
  - Find a function that maps each input $x$ directly onto a class label.

# Inference and decision

There are many reasons for wanting to compute the posterior probabilities:

- Minimizing risk: What if the loss matrix are subjected to revision from time to time?
- Reject option.
- Compensating for class priors: What if one class occupies $99.9\%$ of the cases (we want a balanced data set to find a more accurate model)?
- Combining models:
  - Combine the outputs of smaller models use the rules of probability.
  - Models can easily be made differentiable with respect to adjustable parameters, which allows them to be composed and trained jointly.

# Classifier accuracy

Consider a cancer screening example:

- True positive: The classifier predicts that a person has cancer and is correct.
- False positive (type $1$ errors): The classifier predicts that a person has cancer and is wrong.
- True negative: The classifier predicts that a person does not have cancer and is correct.
- False negative (type $2$ errors): The classifier predicts that a person does not have cancer and is wrong.

# Classifier accuracy

$$\text{Accuracy} = \frac{N_{TP} + N_{TN}}{N_{TP} + N_{FP} + N_{TN} + N_{FN}}$$

$$\text{Precision} = \frac{N_{TP}}{N_{TP} + N_{FP}}$$

$$\text{Recall} = \frac{N_{TP}}{N_{TP} + N_{FN}}$$

$$\text{False positive rate} = \frac{N_{FP}}{N_{FP} + N_{TN}}$$

$$\text{False discovery rate} = \frac{N_{FP}}{N_{FP} + N_{TP}}$$

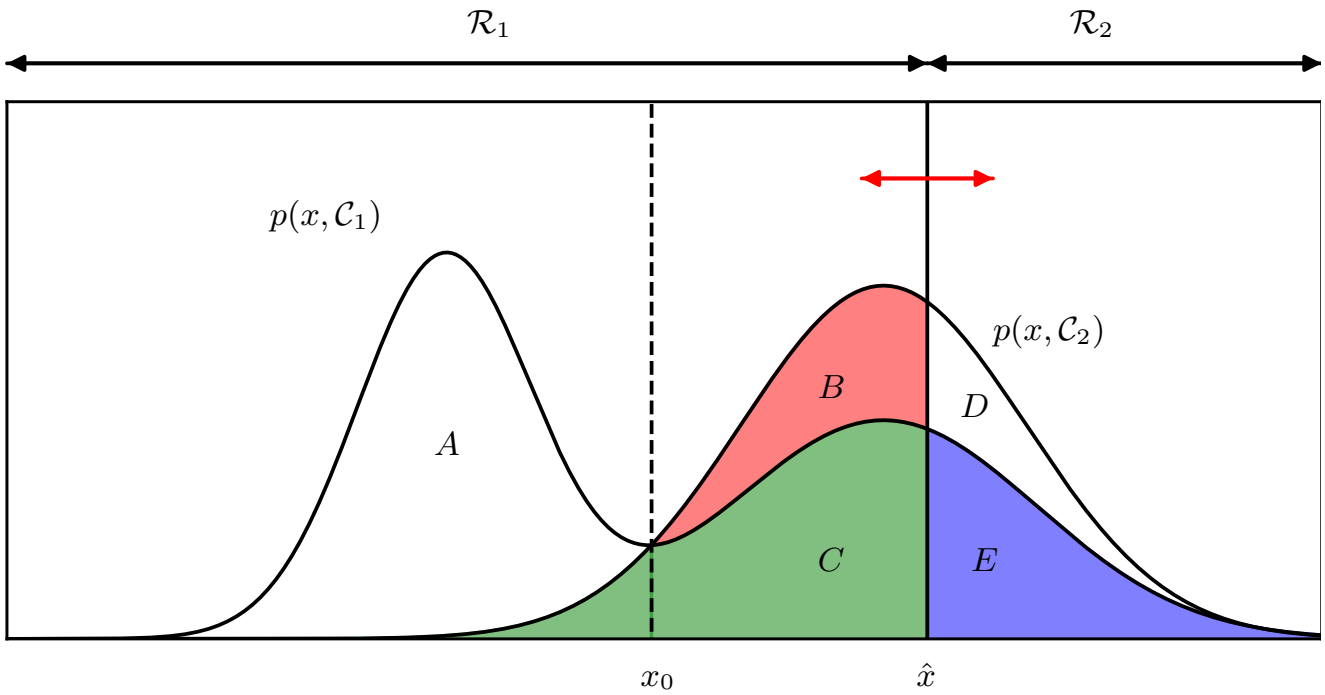
# ROC curve

There is a trade-off between type $1$ errors and type $2$ errors. To better understand this trade-off, it is useful to plot the ROC (receiver operating characteristic) curve:

- $x$-axis: $\text{False positive rate} = \frac{N_{FP}}{N_{FP}+N_{TN}}$.
- $y$-axis: $\text{True positive rate} = \frac{N_{TP}}{N_{TP}+N_{FN}}$.

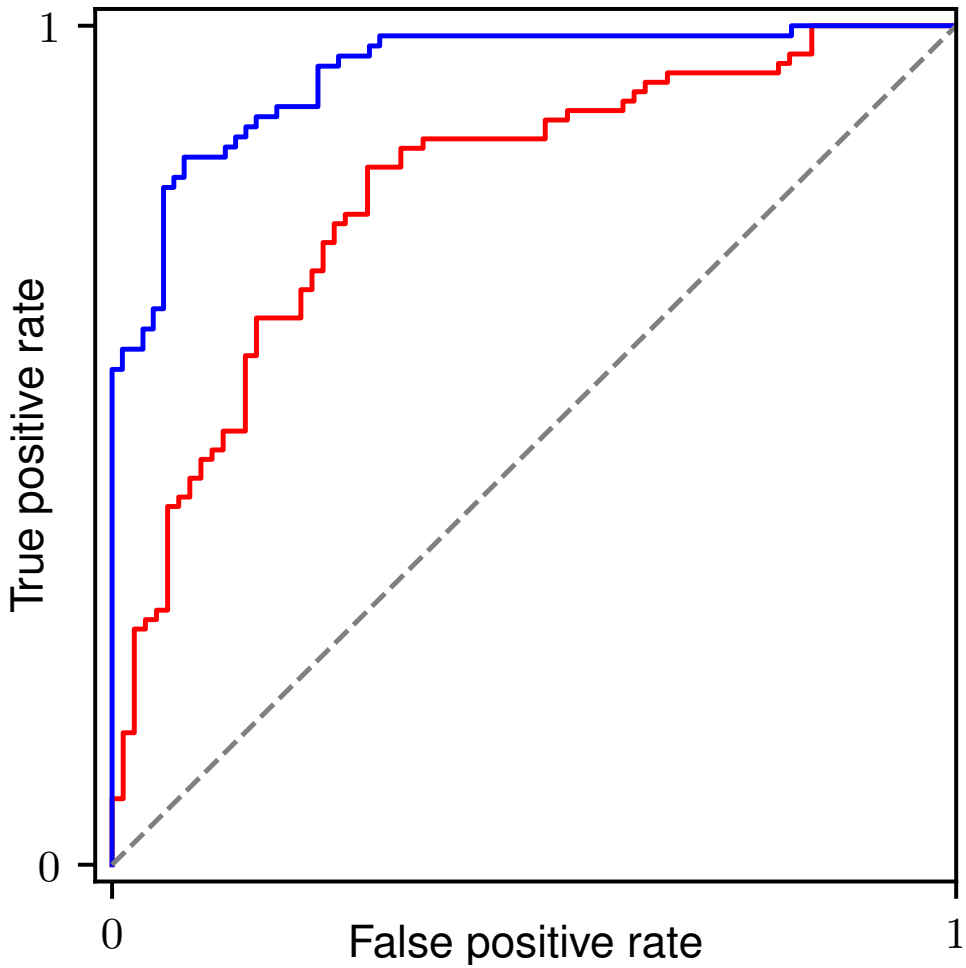
# ROC curve

Figure: As the decision boundary is moved from $\infty$ to $-\infty$, the ROC curve is traced out



# ROC curve

Figure: The ROC (receiver operating characteristic) curve



# ROC curve

Some observations:

- The bottom left corner represents a classifier that always outputs negative.
- The top left corner represents the best possible classifier.
- The top right corner represents a classifier that always outputs positive.
- The diagonal line represents a simple random classifier.

Sometimes it is useful to have a single number that characterises the whole ROC curve:

- The AUC (area under the curve):
  - $0.5$: Random guessing.
  - $1.0$: Perfect classifier.
- The F-score: $F = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = \frac{2N_{TP}}{2N_{TP} + N_{FP} + N_{FN}}$.