Classification problem

Imagine that we have a classification problem: From features X, we predict one of K classes.

For example,

2. In the ER, a person with a certain set of symptoms could have one of three `medical condition`s: Stroke, drug overdose, or epileptic seizure. Which of the three conditions does the individual have?

    - The outcome `medical condition` has three categories/classes.

Let's refer to the classes as $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$.

Q: If we know $p(X, \mathcal{C}_k)$, how can we build a classifier?

Bayes in classification

We are looking to use the observed features, X, to predict a class, $\mathcal{C}_k$.

As with Logistic Regression, it is useful to consider the probabilities,

$$p(\mathcal{C}_k|X).$$

Hope: If we know these conditional probabilities, and someone gives us a datapoint with features X. We could predict that the data point belongs to the most-likely class at X.

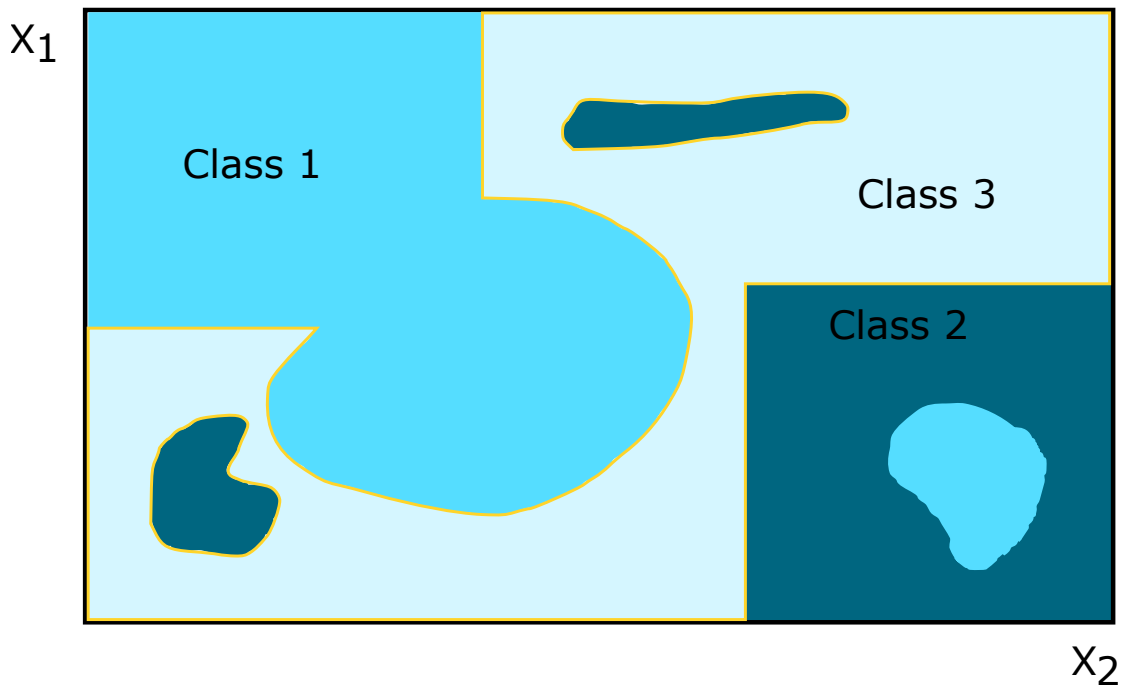
We can estimate $p(\mathcal{C}_k|X)$ from training data using Bayes' Theorem

$$p(\mathcal{C}_k|X) = \frac{p(X|\mathcal{C}_k)p(\mathcal{C}_k)}{p(X)}.$$

Bayes in classification

Let's use $p(\mathcal{C}_k|X)$ to classify *aiming to make as few misclassifications* as possible.

- We need to define a *decision rule:* When do we classify a point as $\mathcal{C}_k$?
- Any decision rule divides the feature space into *decision regions* $\mathcal{R}_k$, $k \in 1, 2, \ldots, K$.
- (These decision regions are separated by *decision boundaries*)



Bayes in classification

Aim to make as few misclassifications as possible. Minimize:

$$p(\text{mistake}) = p(X \in \mathcal{R}_1, \mathcal{C}_2) + p(X \in \mathcal{R}_2, \mathcal{C}_1) \tag{1}$$

$$= \int_{\mathcal{R}_1} p(X, \mathcal{C}_2) dX + \int_{\mathcal{R}_2} p(X, \mathcal{C}_1) dX \tag{2}$$

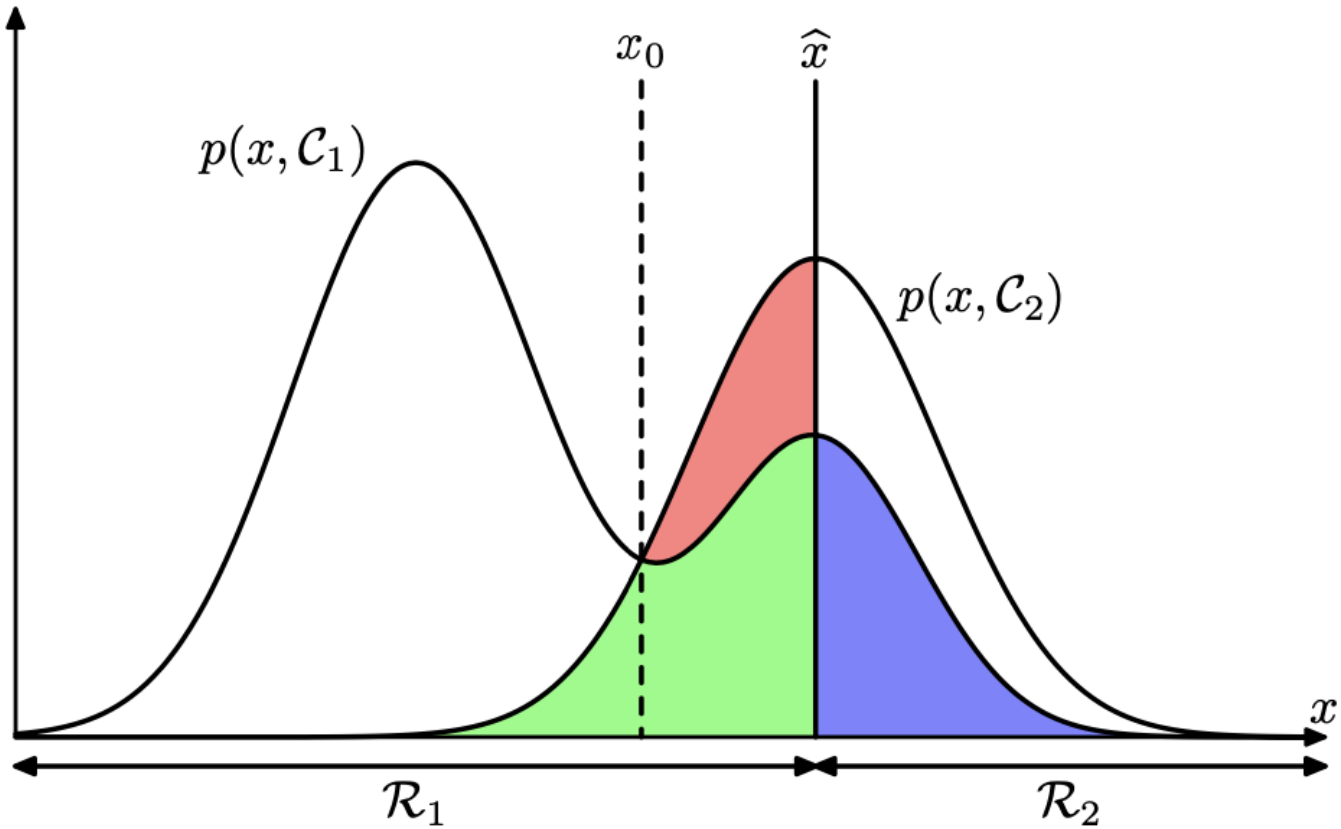
From the product rule, $p(X, \mathcal{C}_k) = p(X)p(\mathcal{C}_k|X)$, so we get,

$$p(\text{mistake}) = \int_{\mathcal{R}_1} p(X)p(\mathcal{C}_2|X) dX + \int_{\mathcal{R}_2} p(X)p(\mathcal{C}_1|X) dX$$

$p(X)$ contributes to both terms integrals, so we limit mistakes the most by assigning the class with the highest $p(\mathcal{C}_k|X)$ at X. **Aligning with our intuition!**

Bayes in classification

We limit mistakes the most by assigning the class with the highest $p(\mathcal{C}_k|X)$ at X.

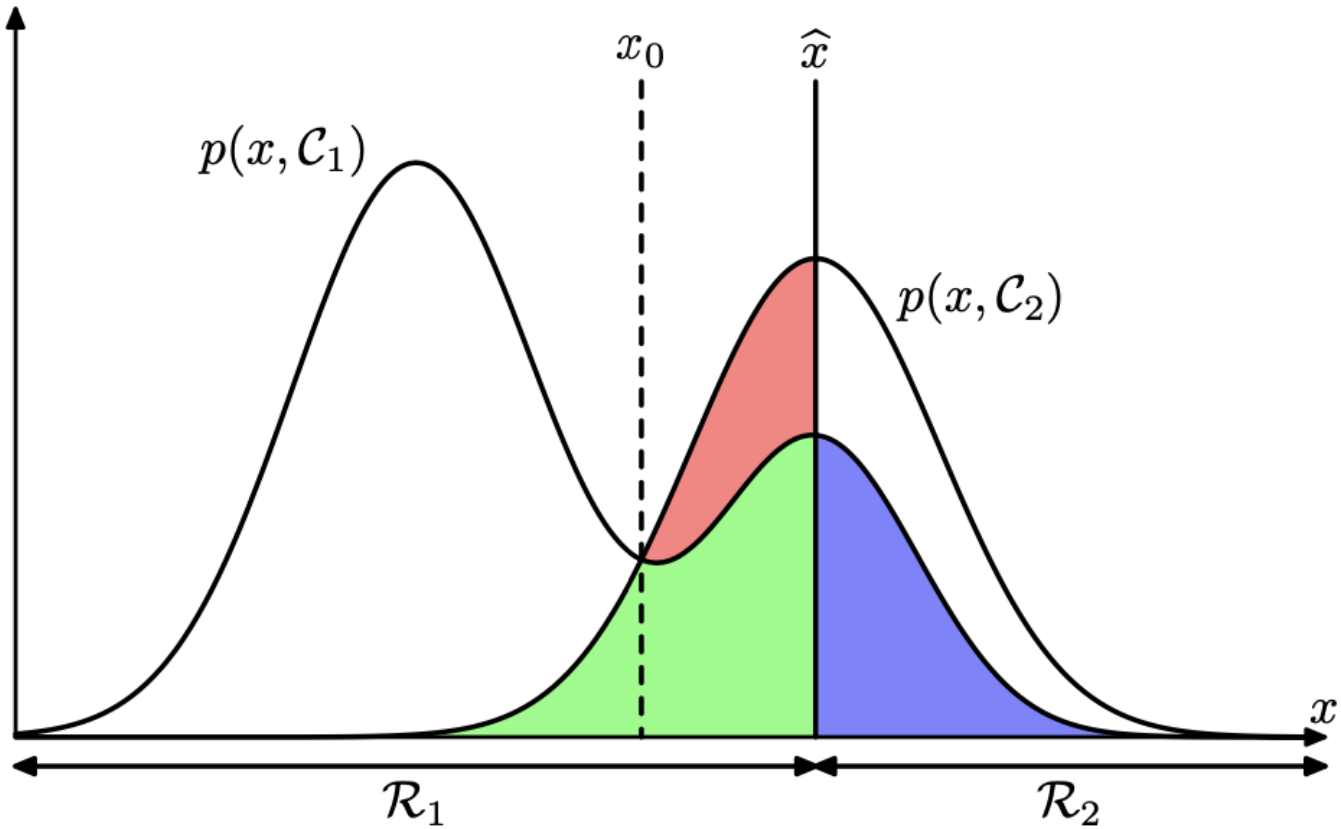


(Argument generalizes to several classes)

Aligning with our intuition!

Bayes in classification

We limit mistakes the most by assigning the class with the highest $p(\mathcal{C}_k|X)$ at X.



(Argument generalizes to several classes)

Aligning with our intuition! But....

Bayes Classifier

The classifier we just developed is the *Bayes classifier*.

With the Bayes classifier we classify to the class k with the *highest posterior probability*, i.e. set

$$d(x) = \arg\max_y \mathrm{P}(Y = y \,|\, x)$$

The Bayes Classifier is very important. We will get back to it later today.

Not all mistakes are equal

Sometimes, some mistakes are worse to make than others.

For example: When screening for cancer, a False Positive causes stress to the affected patient. A False Negative may cause the death of the patient. We *really* need to limit False Negatives!

So our goal may not be to limit mistakes, but to limit *certain* mistakes.

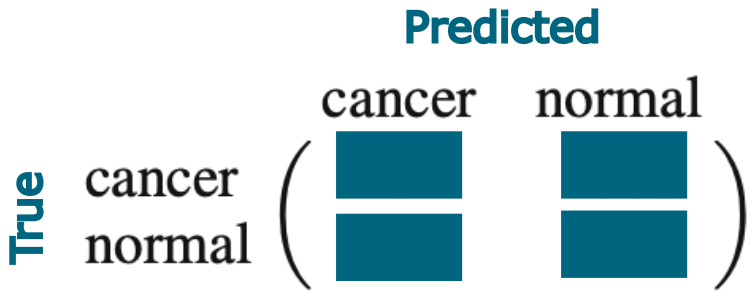
Loss and Loss matrix

Let's punish the algorithm for every classification it makes.

We do it by assigning it a **loss** L_{kj} (usually ≥ 0), which depends on k, the true class, and j, the assigned class of the data point. L_{kj} is an entry in a *loss matrix*.

Bigger loss for a given classification is a bigger punishment.

Q: For cancer detection, where should loss be big in the *loss matrix* ?



Q: Try to write down a loss matrix for the cancer problem.

Loss and Loss matrix

Let's punish the algorithm for every classification it makes.

We do it by assigning it a **loss** L_{kj} (usually ≥ 0), which depends on k, the true class, and j, the assigned class of the data point. L_{kj} is an entry in a *loss matrix*.

Bigger loss for a given classification is a bigger punishment.

Q: For cancer detection, where should loss be big in the *loss matrix* ?

$$\begin{array}{c} \\ \text{cancer} \\ \text{normal} \end{array} \begin{array}{c} \begin{array}{cc} \text{cancer} & \text{normal} \end{array} \\ \begin{pmatrix} 0 & 1000 \\ 1 & 0 \end{pmatrix} \end{array}$$

Q: Try to write down a loss matrix for the cancer problem.

Goal in terms of loss

The concept of loss gives us a new way of expressing our goal:

We want our classification algorithm to *minimize the expected loss*.

$$\mathbb{E}[L] = \sum_k \sum_j \int_{\mathcal{R}_j} L_{kj} p(X, \mathcal{C}_k) dX$$

Again, we can rewrite this to get

$$\mathbb{E}[L] = \sum_k \sum_j \int_{\mathcal{R}_j} L_{kj} p(X) p(\mathcal{C}_k|X) dX.$$

So we minimize loss by assigning a new X to the class j that minimizes

$$\sum_k L_{kj} p(\mathcal{C}_k|X).$$

Readily doable if we know the posterior probabilities $p(\mathcal{C}_k|X)$.

Loss: A few more points

Minimising posterior expected loss is enough

If $d(x)$ minimises the posterior expected loss for each fixed x (averaging over y), $d(x)$ also minimises the expected loss (i.e. averaging over both x and y).

In classification, we often use 0-1 *loss*:

$$L(j,k) = \begin{cases} 1, & j \neq k \\ 0. & j = k \end{cases}$$

In regression, we can also use loss, e.g. *Squared error loss*

$$L(y, d(x)) = (y - d(x))^2.$$

Absolute error loss

$$L(y, d(x)) = |y - d(x)|.$$

Remember: the loss function is usually non-negative.

Expected loss for a prediction mechanism

The expected loss is a theoretical quantity that has many names: test error, generalisation error, risk, prediction error.

We can estimate the expected loss from a specific dataset by the *empirical risk*:

$$\frac{1}{n}\sum_{i=1}^{n} L(Y_i, d(X_i))$$

where n is the number of observations in the dataset.

Test and training error

The *training error* is the empirical risk computed from the training set. Generally a bad estimator of the expected loss.

The term *test error* is, in practice, used to denote both

- the true expected loss
- the estimate that is the empirical risk computed from the test data.

The test error you compute by cross-validation is also an estimate of the expected loss for your prediction model.

Bayes Classifier

In the language of Loss:

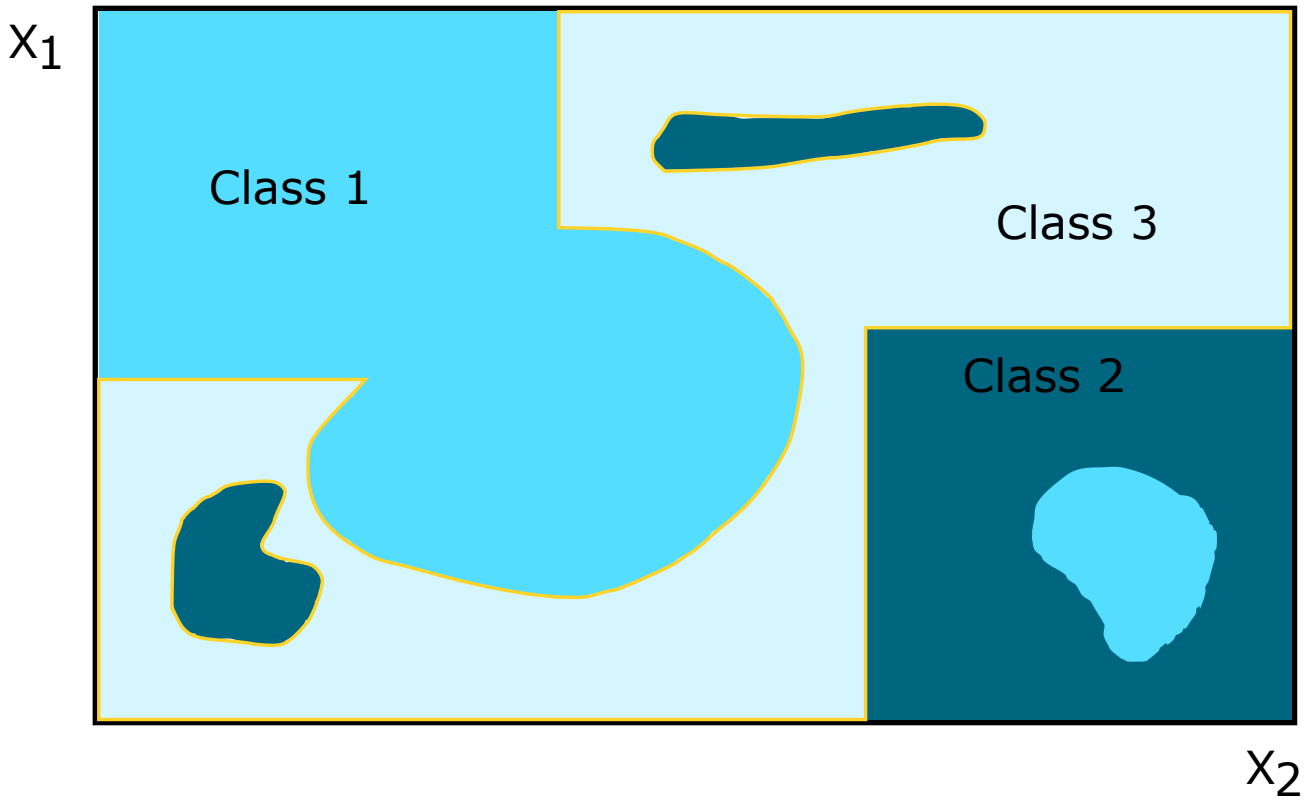
The *Bayes classifier* minimises the expected loss under the specific choice of 0-1 loss (misclassification error).

The associated error, the *Bayes error rate*, is a theoretical lower bound – a bit like the irreducible error in the bias-variance decomposition.

Remember: with a Bayes classifier, we classify to the class k with the *highest posterior probability*, i.e. set

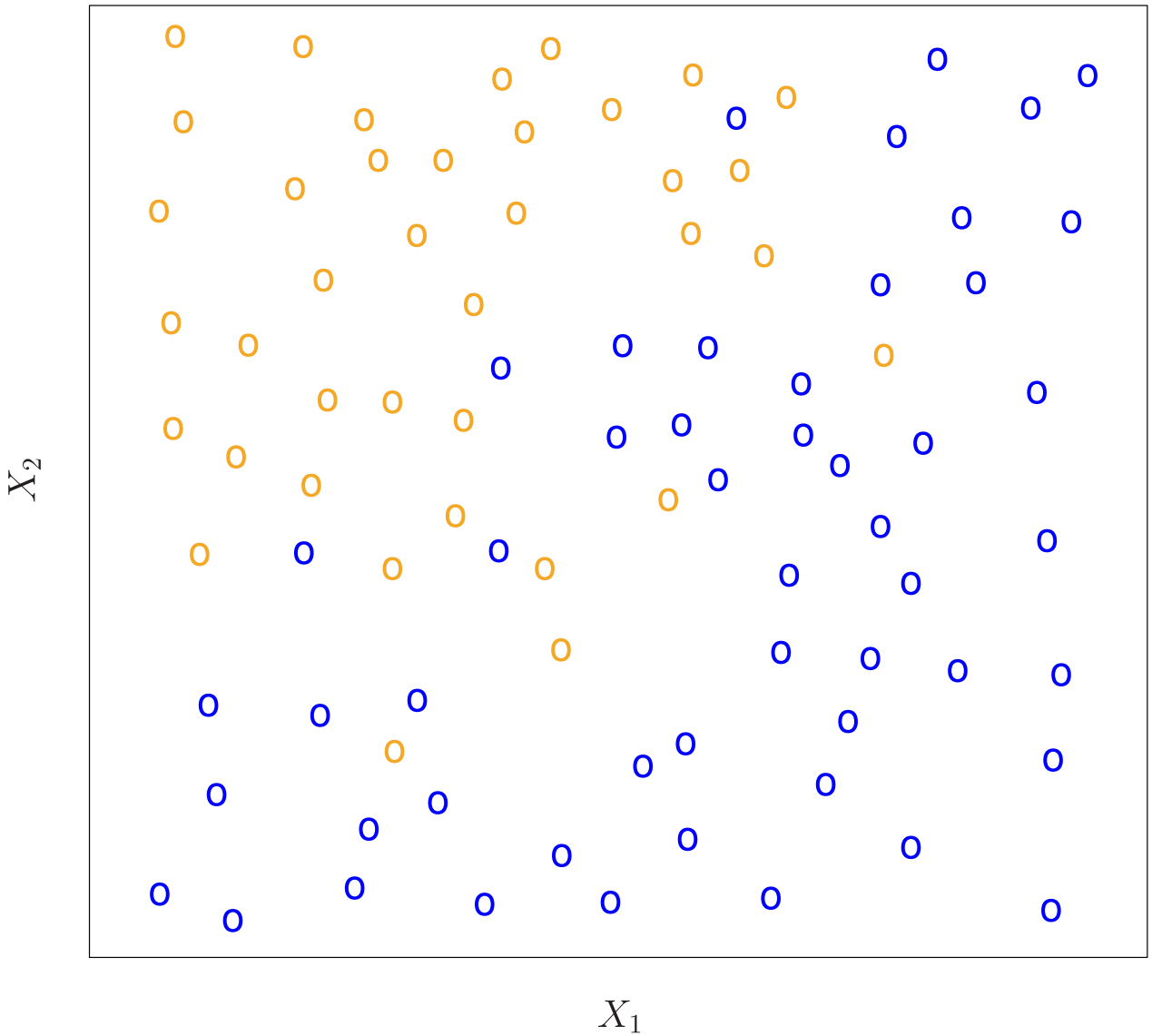
$$d(x) = \arg\max_y \mathrm{P}(Y = y \,|\, x)$$

Bayes Classifier



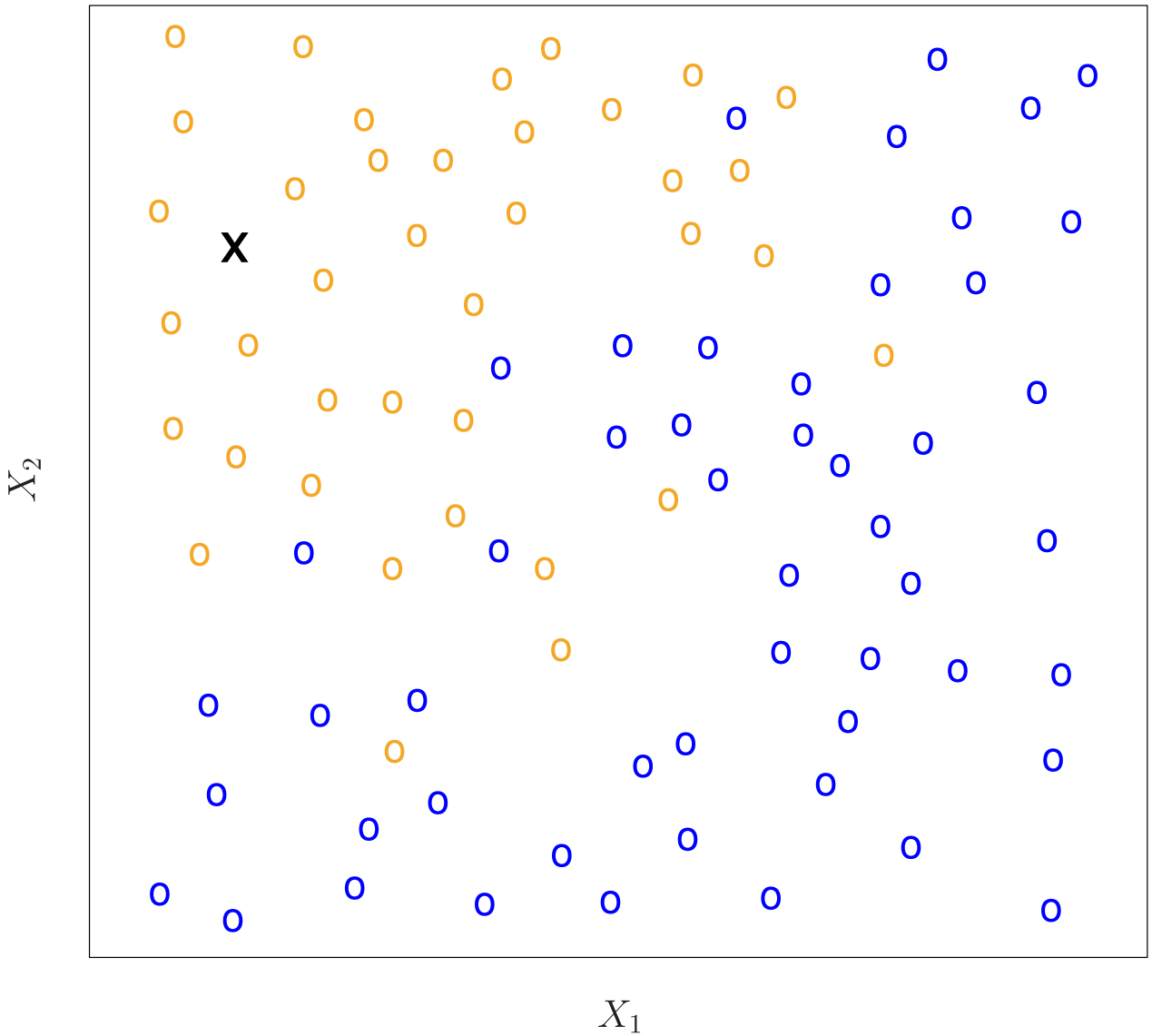
Entirely new X

Sometimes there may be many X that we have never observed. How to classify?



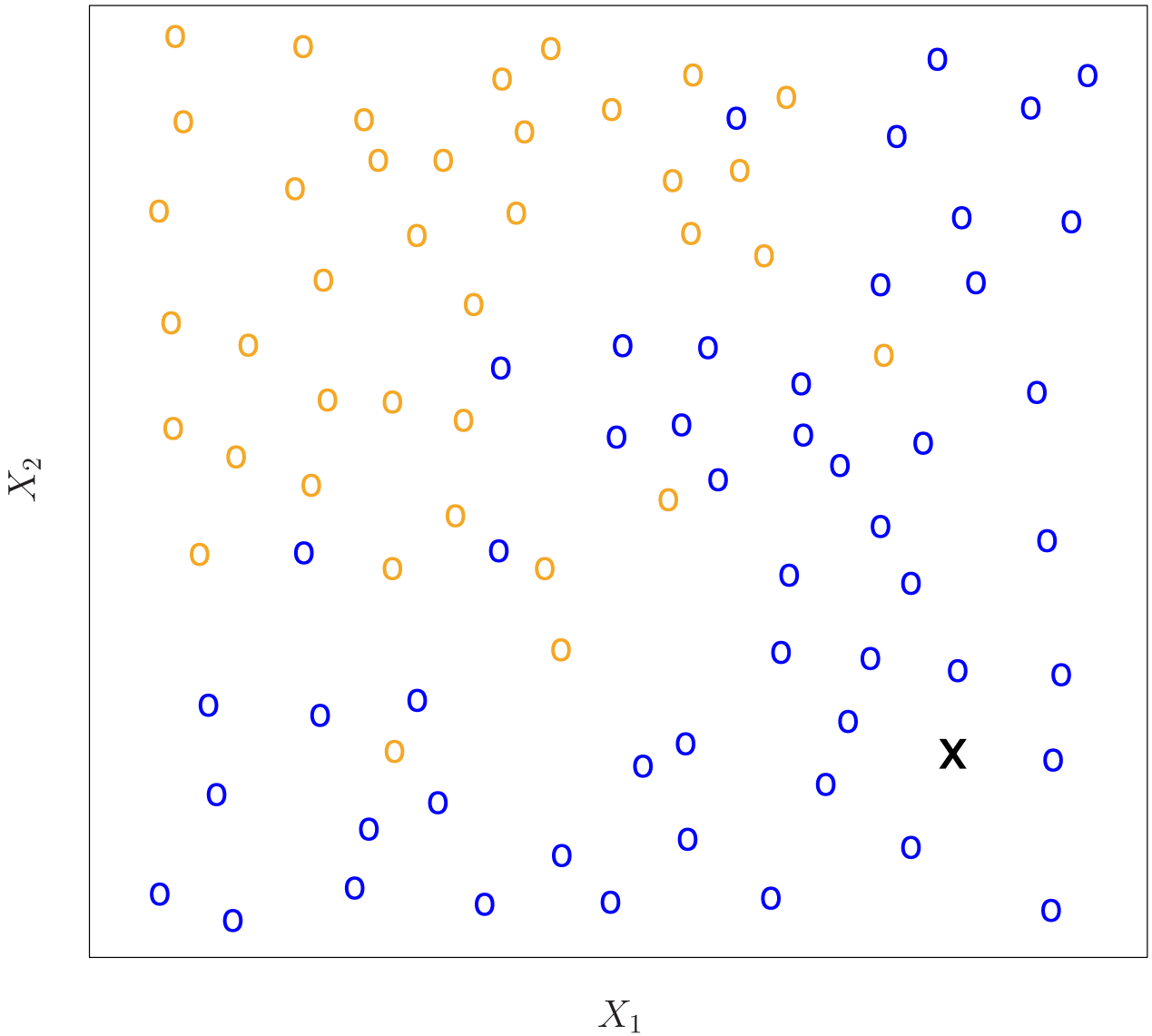
Entirely new X

Sometimes there may be many X that we have never observed. How to classify?



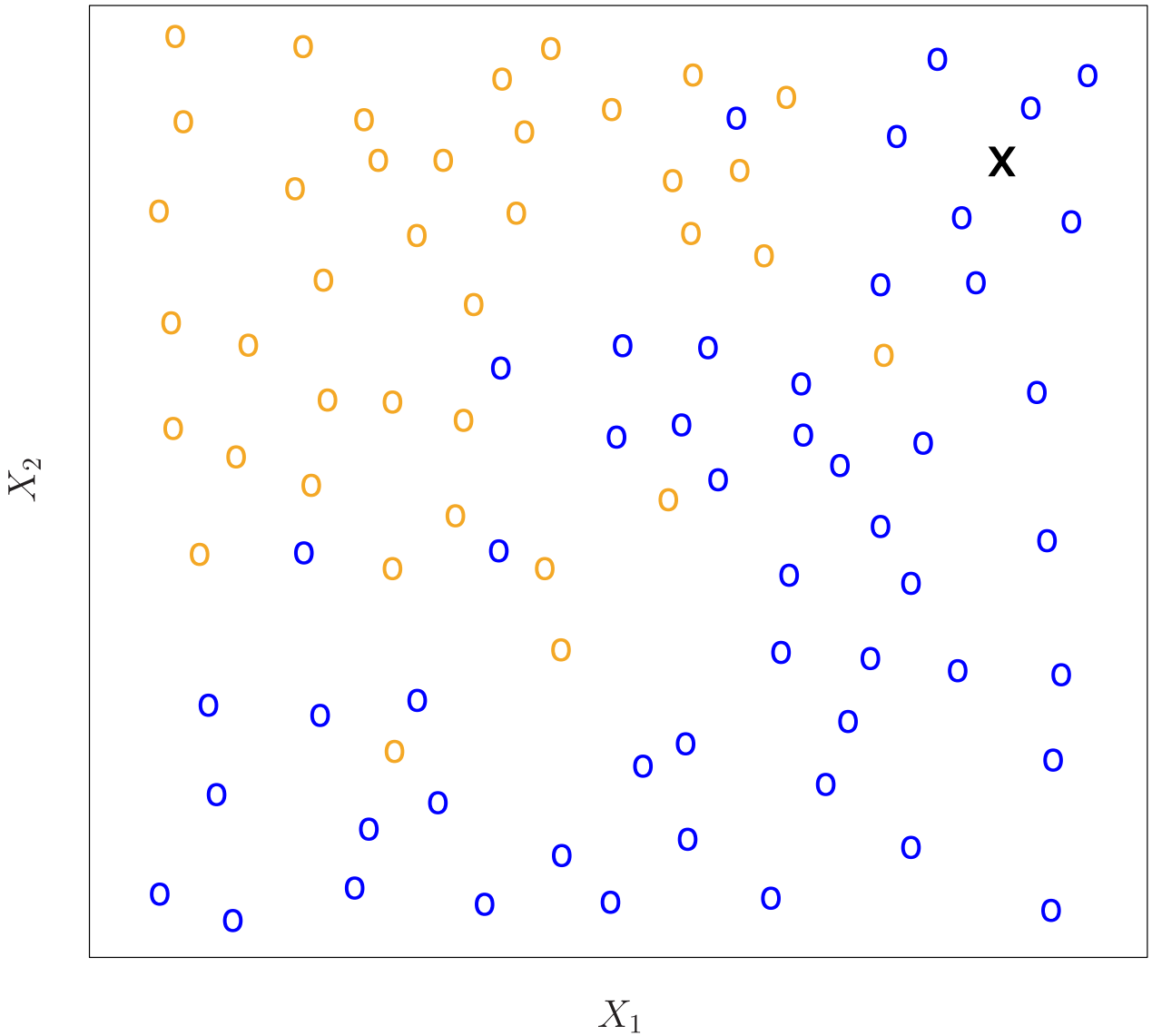
Entirely new X

Sometimes there may be many X that we have never observed. How to classify?



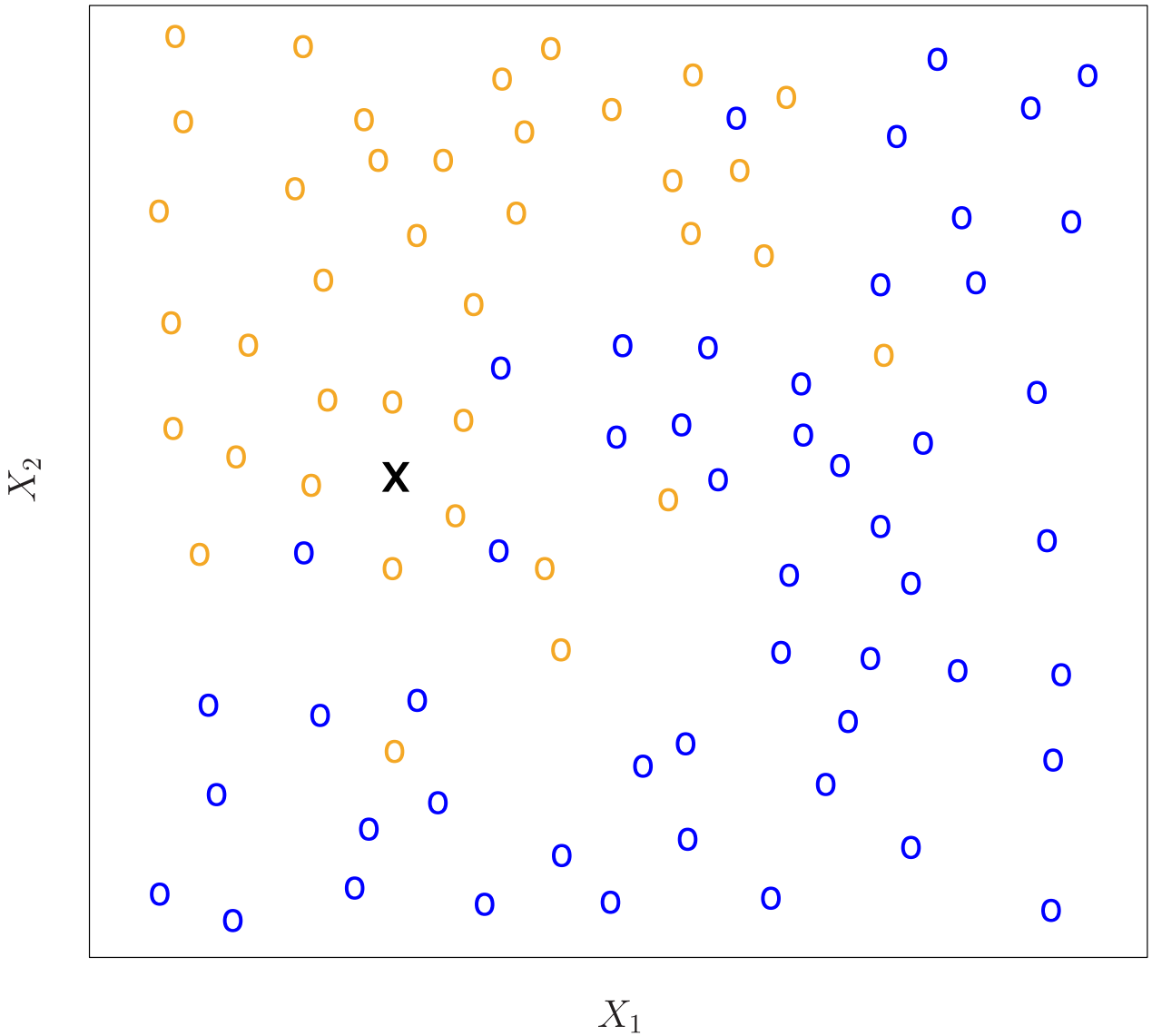
Entirely new X

Sometimes there may be many X that we have never observed. How to classify?



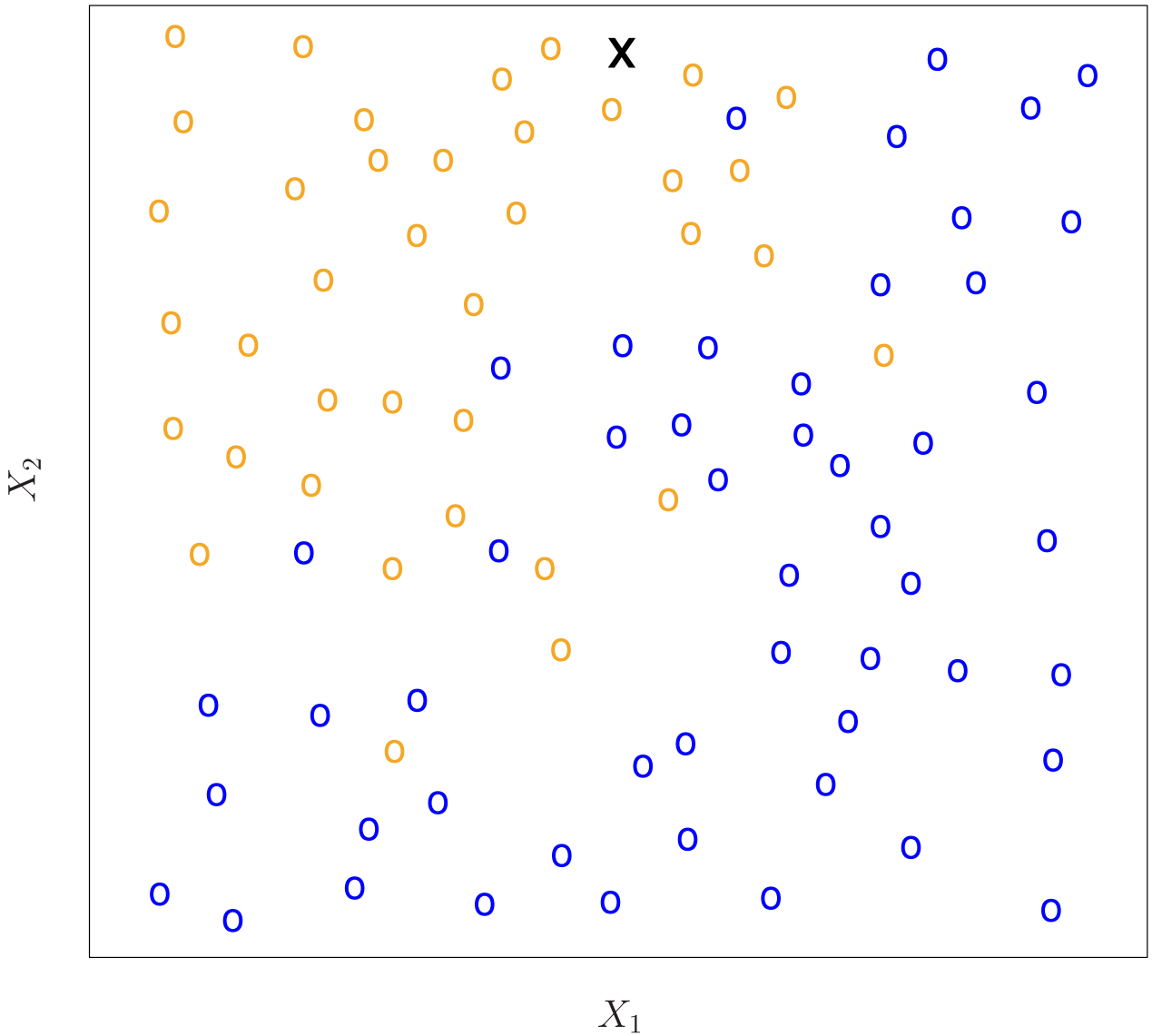
Entirely new X

Sometimes there may be many X that we have never observed. How to classify?



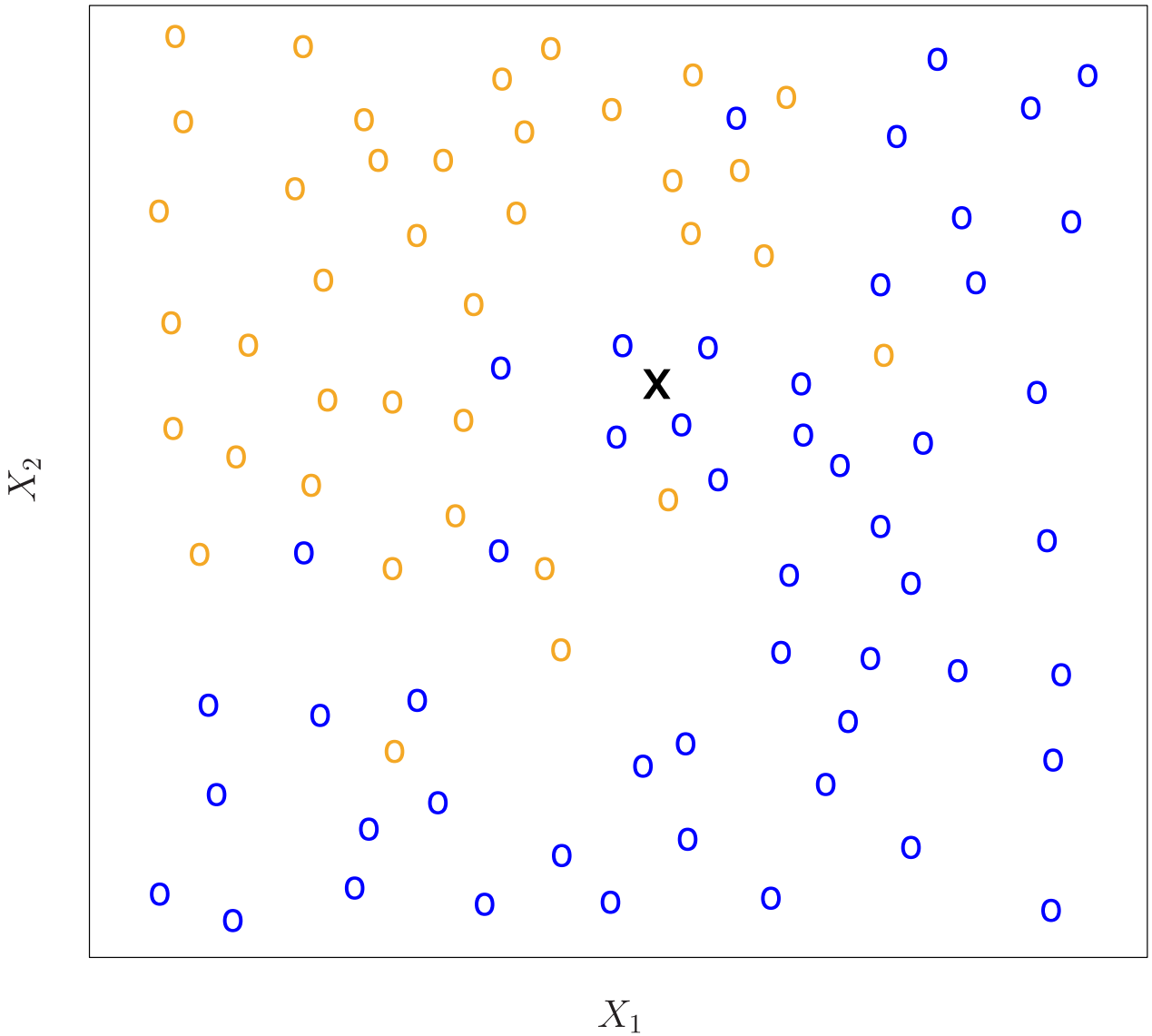
Entirely new X

Sometimes there may be many X that we have never observed. How to classify?



Entirely new X

Sometimes there may be many X that we have never observed. How to classify?



K-nearest neighbours (KNN) - A simple approximation to the Bayes classifier.

KNN approximates the posterior class probabilities.

K-nearest neighbours classification of point x_0

1. Find the K points in the training data that are closest to x_0 (call this set $\mathcal{N}_0$)
2. *Estimate the posterior probability* for class j as the fraction of points in $\mathcal{N}_0$ from class j:

$$\mathrm{P}(Y = j \,|\, X = x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} I(y_i = j).$$

3. Choose the class with highest posterior probability.

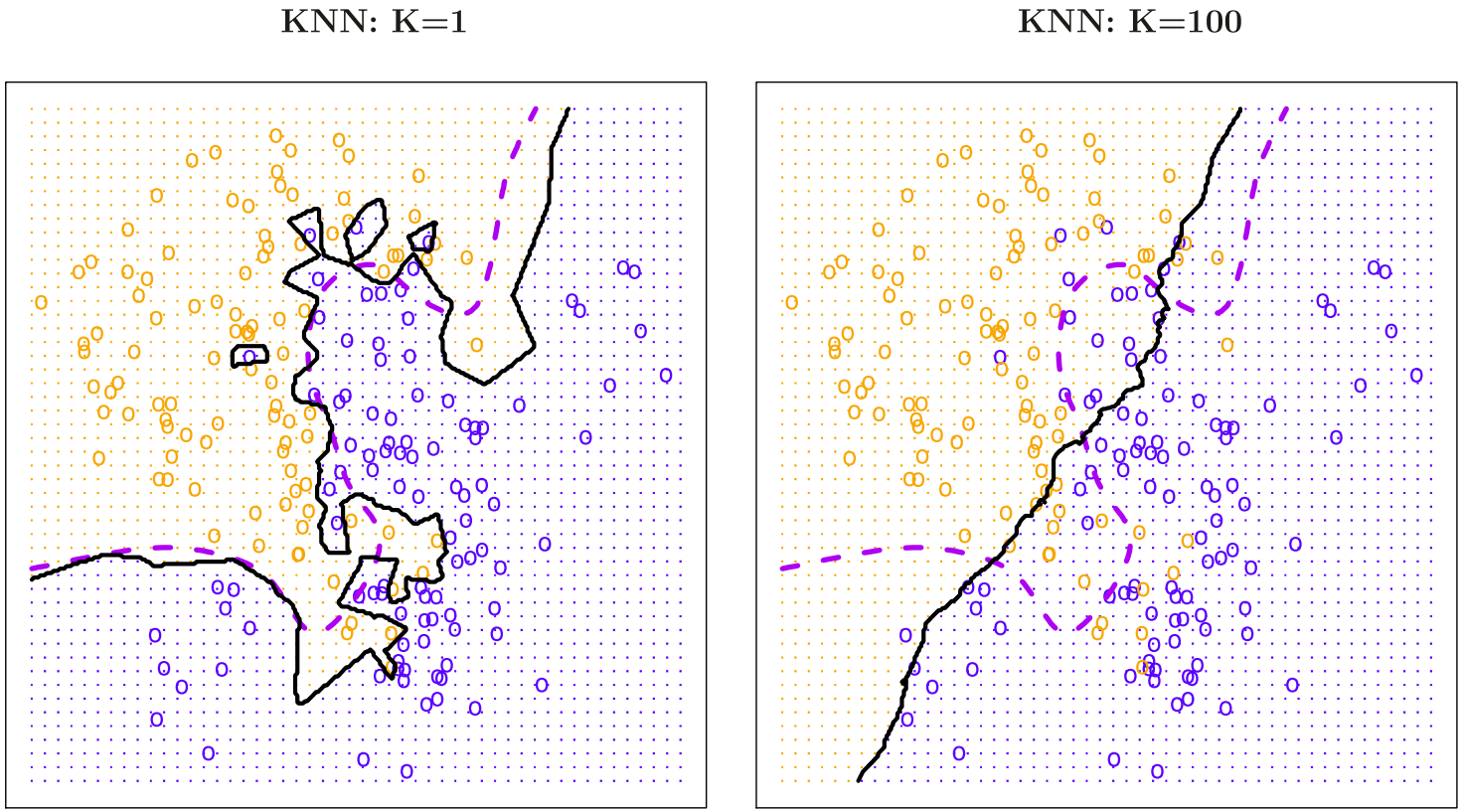


FIGURE 2.16. *A comparison of the KNN decision boundaries (solid black curves) obtained using $K = 1$ and $K = 100$ on the data from Figure 2.13. With $K = 1$, the decision boundary is overly flexible, while with $K = 100$ it is not sufficiently flexible. The Bayes decision boundary is shown as a purple dashed line.*

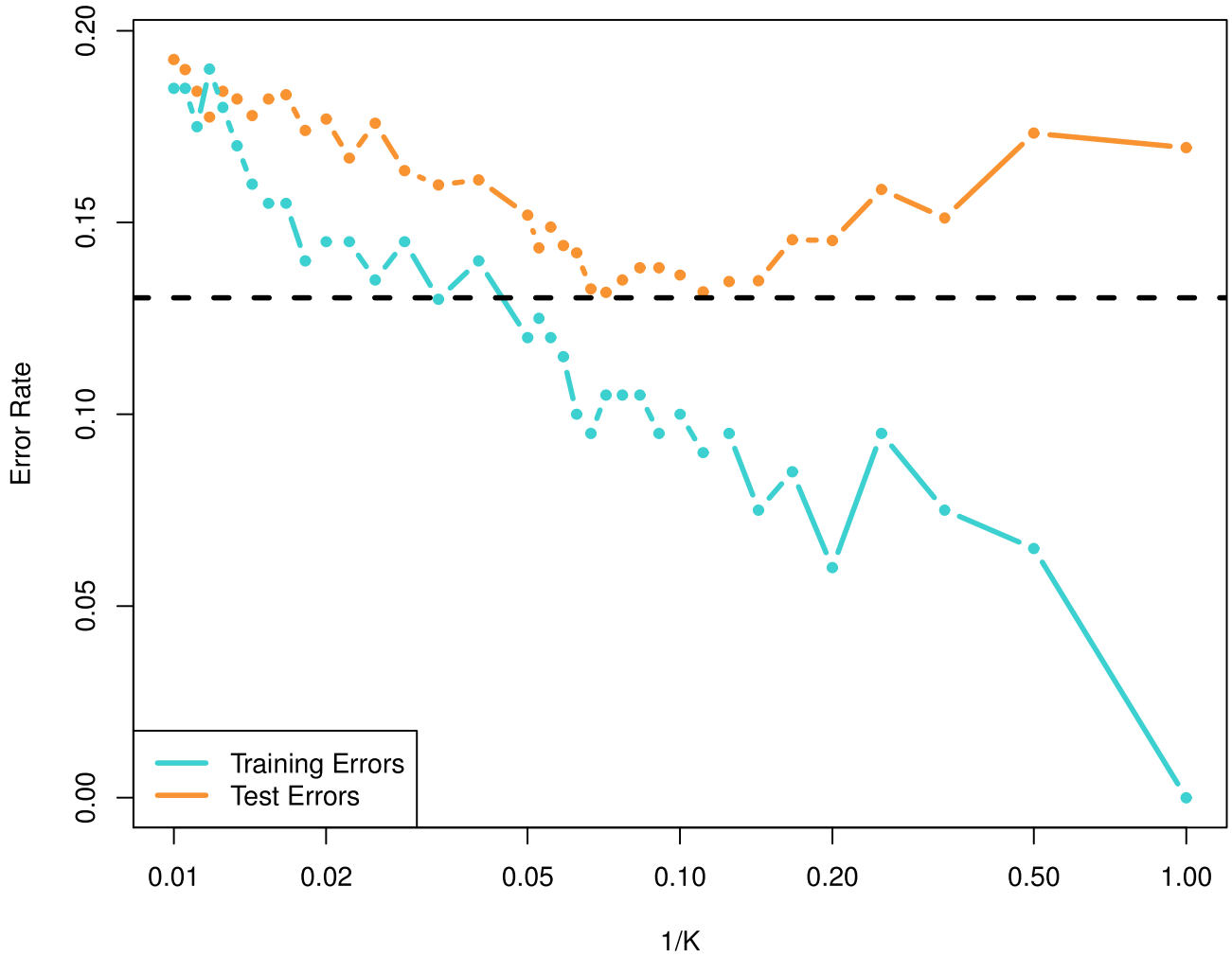


FIGURE 2.17. *The KNN training error rate (blue, 200 observations) and test error rate (orange, 5,000 observations) on the data from Figure 2.13, as the level of flexibility (assessed using $1/K$ on the log scale) increases, or equivalently as the number of neighbors K decreases. The black dashed line indicates the Bayes error rate. The jumpiness of the curves is due to the small size of the training data set.*

K-nearest neighbours (KNN): Summary

- The resulting decision rule is simply that KNN assigns a class according to a majority vote among the K closest training points.
- Gives extremely flexible boundaries
- Often a good 'baseline' classifier with error rate close to Bayes error rate
- Often does not work well in high dimension feature space
- K can be chosen by cross-validation