

# Logistic Regression for Classification

Fraida Fund

## Contents

In this lecture . . . . .	3
Classification . . . . .	3
Linear classifiers . . . . .	3
Binary classification with linear decision boundary . . . . .	3
Linear classification rule . . . . .	3
Multi-class classification: illustration . . . . .	4
Linear separability . . . . .	4
Non-uniqueness of separating hyperplane . . . . .	4
Non-existence of perfectly separating hyperplane . . . . .	5
Choosing a hyperplane . . . . .	5
Logistic regression . . . . .	5
Probabilistic model for binary classification . . . . .	5
Logistic/sigmoid function . . . . .	5
Logistic function for binary classification . . . . .	6
Logistic function with threshold . . . . .	6
Logistic model as a “soft” classifier . . . . .	7
Logistic classifier properties (1) . . . . .	7
Logistic classifier properties (2) . . . . .	7
Logistic regression - illustration . . . . .	7
Multi-class logistic regression . . . . .	8
Softmax function . . . . .	8
Softmax function as a PMF . . . . .	8
Softmax function for multi-class logistic regression (1) . . . . .	8
Softmax function for multi-class logistic regression (2) . . . . .	8
Fitting logistic regression model . . . . .	9
Learning logistic model parameters . . . . .	9
Maximum likelihood estimation (1) . . . . .	9
Maximum likelihood estimation (2) . . . . .	9
Maximum likelihood estimation (3) . . . . .	9
Maximum likelihood estimation (4) . . . . .	10
Binary cross-entropy loss (1) . . . . .	10
Binary cross-entropy loss (2) . . . . .	10
Cross-entropy loss for multi-class classification (1) . . . . .	10
Cross-entropy loss for multi-class classification (2) . . . . .	11
Minimizing cross-entropy loss . . . . .	11
Transformations, bias, variance . . . . .	12
“Recipe” for logistic regression (binary classifier) . . . . .	12
“Recipe” for logistic regression (multi-class classifier) . . . . .	12
Naive Bayes classifier . . . . .	13
Probabilistic models (1) . . . . .	13
Probabilistic models (2) . . . . .	13
Probabilistic models (3) . . . . .	13

Bayes rule . . . . .	13
Class conditional probability (1) . . . . .	14
Class conditional probability (2) . . . . .	14
Prior probability . . . . .	14
Evidence . . . . .	14
Naive bayes decision boundary . . . . .	15
Why generative model? . . . . .	15

## In this lecture

- Linear classifiers
- Logistic regression
- Fitting logistic regression
- Naive Bayes classifier

## Classification

Suppose we have a series of data points  $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)\}$  and there is some (unknown) relationship between  $\mathbf{x}_i$  and  $y_i$ .

- **Classification:** The output variable  $y$  is constrained to be  $\in 1, 2, \dots, K$
- **Binary classification:** The output variable  $y$  is constrained to be  $\in 0, 1$

## Linear classifiers

### Binary classification with linear decision boundary

- Plot training data points
- Draw a line (**decision boundary**) separating 0 class and 1 class
- If a new data point is in the **decision region** corresponding to class 0, then  $\hat{y} = 0$ .
- If it is in the decision region corresponding to class 1, then  $\hat{y} = 1$ .

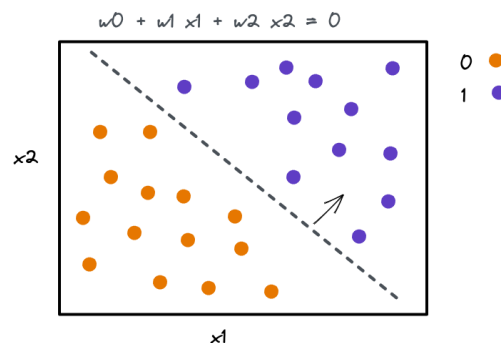


Figure 1: Binary classification problem with linear decision boundary.

### Linear classification rule

- Given a **weight vector**:  $\mathbf{w} = (w_0, \dots, w_d)$
- Compute linear combination  $z = w_0 + \sum_{j=1}^d w_d x_d$
- Predict class:

$$\hat{y} = \begin{cases} 1, & z > 0 \\ 0, & z \leq 0 \end{cases}$$

## Multi-class classification: illustration

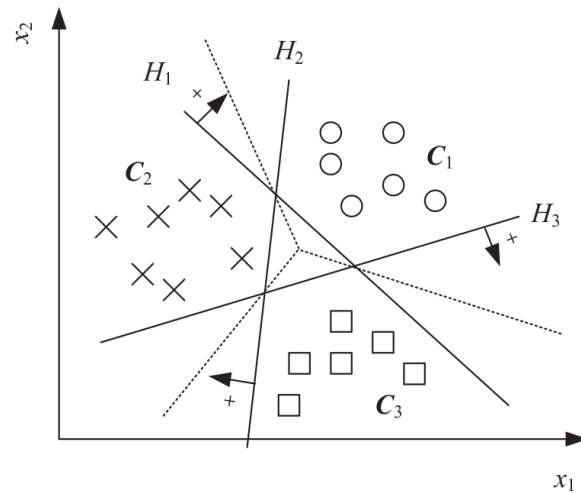


Figure 2: Each hyperplane  $H_i$  separates the examples of  $C_i$  from the examples of all other classes.

## Linear separability

Given training data

$$(\mathbf{x}_i, y_i), i = 1, \dots, N$$

The problem is **perfectly linearly separable** if there exists a **separating hyperplane**  $H_i$  such that all  $\mathbf{x} \in C_i$  lie on its positive side, and all  $\mathbf{x} \in C_j, j \neq i$  lie on its negative side.

## Non-uniqueness of separating hyperplane

When a separating hyperplane exists, it is not unique (there are in fact infinitely many such hyperplanes.)

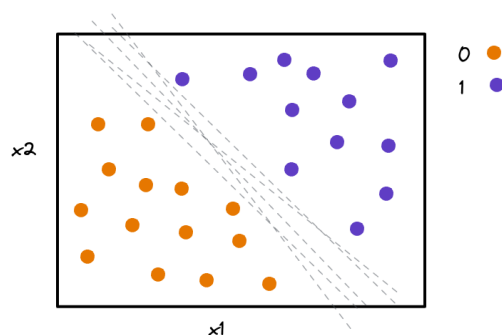


Figure 3: Several separating hyperplanes.

## Non-existence of perfectly separating hyperplane

Many datasets *not* linearly separable - some points will be misclassified by *any* possible hyperplane.

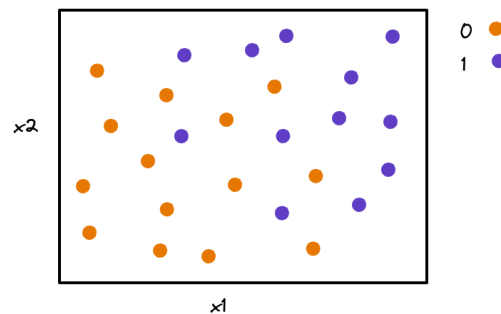


Figure 4: This data is not separable.

## Choosing a hyperplane

Which hyperplane to choose?

We will try to find the hyperplane that minimizes loss according to some **loss function**.

Will revisit several times this semester.

## Logistic regression

### Probabilistic model for binary classification

Instead of looking for a model  $f$  so that

$$y_i \approx f(x_i)$$

we will look for an  $f$  so that

$$P(y_i = 1|x_i) = f(x_i), P(y_i = 0|x_i) = 1 - f(x_i)$$

We need a function that takes a real value and maps it to range  $[0, 1]$ . What function should we use?

### Logistic/sigmoid function

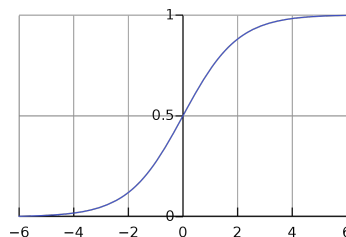


Figure 5:  $\sigma(z) = \frac{1}{1+e^{-z}}$  is a classic “S”-shaped function.

Note the intuitive relationship behind this function’s output and the distance from the linear separator (the argument that is input to the function).

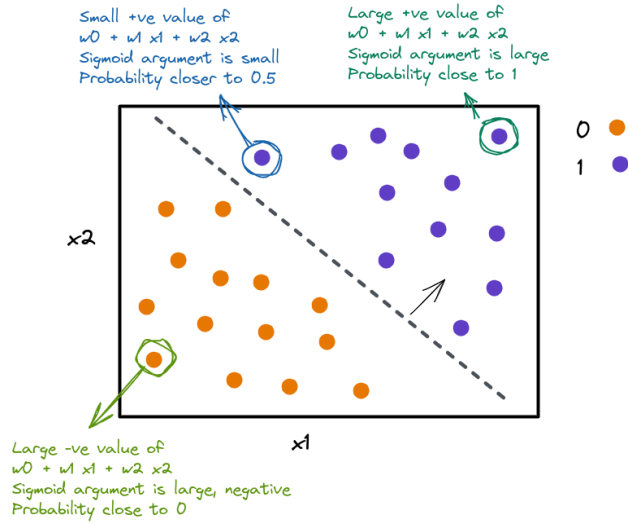


Figure 6: Output is close to 0 or 1 if the argument to the  $\sigma$  has large magnitude (point is far from separating hyperplane, but closer to 0.5 if the argument is small (point is near separating hyperplane)).

### Logistic function for binary classification

Let  $z = w_0 + \sum_{j=1}^d w_d x_d$ , then

$$P(y = 1|\mathbf{x}) = \frac{1}{1 + e^{-z}}, \quad P(y = 0|\mathbf{x}) = \frac{e^{-z}}{1 + e^{-z}}$$

(note:  $P(y = 1) + P(y = 0) = 1$ )

### Logistic function with threshold

Choose a threshold  $t$ , then

$$\hat{y} = \begin{cases} 1, & P(y = 1|\mathbf{x}) \geq t \\ 0, & P(y = 1|\mathbf{x}) < t \end{cases}$$

## Logistic model as a “soft” classifier

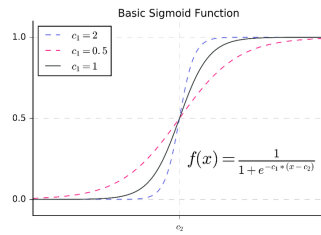


Figure 7: Plot of  $P(y = 1|x) = \frac{1}{1+e^{-z}}$ ,  $z = w_1 x$ . As  $w_1 \rightarrow \infty$  the logistic model becomes a “hard” rule.

### Logistic classifier properties (1)

- Class probabilities depend on distance from separating hyperplane
- Points far from separating hyperplane have probability  $\approx 0$  or  $\approx 1$
- When  $\|\mathbf{w}\|$  is larger, class probabilities go towards extremes (0,1) more quickly

### Logistic classifier properties (2)

- Unlike linear regression, weights do *not* correspond to change in output associated with one-unit change in input.
- Sign of weight *does* tell us about relationship between a given feature and target variable.

### Logistic regression - illustration

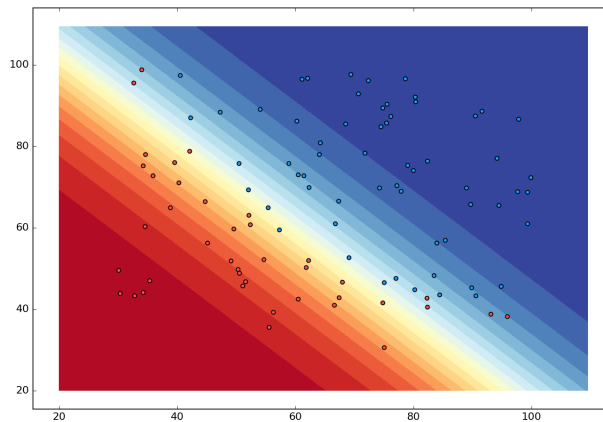


Figure 8: Logistic regression, illustrated with contour plot.

### Multi-class logistic regression

Suppose  $y \in 1, \dots, K$ . We use:

- $\mathbf{W} \in R^{K \times d}$  (parameter matrix)
- $\mathbf{z} = \mathbf{W}\mathbf{x}$  ( $K$  linear functions)

Assume we have stacked a 1s column so that the intercept is rolled into the parameter matrix.

### Softmax function

$$g_k(\mathbf{z}) = \frac{e^{z_k}}{\sum_{\ell=1}^K e^{z_\ell}}$$

- Takes as input a vector of  $K$  numbers
- Outputs  $K$  probabilities proportional to the exponentials of the input numbers.

### Softmax function as a PMF

Acts like a probability mass function:

- $g_k(\mathbf{z}) \in [0, 1]$  for each  $k$
- $\sum_{k=1}^K g_k(\mathbf{z}) = 1$
- larger input corresponds to larger “probability”

### Softmax function for multi-class logistic regression (1)

Class probabilities are given by

$$P(y = k|\mathbf{x}) = \frac{e^{z_k}}{\sum_{\ell=1}^K e^{z_\ell}}$$

### Softmax function for multi-class logistic regression (2)

When  $z_k \gg z_\ell$  for all  $\ell \neq k$ :

- $g_k(\mathbf{z}) \approx 1$
- $g_\ell(\mathbf{z}) \approx 0$  for all  $\ell \neq k$

Assign highest probability to class  $k$  when  $z_k$  is largest.



## Fitting logistic regression model

We know that to fit weights, we need

- a loss function,
- and a training algorithm to find the weights that minimize the loss function.

### Learning logistic model parameters

Weights  $\mathbf{W}$  are the unknown **model parameters**:

$$\mathbf{z} = \mathbf{W}\mathbf{x}, \mathbf{W} \in R^{K \times d}$$

$$P(y = k|\mathbf{x}) = g_k(\mathbf{z}) = g_k(\mathbf{W}\mathbf{x})$$

Given training data  $(\mathbf{x}_i, y_i), i = 1, \dots, n$ , we must learn  $\mathbf{W}$ .

### Maximum likelihood estimation (1)

Let  $P(\mathbf{y}|\mathbf{X}, \mathbf{W})$  be the probability of observing class labels  $\mathbf{y} = (y_1, \dots, y_n)^T$  given inputs  $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)^T$  and weights  $\mathbf{W}$ .

The **maximum likelihood estimate** is

$$\hat{\mathbf{W}} = \underset{\mathbf{W}}{\operatorname{argmax}} P(\mathbf{y}|\mathbf{X}, \mathbf{W})$$

It is the estimate of parameters for which these observations are most likely.

### Maximum likelihood estimation (2)

Assume outputs  $y_i$  are independent of one another,

$$P(\mathbf{y}|\mathbf{X}, \mathbf{W}) = \prod_{i=1}^n P(y_i|\mathbf{x}_i, \mathbf{W})$$

We take the log of both sides, because then the product turns into a sum...

### Maximum likelihood estimation (3)

Define the **negative log likelihood**:

$$\begin{aligned} L(\mathbf{W}) &= -\ln P(\mathbf{y}|\mathbf{X}, \mathbf{W}) \\ &= -\sum_{i=1}^n \ln P(y_i|\mathbf{x}_i, \mathbf{W}) \end{aligned}$$

(the term in the sum is also called cross-entropy)

Note that maximizing the likelihood is the same as minimizing the negative log likelihood.

#### Maximum likelihood estimation (4)

Now we can re-write max likelihood estimator with a loss function to minimize:

$$\hat{\mathbf{W}} = \underset{\mathbf{W}}{\operatorname{argmax}} P(\mathbf{y}|\mathbf{X}, \mathbf{W}) = \underset{\mathbf{W}}{\operatorname{argmin}} L(\mathbf{W})$$

The next step will be to plug in our sigmoid function.

#### Binary cross-entropy loss (1)

For binary classification with class labels 0, 1:

$$\begin{aligned} \ln P(y_i|\mathbf{x}_i, \mathbf{w}) &= y_i \ln P(y_i = 1|\mathbf{x}_i, \mathbf{w}) + (1 - y_i) \ln P(y_i = 0|\mathbf{x}_i, \mathbf{w}) \\ &= y_i \ln \sigma(z_i) + (1 - y_i) \ln(1 - \sigma(z_i)) \\ &= y_i (\ln \sigma(z_i) - \ln \sigma(-z_i)) + \ln \sigma(-z_i) \\ &= y_i \ln \frac{\sigma(z_i)}{\sigma(-z_i)} + \ln \sigma(-z_i) \\ &= y_i \ln \frac{1 + e^{z_i}}{1 + e^{-z_i}} + \ln \sigma(-z_i) \\ &= y_i \ln \frac{e^{z_i}(e^{-z_i} + 1)}{1 + e^{-z_i}} + \ln \sigma(-z_i) \\ &= y_i z_i - \ln(1 + e^{z_i}) \end{aligned} \tag{1}$$

(Note:  $\sigma(-z) = 1 - \sigma(z)$ )

#### Binary cross-entropy loss (2)

Binary cross-entropy loss function (negative log likelihood):

$$\sum_{i=1}^n \ln(1 + e^{z_i}) - y_i z_i$$

#### Cross-entropy loss for multi-class classification (1)

Define “one-hot” vector - for a sample from class  $k$ , all entries in the vector are 0 except for the  $k$ th entry which is 1:

$$r_{ik} = \begin{cases} 1 & y_i = k \\ 0 & y_i \neq k \end{cases}$$

$$i = 1, \dots, n, \quad k = 1, \dots, K$$

## Cross-entropy loss for multi-class classification (2)

Then,

$$\ln P(y_i | \mathbf{x}_i, \mathbf{W}) = \sum_{k=1}^K r_{ik} \ln P(y_i = k | \mathbf{x}_i, \mathbf{W})$$

Cross-entropy loss function is

$$\sum_{i=1}^n \left[ \ln \left( \sum_k e^{z_{ik}} \right) - \sum_k z_{ik} r_{ik} \right]$$

### Minimizing cross-entropy loss

To minimize, we would take the partial derivative:

$$\frac{\partial L(W)}{\partial W_{kj}} = 0$$

for all  $W_{kj}$

**But**, there is no closed-form expression - can only estimate weights via numerical optimization (e.g. gradient descent)

## Transformations, bias, variance

- Can use basis functions to map problem to transformed feature space (if “natural” decision boundary is non-linear)
- Variance increases with  $d$  and decreases with  $n$
- Can add a regularization penalty to loss function

## “Recipe” for logistic regression (binary classifier)

- Choose a **model**:

$$P(y = 1|x, w) = \sigma \left( w_0 + \sum_{i=1}^d w_i x_i \right)$$

$$\hat{y} = \begin{cases} 1, & P(y = 1|\mathbf{x}) \geq t \\ 0, & P(y = 1|\mathbf{x}) < t \end{cases}$$

- Get **data** - for supervised learning, we need **labeled** examples:  $(x_i, y_i), i = 1, 2, \dots, n$
- Choose a **loss function** that will measure how well model fits data: binary cross-entropy

$$\sum_{i=1}^n \ln(1 + e^{z_i}) - y_i z_i$$

- Find model **parameters** that minimize loss: use numerical optimization to find weight vector  $w$
- Use model to **predict**  $\hat{y}$  for new, unlabeled samples.

## “Recipe” for logistic regression (multi-class classifier)

- Choose a **model**: find probability of belonging to each class, then choose the class for which the probability is highest.

$$P(y = k|\mathbf{x}) = \frac{e^{z_k}}{\sum_{\ell=1}^K e^{z_\ell}} \text{ where } \mathbf{z} = \mathbf{W}\mathbf{x}$$

- Get **data** - for supervised learning, we need **labeled** examples:  $(x_i, y_i), i = 1, 2, \dots, n$
- Choose a **loss function** that will measure how well model fits data: cross-entropy

$$\sum_{i=1}^n \left[ \ln \left( \sum_k e^{z_{ik}} \right) - \sum_k z_{ik} r_{ik} \right] \text{ where}$$

$$r_{ik} = \begin{cases} 1 & y_i = k \\ 0 & y_i \neq k \end{cases}$$

- Find model **parameters** that minimize loss: use numerical optimization to find weight vector  $w$
- Use model to **predict**  $\hat{y}$  for new, unlabeled samples.

## Naive Bayes classifier

A quick look at a different type of model!

### Probabilistic models (1)

For logistic regression, minimizing the cross-entropy loss finds the parameters for which

$$P(\mathbf{y}|\mathbf{X}, \mathbf{W})$$

is maximized.

### Probabilistic models (2)

For linear regression, assuming normally distributed stochastic error, minimizing the **squared error** loss finds the parameters for which

$$P(\mathbf{y}|\mathbf{X}, \mathbf{w})$$

is maximized.

Surprise! We've been doing maximum likelihood estimation all along.

### Probabilistic models (3)

ML models that try to

- get a good fit for  $P(y|X)$ : **discriminative** models.
- fit  $P(X, y)$  or  $P(X|y)P(y)$ : **generative** models.

Linear regression and logistic regression are both considered discriminative models; they say “given that we have this data, what's the most likely label?” (e.g. learning a mapping from an input to a target variable).

Generative models try to learn “what does data for each class look like” and then apply Bayes rule.

### Bayes rule

For a sample  $\mathbf{x}_i$ ,  $y_k$  is label of class  $k$ :

$$P(y_k|\mathbf{x}_i) = \frac{P(\mathbf{x}_i|y_k)P(y_k)}{P(\mathbf{x}_i)}$$

- $P(y_k|\mathbf{x}_i)$ : posterior probability. “What is the probability that this sample belongs to class  $k$ , given its observed feature values are  $\mathbf{x}_i$ ?”
- $P(\mathbf{x}_i|y_k)$ : conditional probability: “What is the probability of observing the feature values  $\mathbf{x}_i$  in a sample, given that the sample belongs to class  $k$ ?”
- $P(y_k)$ : prior probability
- $P(\mathbf{x}_i)$ : evidence

### Class conditional probability (1)

“Naive” assumption conditional independence of features:

$$\begin{aligned} P(\mathbf{x}_i|y_k) &= P(x_{i,1}|y_k)P(x_{i,2}|y_k) \dots P(x_{i,d}|y_k) \\ &= \prod_{j=1}^d P(x_{i,j}|y_k) \end{aligned}$$

This is called “naive” because this assumption is probably not true in most realistic situations.

(But the classifier may still work OK!)

Also assumes samples are i.i.d.

### Class conditional probability (2)

Example: for binary/categorical features, we could compute

$$\hat{P}(x_{i,j}|y_k) = \frac{N_{x_{i,j},y_k}}{N_{y_k}}$$

- $N_{x_{i,j},y_k}$  is the number of samples belonging to class  $k$  that have feature  $j$ .
- $N_{y_k}$  is the total number of samples belonging to class  $k$ .

Example: for cat photo classifier,

$$\hat{P}(\mathbf{x}_i = [\text{has tail, has pointy ears, has fur, purrs when petted, likes to eat fish}]|y = \text{cat})$$

$$\begin{aligned} \rightarrow P\left(\frac{N_{\text{tail, cat}}}{N_{\text{cat}}}\right)P\left(\frac{N_{\text{pointy ears, cat}}}{N_{\text{cat}}}\right)P\left(\frac{N_{\text{fur, cat}}}{N_{\text{cat}}}\right)P\left(\frac{N_{\text{purrs, cat}}}{N_{\text{cat}}}\right)P\left(\frac{N_{\text{eats fish, cat}}}{N_{\text{cat}}}\right) \\ \rightarrow \frac{20}{20} \frac{18}{20} \frac{17}{20} \frac{5}{20} \frac{15}{20} \end{aligned}$$

### Prior probability

Can estimate prior probability as

$$\hat{P}(y_k) = \frac{N_{y_k}}{N}$$

Prior probabilities: probability of encountering a particular class  $k$ .

Example:  $\frac{20}{1500}$  photos are cats.

### Evidence

We don't actually need  $P(\mathbf{x}_i)$  to make decisions, since it is the same for every class.

## Naive bayes decision boundary

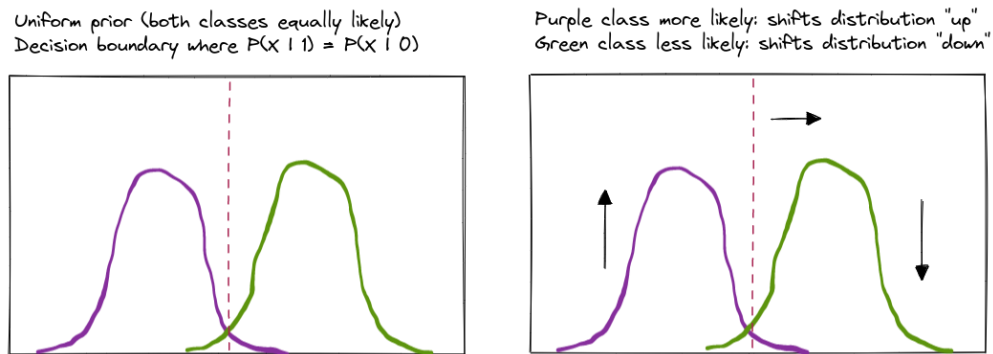


Figure 9: Naive bayes decision boundary.

## Why generative model?

The generative model solves a more general problem than the discriminative model!

But, only the generative model can be used to **generate** new samples similar to the training data.

Example: "generate a new sample that is probably a cat."