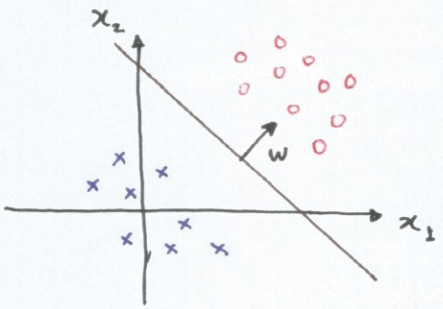
LOGISTIC REGRESSION

TWO-CLASS LINEAR CLASSIFIERS

→ For each class, there is a region of feature space in which the classifier selects one class over the other

↳ The boundary of this region is the DECISION BOUNDARY

→ In linear classifiers, the DB is a straight line



→ In two-class linear classifier, we learn a fn

$$F(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T\mathbf{x} + w_0$$ (w_0: Bias)

that represents how aligned the instance is w/ $y = 1$

→ $\mathbf{w}$ are parameters of the classifier tht. we learn from data

→ To do classification of an input $\mathbf{x}$:

$$\mathbf{x} \mapsto (y = 1) \quad \text{if} \quad F(\mathbf{x}, \mathbf{w}) > 0$$

↳ The DB in this case is $\{\mathbf{x} \mid \mathbf{w}^T\mathbf{x} + w_0 = 0\}$

↳ normal vector to the surface

TWO-CLASS DISCRIMINATION

→ Consider a two-class case $y \in \{0, 1\}$, $\mathbf{x} = (1, x_1, \ldots, x_d)$ and $\mathbf{w} = (w_0, \ldots, w_d)$

→ We want a linear, probabilistic model:

$$P(y = 1 \mid \mathbf{x}) = f(\mathbf{w}^T\mathbf{x})$$

↳ will 'squash' the real line into $[0, 1]$

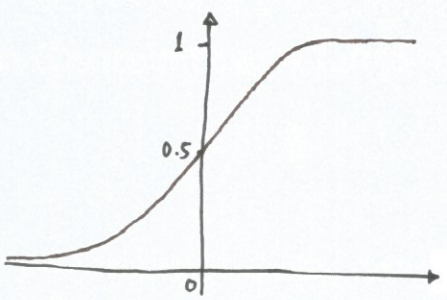
THE LOGISTIC FN

→

$$f(z) = \sigma(z) = \frac{1}{1 + e^{-z}}$$

→ As z goes from $-\infty$ to ∞, f goes from 0 to 1

→ Has a 'sigmoid' shape



→ Purpose of a sigmoid fn:

- To produce values that can be used as probabilities
- To produce values that sum up to 1

LINEAR WEIGHTS

→ Linear weights + logistic fn = LOGISTIC REGRESSION

→ We model the class probabilities as:

$$p(y=1 \mid x) = \sigma\left(\sum_{j=0}^{D} w_j x_j\right) = \sigma(w^T x)$$

→ $\sigma(z) = 0.5$ when $z = 0$

Hence, the **DB is given by $w^T x = 0$.**

→ DB is a M−1 hyperplane for a M dimensional problem.

→ We write $\tilde{w} = (w_1, w_2, \ldots, w_d)$

DIRECTION
of vector $\tilde{w}$ affects the angle of the hyperplane, which is ⊥ to $\tilde{w}$

MAGNITUDE
of vector $\tilde{w}$ affects how certain the classifications are:
- Small $\tilde{w}$: Most of the probabilities within the region of the DB will be near to 0.5
- Large $\tilde{w}$: Probabilities in the same region will be close to 0 or 1

→ The bias parameter w_0 shifts the position of the hyperplane, but does not alter the angle.

LEARNING THE PARAMETERS

→ We want to set the parameters **w** using training data

→ Assume data is independent & identically distributed.
Call the dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$

→ The likelihood is:

$$p(D \mid w) = \prod_{i=1}^{n} p(y = y_i \mid x_i, w) = \prod_{i=1}^{n} p(y=1 \mid x_i, w)^{y_i} \left(1 - p(y=1 \mid x_i, w)\right)^{1-y_i}$$

Hence the log likelihood is: (Taking log will make things easier bc. it converts the product into summation)

$$L(w) = \sum_{i=1}^{n} y_i \log \sigma(w^T x_i) + (1 - y_i) \log\left(1 - \sigma(w^T x_i)\right)$$

→ has a unique optimum
→ It is convex

To maximize, take gradient

$$\frac{\partial L}{\partial w_j} = \sum_{i=1}^{n} \left(y_i - \sigma(w^T x_i)\right) x_{ij}$$

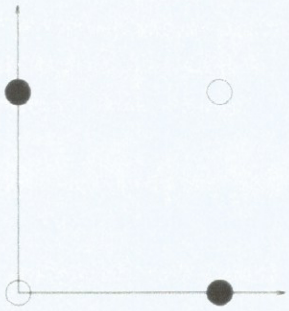
But unlike linear regression, you cannot maximize $L(w)$ explicitly.
We need to use a numerical optimisation method.

BASIS FNS

→ A problem is linearly separable if we can find weights so that

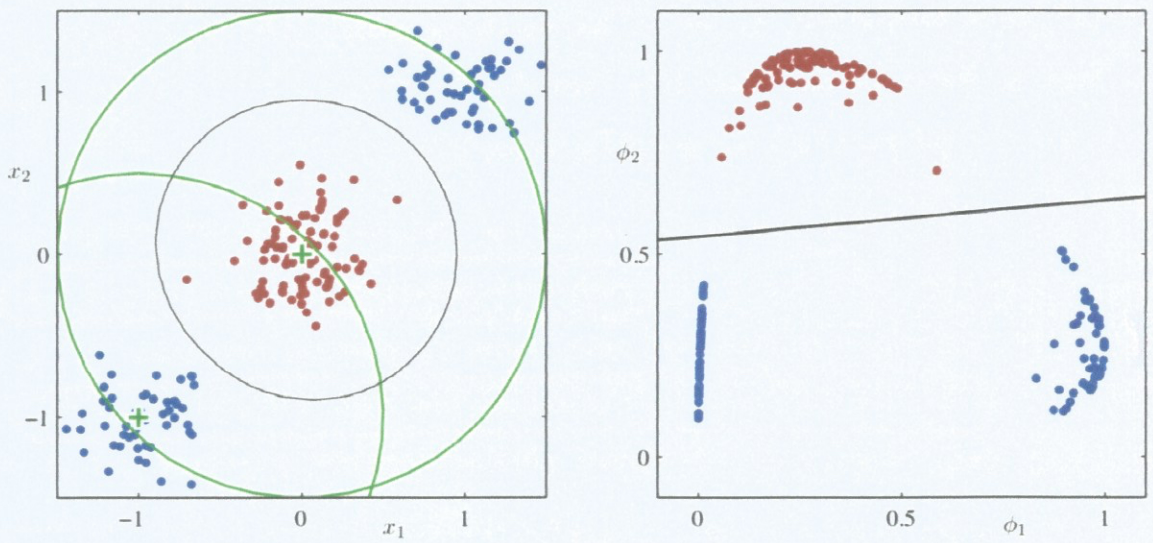
- $\tilde{\mathbf{w}}^T\mathbf{x} + w_0 > 0$ for all positive cases (where $y = 1$), and
- $\tilde{\mathbf{w}}^T\mathbf{x} + w_0 \leq 0$ for all negative cases (where $y = 0$)

i.e. XOR



XOR becomes linearly separable if we apply a non-linear tranformation $\phi(\mathbf{x})$ of the input — what is one?

GAUSSIAN BASIS FN



Using two Gaussian basis functions $\phi_1(\mathbf{x})$ and $\phi_2(\mathbf{x})$

DISCRIMINATIVE MODEL

→ Logistic regression is a <u>discriminative</u> approach

↳ Estimate parameters of $P(Y|X)$ directly from training data

→ Recall that Naive Bayes is a generative approach

↳ Estimate parameters of $p(x|y)$ and $p(y)$ from training data

↳ Use Bayes rule to calculate $p(y|x)$

ADVANTAGE

- DISCRIMINATIVE → You don't have to waste time modelling the prob. of input data $p(x)$ since we're always given the input
- GENERATIVE → Can be good w/ missing data (how Naive Bayes handles missing data)
  - ↳ Good for detecting outliers
  - ↳ Sometimes you really do want to generate the input

→ Generative classifiers like Naive Bayes can be linear as well in particular circumst.:

1) Gaussian data w/ equal covariance

- If $p(x|y=1) \sim N(\mu_1, \Sigma)$ and $p(x|y=0) \sim N(\mu_2, \Sigma)$ then $p(y=1|x) = \sigma(\tilde{w}^T x + w_0)$

2) Binary data

- Let each component x_j be a Bernoulli variable, i.e. $x_j \in \{0, 1\}$

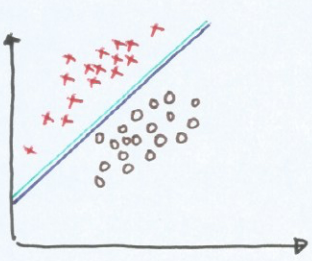
Then a Naive Bayes classifier has the form

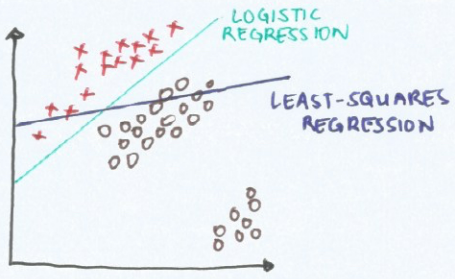
$$p(y=1|x) = \sigma(\tilde{w}^T x + w_0)$$

LEAST-SQUARES CLASSIFICATION

→ Logistic regression is more complicated algorithmically than linear regression

→ So why not just use linear regression w/ 0/1 targets?





MULTICLASS CLASSIFICATION

→ If we are building a classifier for $K > 2$ classes, we will build K functions, each parameterised by a set of linear weights, each of which outputs a real value (They don't resemble logistic regression models; no sigmoid)

→ Then, use the **softmax fn** to combine the parameters produced by the K models (to create one model)

$$p(y = k \mid x) = \frac{e^{w_k^T x}}{\sum_{j=1}^{C} e^{w_j^T x}}$$

→ The softmax fn takes the place of the sigmoid for logistic regression

It normalises the exponent of its input across the K classes.

QUIZ QNs

1) Logistic regression is a method for:

a. Classification
b. regression

→ The regression part comes from the fact that it is predicting a real no., which is interpreted as the probability of being in the (+)ve class

2) How many classes can a logistic regression model choose between:

a. infinite no.
b. many
c. two
d. three

→ There is, however, an extension for multiclass problems

3) In a logistic regression model, we learn a model of:

a. A straight line decision boundary
b. A decision boundary which can be a straight line but doesn't have to be
c. The probability that an instance is in the (+)ve class
d. The likelihood of the training data

Boundary For vector w

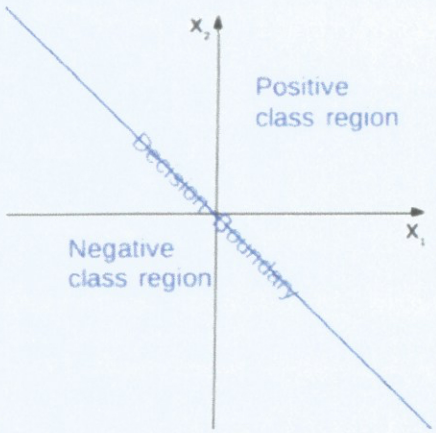
We have two two-dimensional datasets D_1 and D_2. We use logistic regression to learn decision boundaries, and find that they are $bfw_1 = (0,1,1)$ and $bfw_2 = (0,2,2)$. Is the following statement true or false?

For any new datapoint, the two models will classify the datapoint into the same class.

✓ True
False

W_2 is a multiple of W_1, so they represent the same line.

For dataset D_2 in the previous question, which of the following diagrams indicates the orientation of the decision boundary and how it distinguishes between positive and negative instances?



→ The DB is a line for which the probability is 0.5

→ Looking at the sigmoid fn, the probability is 0.5 when the value passed to the sigmoid fn is zero

→ In this case, the value passed to the sigmoid is:

$$w_0 + w^T x = 0 + 2x_1 + 2x_2$$

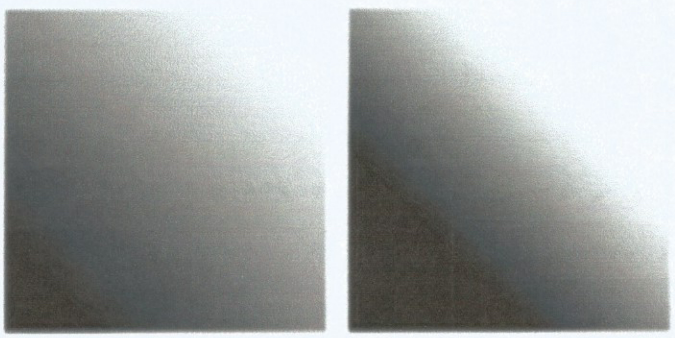
Solving this gives

$$x_1 = -x_2$$

→ To find the (+)ve class region, we want to increase the value passed to the sigmoid, so I need to make $2x_1 + 2x_2$ bigger.

→ This involves moving in the direction of the vector (1, 1)
↳ towards the top right.

Assume the same problem setting as the previous two questions, and label the model for D_1 with M_1, and for D_2 with M_2. In the following diagrams set in X_1, X_2 space with the horizontal axis being X_1, white represents certainty of one class, black represents certainty of the other class, and greys represent degree of uncertainty. You can see there is a smooth transition from certainty of one class to the other. Which pair of diagrams represents better the change in probability from one class to the other as the new datapoint moves away from and towards the decision boundary?



M1 M2

Model M_2 has double the param. values of model M_1.

↳ This means that a small change in (x_1, x_2) results in twice as large a change in $w^T x$ in model M_2 as in model M_1, and hence a larger change in probability

↳ As $W_2^T x$ is larger, we should expect M_2 to have a more rapid transition between 0 and 1