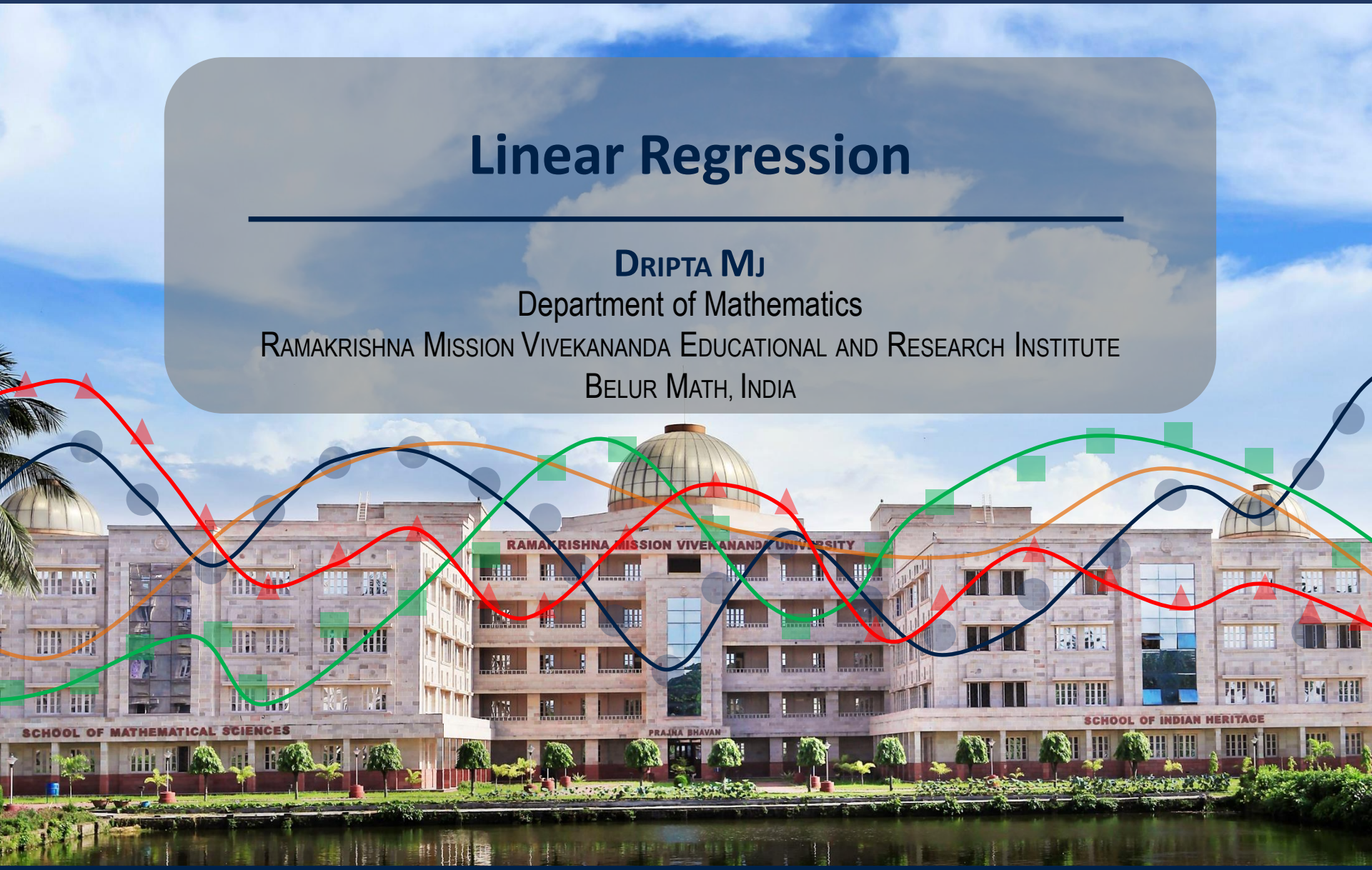
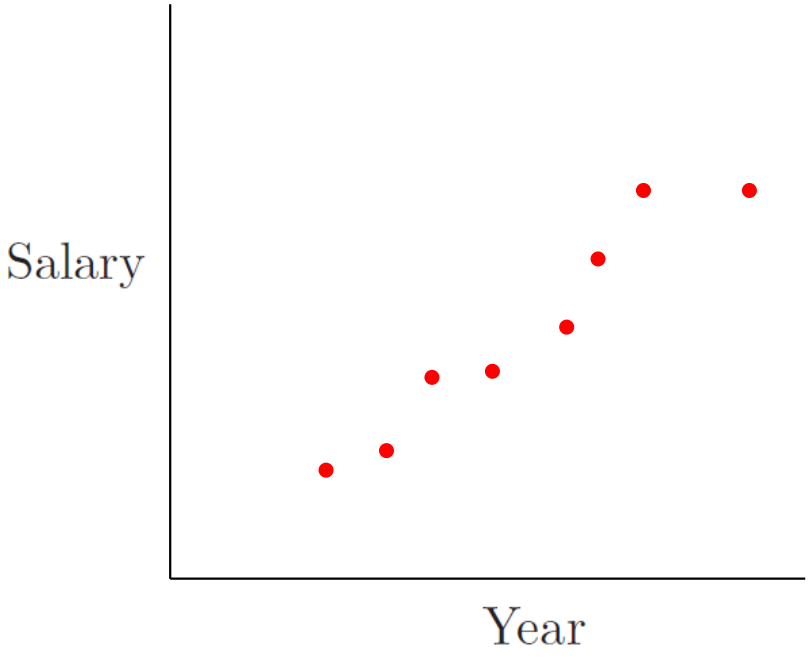
Linear Regression

Dripta Mj
Department of Mathematics
Ramakrishna Mission Vivekananda Educational and Research Institute
Belur Math, India



Introduction

- Variation of salary with time:



Notation

- Given a data set of N points.
- Input data comprise D features, suppose they are $x_1, x_2, ..., x_D$.
- Representation of the ith input data point:

$$\mathbf{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}, ..., x_D^{(i)}]^{\mathrm{T}}$$

- Set of input data points:

$$\mathbf{X} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ...\mathbf{x}^{(N)}\}$$

- Set of outputs for the N data points:

$$\mathbf{y} = \{y^{(1)}, y^{(2)}, ..., y^{(N)}\}$$

Linear Regression

- Input variables can be written in vectorial form as:

$$\mathbf{x} = [x_1, x_2, ..., x_D]^T$$

- Prediction: Linear combination of input variables

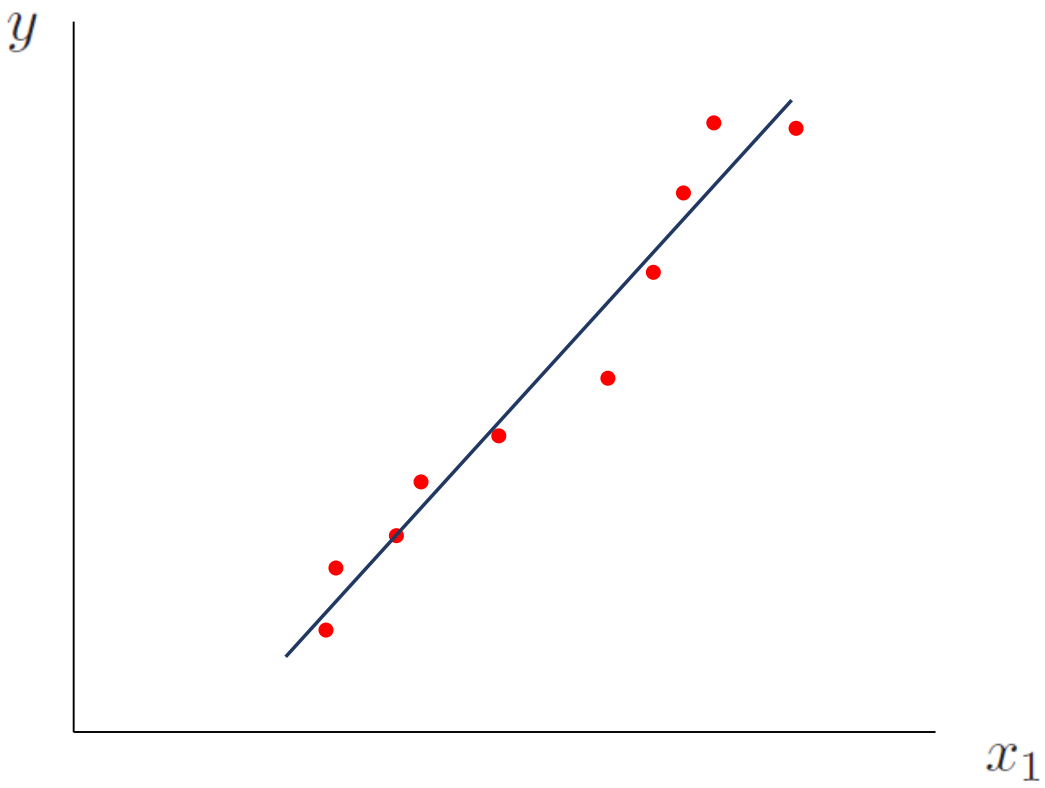
$$z(\mathbf{x}, p_0, p_1, ..., p_D) = p_0 + p_1 x_1 + p_2 x_2 + ... + p_D x_D$$

- Vectorial representation of weights:

$$\mathbf{p} = [p_0, p_1, p_2, ..., p_D]^T$$

Simplest case

- One input variable: x_1



$$z = p_0 + p_1 x_1$$

Using basis functions

- Output: Linear combination of functions of input variables

$$z(\mathbf{x}, p) = p_0 + \sum_{i=1}^{Q} p_i f_i(\mathbf{x})$$

where $f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_Q(\mathbf{x})$ are the basis functions.

- Taking $f_0 = 1$, can write

$$z(\mathbf{x}, p) = \sum_{i=0}^{Q} p_i f_i(\mathbf{x})$$

- Vectorial representation:

$$z(\mathbf{x}, \mathbf{p}) = \mathbf{p}^T \mathbf{f}(\mathbf{x})$$

where $\mathbf{p} = [p_0, p_1, ..., p_Q]^{\mathrm{T}}$ and $\mathbf{f}(\mathbf{x}) = [f_0(\mathbf{x}), f_1(\mathbf{x}), ..., f_Q(\mathbf{x})]^{\mathrm{T}}$

Basis functions

- Polynomial basis functions (with single input x_1) have the form:

$$f_0(x_1) = 1, \qquad f_1(x_1) = x_1, \qquad f_2(x_1) = x_1^2, \quad \ldots. \quad f_Q(x_1) = x_1^Q,$$

and therefore

$$\mathbf{f}(x_1) = \left[1,\ x_1,\ x_1^2, \ldots.,\ x_1^Q\right]^{\mathrm{T}}.$$

- More complex functions can be generated by increasing the degree of the polynomial Q.

- Some other popular choices of basis functions:
  - Gaussian basis functions

Loss function

- Dataset comprise N data points.
- Outputs at the training locations: $\mathbf{y} = \{y^{(1)}, y^{(2)}, ..., y^{(N)}\}$.
- Predictions at the training locations: $\mathbf{z} = \{z^{(1)}, z^{(2)}, ..., z^{(N)}\}$.
- Sum of squared errors:

$$\begin{aligned} \text{SE} &= \sum_{n=1}^{N} \left(y^{(n)} - z^{(n)}\right)^2 \\ &= \sum_{n=1}^{N} \left(y^{(n)} - \mathbf{p}^T \mathbf{f}(\mathbf{x}^{(n)})\right)^2 \end{aligned}$$

- Mean squared error $\text{MSE} = \dfrac{\text{SE}}{N}$
- Often the error function is written in the following form that is more amenable to differentiation

$$E = \frac{1}{2} \sum_{n=1}^{N} \left(y^{(n)} - \mathbf{p}^T \mathbf{f}(\mathbf{x}^{(n)})\right)^2$$

Parameter values

- Optimal weights – Setting the gradient of $E(\mathbf{p})$ to zero:

$$\nabla E(\mathbf{p}) = 0$$

which yields

$$\sum_{n=1}^{N} y^{(n)} \mathbf{f}(\mathbf{x}^{(n)})^{\mathrm{T}} - \mathbf{p}^{\mathrm{T}} \left(\sum_{n=1}^{N} \mathbf{f}(\mathbf{x}^{(n)}) \mathbf{f}(\mathbf{x}^{(n)})^{\mathrm{T}} \right) = 0$$

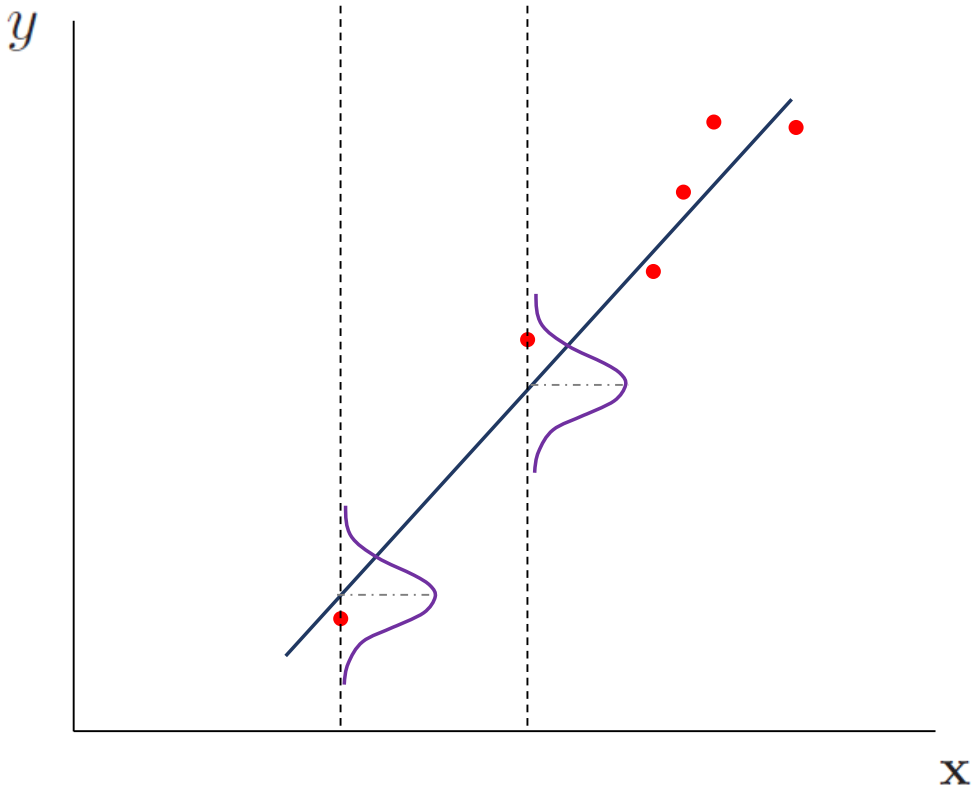
Let

$$\mathcal{F} = \begin{bmatrix} \mathbf{f}(\mathbf{x}^{(1)})^{\mathrm{T}} \\ \mathbf{f}(\mathbf{x}^{(2)})^{\mathrm{T}} \\ \cdot \\ \cdot \\ \mathbf{f}(\mathbf{x}^{(N)})^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} f_0(\mathbf{x}^{(1)}) & f_1(\mathbf{x}^{(1)}) & \cdots & f_Q(\mathbf{x}^{(1)}) \\ f_0(\mathbf{x}^{(2)}) & f_1(\mathbf{x}^{(2)}) & \cdots & f_Q(\mathbf{x}^{(2)}) \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ f_0(\mathbf{x}^{(N)}) & f_1(\mathbf{x}^{(N)}) & \cdots & f_Q(\mathbf{x}^{(N)}) \end{bmatrix}$$

- Finally, the value of the weights $\mathbf{p}$ are obtained as:

$$\mathbf{p} = (\mathcal{F}^{\mathrm{T}} \mathcal{F})^{-1} \mathcal{F}^{\mathrm{T}} \mathbf{y}$$

Probabilistic approach



- Assumption: Gaussian noise model

$$y^{(n)} \sim \mathcal{N}(\mathbf{p}^{\mathrm{T}}\mathbf{x}^{(n)}, \sigma^2)$$

Maximum likelihood estimator

- Training data comprise N data points:

$$\mathbf{X} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(N)}\}, \qquad \mathbf{y} = \{y^{(1)}, y^{(2)}, ..., y^{(N)}\}$$

- Likelihood function

$$p(\mathbf{y}|\mathbf{X}, \mathbf{p}, \sigma^2) = \prod_{n=1}^{N} \mathcal{N}(y^{(n)}|\mathbf{p}^T\mathbf{f}(\mathbf{x}^{(n)}), \sigma^2)$$

- Taking log on both sides:

$$\ln p(\mathbf{y}|\mathbf{X}, \mathbf{p}, \sigma^2) = \sum_{n=1}^{N} \ln \mathcal{N}(y^{(n)}|\mathbf{p}^T\mathbf{f}(\mathbf{x}^{(n)}), \sigma^2)$$

$$= -\frac{N}{2}\ln(2\pi) - \frac{N}{2}\ln(\sigma^2) - \frac{1}{2\sigma^2}\sum_{n=1}^{N}\left(y^{(n)} - \mathbf{p}^T\mathbf{f}(\mathbf{x}^{(n)})\right)^2$$

- For a linear model with conditional Gaussian noise distribution, maximizing likelihood with respect to weights is equivalent to minimizing the sum-of-squared error.

Maximum likelihood estimator

- Weights for maximum likelihood – setting the gradient of log likelihood to zero:

$$\nabla \ln p(\mathbf{y}|\mathbf{X}, \mathbf{p}, \sigma^2) = \sum_{n=1}^{N} \left(y^{(n)} - \mathbf{p}^{\mathrm{T}} \mathbf{f}(\mathbf{x}^{(n)})\right) \mathbf{f}(\mathbf{x}^{(n)})^{\mathrm{T}} = 0$$

finally yields

$$\mathbf{p} = (\mathcal{F}^{\mathrm{T}} \mathcal{F})^{-1} \mathcal{F}^{\mathrm{T}} \mathbf{y}$$

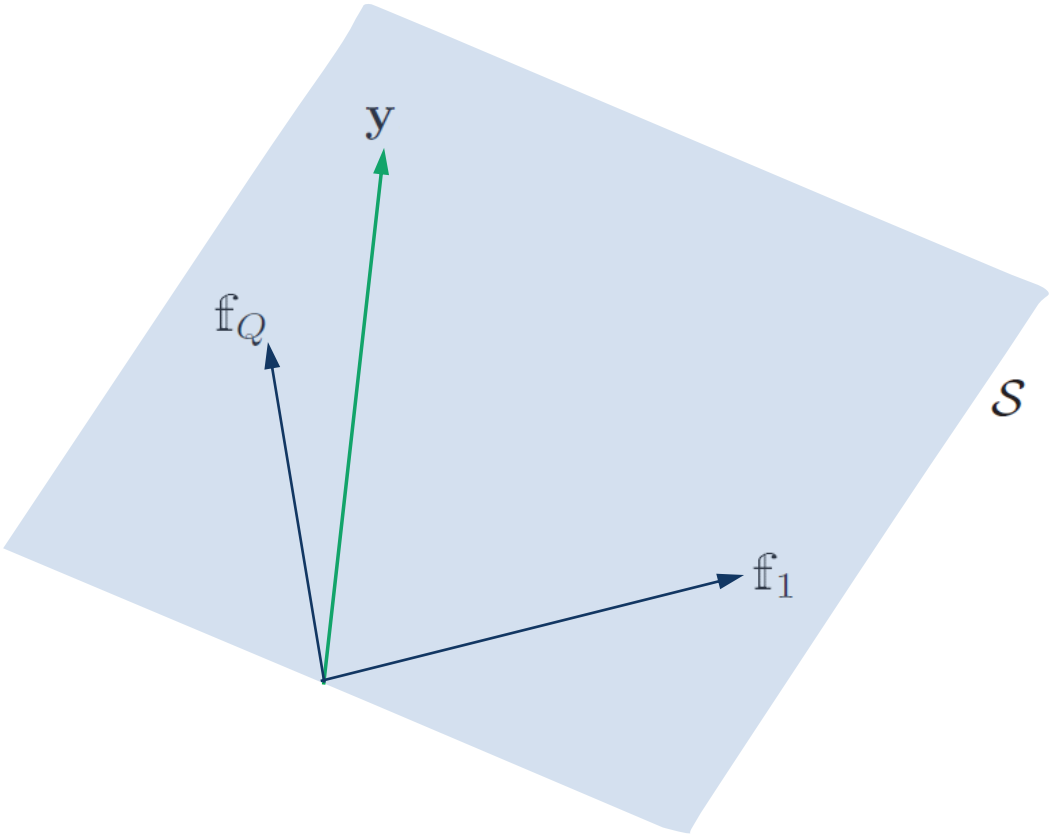
- The noise variance σ^2 can also determined by maximizing (log) likelihood with respect to σ^2, yielding:

$$\sigma^2 = \frac{1}{N} \sum_{n=1}^{N} \left(y^{(n)} - \mathbf{p}^{\mathrm{T}} \mathbf{f}(\mathbf{x}^{(n)})\right)^2$$

- This is the residual variance of the outputs around the predictions

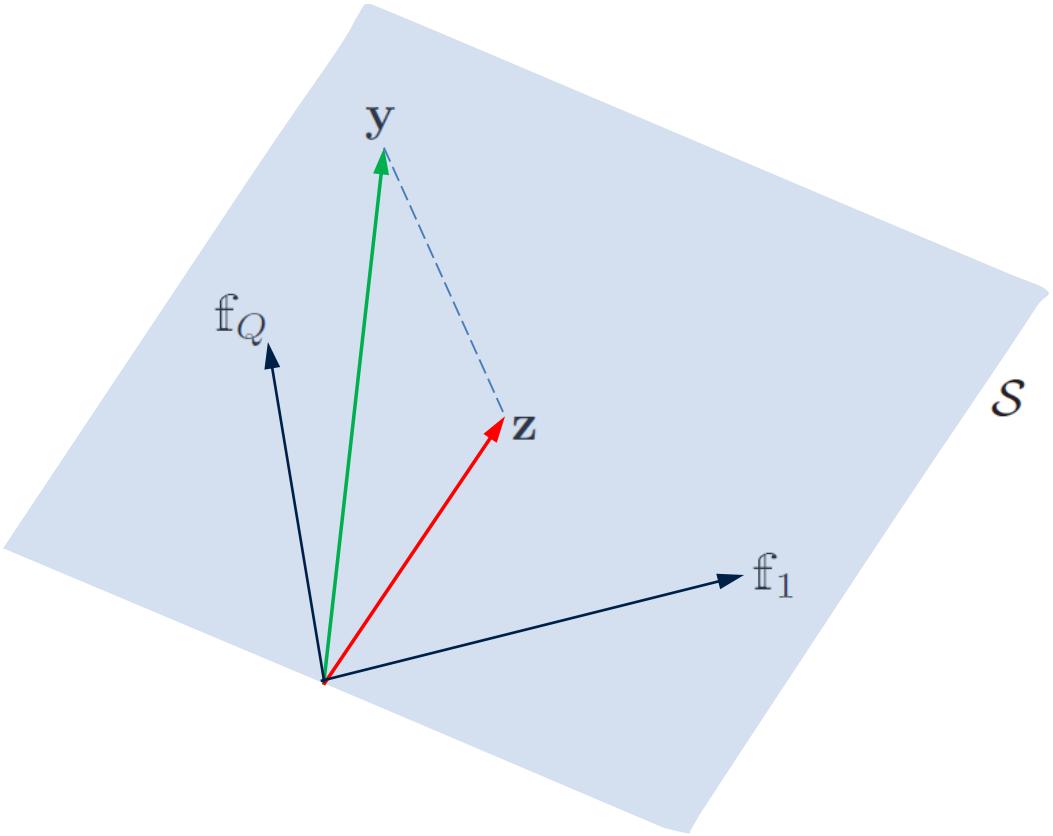
Geometrical interpretation

- Consider an N-dimensional space, whose axes indicate the N output values of the training dataset.
- Let $\mathbb{f}_j$ be the j-th column of $\mathcal{F}$, i.e. $\mathbb{f}_j = [f_j(\mathbf{x}^{(1)}), f_j(\mathbf{x}^{(2)}), .., f_j(\mathbf{x}^{(N)})]^{\mathrm{T}}$.
- In the N-dimensional space consider a subspace $\mathcal{S}$ spanned by Q basis vectors $\{\mathbb{f}_1, \mathbb{f}_2, .., \mathbb{f}_Q\}$.



Geometrical interpretation

- Let $\mathbf{z}$ be an arbitrary linear combination of the basis vectors.
- The vector $\mathbf{z}$ can lie anywhere in the subspace $\mathcal{S}$.
- The least squares algorithm yields solution $\mathbf{z}$ lying on the subspace $\mathcal{S}$ which is closest to the vector $\mathbf{y}$.



R^2

- R^2: Coefficient of determination
- Fraction of the total variation in the outputs that is accounted for by preforming regression of the outputs w.r.t.input variables.
- Mathematically R^2 is defined as

$$R^2 = 1 - \frac{S_R}{S_T}$$

- The total variation S_T is taken as the sum of squared deviations of each y value from the mean of y.
  - The mean $\overline{y}$ is the best guess when the values of the input variables are unknown.
  - The mean $\overline{y}$ can be computed as

$$\overline{y} = \frac{1}{N} \sum_{n=1}^{N} y^{(n)}$$

  - The total variation can be thought of as how good the predictions of y can be without the knowledge of $\mathbf{x}$.

R^2

- R^2: Coefficient of determination
- Fraction of the total variation in the outputs that is accounted for by preforming regression of the outputs w.r.t.input variables.
- Mathematically R^2 is defined as

$$R^2 = 1 - \frac{S_R}{S_T}$$
$$= 1 - \frac{\sum_{n=1}^{N} \left(z^{(n)} - y^{(n)}\right)^2}{\sum_{n=1}^{N} \left(y^{(n)} - \overline{y}\right)^2}$$

- The sum of square of residuals S_R is a measure of "unexplained variability" and is defined as

$$S_R = \sum_{n=1}^{N} \left(z^{(n)} - y^{(n)}\right)^2$$

- The difference between S_T and S_R can be considered as the "explained variability" i.e. the amount of variability in the outputs that is explained by performing regression on the inputs.

LOGISTIC REGRESSION

Introduction

- Binary classification where output y can take two values, e.g. $y \in \{0, 1\}$.
- Probabilistic model for binary classification.
- Discriminative model: Output y is modelled, while x is considered fixed.
- Can be considered as extension of linear regression to classification problems.
- Examples:
  - Spam classifier: Positive class if spam, negative class otherwise.
  - Classify cancer patients into high or low risk groups.

Binary classification

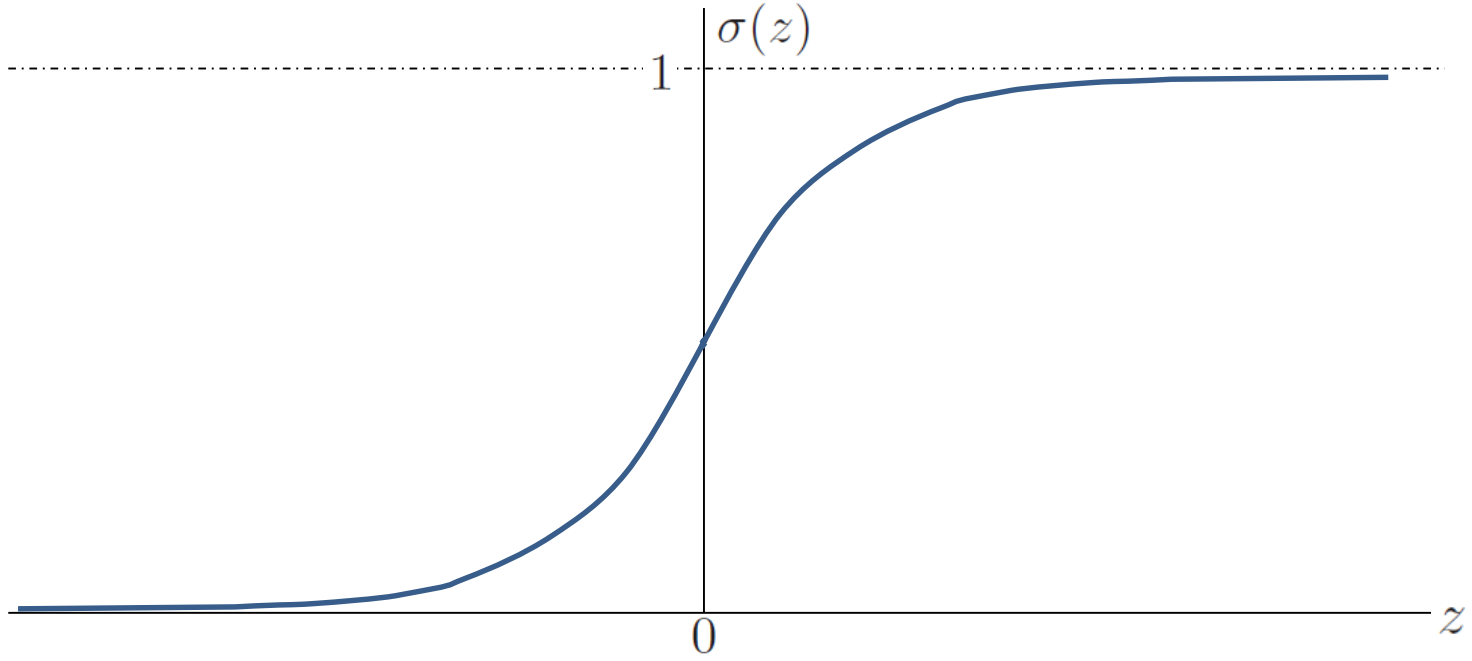
- Dataset: $\{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), ..., (\mathbf{x}^{(N)}, y^{(N)})\}$.
- Augmented vectorial representation of the inputs:

$$\mathbf{x} = \left[x_0, x_1, x_2, ..., x_D\right]^{\mathrm{T}}$$

  where $x_0 = 1$ and $x_1, x_2, ..., x_D$ are the features of the problem.
- Outputs: $y \in \{0, 1\}$.
- Idea: Use a nonlinear function that maps the outputs between 0 and 1.
- For an input $\mathbf{x}$, the weight vector $\mathbf{w} = [w_0, w_1, w_2, ... w_D]^{\mathrm{T}}$ is used to compute a score $\mathbf{w}^{\mathrm{T}}\mathbf{x}$. The nonlinear function then squashes the score between 0 and 1.
- A popular nonlinear function: Sigmoid.

Sigmoid function

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$



- Sigmoid is a smooth function of the inputs.
- Formulation enables gradient-based parameter learning.

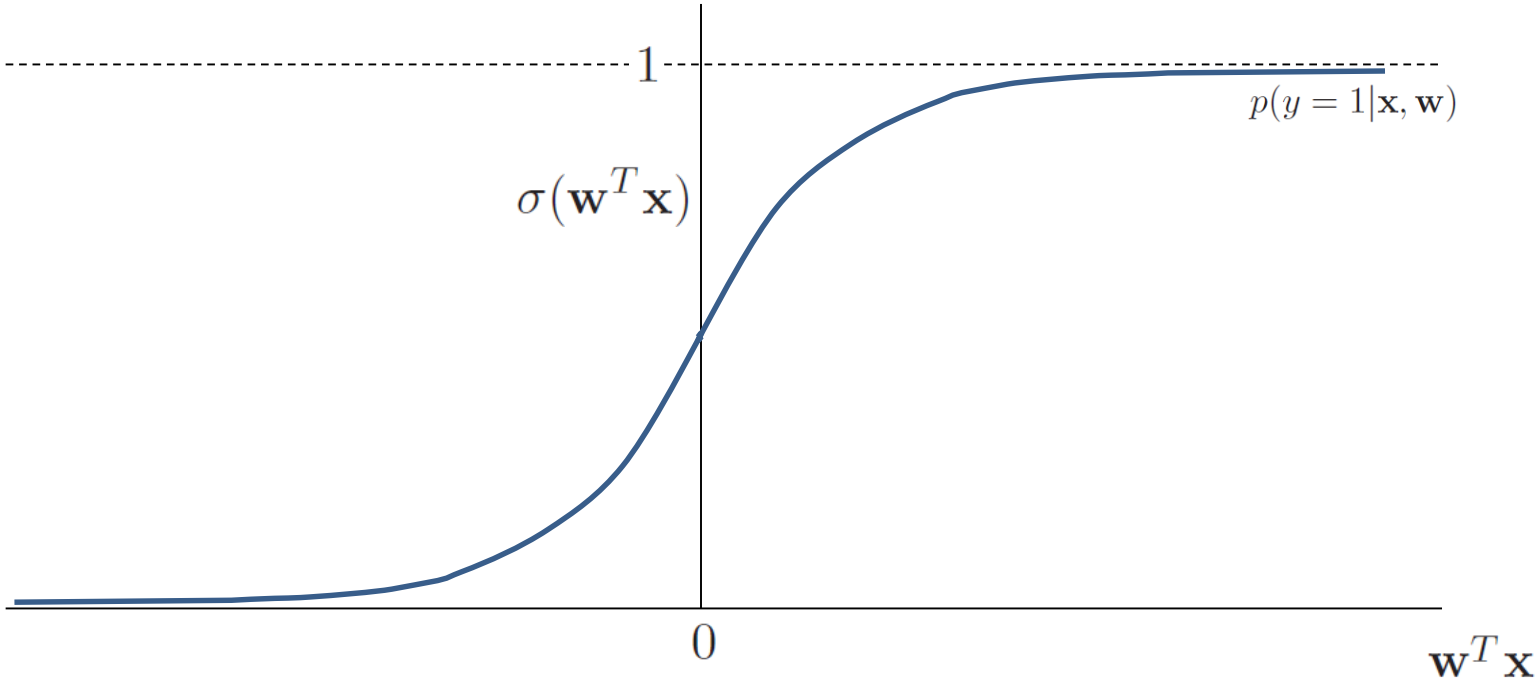
Probabilistic modelling

- Use sigmoid function to model probability of the class $y = 1$ (say).

$$p(y = 1|\mathbf{x}, \mathbf{w}) = \sigma(\mathbf{w}^T\mathbf{x})$$

where

$$\mathbf{w}^T\mathbf{x} = w_0 + \sum_{j=1}^{D} w_j x_j$$

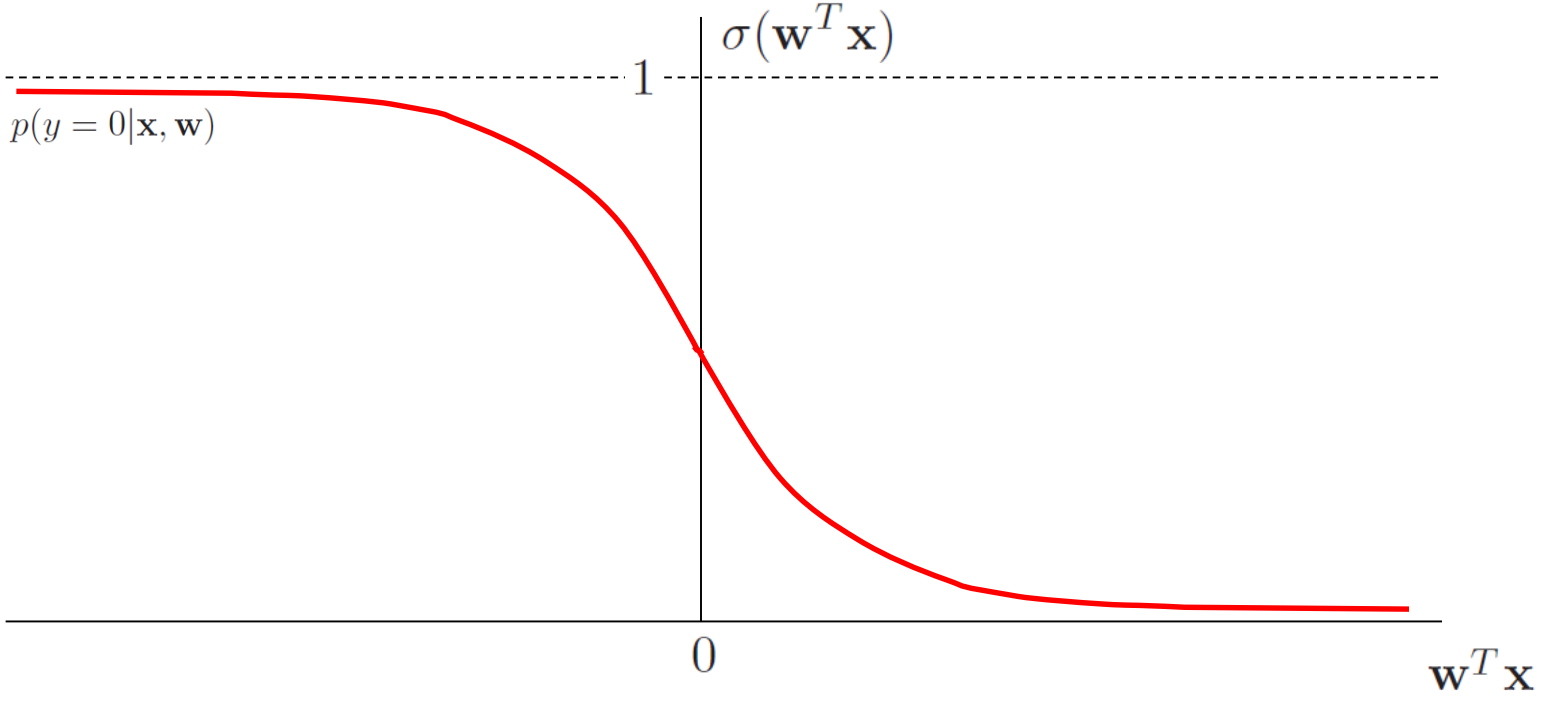


Marginalization

- Performing marginalization gives the probability of the other class, i.e. $y = 0$:

$$p(y = 0|\mathbf{x}, \mathbf{w}) + p(y = 1|\mathbf{x}, \mathbf{w}) = 1$$

$$\Rightarrow p(y = 0|\mathbf{x}, \mathbf{w}) = 1 - p(y = 1|\mathbf{x}, \mathbf{w})$$



Decision boundary

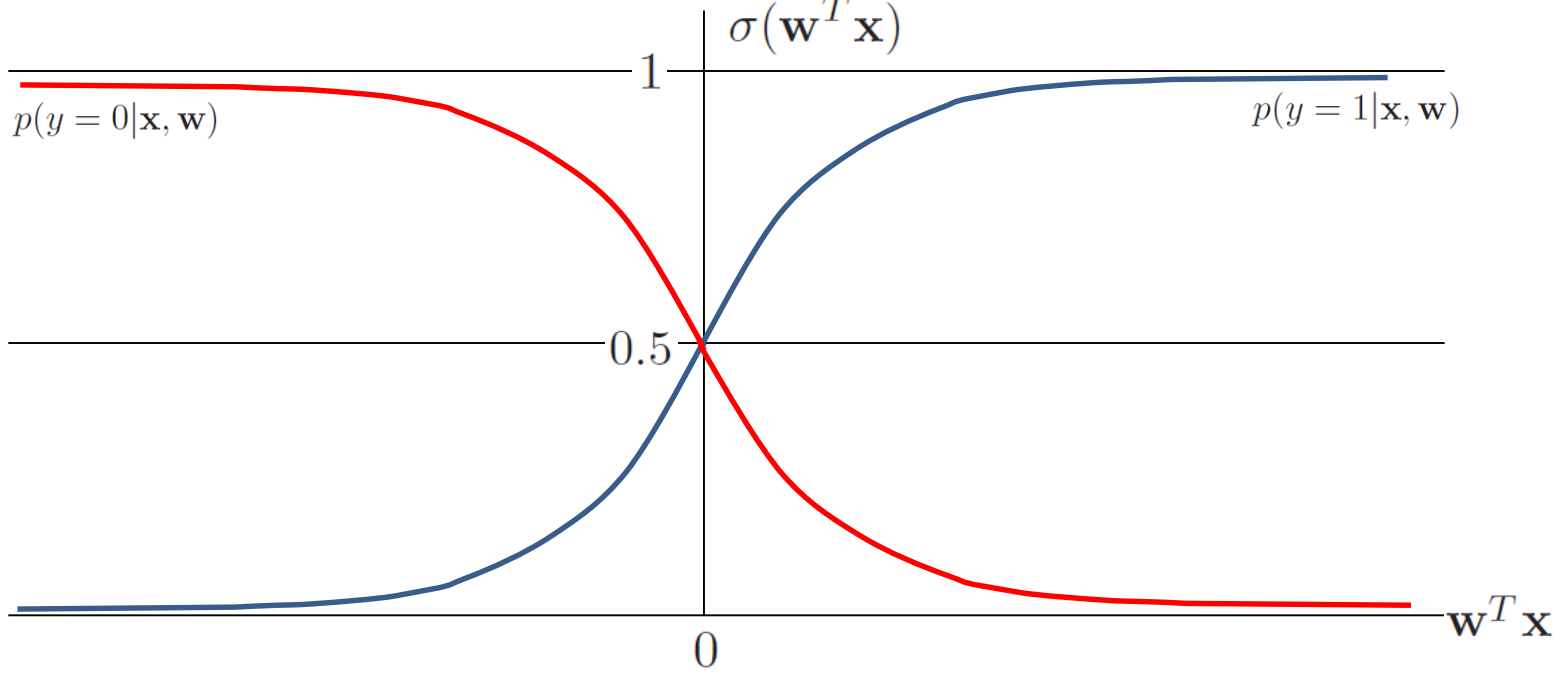
- Decision boundary is the curve separating predictions of different classes.
- For binary classification, decision boundary can be obtained by solving:

$$p(y = 0|\mathbf{x}, \mathbf{w}) = p(y = 1|\mathbf{x}, \mathbf{w}) = 0.5$$

$$\Rightarrow \sigma(\mathbf{w}^T\mathbf{x}) = \frac{1}{2} \qquad \Rightarrow \frac{1}{1 + \exp(-(\mathbf{w}^T\mathbf{x}))} = \frac{1}{2}$$

$$\Rightarrow \mathbf{w}^T\mathbf{x} = 0$$

- This is a linear decision boundary.



Compact representation

- Probabilities of the two classes can be expressed as:

$$p(y = 1|\mathbf{x}, \mathbf{w}) = \sigma(\mathbf{w}^T \mathbf{x})$$

$$p(y = 0|\mathbf{x}, \mathbf{w}) = 1 - \sigma(\mathbf{w}^T \mathbf{x})$$

- The likelihood of a single data point can be compactly expressed as

$$p(y|\mathbf{x}, \mathbf{w}) = p(y = 0|\mathbf{x}, \mathbf{w})^{(1-y)} p(y = 1|\mathbf{x}, \mathbf{w})^y$$

$$= \big(1 - p(y = 1|\mathbf{x}, \mathbf{w})\big)^{(1-y)} p(y = 1|\mathbf{x}, \mathbf{w})^y$$

Likelihood function

- Assuming the data points were generated independently, the likelihood can be written as

$$
\begin{aligned}
L(\mathbf{w}) &= p(y^{(1)}, y^{(2)}, ..., y^{(N)} | \mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(N)}, \mathbf{w}) \\
&= \prod_{i=1}^{N} p(y^{(i)} | \mathbf{x}^{(i)}, \mathbf{w}) \\
&= \prod_{i=1}^{N} \left(1 - p(y^{(i)} = 1 | \mathbf{x}^{(i)}, \mathbf{w})\right)^{(1-y^{(i)})} \left(p(y^{(i)} = 1 | \mathbf{x}^{(i)}, \mathbf{w})\right)^{y^{(i)}}
\end{aligned}
$$

- Taking log on both sides we get

$$
\log L(\mathbf{w}) = \sum_{i=1}^{N} \left[(1 - y^{(i)}) \log \left(1 - p(y^{(i)} = 1 | \mathbf{x}^{(i)}, \mathbf{w})\right) + y^{(i)} \log p(y^{(i)} = 1 | \mathbf{x}^{(i)}, \mathbf{w}) \right]
$$

- Negative of the log likelihood, i.e. $-\log L(\mathbf{w})$, is the cross-entropy loss.

Gradient ascent

- Gradient ascent: Compute the direction of steepest ascent towards the optimum, and update the weights by moving in that direction with step-size λ

$$w_j^{(t+1)} := w_j^{(t)} + \lambda \frac{\partial \log L(\mathbf{w})}{\partial w_j}$$

where λ is the learning rate.

- Vectorial representation:

$$\mathbf{w}^{(t+1)} := \mathbf{w}^{(t)} + \lambda \nabla \log L(\mathbf{w})$$

where

$$\nabla \log L(\mathbf{w}) = \left[\frac{\partial \log L(\mathbf{w})}{\partial w_0}, ..., \frac{\partial \log L(\mathbf{w})}{\partial w_D} \right]^T$$

Sigmoid function – property

$$\sigma'(z) = \frac{\mathrm{d}}{\mathrm{d}z}\left[\frac{1}{(1+\exp(-z))}\right]$$

$$= -\frac{1}{(1+\exp(-z))^2}(-\exp(-z))$$

$$= \frac{1}{(1+\exp(-z))}\left(\frac{1+\exp(-z)-1}{(1+\exp(-z))}\right)$$

$$= \frac{1}{(1+\exp(-z))}\left(1-\frac{1}{(1+\exp(-z))}\right)$$

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

Gradient of log likelihood

- Gradient of the log likelihood $\log L(\mathbf{w})$ with respect to a particular weight w_j can be computed as

$$\frac{\partial}{\partial w_j}\log(L(\mathbf{w})) = \sum_{i=1}^{N}\left(-\frac{1-y^{(i)}}{1-\sigma(\mathbf{w}^T\mathbf{x}^{(i)})}+\frac{y^{(i)}}{\sigma(\mathbf{w}^T\mathbf{x}^{(i)})}\right)\frac{\partial\sigma(\mathbf{w}^T\mathbf{x}^{(i)})}{\partial w_j}$$

$$= \sum_{i=1}^{N}\left(-\frac{1-y^{(i)}}{1-\sigma(\mathbf{w}^T\mathbf{x}^{(i)})}+\frac{y^{(i)}}{\sigma(\mathbf{w}^T\mathbf{x}^{(i)})}\right)\frac{\partial\sigma(\mathbf{w}^T\mathbf{x}^{(i)})}{\partial(\mathbf{w}^T\mathbf{x}^{(i)})}\frac{\partial(\mathbf{w}^T\mathbf{x}^{(i)})}{\partial w_j}$$

$$= \sum_{i=1}^{N}\left(\frac{-(1-y^{(i)})\sigma(\mathbf{w}^T\mathbf{x}^{(i)})+y^{(i)}(1-\sigma(\mathbf{w}^T\mathbf{x}^{(i)}))}{(1-\sigma(\mathbf{w}^T\mathbf{x}^{(i)}))\sigma(\mathbf{w}^T\mathbf{x}^{(i)})}\right)(1-\sigma(\mathbf{w}^T\mathbf{x}^{(i)}))\sigma(\mathbf{w}^T\mathbf{x}^{(i)})\frac{\partial(\mathbf{w}^T\mathbf{x}^{(i)})}{\partial w_j}$$

$$= \sum_{i=1}^{N}\left(-\sigma(\mathbf{w}^T\mathbf{x}^{(i)})+y^{(i)}\sigma(\mathbf{w}^T\mathbf{x}^{(i)})+y^{(i)}-y^{(i)}\sigma(\mathbf{w}^T\mathbf{x}^{(i)})\right)x_j^{(i)}$$

$$= \sum_{i=1}^{N}\left(y^{(i)}-\sigma(\mathbf{w}^T\mathbf{x}^{(i)})\right)x_j^{(i)}$$

Training & Prediction

- Weights optimization using gradient ascent:
  - Initialize weights $\mathbf{w}$ randomly.
  - Iterate weights until convergence

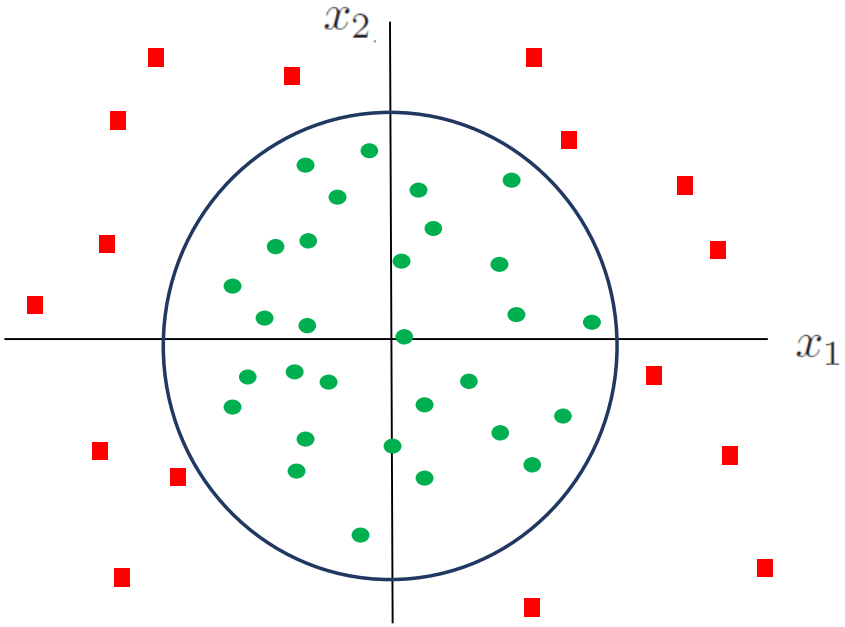
$$w_j^{(t+1)} = w_j^{(t)} + \lambda \sum_{i=1}^{N} \left(y^{(i)} - \sigma(\mathbf{w}^T \mathbf{x}^{(i)})\right) x_j^{(i)}$$

- The optimized weights can then be used for class prediction of a new unobserved example.

- The example is assigned to the class yielding the largest posterior probability:

$$y_* = \arg \max_{k=\{0,1\}} p(y_* = k | \mathbf{x}_*, \mathbf{y})$$

Non-linear decision boundary

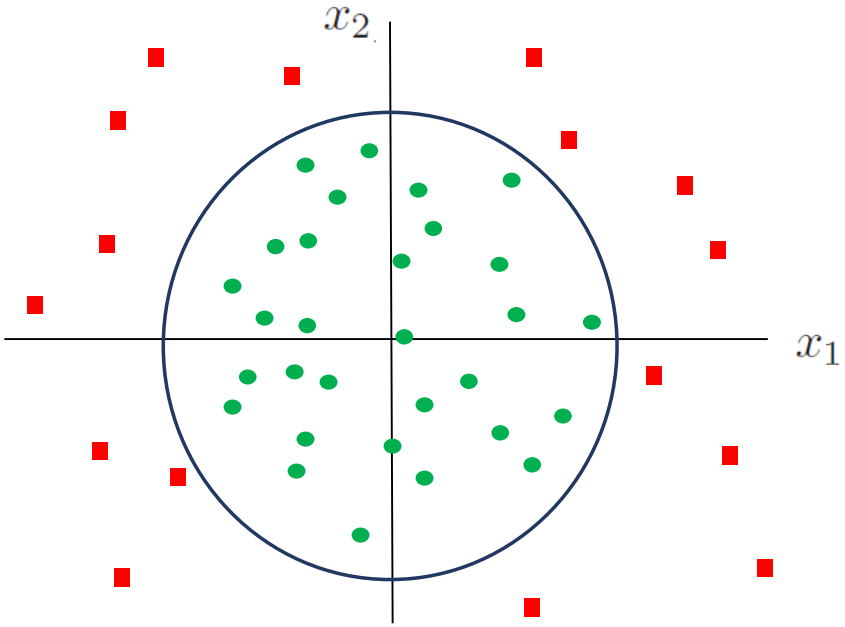


- Non-linear decision boundaries can be achieved using basis functions:

$$p(y = 1|\mathbf{w}, \mathbf{x}) = \sigma\Big(\sum_{i=0}^{Q} w_i f_i(\mathbf{x})\Big)$$
$$= \sigma\big(\mathbf{w}^{\mathrm{T}}\mathbf{f}(\mathbf{x})\big)$$

where $f_0 = 1$.

Non-linear decision boundary

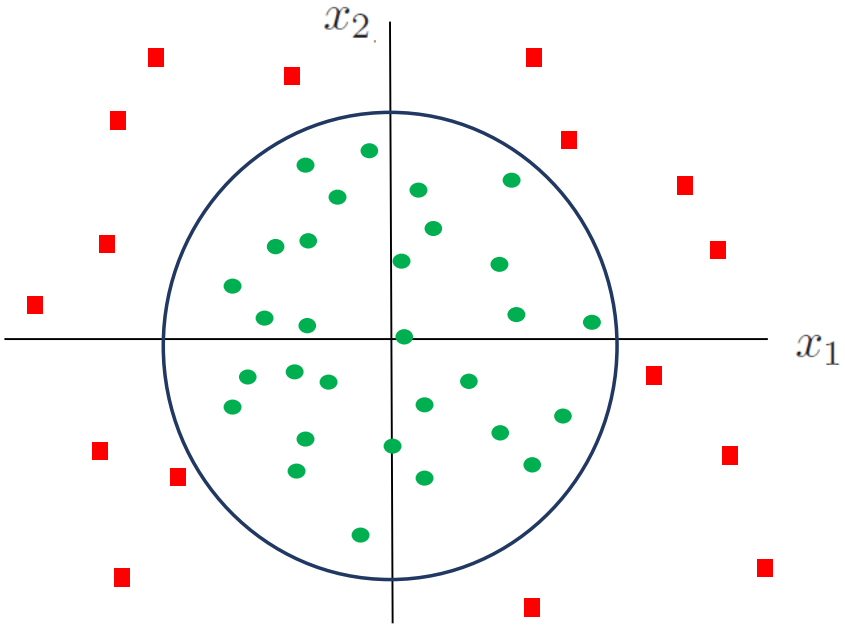


- Take $f_1(\mathbf{x}) = x_1$, $f_2(\mathbf{x}) = x_2$, $f_3(\mathbf{x}) = x_1^2$, $f_4(\mathbf{x}) = x_2^2$ and $f_5(\mathbf{x}) = x_1 x_2$, then

$$\begin{aligned} p(y = 1|\mathbf{w}, \mathbf{x}) &= \sigma\big(w_0 f_0 + w_1 f_1 + w_2 f_2 + w_3 f_3 + w_4 f_4 + w_5 f_5\big) \\ &= \sigma\big(w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2 + w_5 x_1 x_2\big) \end{aligned}$$

- Decision boundary: $\mathbf{w}^{\mathrm{T}}\mathbf{f}(\mathbf{x}) = 0$.

Non-linear decision boundary



- If for the given problem we have $\mathbf{w} = [-4, 0, 0, 1, 1, 0]^{\mathrm{T}}$ then $y = 1$ if

$$-4 + x_1^2 + x_2^2 \geq 0.$$

- The decision boundary is $x_1^2 + x_2^2 = 4$.
- Highly non-linear basis functions may lead to overfitting of data.

Confusion matrix

- A table used to describe/visualize the performance of a classification algorithm.
- Confusion matrix for a binary classification problem:

		Prediction	
		Negative	Positive
Actual	Negative	980	6
	Positive	4	10

- Standard metric: Accuracy
  - Ratio of number of correct predictions to all predictions.

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{True Positive} + \text{False Positive} + \text{True Negative} + \text{False Negative}}$$

Confusion matrix

		Prediction	
		Negative	Positive
Actual	Negative	True Negative (TN)	False Positive (FP)
	Positive	False Negative (FN)	True Positive (TP)

- Terminology:
  - True Positive (TP): Actual observation is positive, and prediction is also positive.
  - False Negative (FN): Actual observation is positive, but prediction is negative.
  - True Negative (TN): Actual observation is negative, and prediction is also negative.
  - False Positive (FP): Actual observation is negative, but prediction is positive.

Precision

		Prediction	
		Negative	Positive
Actual	Negative	True Negative (TN)	False Positive (FP)
	Positive	False Negative (FN)	True Positive (TP)

- A metric to measure the quality of the positive predictions.
- Computes the fraction of the +ve predictions which are accurate.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

$$= \frac{\text{True Positive}}{\text{Total Positive Predictions}}$$

- Use precision for more confidence on accurate +ve (TP) predictions.
  - Example: In spam detection, the classifier should be confident while labelling an email as spam (TP). Some actually spam emails going into inbox (FN) can be accepted, but do not want any important email to go into spam folder (FP), i.e. we are more concerned about the percentage of accurate +ve predictions.

Recall

		Prediction	
		Negative	Positive
Actual	Negative	True Negative (TN)	False Positive (FP)
	Positive	False Negative (FN)	True Positive (TP)

- Computes the fraction of actual +ve examples which are accurately predicted:

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

$$= \frac{\text{True Positive}}{\text{Total Actual Positive}}$$

- Use recall if FN is unacceptable.
  - Example: Classify cancer patients into high and low risk groups. Want high risk group (+ve) to be accurately predicted, i.e. the ratio of TP to (TP+FN) should be high.
- +ve recall is also known as sensitivity.

F1 score

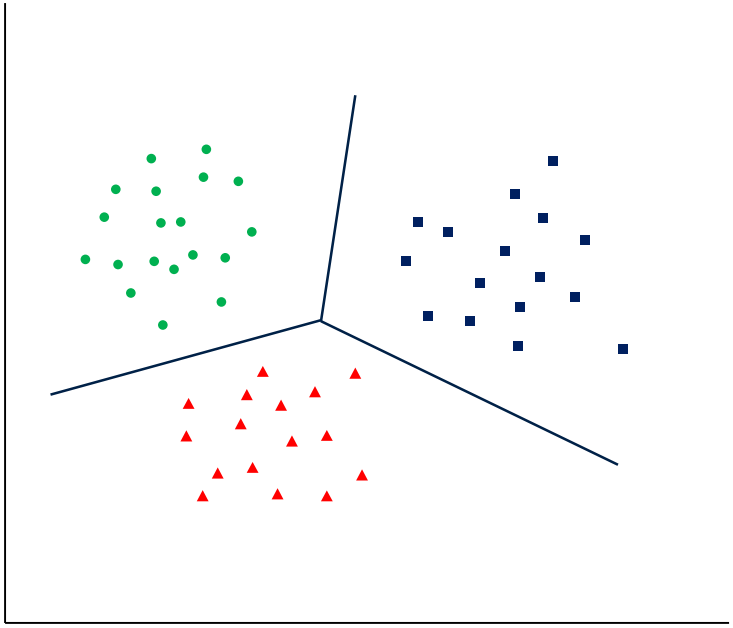
		Prediction	
		Negative	Positive
Actual	Negative	True Negative (TN)	False Positive (FP)
	Positive	False Negative (FN)	True Positive (TP)

- Is preferred when we want a balance between precision and recall.
- Is defined as the harmonic mean of precision and recall:

$$\text{F1} = \left(\frac{\text{Recall}^{-1} + \text{Precision}^{-1}}{2}\right)^{-1}$$

$$= 2\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Softmax regression



- Also known as multinomial logistic regression.
- Extension of logistic regression to classification problems involving multiple (more than 2).
- The decision boundary between classes is still linear.

Softmax regression

- Suppose there are K output classes, e.g. $y \in \{1, 2, ..., K\}$.

- Instead of a vector of weights, we now have a matrix of weights

$$\mathbf{W} = \left[\mathbf{w}_1,\ \mathbf{w}_2,\ .... \mathbf{w}_K\right]$$

  where the vector $\mathbf{w}_k$ corresponds to weights for class k.

- Probability of a class is modelled using the softmax function:

$$p(y = j|\mathbf{x}, \mathbf{W}) = \frac{\exp(\mathbf{w}_j^{\mathrm{T}}\mathbf{x})}{\sum_{k=1}^{K} \exp(\mathbf{w}_k^{\mathrm{T}}\mathbf{x})}$$

- Normalization constant is same for every class.

Softmax regression

- Sum of probabilities of an example over different classes is equal to one:

$$\sum_{k=1}^{K} p(y=k|\mathbf{x},\mathbf{W}) = p(y=1|\mathbf{x},\mathbf{W}) + p(y=2|\mathbf{x},\mathbf{W}) + ... + p(y=K|\mathbf{x},\mathbf{W})$$

$$= \frac{\exp(\mathbf{w}_1^{\mathrm{T}}\mathbf{x})}{\sum_{k=1}^{K}\exp(\mathbf{w}_k^{\mathrm{T}}\mathbf{x})} + \frac{\exp(\mathbf{w}_2^{\mathrm{T}}\mathbf{x})}{\sum_{k=1}^{K}\exp(\mathbf{w}_k^{\mathrm{T}}\mathbf{x})} + ... + \frac{\exp(\mathbf{w}_K^{\mathrm{T}}\mathbf{x})}{\sum_{k=1}^{K}\exp(\mathbf{w}_k^{\mathrm{T}}\mathbf{x})}$$

$$= 1$$

- The likelihood of a single data point can be expressed as

$$p(y|\mathbf{x},\mathbf{W}) = \prod_{k=1}^{K} \left[p(k|\mathbf{x},\mathbf{W})\right]^{\mathbb{1}_{y=k}}$$

Likelihood

- Likelihood function:

$$p(\mathbf{y}|\mathbf{X}, \mathbf{W}) = \prod_{n=1}^{N} p(y^{(n)}|\mathbf{x}^{(n)}, \mathbf{W})$$
$$= \prod_{n=1}^{N} \prod_{k=1}^{K} \left[p(k|\mathbf{x}^{(n)}, \mathbf{W})\right]^{\mathbb{1}_{y^{(n)}=k}}$$

- Log-likelihood:

$$\log p(\mathbf{y}|\mathbf{X}, \mathbf{W}) = \log \prod_{n=1}^{N} \prod_{k=1}^{K} \left[p(k|\mathbf{x}^{(n)}, \mathbf{W})\right]^{\mathbb{1}_{y^{(n)}=k}}$$
$$= \sum_{n=1}^{N} \log \prod_{k=1}^{K} \left[p(k|\mathbf{x}^{(n)}, \mathbf{W})\right]^{\mathbb{1}_{y^{(n)}=k}}$$
$$= \sum_{n=1}^{N} \log \prod_{k=1}^{K} \left[\frac{\exp(\mathbf{w}_k^{\mathrm{T}}\mathbf{x}^{(k)})}{\sum_{m=1}^{K} \exp(\mathbf{w}_m^{\mathrm{T}}\mathbf{x}^{(m)})}\right]^{\mathbb{1}_{y^{(n)}=k}}$$

Training and Prediction

- Maximize log-likelihood $\log p(\mathbf{y}|\mathbf{X}, \mathbf{W})$ with respect the weights $\mathbf{W}$ using gradient ascent (or other such method).
- Use the optimized weights to compute the posterior probabilities (pertaining to different classes) for a new example.
- The test example is assigned to the class yielding the highest posterior probability.