

Turing Machine and Deep Learning

Lecture 2: Supervised Machine Learning

Satchit Chatterji

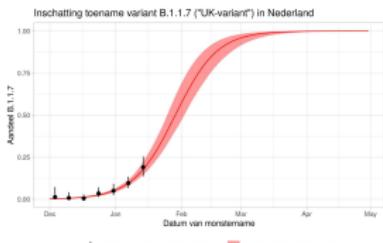
satchit.chatterji@gmail.com

MSc Artificial Intelligence
University of Amsterdam

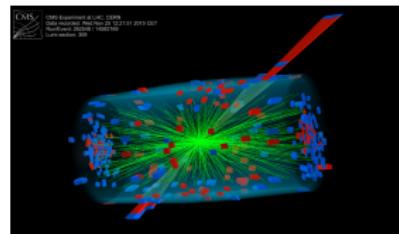
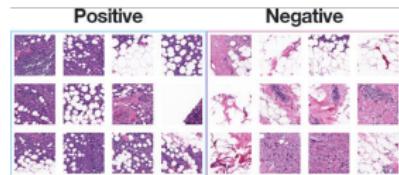
19th May 2023

Types of Tasks

Regression



Classification



Logistic Regression

Regression → Classification

For a given 1D data point $i \in \{1, \dots, N\}$, we have an estimated line:

$$\hat{y}^i = mx^i + c$$

Notation

Keep in mind here that for the next few slides, x^i means *the i^{th} data point, rather than x to the exponent i .* This will be justified later.

We can also write this as:

$$\hat{y}^i = w_0 + w_1 x^i$$

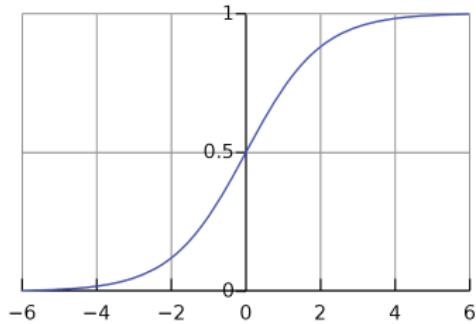
Regression → Classification

$$\hat{y}^i = \sigma(w_0 + w_1 x_1^i)$$

Regression → Classification

$$\hat{y}^i = \sigma(w_0 + w_1 x_1^i)$$

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

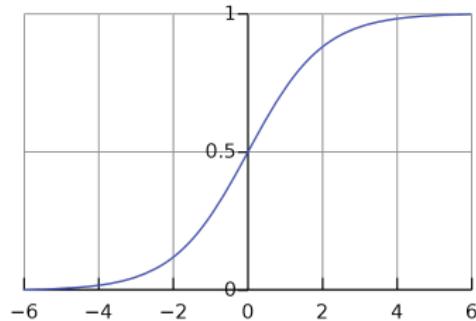


Regression → Classification

If we have k inputs,

$$\hat{y}^i = \sigma(w_0 + w_1 x_1^i + w_2 x_2^i + \dots + w_k x_k^i)$$

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



Tangent: Vectorization

Machine learning is (currently) fastest when run on GPU and is vectorized, i.e. when the math is re-written as matrix/vector calculus.

$$\hat{y}^i = \sigma(w_0 + w_1x_1^i + w_2x_2^i + \dots + w_kx_k^i)$$

Tangent: Vectorization

Machine learning is (currently) fastest when run on GPU and is vectorized, i.e. when the math is re-written as matrix/vector calculus.

$$\hat{y}^i = \sigma(w_0 + w_1 x_1^i + w_2 x_2^i + \dots + w_k x_k^i)$$

Let $\vec{w} =$

Tangent: Vectorization

Machine learning is (currently) fastest when run on GPU and is vectorized, i.e. when the math is re-written as matrix/vector calculus.

$$\hat{y}^i = \sigma(w_0 + w_1 x_1^i + w_2 x_2^i + \dots + w_k x_k^i)$$

Let $\vec{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_k \end{bmatrix}$,

Tangent: Vectorization

Machine learning is (currently) fastest when run on GPU and is vectorized, i.e. when the math is re-written as matrix/vector calculus.

$$\hat{y}^i = \sigma(w_0 + w_1 x_1^i + w_2 x_2^i + \dots + w_k x_k^i)$$

Let $\vec{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_k \end{bmatrix}$, and $\vec{x}^i =$

Tangent: Vectorization

Machine learning is (currently) fastest when run on GPU and is vectorized, i.e. when the math is re-written as matrix/vector calculus.

$$\hat{y}^i = \sigma(w_0 + w_1 x_1^i + w_2 x_2^i + \dots + w_k x_k^i)$$

Let $\vec{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_k \end{bmatrix}$, and $\vec{x}^i = \begin{bmatrix} 1 \\ x_1^i \\ \vdots \\ x_k^i \end{bmatrix}$

Rewritten as a vector dot product:

Tangent: Vectorization

Machine learning is (currently) fastest when run on GPU and is vectorized, i.e. when the math is re-written as matrix/vector calculus.

$$\hat{y}^i = \sigma(w_0 + w_1 x_1^i + w_2 x_2^i + \dots + w_k x_k^i)$$

Let $\vec{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_k \end{bmatrix}$, and $\vec{x}^i = \begin{bmatrix} 1 \\ x_1^i \\ \vdots \\ x_k^i \end{bmatrix}$

Rewritten as a vector dot product:

$$\hat{y}^i = \sigma(\vec{w}^T \vec{x}^i)$$

Logistic Regression

$$\hat{y}^i = \sigma(\vec{w}^T \vec{x}^i)$$

Goal (Binary Logistic Regression)

Estimate a probability over features parameterised by a weight vector.

To classify binary class labels, set a threshold to predict (commonly 0.5):

$$\hat{p}^i = P[y^i = 1] = \frac{1}{1 + e^{\vec{w}^T \vec{x}^i}} \in [0, 1]$$

$$\hat{y}^i = \begin{cases} 1 & \hat{p}^i \geq 0.5; \\ 0 & \hat{p}^i < 0.5. \end{cases}$$

Logistic Regression: Building blocks

Hypothesis space \mathcal{H} :

Logistic Regression: Building blocks

Hypothesis space \mathcal{H} : All functions of the form:

$$h(\vec{x}) = \sigma(\vec{w}^T \vec{x})$$

where \vec{w} are weights and \vec{x} are features of a data point.

Logistic Regression: Building blocks

Hypothesis space \mathcal{H} : All functions of the form:

$$h(\vec{x}) = \sigma(\vec{w}^T \vec{x})$$

where \vec{w} are weights and \vec{x} are features of a data point.

Loss function: Standard in binary classification tasks to use *binary cross-entropy* (derivable from σ):

$$\text{BCE}(y, \hat{y}) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})$$

Logistic Regression: Building blocks

Hypothesis space \mathcal{H} : All functions of the form:

$$h(\vec{x}) = \sigma(\vec{w}^T \vec{x})$$

where \vec{w} are weights and \vec{x} are features of a data point.

Loss function: Standard in binary classification tasks to use *binary cross-entropy* (derivable from σ):

$$\text{BCE}(y, \hat{y}) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})$$

Thus, we must find h_{opt} that minimises our **empirical risk**:

$$\hat{f} = h_{opt} = \arg \min_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^N (y^i - h(\vec{x}^i))^2$$

Classification Metrics (*a selection*)

Confusion Matrix

		Ground Truth
Predicted	Alien	True Positives
	Fish	False Positives
	Alien	False Negatives
	Fish	True Negatives

Accuracy

		Ground Truth
Predicted	True Positives	False Positives
	False Negatives	True Negatives

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

Accuracy

		Ground Truth
Predicted	Cat	Alien
	Alien	Cat
Cat	10	6
Alien	4	11

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

Accuracy



Ground Truth

Predicted	Alien	Robot
Alien	10	6
Robot	4	11

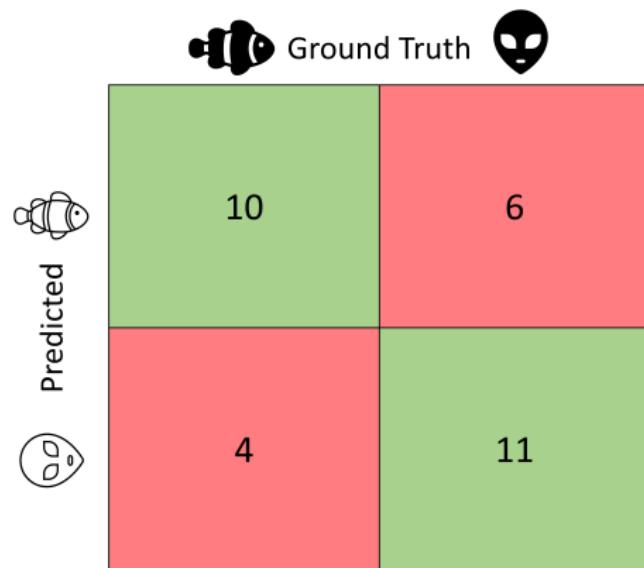
$$\begin{aligned}\text{Accuracy} &= \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \\ &= \frac{10 + 11}{10 + 11 + 6 + 4} \\ &= 0.677\dots\end{aligned}$$

Precision

		Ground Truth
Predicted	True Positives	False Positives
	False Negatives	True Negatives

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

Precision



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

Precision

		Ground Truth
Predicted	Alien	Not Alien
	Alien	Not Alien
Alien	10	6
Not Alien	4	11

$$\begin{aligned}\text{Precision} &= \frac{\text{TP}}{\text{TP}+\text{FP}} \\ &= \frac{10}{10+6} \\ &= 0.625\end{aligned}$$

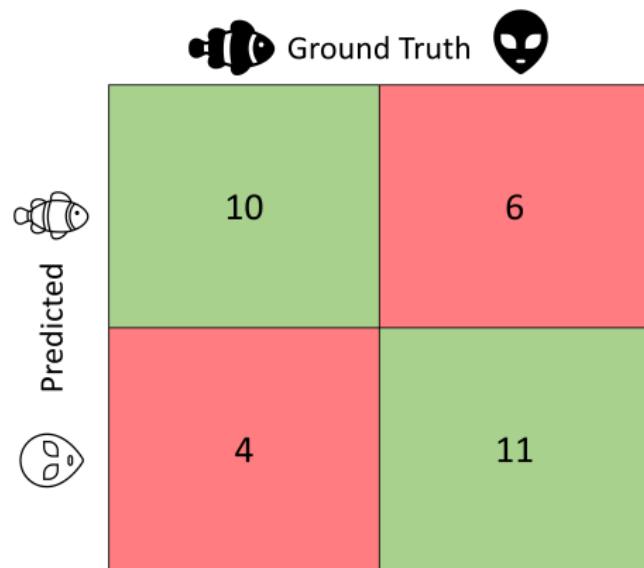
Recall

 Ground Truth 

Predicted	 True Positives	 False Positives
 False Negatives		 True Negatives
 True Negatives	 False Negatives	

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

Recall



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

Recall

Ground Truth

Ground Truth		
Predicted	Alien	Not Alien
Alien	10	6
Not Alien	4	11

$$\begin{aligned} \text{Recall} &= \frac{\text{TP}}{\text{TP} + \text{FN}} \\ &= \frac{10}{10 + 4} \\ &= 0.714... \end{aligned}$$

F_β score

		Ground Truth
Predicted	True Positives	False Positives
	False Negatives	True Negatives

$$F_\beta = (1 + \beta)^2 \cdot \frac{\text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}}$$

F_1 score



Predicted	True Positives	False Positives
	True Negatives	

$$F_1 = 2 \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

F_1 score

		Ground Truth
Predicted	Alien	Not Alien
	Alien	Not Alien
Alien	10	6
Not Alien	4	11

$$F_1 = 2 \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

F_1 score

		Ground Truth
Predicted	Alien	Not Alien
	Alien	Not Alien
Alien	10	6
Not Alien	4	11

$$\begin{aligned} F_1 &= 2 \frac{0.625 \cdot 0.714}{0.625 + 0.714} \\ &= 0.666\dots \end{aligned}$$

Collected Measures

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

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

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

$$F_1 = 2 \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

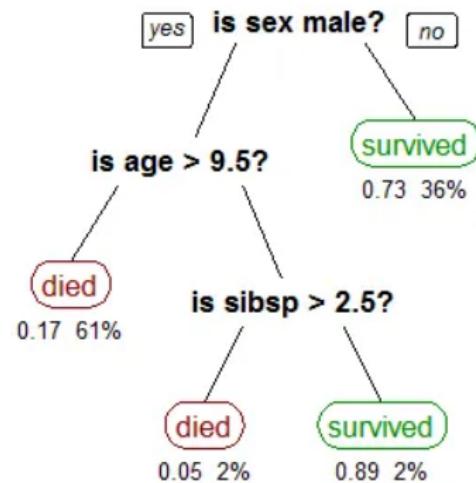
$$F_\beta = (1 + \beta)^2 \cdot \frac{\text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}}$$

Notebook time!

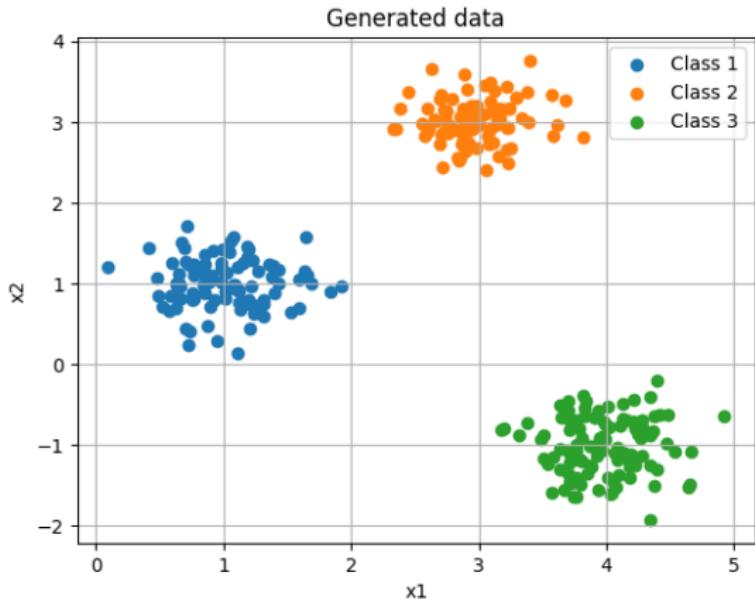
Decision Trees

Decision Trees

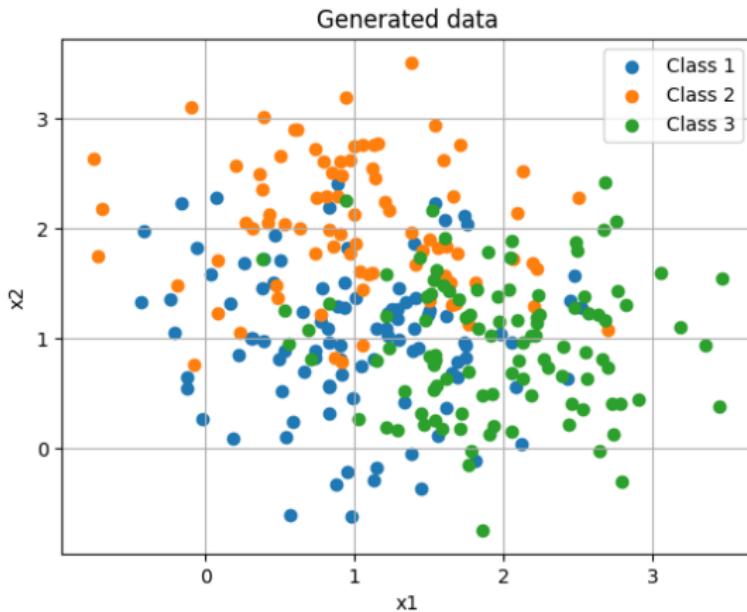
- Partition feature space into regions and assign labels/values to those regions.
- Each internal node represents a test on a feature, leading to different branches based on the outcome of the test.
- The leaves of the tree represent the final predicted labels or values.
- Visually appealing and (sort of) interpretable!
- Loss function for each node:
Gini index.



Decision Trees: Example



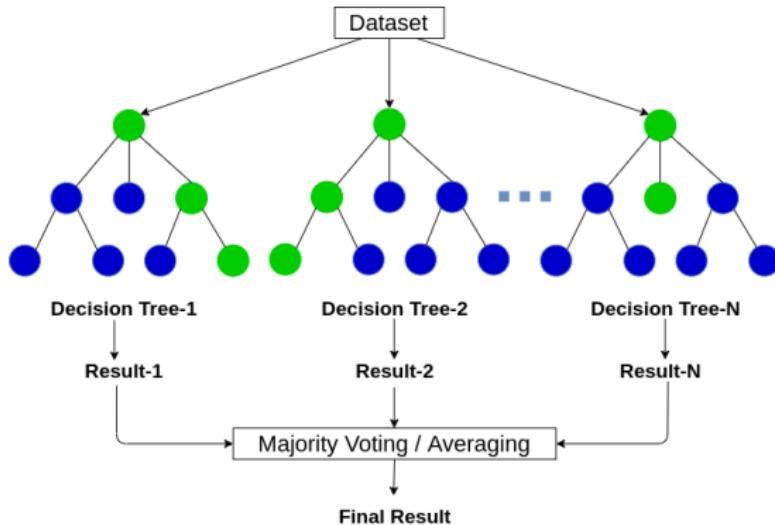
Decision Trees: Example



Random Forests

Random Forests

Random forest is an **ensemble learning** method that combines multiple decision trees to make predictions.



Ensemble Learning

An ML method which combines the predictions of multiple models to improve overall performance.

Random Forest: Training Process

- ① Data Preparation
- ② Random Sampling (Bootstrapping)
- ③ Decision Tree Construction + Random Feature Selection
- ④ Voting and Aggregation
- ⑤ Evaluation

We get better/more generalizable results (harder to overfit) *but*, we lose out on interpretability.

Support Vector Machines

SVMs: Math

SVMs: Math

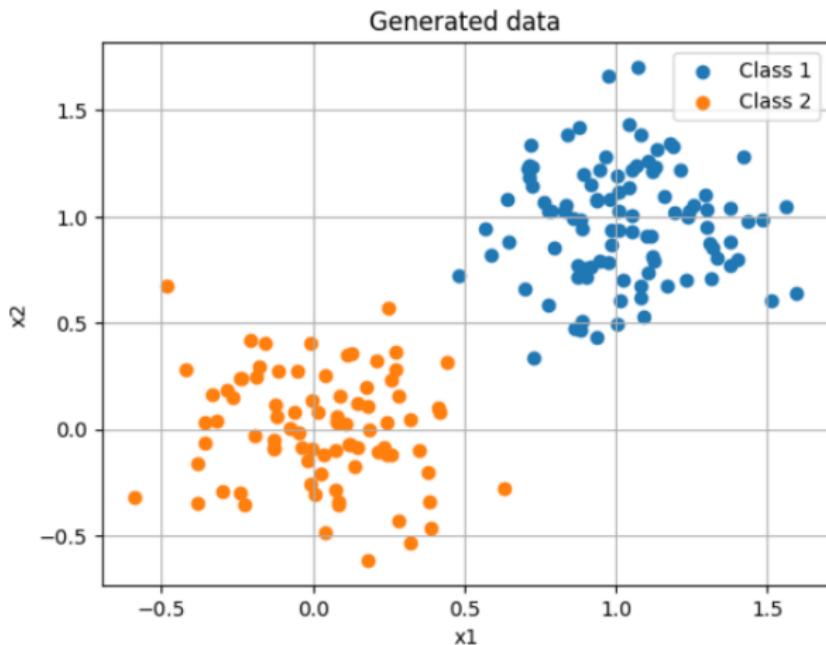
No

SVMs: Math

No

But if you're interested, look at UvA's ML1 Lecture 11: <https://youtu.be/QTzo2whcG2g>.

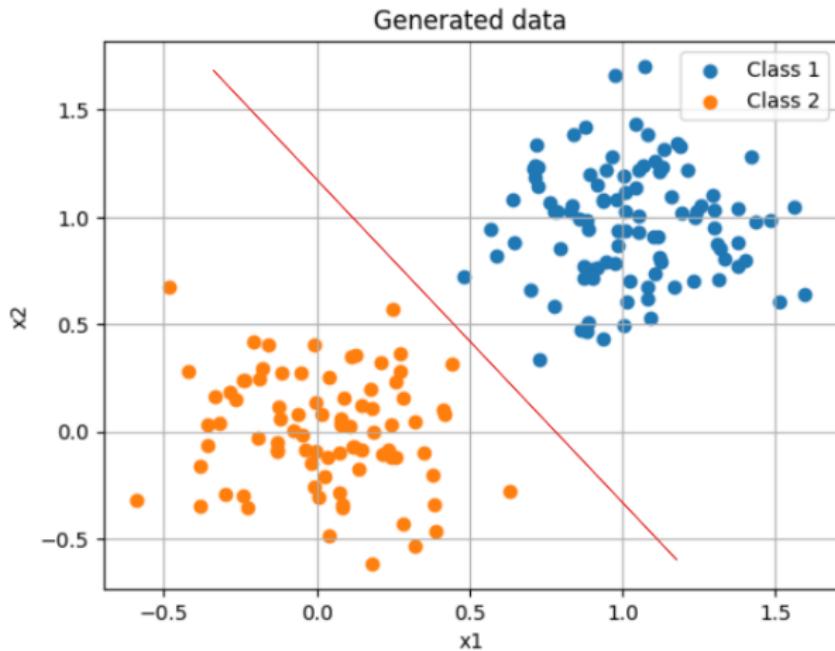
SVMs: *Intuition*



Goal

Find a *decision boundary* that maximally separates the two classes.

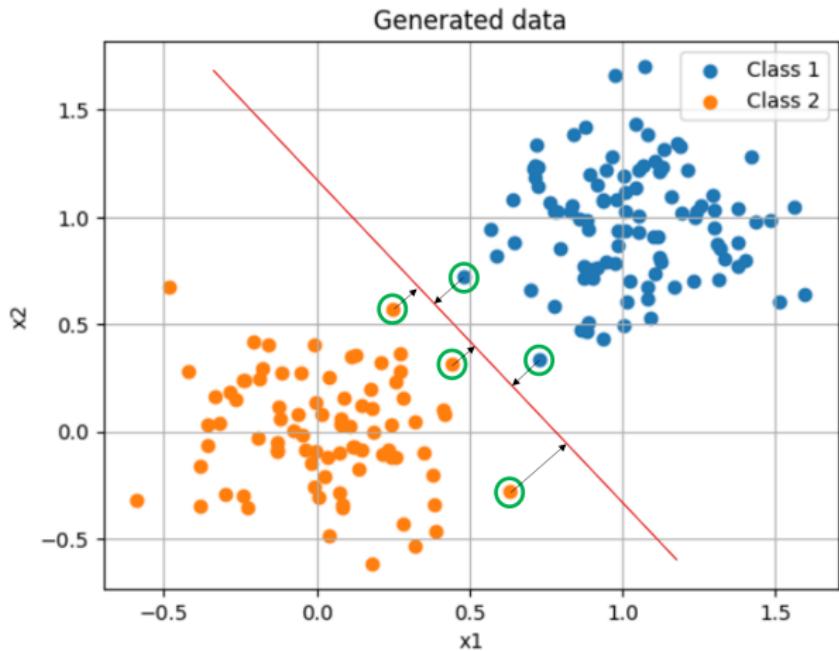
SVMs: *Intuition*



Goal

Find a *decision boundary* that maximally separates the two classes.

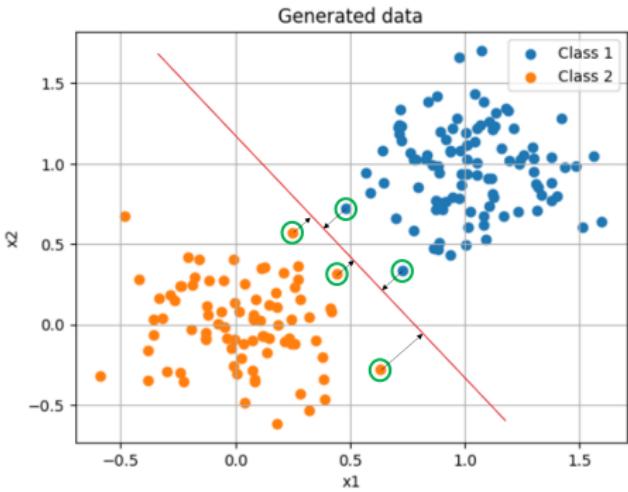
SVMs: *Intuition*



Goal

Find a *decision boundary* that maximally separates the two classes.

SVMs: *Intuition*

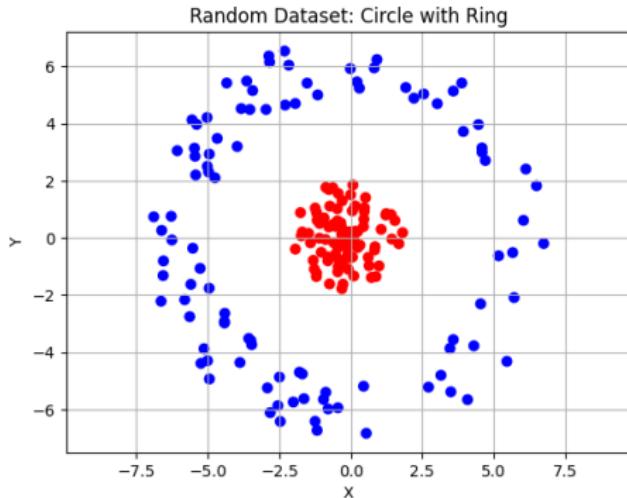


Goal

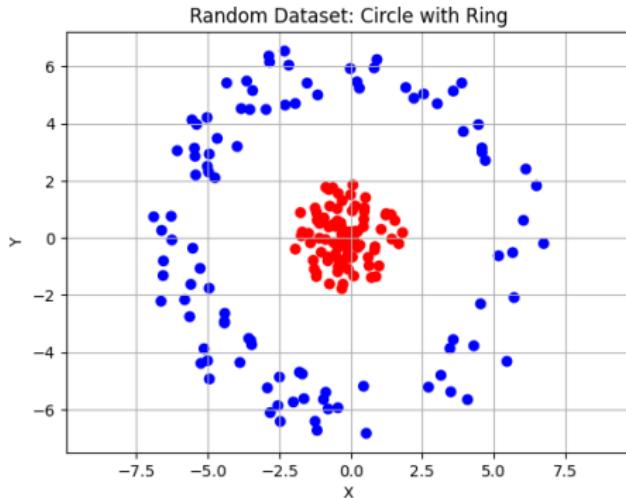
Find a *decision boundary* that maximally separates the two classes.

What about data sets that are *non-linearly separable*?

SVMs: Intuition II – *The Kernel Trick*

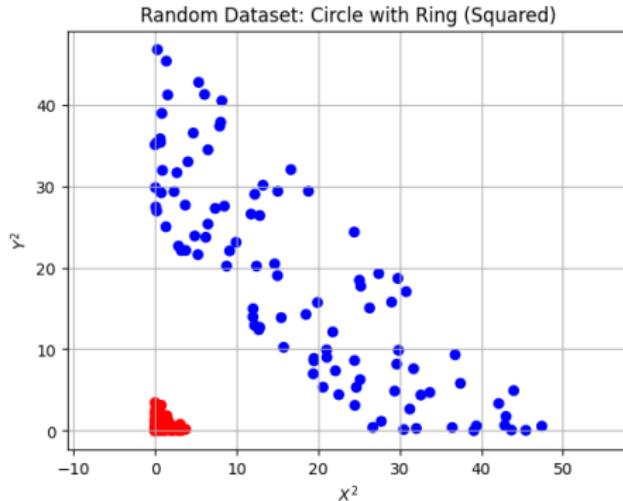


SVMs: Intuition II – *The Kernel Trick*

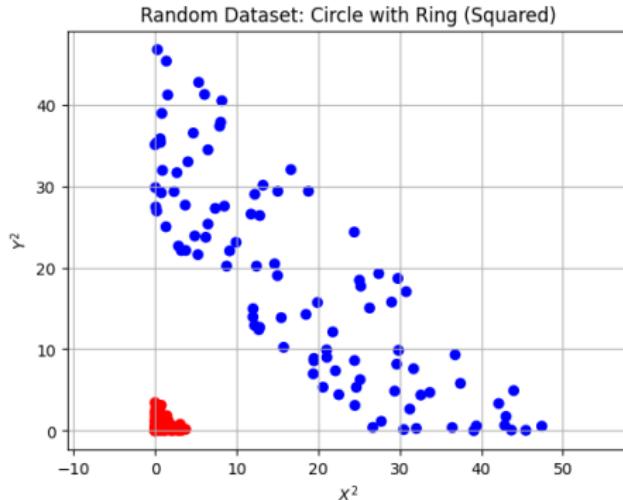


What if we could modify the data to be linearly separable?

SVMs: Intuition II – *The Kernel Trick*



SVMs: Intuition II – *The Kernel Trick*



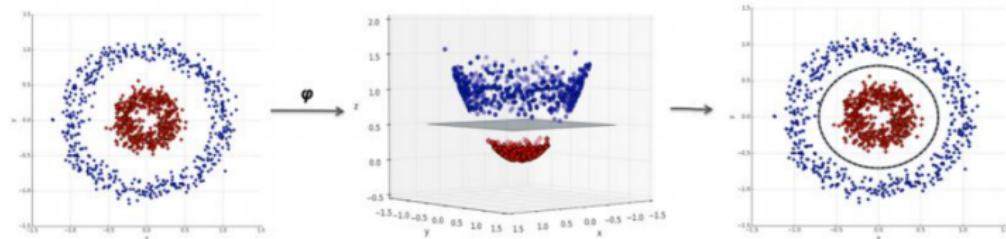
Problems

- Computationally intensive
- Hard to find a good “basis function”

SVMs: Intuition II – *The Kernel Trick*

Solution: The Kernel Trick

Learn a ‘kernel’ $K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$, representing *some* similarity measure in a higher-dimensional space between data points x_i and x_j without transforming the data itself.



Use this to learn a (hyper-)plane decision boundary.

⇒ SVMs are *non-parametric*.

SVMs: Types of Kernels

- Linear:

$$K(x_i, x_j) = x_i^T x_j$$

- Polynomial:

$$K(x_i, x_j) = (1 + x_i^T x_j)^d$$

- Radial Basis:

$$K(x_i, x_j) = e^{-\gamma \|x_i^T x_j\|^2}, \text{ where } \gamma > 0$$

Careful!

- Different kernels have different hyperparameters.
- May not scale with data ($\approx \mathcal{O}(n^2)$ to $\approx \mathcal{O}(n^3)$ with n data points.)
- As always, beware of overfitting!

TURINGSTUDENTSROTTERDAM

Image credits

Slide 2

- ① <https://medium.com/analytics-vidhya/house-price-prediction-using-linear-regression-from-scratch-b2b48fd73689>
- ② <https://weather.com>
- ③ <https://www.rivm.nl/en/coronavirus-covid-19/modelling>
- ④ <https://gsurma.medium.com/image-classifier-cats-vs-dogs-with-convolutional-neural-networks-cnns-and-google-colabs-4e9af21ae7a8>
- ⑤ <https://pyimagesearch.com/2019/02/18/breast-cancer-classification-with-keras-and-deep-learning/>
- ⑥ <https://home.cern/news/news/accelerators/lhc-collides-ions-new-record-energy>

Slide 32

- <https://www.analyticsvidhya.com/blog/2020/05/decision-tree-vs-random-forest-algorithm/>