ieeta instituto de engenharia electrónica e telemática de aveiro

universidade de aveiro

Departamento de Eletrónica, Telecomunicações e Informática

Machine Learning

Lecture 5:

Support Vector Machine (SVM)

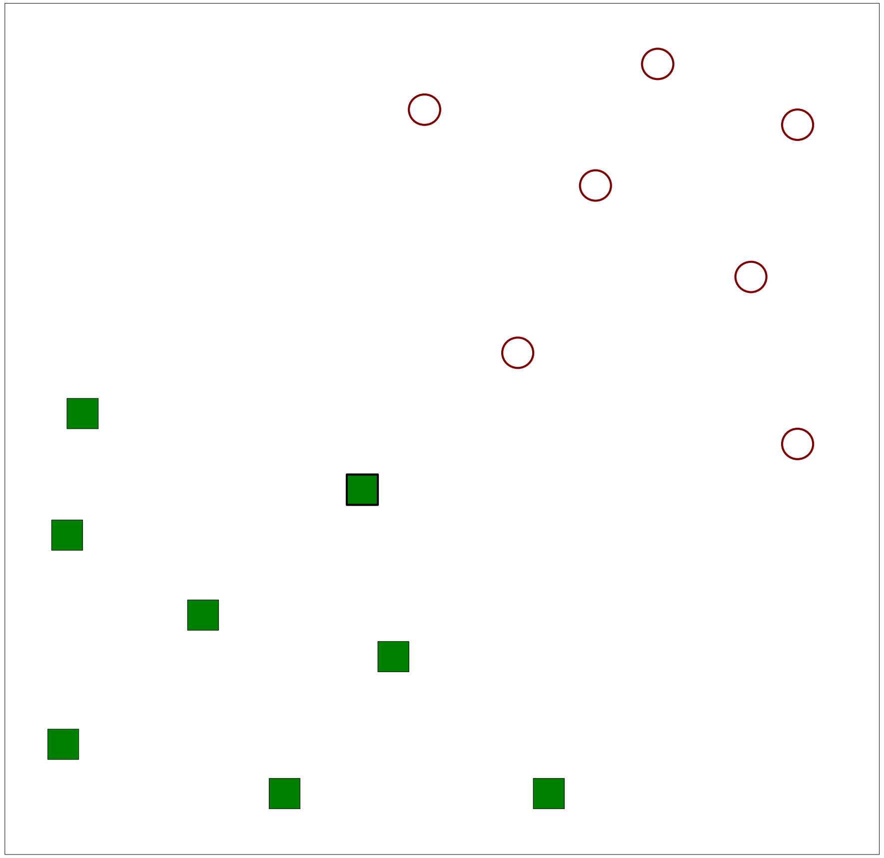
Petia Georgieva
(petia@ua.pt)

universidade de aveiro

LECTURE outline

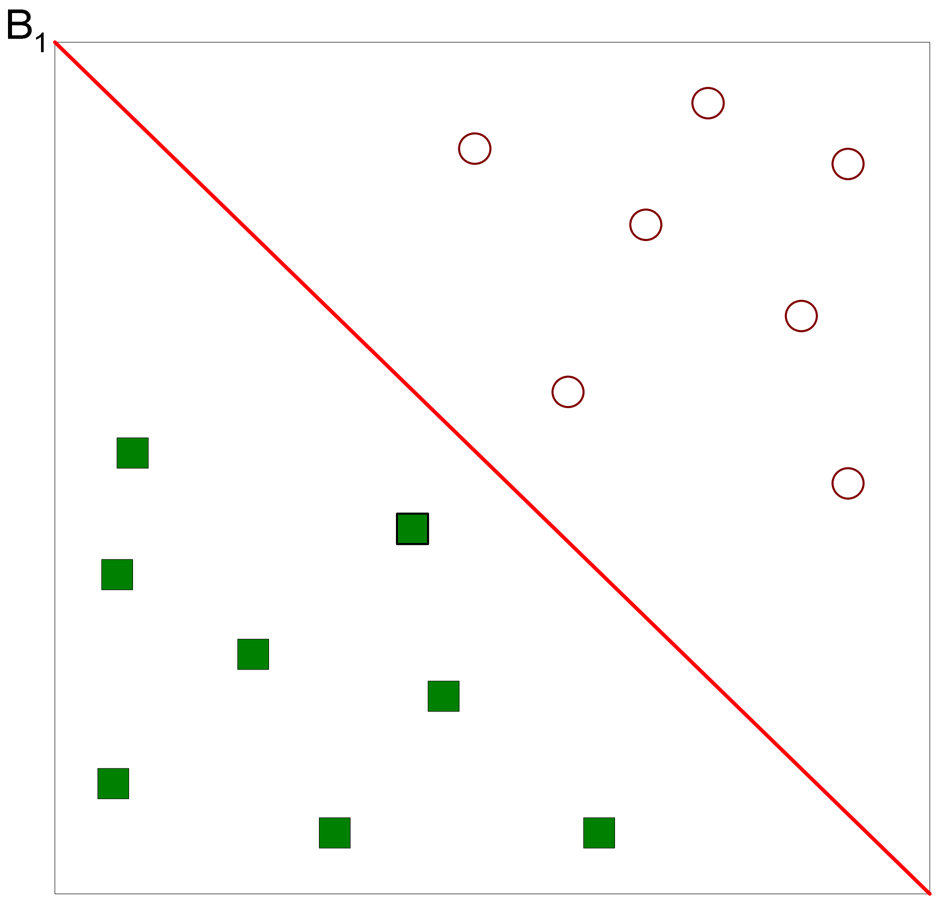
1. **Linear Support Vector Machine (SVM)**
2. **Nonlinear SVM - Gaussian RBF Kernel**
3. **Performance evaluation – confusion matrix**
4. **Class imbalance problem**

Linearly separable classes



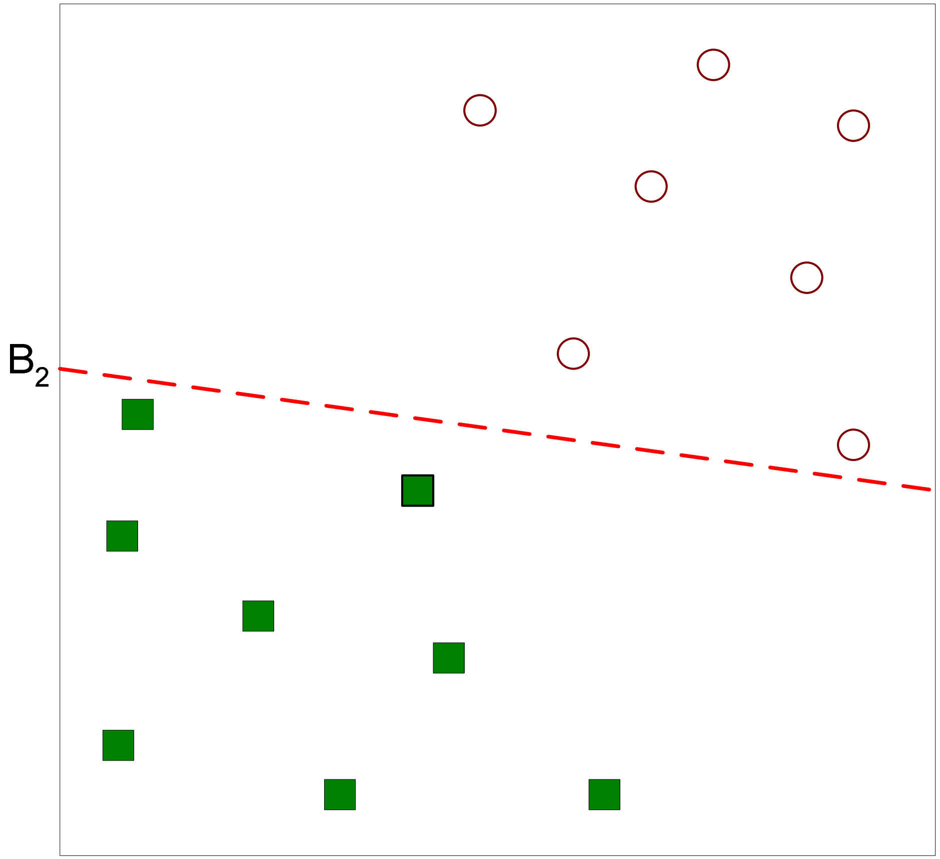
Find a decision boundary to separate data

Linearly separable classes



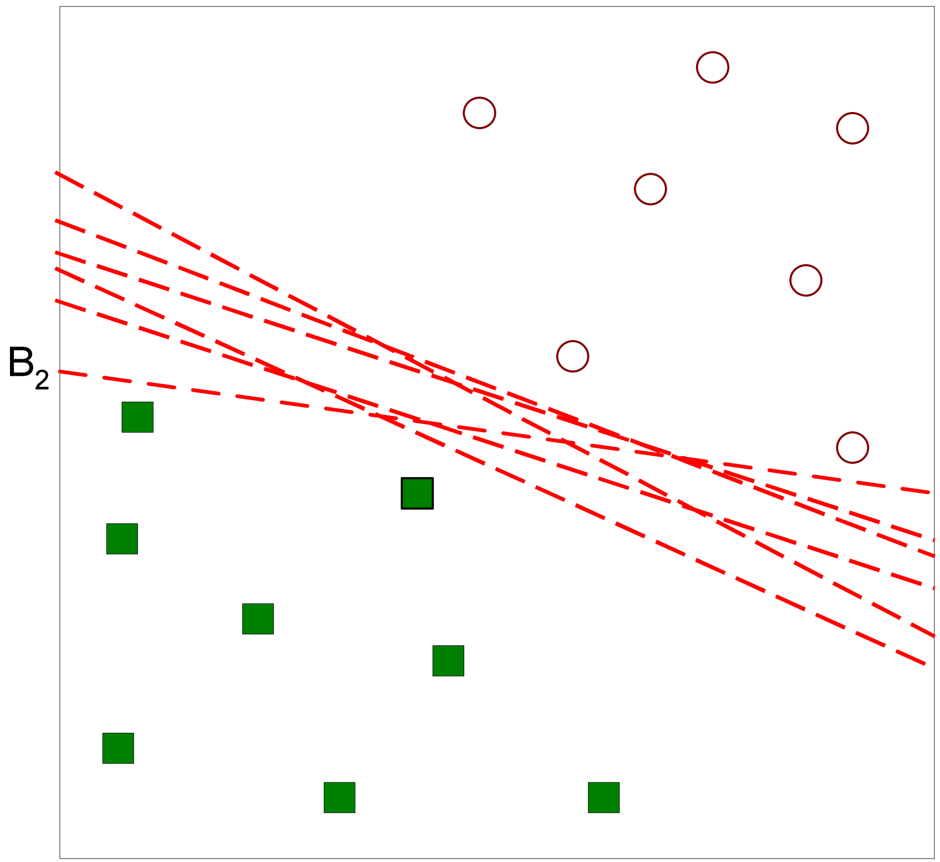
One Possible Solution

Linearly separable classes



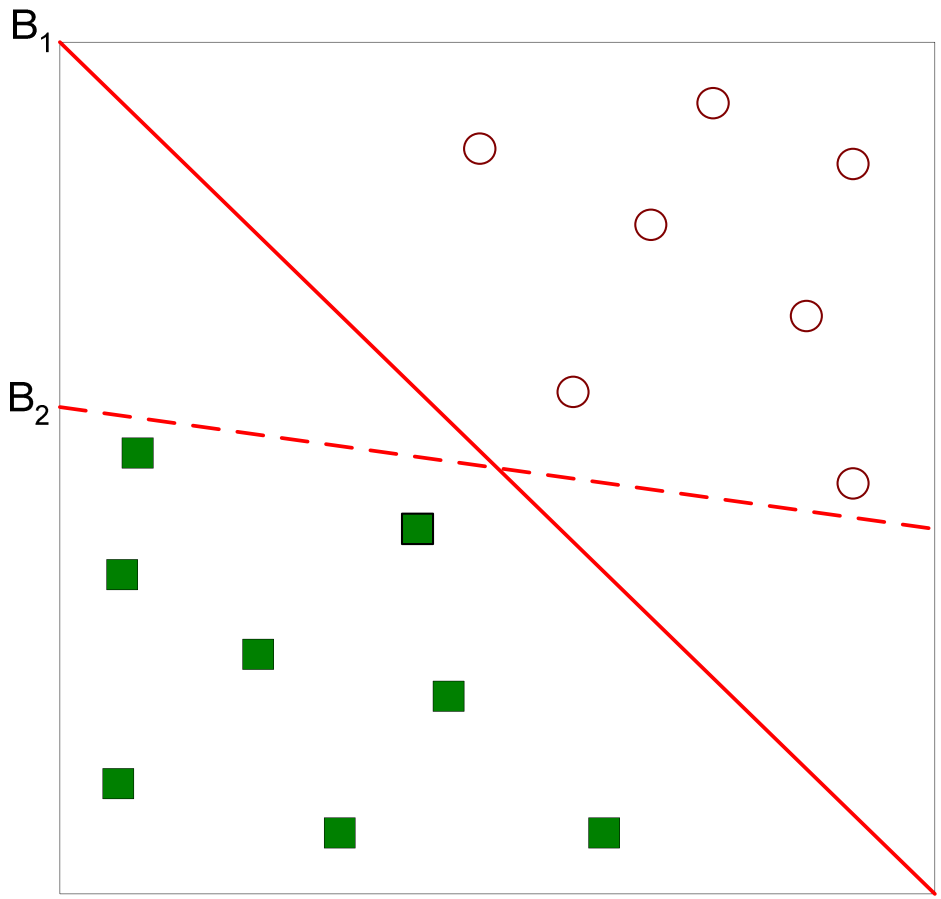
Another possible solution

Linearly separable classes



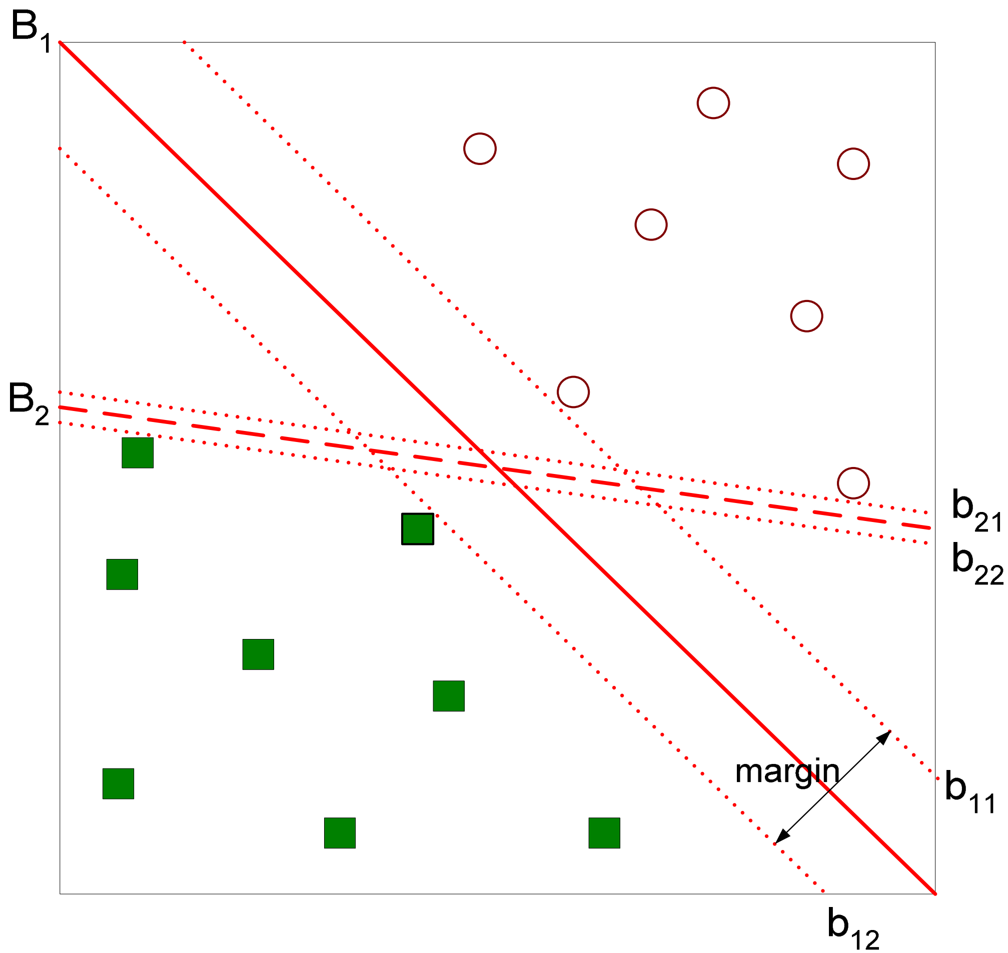
Many possible solutions

Linearly separable classes



Which one is better? B1 or B2?

SVM - Large margin classifier

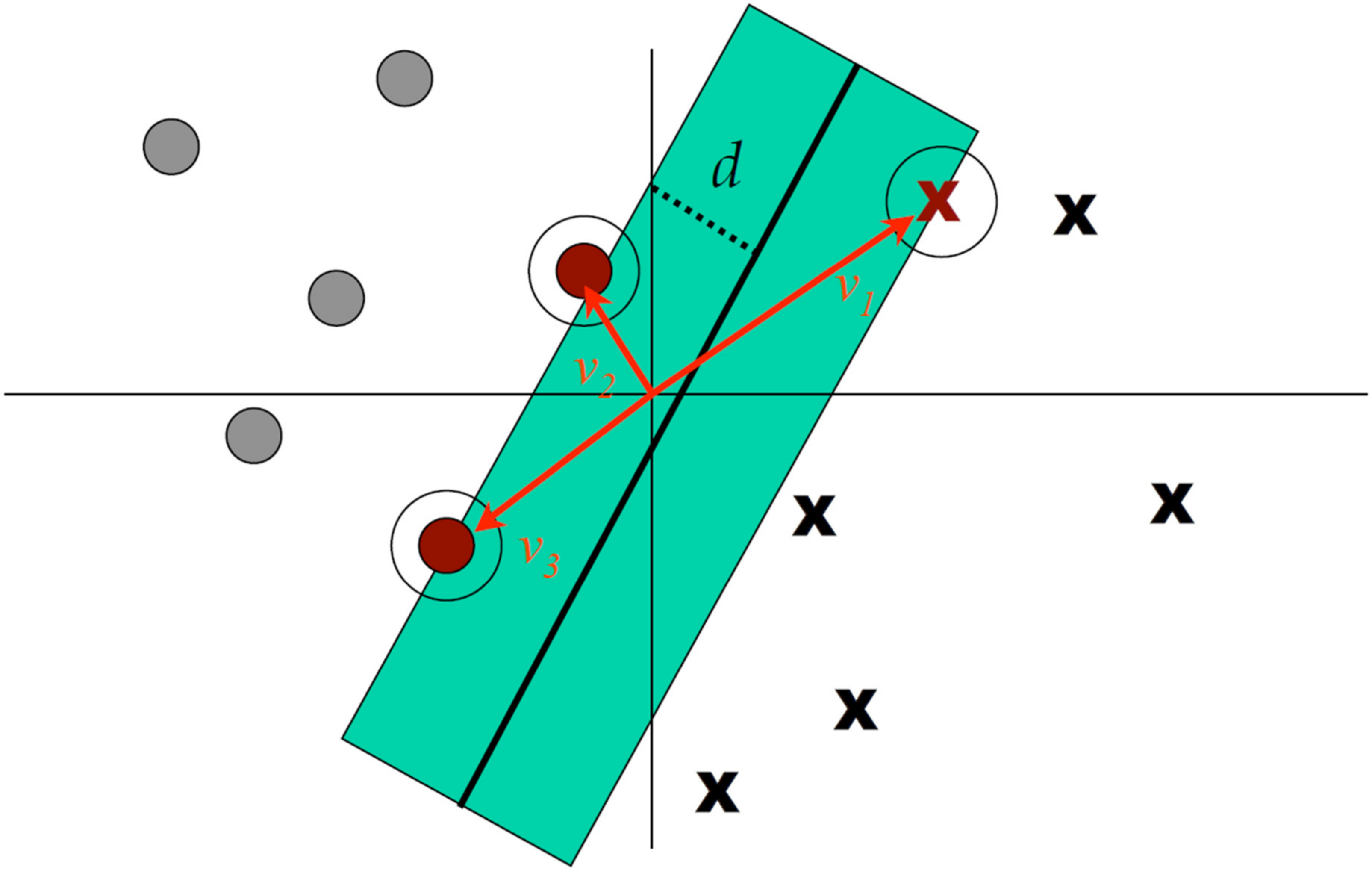


Find a boundary that maximizes the margin => B1 is better than B2

Proposed by Vladimir N. Vapnik and Alexey Chervonenkis, 1963

SUPPORT VECTORS (v1,v2,v3)

Only the closest points (support vectors) from each class are used to decide which is the optimum (the largest) margin between the classes.



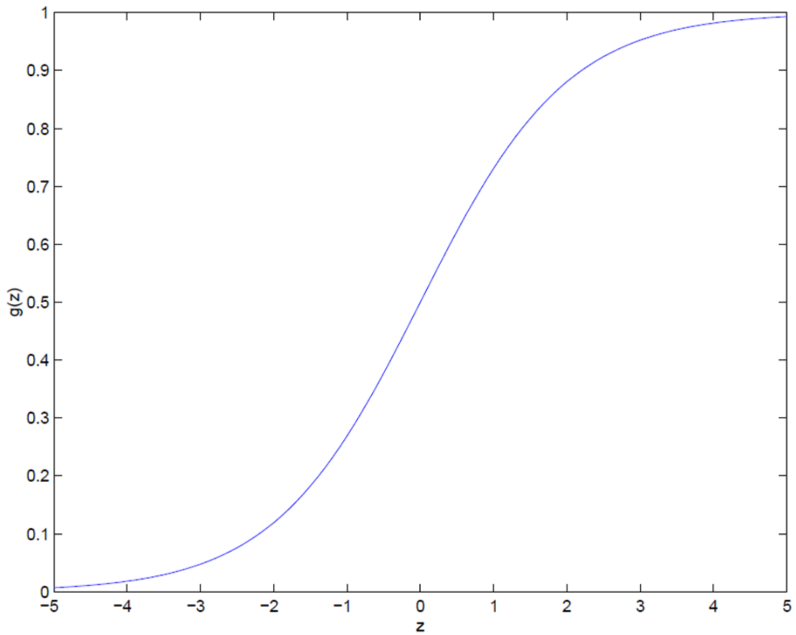
Logistic Regression (LogReg) -revised

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$$

$$\theta^T x = \theta_0 + \sum_{j=1}^{n} \theta_j x_j$$

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

Logistic (sigmoid) function



if $y = 1$, we want $h_\theta(x) \approx 1$, $\theta^T x >> 0$

if $y = 0$, we want $h_\theta(x) \approx 0$, $\theta^T x << 0$

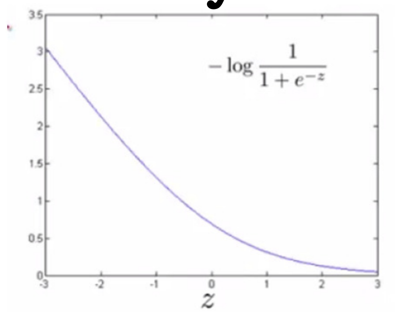
universidade de aveiro

SVM cost function

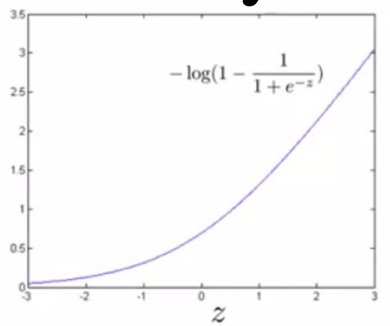
Regularized LogReg cost function:

$$\min_{\theta} \frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)}\left(-\log h_\theta(x^{(i)})\right) + (1-y^{(i)})\left((-\log(1-h_\theta(x^{(i)}))\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

for y=1

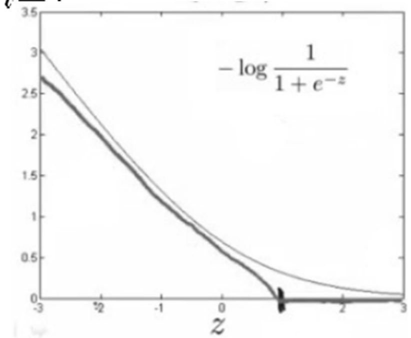


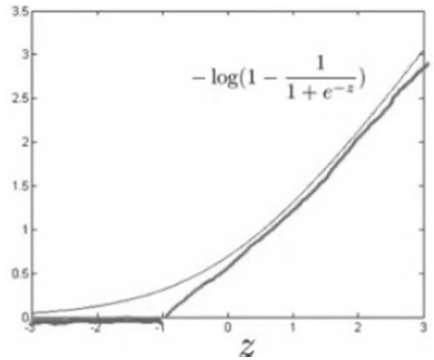
for y=0



Regularized SVM cost function (Modification of LogReg cost function. ***cost0*** & ***cost1*** are assimptotic safety margins with computational advantages)

$$\min_{\theta} C\sum_{i=1}^{m}\left[y^{(i)} cost_1(\theta^T x^{(i)}) + (1-y^{(i)}) cost_0(\theta^T x^{(i)})\right] + \frac{1}{2}\sum_{i=1}^{n}\theta_j^2$$





$$z = \theta^T x$$

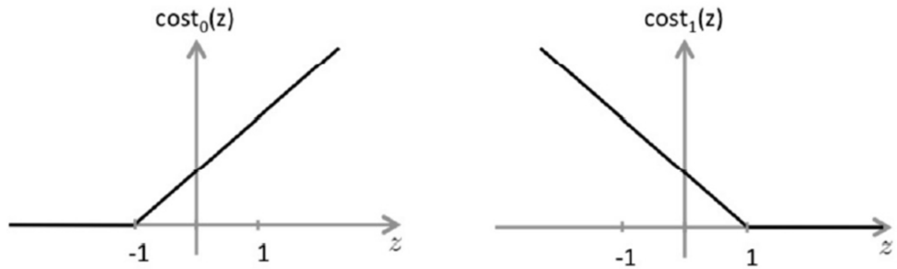
SVM cost function

Regularized LogReg cost function:

$$\min_{\theta} \frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)}\left(-\log h_\theta(x^{(i)})\right) + (1-y^{(i)})\left((-\log(1-h_\theta(x^{(i)}))\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

Regularized SVM cost function

$$\min_{\theta} C\sum_{i=1}^{m}\left[y^{(i)} cost_1(\theta^T x^{(i)}) + (1-y^{(i)}) cost_0(\theta^T x^{(i)})\right] + \frac{1}{2}\sum_{i=1}^{n}\theta_j^2$$



$$z = \theta^T x$$

Different way of parameterization: C is equivalent to $1/\lambda$.

C >0 - parameter that controls the penalty for misclassified training examples.
Increase C more importance to training data fitting.
Decrease C – more importance to generalization properties (combat overfitting).

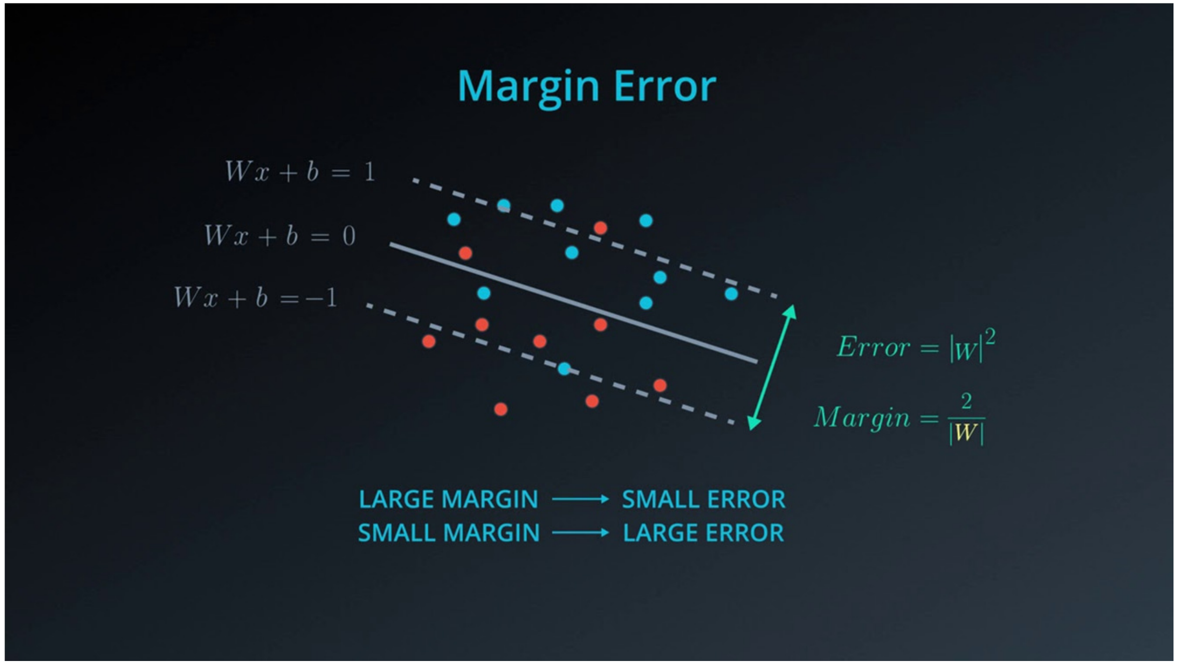
universidade de aveiro

SVM Algorithm – soft margin

Two quantities to optimize : classification error (how many points are wrongly classified) and "margin error" (optimize the margin between the two classes)

Search for the largest margin that minimizes the classification error.

C controls how wide is the "street".



$$\theta^T x => Wx + b$$

$$\min_{\theta} \sum_{j=1}^{n} W_j^2 = |W|^2$$

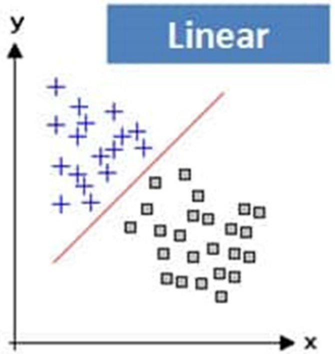
$(L_2 \text{ norm})$ such that

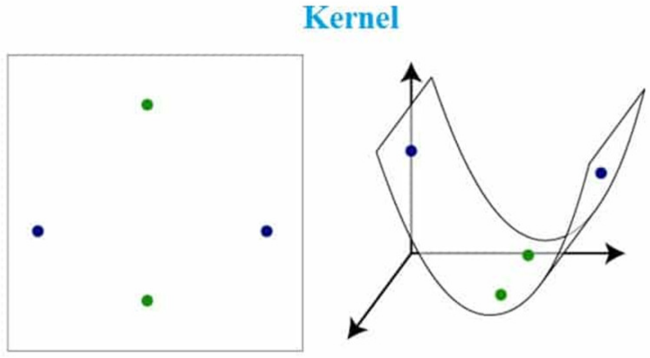
$$Wx^{(i)} + b \geq 1, \quad \text{if } y = 1$$

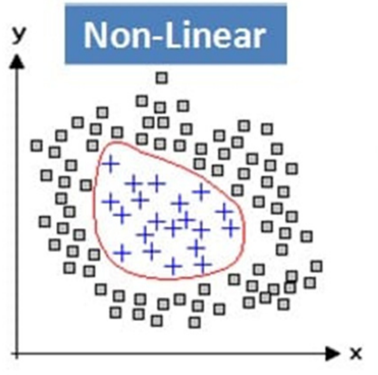
$$Wx^{(i)} + b \leq -1 \quad \text{if } y = 0$$

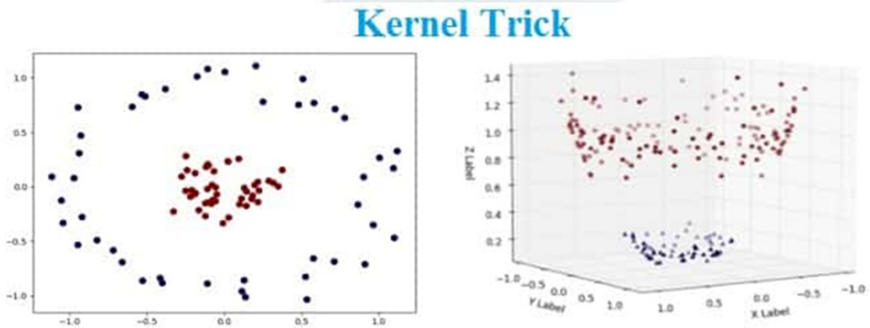
universidade de aveiro

Nonlinearly separable data – kernel SVM









Kernel: function which maps a lower-dimensional data into higher dimensional data.

Tipical Kernels:

- Polynomial Kernel - adding extra polynomial terms
- Gaussian Radial Basis Function (RBF) kernel – <u>the most used kernel</u>
- Laplace RBF kernel
- Hyperbolic tangent kernel
- Sigmoid kernel, etc.

Nonlinear SVM – Gaussian RBF Kernel

$$k(x_i, x_j) = e^{\left(-\gamma \left\| x^{(i)} - x^{(j)} \right\|^2\right)}, \quad \gamma > 0, \gamma = 1/2\sigma^2, \quad \sigma - \text{variance}$$

The RBF kernel is a metric of similarity between examples, $x^{(i)}$ and $x^{(j)}$
Substitute the original features with similarity features (kernels).

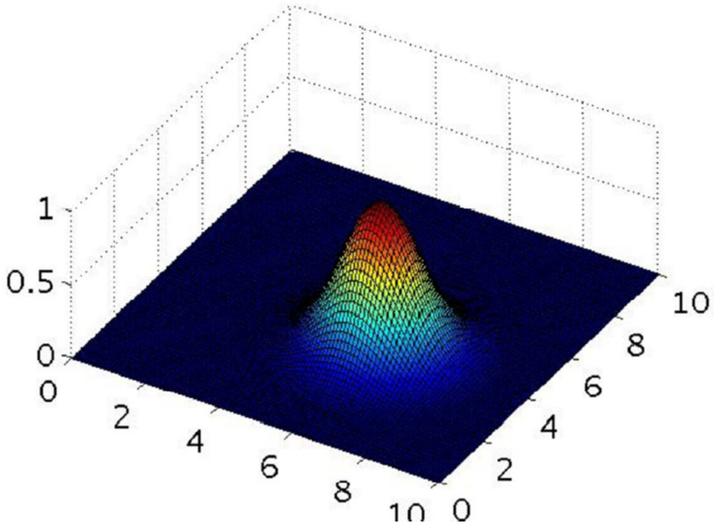
Note: the original (n+1 dimensional) feature vector is substituted by the new (m+1 dimensional) similarity feature vector.

m –number of examples, **m>>n !!!**

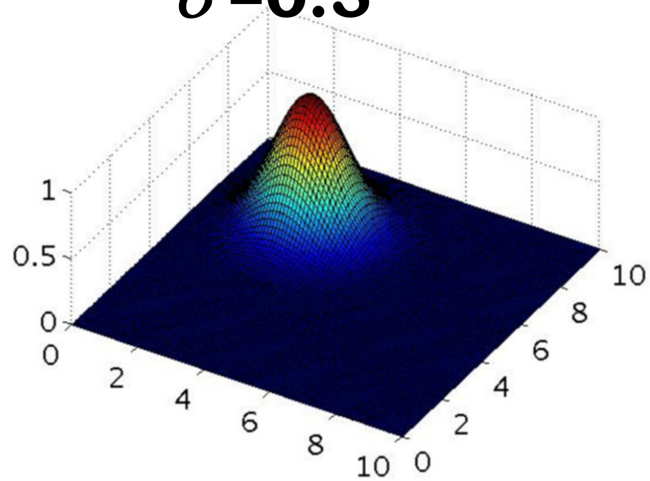
Gaussian RBF Kernel – Parameter σ

$$k(x_i, x_j) = e^{-\gamma\|x_i - x_j\|^2}, \quad \gamma = \frac{1}{2\sigma^2} > 0$$

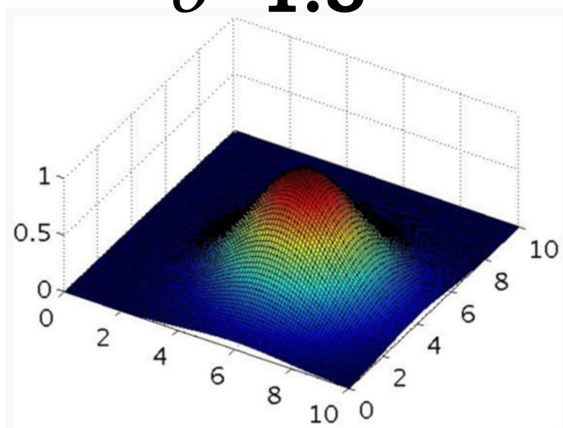
σ=1



σ=0.5



σ=1.5



σ determines how fast the similarity metric decreases to 0 as the examples go away of each other.

Large σ: kernels vary more smoothly (combat overfitting)

Small σ: kernels vary less smoothly (more importance to training data fitting)

universidade de aveiro

SVM parameters

<u>How to **choose hyper-parameter C:**</u>

Large C: lower bias, high variance (equivalent to small regular. param. λ)

Small C: higher bias, lower variance (equivalent to large regular. param. λ)

<u>How to **choose hyper-parameter** σ**:**</u>

Large σ**:** features vary more smoothly. Higher bias, lower variance

Small σ**:** features vary less smoothly. Lower bias, higher variance

SVM implementation

Use SVM software packages to solve SVM optimization !!!

In Python, use Scikit-learn (sklearn) machine learning library and

Import SVC (Support Vector Classification):

from sklearn.svm import SVC

classifier = SVC(kernel="rbf",gamma =?)

"rbf“ (Radial Basis Function) corresponds to the Gaussian kernel.

gamma = 1/σ.

SVM math explained: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

https://datamites.com/blog/support-vector-machine-algorithm-svm-understanding-kernel-trick/#:~:text=A%20Kernel%20Trick%20is%20a,Lagrangian%20formula%20using%20Lagrangian%20multipliers.%20(

Logistic Reg versus SVM

n = number of features, m=number of examples

- If n is large (relative to m) (e.g. n=10000; m=10-1000) =>

use logistic regression or SVM without kernel ("linear kernel")

- If n is small, m is intermediate (n=1-1000; m=10-10000) =>

Use SVM with Gaussian kernel

- If n is small, m is large (n=1-1000; m=50000)

Create more features, then use logistic regression or SVM without a kernel.

- Neural Networks likely to work well for most of these setting, but may be slower to train.

Performance Evaluation – Confusion Matrix

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

a: TP (true positive)

b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

Performance metric - Accuracy

	PREDICTED CLASS	
ACTUAL CLASS	Class=Yes	Class=No
Class=Yes	(TP)	(FN)
Class=No	(FP)	(TN)

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

Accuracy - fraction of examples correctly classified.

1-Accuracy: Error rate (misclassification rate)

Limitation of Accuracy

- Consider binary classification (**Unbalanced data set**)
  - Class 0 has 9990 examples
  - Class 1 has 10 examples

- If model classify all examples as class 0, accuracy is 9990/10000 = 99.9 %

- Accuracy is misleading because model does not classify correctly any example of class 1 => Need to find a way to balance the data set !!!

Other performance metrics

Sensitivity (recall) – true positive rate, of all positive examples the fraction of correctly classified

$$\text{Recall (r)} = \frac{TP}{TP + FN}$$

Specificity - true negative rate, of all negative examples the fraction of correctly classified

$$\text{Specificity(s)} = \frac{TN}{TN + FP}$$

Precision - the fraction of correctly classified positive samples from all classified as positive

$$\text{Precision (p)} = \frac{TP}{TP + FP}$$

F1 Score - weighted average of Precision and Recall

F1=2*(Recall * Precision) / (Recall + Precision)

Balanced Accuracy= (Recall+Specificity)/2

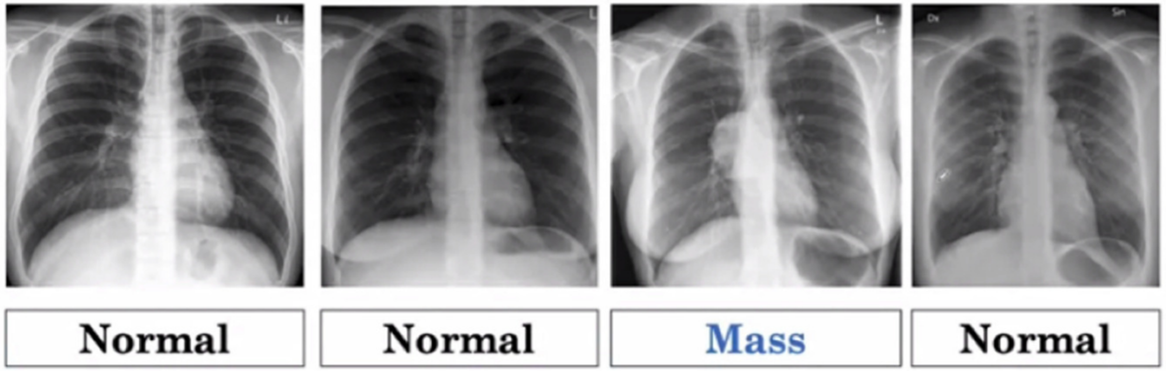
Performance metrics – example

	predicted	
	Positive	Negative
Positive	500	100
Negative	500	10000

- Accuracy $\frac{500+10000}{500+500+100+10000} = 0.95$
- Precision $\frac{500}{500+500} = 0.5$
- Recall $\frac{500}{500+100} = 0.83$
- Specificity $\frac{10000}{10000+500} = 0.95$

- Positive class is predicted poorly
- Accuracy is not a reliable measure for un-balanced datasets
- If # of examples of one class is much lower than # of examples of the other class => **F1 score and balanced accuracy are better measures.**

Class Imbalance problem



Solution 1: Weighted Binary Cross Entropy Loss

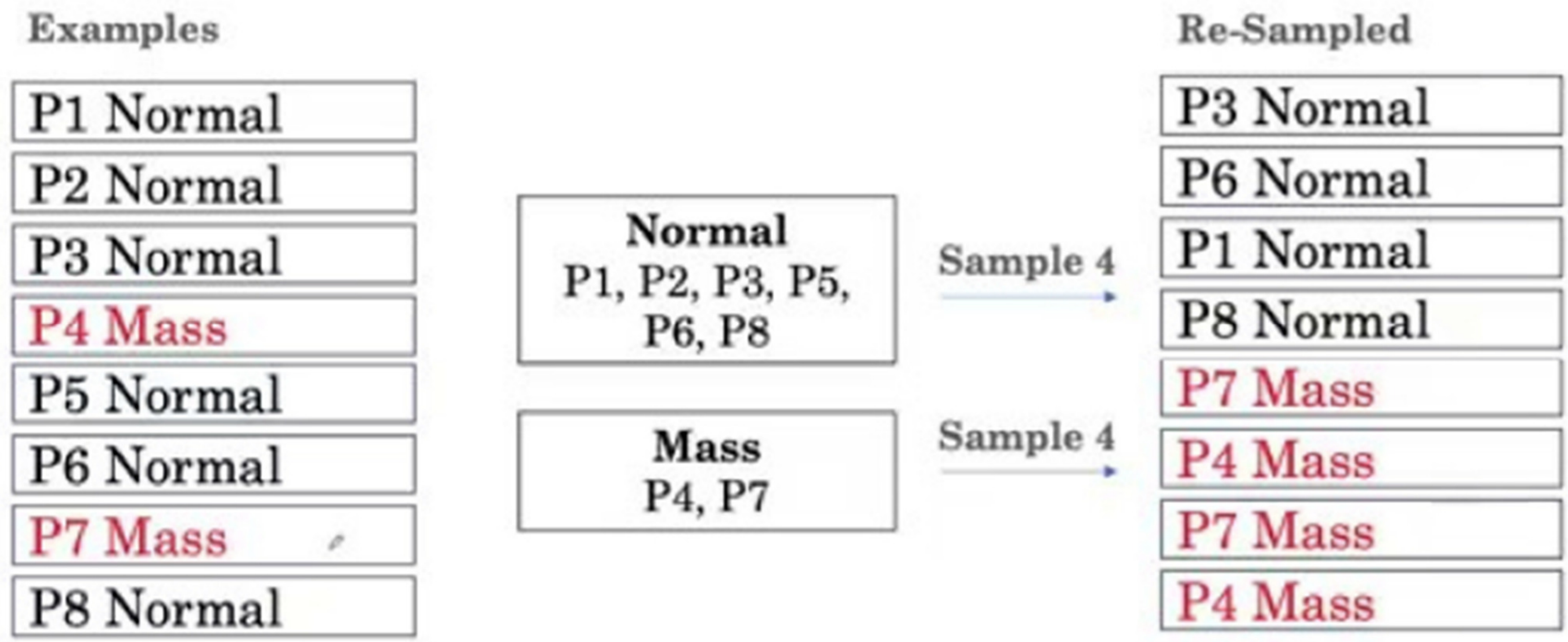
Weights:

$$w_p = \frac{\text{num negative}}{\text{num total}} \qquad w_n = \frac{\text{num positive}}{\text{num total}}$$

$$\mathcal{L}^{w}_{cross-entropy}(x) = -(w_p y \log(f(x)) + w_n(1-y)\log(1-f(x))).$$

Class Imbalance problem

Solution 2: Re-sampling methods (under-sampling, oversampling)



Epoch /Batch Size / Iterations / Train step

One Epoch is when an ENTIRE dataset is passed through the model (e.g. forward and backward in a neural network) only ONCE.
If data is too big to feed to the computer at once one epoch is divided in several smaller batches.

Batch Size: Total number of training examples present in a single batch.

Iterations is the number of batches needed to complete one epoch.

Example: Let's say we have 2000 training examples.
We can divide the dataset of 2000 examples into batches of 500 then it will take 4 iterations to complete 1 epoch.

Training run/step - is one update of the model parameters.
We update the parameters after one batch or after one epoch.

ML lab (part2) - Spam Detector

- Labelled data set : SpamAssassin Public Corpus

- Convert the email into a binary feature vector:

- Clean (remove slash , dots, coms)
- Tokenize (parse) into words
- Count the word frequency
- Create dictionary with most frequent words (e.g. 10000 to 50000 words) -

Feature space.

- Substitute the words with the dictionary indices.
- Extract binary features from emails - binary (sparse) feature for an email corresponds to whether the i-th word in the dictionary occurs in the email.

- **Apply classifier** (e.g. SVM)

Dictionary => Email with dictionary indices => binary features

```
1 aa
2 ab
3 abil
...
86 anyon
...
916 know
...
1898 zero
1899 zip
```

```
86 916 794 1077 883
370 1699 790 1822
1831 883 431 1171
794 1002 1893 1364
592 1676 238 162 89
688 945 1663 1120
1062 1699 375 1162
479 1893 1510 799
1182 1237 810 1895
1440 1547 181 1699
1758 1896 688 1676
992 961 1477 71 530
1699 531
```

$$x = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{R}^n$$

universidade de aveiro