Machine Learning and Artifical Intelligence

Lab 05 – SVMs and Evaluation Metrics

12/04/2022

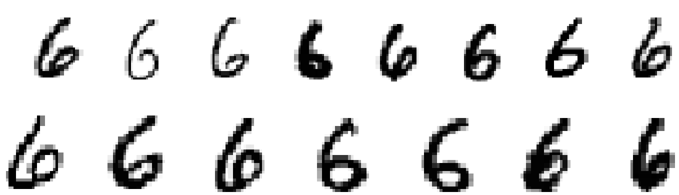
Practical problem

We want to recognize and classify images of handwritten digits:
https://en.wikipedia.org/wiki/MNIST_database

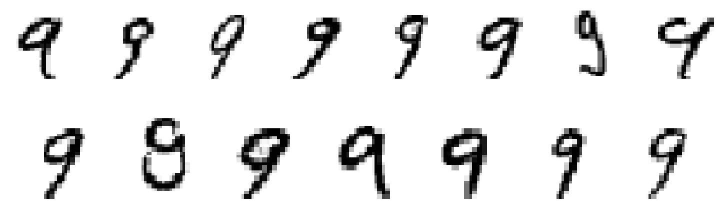
Let's consider this time images belonging to classes '6' and '9':

- They are quite similar between each other (rotation)

«6»



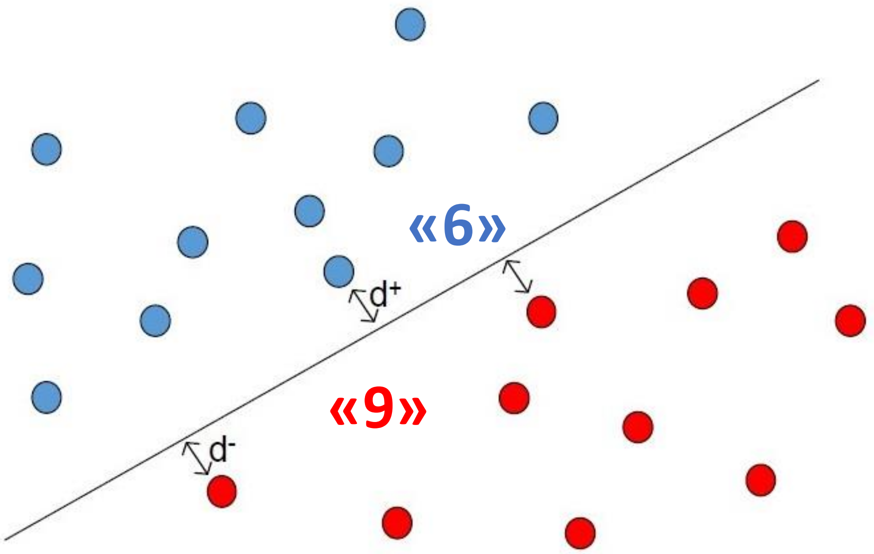
«9»



Support Vector Machines

- Find the separation hyperplane between the 2 classes in order to maximize the margin, i.e., the minimum distances d+ and d- between the nearest positive and negative samples respectively

Linearly Separable



- The lines containing the points with these margins (d+ and d-) are called support vectors

- Linearly classify new data x:

$$f(\mathbf{x}) = sign(\mathbf{w} \cdot \mathbf{x} + b)$$

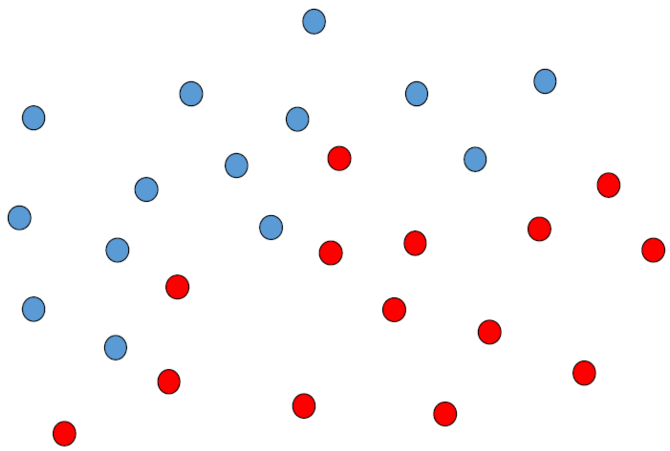
What if the problem is non-linear?

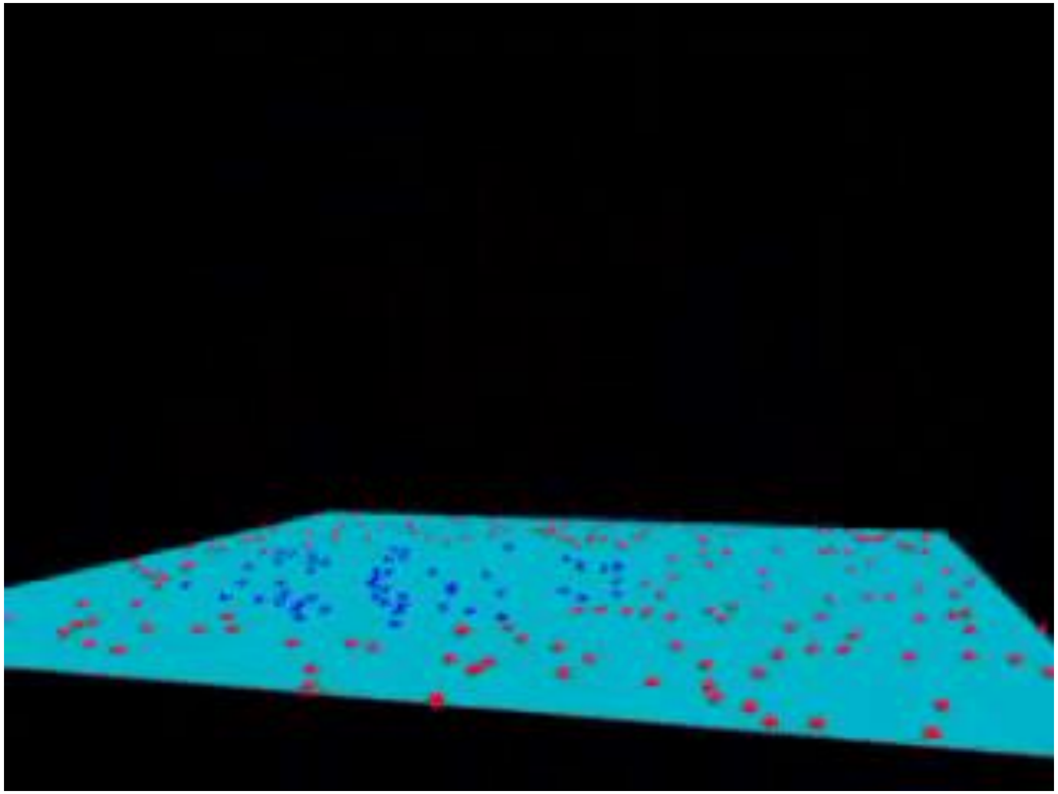
- We can't find a separation hyperplane that divides all the samples correctly!

- Introduction of Slack variables ξ_i, that allow some SV to exceed the margin, along and a parameter C indicating the cost for a wrong classification.

- Kernel trick: Map data in another, higher-dimensional space through functions called kernels.

Not Linearly Separable





Examples of kernel functions

Linear	$K(x,z) = \langle x,z \rangle$
Polynomial	$K(x,z) = \left(\langle x,z \rangle + 1\right)^p$
Radial basis functions	$K(x,z) = e^{-\frac{\lVert x-z \rVert^2}{2\sigma^2}}$
Sigmoid	$K(x,z) = \tanh(a\langle x,z \rangle + b)$

SVM in Scikit-learn

- The SVM classification algorithm can be implemented through scikit-learn (https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC)

- How?
  1. Model initialization (kernel specific): model = *SVC(...)*
  2. Model fit (training to get w and b): *model.fit()*
  3. Classification of test elements: *model.predict(x)*

Evaluation

- It is imperative to understand how a classifier behaves quantitatively.

- We need this information to:
  - *Have absolute feedback*: the goodness of a classifier/regressor.
  - *Have relative feedback*: the goodness of a classifier/regressor compared to another.

Classification evaluation metrics

- Accuracy: Number of correct predictions divided by the total number of predictions (dimensionality of the test set):

$$accuracy = \frac{\#\ correct\ classifications}{\#\ classifications}$$

- Error rate: Number of wrong predictions with respect to the total number of predictions:

$$error\_rate = \frac{\#\ incorrect\ classifications}{\#\ classifications}$$

Confusion Matrix

- It allows you to understand where exactly the classifier makes mistakes.
- Introduced initially for binary classification cases.
- Example:
  - Class A: Dog(positive)
  - Class B: Not-dog (negative)

Confusion matrix for binary classification

Actual value	A	TP	FN Type I error
	B	FP Type II error	TN
		A	B
		Predicted value	

Confusion Matrix – Construction

- With each prediction on the test set, I add +1 in the appropriate box (intersection between the index predicted by the classifier and the GT index)
- In the case of balanced classes, we can normalize the values of the matrix, so that the rows sum to one.
- It's convenient to normalize values by rows and get percentages. The absolute count does contain more information though.

Predicted

Real		P	N
	P	20 (0.66)	10 (0.33)
	N	5 (0.17)	25 (0.83)

Confusion Matrix - Metrics

- In the range [0,1]
- Accuracy $\dfrac{tp + tn}{tp + tn + fp + fn}$
- Precision $\dfrac{tp}{tp + fp}$
  - Portion of cases predicted as positive that actually are positive.
  - (If high) I take as positive only elements that actually are positive.
  - (If low) I say everything is positive.

Confusion Matrix - Metrics

- **Recall** (sensitivity)
  - Portion of all actually positive cases that have actually been classified as such
  - (If high) I don't lose positive elements
  - (If low) I lose positive elements

$$\frac{tp}{tp + fn}$$

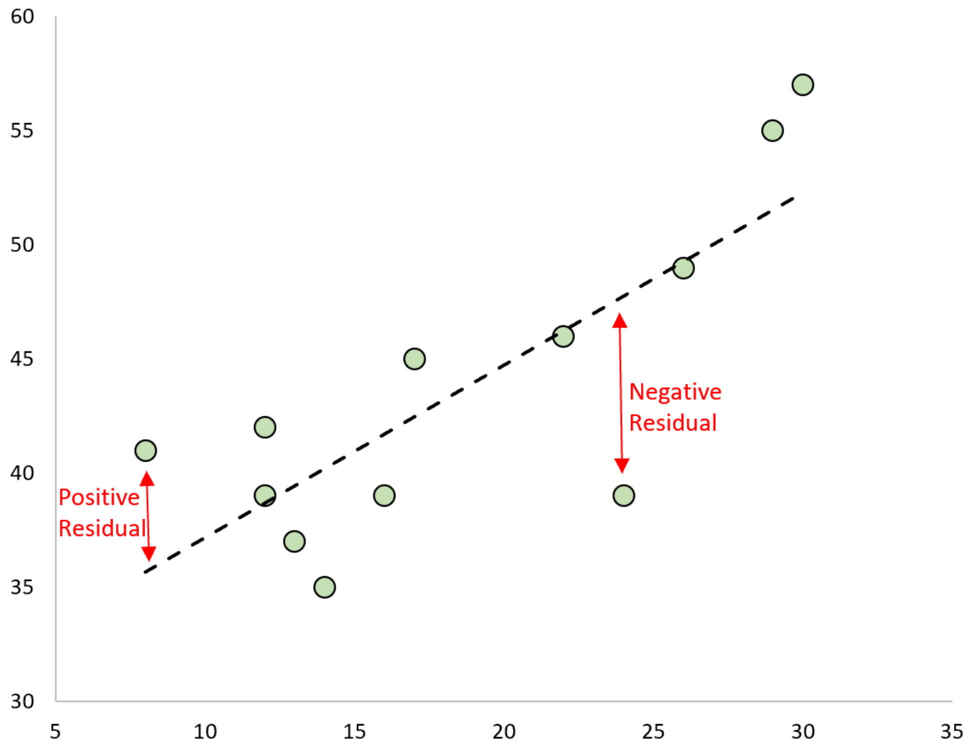
- **F-measure**
  - Combine precision and recall into one measure.

$$\textbf{\textit{F - measure}} = \frac{2*\text{Recall}*\text{Precision}}{\text{Recall} + \text{Precision}}$$

Regression tasks

- In machine learning, we are not limited to classification tasks, even though they are common.
- In regression tasks, the model learns to predict numeric scores, so the model output is a continuous variable.
- In such a case, the metrics we mentioned before cannot be applied, because we are more interested in the magnitude of the error.

Regression evalution metrics



$$\text{MAE} = \underbrace{\frac{1}{n}\sum_{i=1}^{n}}_{\text{test set}} |\underbrace{y_i}_{\text{predicted vaue}} - \underbrace{\hat{y}_i}_{\text{actual value}}|$$

$$MSE = \frac{1}{n}\sum\underbrace{\left(y - \hat{y}\right)^2}_{\text{The square of the difference between actual and predicted}}$$

$$RMSE = \sqrt{\sum_{i=1}^{n}\frac{(\hat{y}_i - y_i)^2}{n}}$$

Evaluation metrics available in Sklearn

https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics

Exercises

A slightly different scenario

- We must no longer classify only images belonging to classes '6' and '9' but all figures from '0' to '9'

From binary to multi-class

- How do we move from binary classification to multi-class classification?

- SVMs (and many other classifiers) don't support multi-class classification *natively*, we need to adopt different strategies.

- One vs Rest:
  - Train K different classifiers, one for each class.
  - Each of the classifiers considers a class as positive and the remaining as negative.
  - Another, less popular approach is One vs One

From binary to multi-class

- One vs Rest strategy:
    1. Given K classes, instantiate K different SVMs.
    2. Train each of the K classifiers to recognize a particular class (the 3rd classifier will be trained to recognize class 3 and so on...)
    3. Given the test elements, each of the K classifiers produces two probabilities:
        1. Of belonging to the k-th class
        2. Of not belonging to the k-th class
    4. The class corresponding to the test element is the one with the highest probability of membership.
    5. P.S: To output probabilities you must use the *‘predict_proba’* function in the Sklearn library.

Confusion Matrix – K classes

Predicted

Real	bend	run	side	wave2	wave1	walk	skip	pjump	jump	jack
bend	0.99	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00
run	0.00	0.84	0.00	0.00	0.00	0.03	0.12	0.00	0.00	0.00
side	0.00	0.00	0.98	0.00	0.00	0.00	0.00	0.00	0.01	0.00
wave2	0.00	0.00	0.00	0.98	0.02	0.00	0.00	0.00	0.00	0.00
wave1	0.02	0.00	0.00	0.01	0.97	0.00	0.00	0.00	0.00	0.00
walk	0.00	0.01	0.00	0.00	0.00	0.98	0.00	0.00	0.00	0.00
skip	0.00	0.16	0.00	0.00	0.00	0.02	0.77	0.00	0.06	0.00
pjump	0.01	0.00	0.01	0.00	0.00	0.00	0.00	0.96	0.00	0.03
jump	0.00	0.00	0.03	0.00	0.00	0.01	0.06	0.00	0.90	0.00
jack	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.00	0.94

Confusion Matrix – K classes

- Precision and Recall are associated with *a single class.*

- Given the confusion matrix C x C

  - Precision: $$\frac{tp}{tp+fp} = \frac{Conf(c,c)}{\sum_d Conf(d,c)}$$

  - Recall $$\frac{tp}{tp+fn} = \frac{Conf(c,c)}{\sum_d Conf(c,d)}$$

Exercises