Support Vector Classifier (SVC)

We introduce another model for the Classification task: the Support Vector Classifier (SVC).

We motivate this classifier by some examples.

Consider the following linearly separable dataset and two separating lines:

Linear separating boundary

Is one separating line "better" than the other ?

Intuitively,the first line feels more fragile

- The distance between the line and the closest example is smaller in the first plot than the second
- If the red point located exactly on the line moved a small amount, we would mis-classify it
- Not as likely to generalize out of sample as well

Something else to consider:

- How sensitive is our classifier to additional examples *that are far from the original boundary*

Consider the two rows in the plot below

- The second plot in each row adds a cluster of examples on the right side, mid-way

Sensitivity to far-from-boundary examples

As you can see

- Logistic Regression is sensitive
- SVC is not

Why should points far away from the original separating boundary affect the fit ?

The answer, of course, is the Loss function.

We will see that the SVC uses a much different loss that has several advantages.

The SVC introduces two additional separating lines that are parallel to the separating boundary

- At a distance m on either side of the boundary
    - measured by length of a line orthogonal to the boundary
- m is called the *margin*
- These two lines define a "buffer" of width twice the margin

We draw two plots with different size margins

Margin

The concept of margin helps to address the two issues raised above.

- Given a choice of two separating boundaries (each with perfect separation)
    - The one with the larger margin is "better"
- An example that is correctly classified and lies *outside* the buffer
- Should not affect the Loss Function

From the above examples

- It may not be possible to have both
    - A large margin and
    - A boundary that separates classes perfectly
    - The left plot, for example

Requiring perfect separation and no examples in the buffer is called *Hard Margin* classification.

- Allowing either condition to be false is called *Soft Margin* classification

A *Soft Margin* classifier allows violations but imposes a *penalty* (by increasing the Loss).

An SVC classifier is a Soft Margin classifier.

The main difference between an SVC and Logistic Regression classifier

- Different Loss functions

The SVC Loss functions differs from Cross Entropy (used in Logistic Regression) in that

- There are examples with **zero** loss
- Those that are correctly classified and outside the buffer

We will dig into the loss function for the SVC shortly.

Support Vector Machines (SVM)

We have seen that many datasets, in raw form, are not linearly separable.

Transformations of the raw data must be applied to induce Linear Separability.

A *Support Vector Machine* combines

- Transformations specialized for the mathematics of an SVC
- An SVC

There is a special subclass of transformation functions called **kernel functions** that

- Uses a clever mathematical trick
- To achieve the effect of applying an expensive transformation
- Without actually creating transformed data !

The SVM helps to automate the transformations.

Key concepts

Just like Logistic Regression, the SVC will:

- Use the features $\mathbf{x}^{(\mathbf{i})}$ to compute a "score" $\hat{s}^{(\mathbf{i})}$
- Compare the predicted score to a threshold
- Predict Positive if the score exceeds the threshold; Negative otherwise

The score is linear in the features

$$s(\mathbf{x}) = \Theta^T \mathbf{x} \quad \text{score}$$

$$s(\mathbf{x}) = 0 \qquad \text{equation of separating boundary}$$

$$\hat{\mathbf{y}}^{(\mathbf{i})} = \begin{cases} \text{Negative} & \text{if } \hat{s}^{(\mathbf{i})} < 0 \\ \text{Positive} & \text{if } \hat{s}^{(\mathbf{i})} \geq 0 \end{cases}$$

The SVC and SVM incorporate several "tricks"

- A clever Loss Function (Hinge Loss)
- Large Margin Classification
- Kernel Transformations

The combined effect can be overwhelming and makes the SVC/SVM seem complex

- and it *does* involve a bit of math

We will try to reduce the complexity by tackling each trick separately.

In [5]:
```
print("Done")
```

```
Done
```