Introduction to Machine Learning

Lecture 2: Classification

Alexis Zubiolo
alexis.zubiolo@gmail.com

Data Science Team Lead @ Adcash

October 20, 2016

Classification in Machine Learning

This lecture is about classification in Machine learning.

Reminder: In classification, the output y is **categorical**.

Examples:

- Mail classification: $y =$ spam or $y =$ not spam
- Object recognition: $y =$ apple, $y =$ car, $\ldots$

Classification in Machine Learning: Applications

Domains of application:

- Medicine
- Image/video classification
- Face recognition/identification
- Spam filtering
- Fraud detection
- Click prediction
- Product recommendation
- Robotics
- Language processing
- Web search
- ...and many others

Support Vector Machines, Decision Trees

In this course, we will see

- Support Vector Machines (SVMs)
- Decision Trees (DTs) / Boosting

Why? Because the concepts behind these classifiers are quite intuitive.

Support Vector Machines

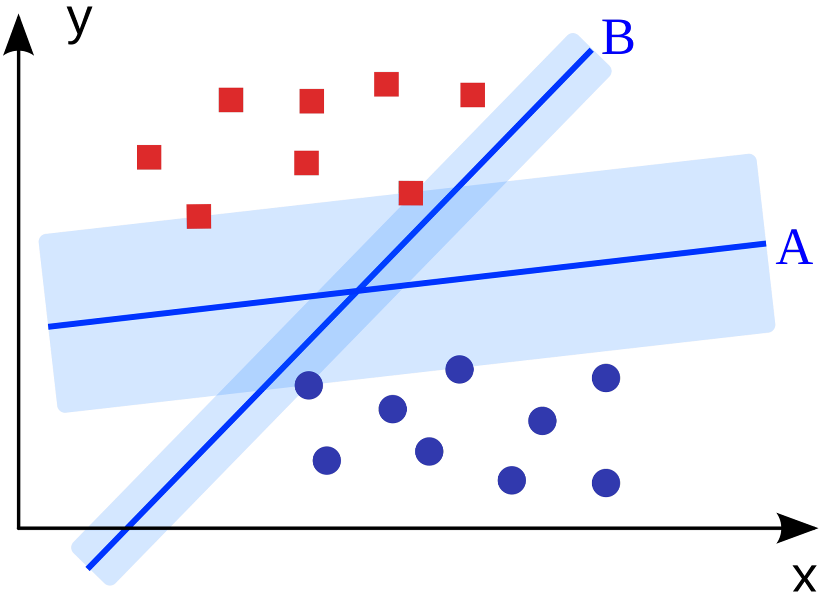
Support Vector Machines: History and intuition

A few historical facts about SVMs:

- First introduced by **Cortes and Vapnik** in 1995 from AT&T Bell labs (paper called Support-Vector Networks)
- Quickly became state of the art in many areas
- Received a lot of attention since then
- Still a widely used classifier

Intuitive idea behind SVMs: Find the linear separator with the widest margin.

SVM: Widest margin



SVM: Widest margin

Among all the possible separators, SVM chooses the one **with the widest margin**.

SVM: Widest margin

Among all the possible separators, SVM chooses the one **with the widest margin**.

Why? Intuitively, a wide margin leads to a good **generalization of the classifier** on new points.

SVM: The kernel trick

So far, the SVM we have seen is a *linear classifier*: It aims at finding a linear separation between the 2 classes. What if they are **not linearly separable**?

Example: See example on the board

What to do in this case?

SVM: The kernel trick

So far, the SVM we have seen is a *linear classifier*: It aims at finding a linear separation between the 2 classes. What if they are **not linearly separable**?

Example: See example on the board

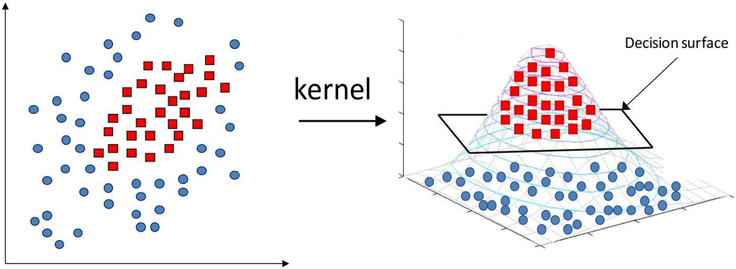
What to do in this case? The **kernel trick** is an option.

SVM: The kernel trick

The kernel trick consists in **mapping the features** x **to a space of higher dimensionality** where the data is (hopefully) linearly separable.

SVM: The kernel trick

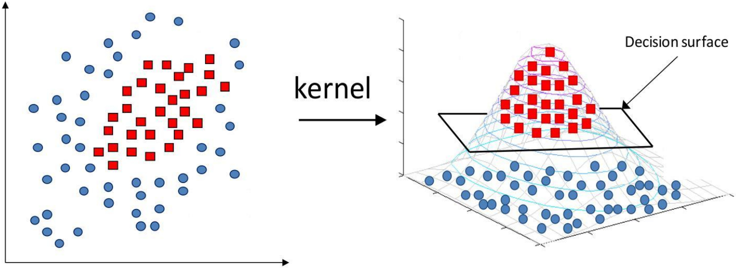
The kernel trick consists in **mapping the features** x **to a space of higher dimensionality** where the data is (hopefully) linearly separable.



Most commonly used kernels: **Gaussian** (aka RBF) and **polynomial** kernels.

SVM: The kernel trick

The kernel trick consists in **mapping the features x to a space of higher dimensionality** where the data is (hopefully) linearly separable.



Most commonly used kernels: **Gaussian** (aka RBF) and **polynomial** kernels.

Note: Adding a kernel leads to **extra parameters**.

SVM: Soft margin

Not tolerating points within the margin may lead to really **“thin” separators**, and then to **bad generalization**.

SVM: Soft margin

Not tolerating points within the margin may lead to really **"thin" separators**, and then to **bad generalization**.

It is possible to tolerate some violation of the margin. The idea is to find a **proper trade-off** between width of the margin and violation tolerance.

SVM: Soft margin

Not tolerating points within the margin may lead to really **"thin" separators**, and then to **bad generalization**.

It is possible to tolerate some violation of the margin. The idea is to find a **proper trade-off** between width of the margin and violation tolerance.

Ideally, we want

- a wide margin
- little to no violation of the margin

SVM: Soft margin

Not tolerating points within the margin may lead to really **“thin” separators**, and then to **bad generalization**.

It is possible to tolerate some violation of the margin. The idea is to find a **proper trade-off** between width of the margin and violation tolerance.

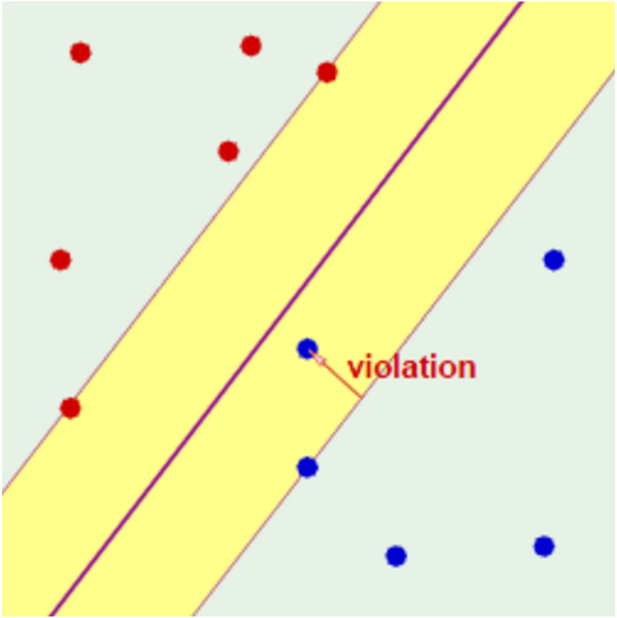
Ideally, we want

- a wide margin
- little to no violation of the margin

but in practice, it is almost not possible, for example because:

- The data may not be linearly separable (even after the kernel trick)
- There is noise in the data

SVM: Soft margin



SVM: Recap

SVM consists in:

- Finding the widest separator
- Using kernels if the data is not linearly separable
- Using soft margin

It has parameters that need to be tuned:

- Soft margin trade-off: C
- Kernel parameter:
  - Degree d for polynomial kernels
  - The standard deviation σ for RBF kernels

SVM

Interactive demonstration (c) Andrej Karpathy

SVM for multiclass classification

What we have seen so far works for *binary classification* (2 classes). What if we have **3 classes or more**?

SVM for multiclass classification

What we have seen so far works for *binary classification* (2 classes). What if we have **3 classes or more**?

Several possible extensions. Some of the most popular ones:

- One against one classification
- One against all classification

Note: These strategies apply to any binary classifier.

One against one classification

Idea:

- Compute a classifier **for all pairs** of classes.
- **Apply all these classifiers** to the new point. Store the predicted classes.
- **Majority vote**: Return the class with the highest number of votes

One against all classification

Note: It is also called *one against rest classification*.

Idea: For each class, split the set of classes into two meta classes

- The considered class
- The union of all the other classes

and compute a classifier for all these possibilities. Apply all these classifiers to a (new) given point.

Final decision: More complicated than for one-versus-one

Decision Trees

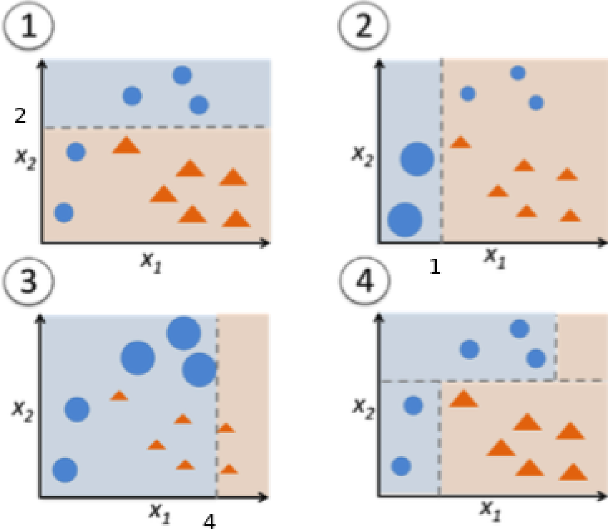
Decision Trees / Boosting

Note: DTs and Boosting are different algorithms, but they share some common ideas.

Idea: **Combining weak classifiers** to get a more robust classifier.

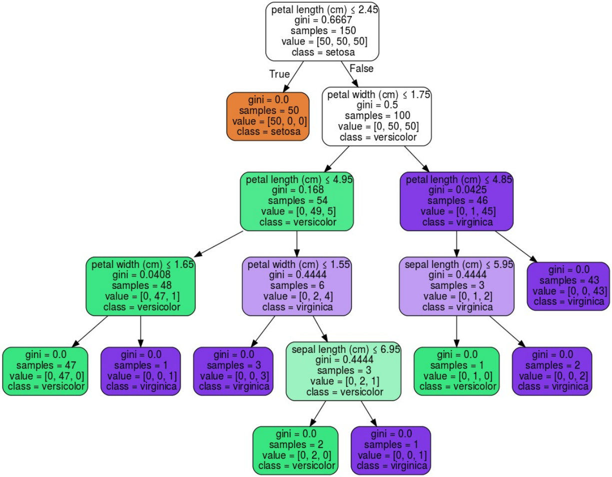
How? By applying these *weak* classifier successively.

DTs/Boosting: Toy example



(step 4 is optional). Try to guess the resulting tree.

DTs: visualization with sklearn



Decision Trees

Interactive demonstration (c) Andrej Karpathy

DTs/boosting recap

Goal: Build a tree of simple classifier to obtain a more sophisticated classifier.
There are **several parameters** that have to be set:

- The **number of trees**
- The **maximum depth** of the trees
- The **number of decision** per node

Conclusion

We've seen two popular classifiers:

- Support Vector Machines
- Decision Trees

A few remarks before we finish:

- These classifiers (and others as well) rely on a few parameters that have **a huge impact** on the quality of the resulting classifier. Example: soft-margin parameter for SVMs, number of trees for DTs.
- Parameters others than accuracy have to be taken into account when choosing the parameters. Example: More tree may lead to a better classifier but it involves more computation also. A proper trade-off has to be found depending on the application.

Thank you! Questions?