# INTRO TO DATA SCIENCE

## LECTURE 11: SUPPORT VECTOR MACHINES

# LAST TIME:

- ENSEMBLE TECHNIQUES
- PROBLEMS IN CLASSIFICATION
- BAGGING, BOOSTING, RANDOM FORESTS

I. SUPPORT VECTOR MACHINES
II. NONLINEAR CLASSIFICATION
III. MAXIMUM MARGIN HYPERPLANES
IV. SLACK VARIABLES

EXERCISE:
V. SVM IN SCIKIT-LEARN

# I. SUPPORT VECTOR MACHINES

Q: What is a support vector machine?

Q: What is a support vector machine?

A: A binary linear classifier whose decision boundary is *explicitly* constructed to minimize generalization error.

Q: What is a support vector machine?

A: A binary linear classifier whose decision boundary is *explicitly* constructed to minimize generalization error.

recall:

**binary classifier** – solves two-class problem

**linear classifier** – creates linear decision boundary (in 2d)

Q: How is the decision boundary derived?

Q: How is the decision boundary derived?

A: Using *geometric reasoning* (as opposed to the algebraic reasoning we've used to derive other classifiers).

Q: How is the decision boundary derived?
A: Using *geometric reasoning* (as opposed to the algebraic reasoning we've used to derive other classifiers).

The generalization error is equated with the geometric concept of **margin**, which is the region along the decision boundary that is free of data points.

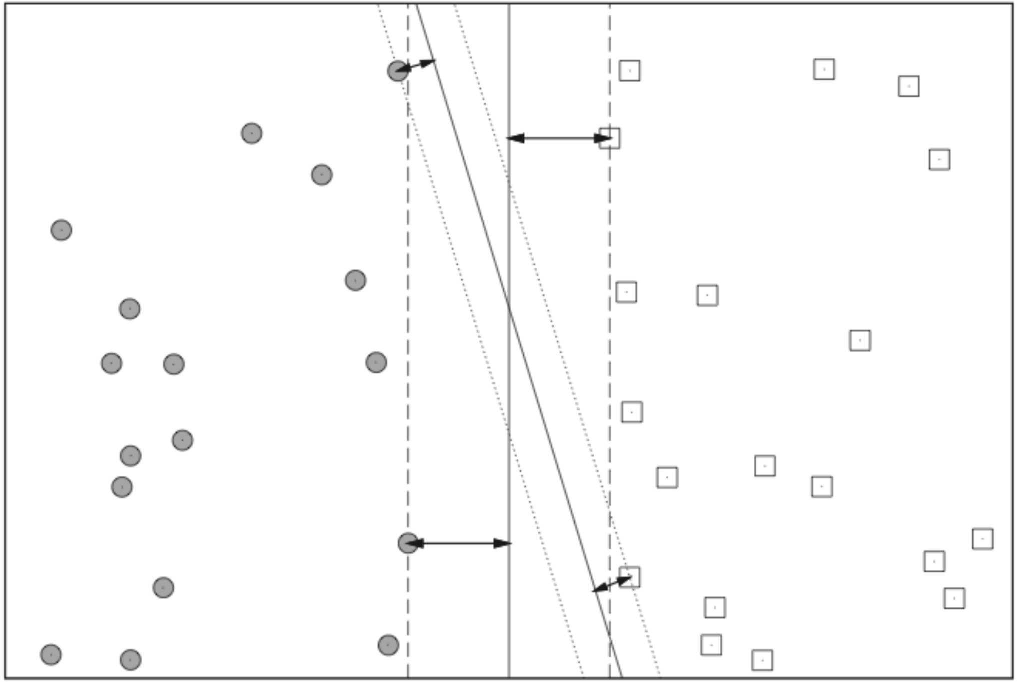


*FIGURE 18-4. Two decision boundaries and their margins. Note that the vertical decision boundary has a wider margin than the other one. The arrows indicate the distance between the respective support vectors and the decision boundary.*

*source: Data Analysis with Open Source Tools, by Philipp K. Janert. O'Reilly Media, 2011.*

Q: How is the decision boundary derived?
A: Using *geometric reasoning* (as opposed to the algebraic reasoning we've used to derive other classifiers).

The goal of an SVM is to create the linear decision boundary with the largest margin. This is commonly called the **maximum margin hyperplane**.

Q: How is the decision boundary derived?

A: Using *geometric reasoning* (as opposed to the algebraic we've used to derive other classifiers).

> NOTE
>
> A *hyperplane* is just a high-dimensional generalization of a line.

The goal of an SVM is to create the linear decision boundary with the largest margin. This is commonly called the **maximum margin hyperplane**.

Q: If SVM is a linear classifier, how can you use it for nonlinear classification?

Q: If SVM is a linear classifier, how can you use it for nonlinear classification?

A: Using a clever maneuver called the **kernel trick**.

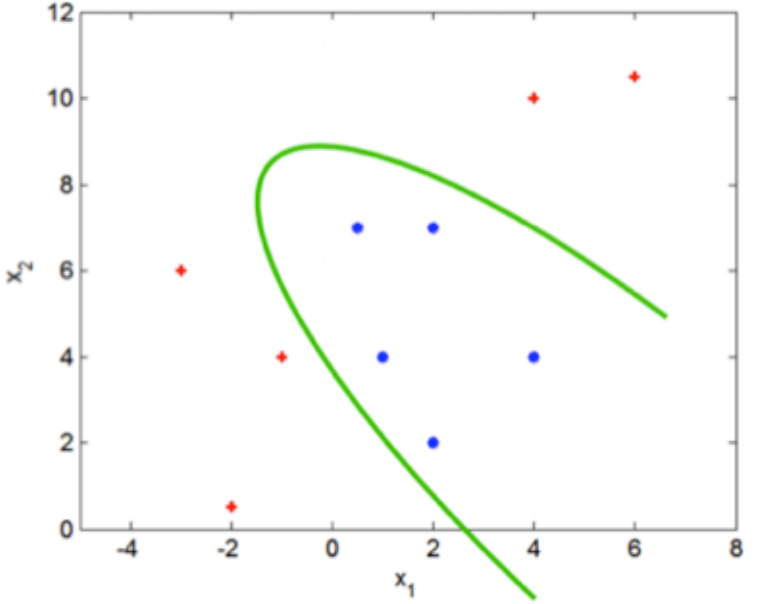
Nonlinear applications of SVM rely on an implicit (nonlinear) mapping $\Phi$ that sends vectors from the original feature space $K$ into a higher-dimensional feature space $K$'.

Nonlinear applications of SVM rely on an implicit (nonlinear) mapping $\Phi$ that sends vectors from the original feature space $K$ into a higher-dimensional feature space $K$'.

Nonlinear classification in $K$ is then obtained by creating a linear decision boundary in $K$'.

# II. NONLINEAR CLASSIFICATION

Suppose we need a more complex classifier than a linear decision boundary allows.



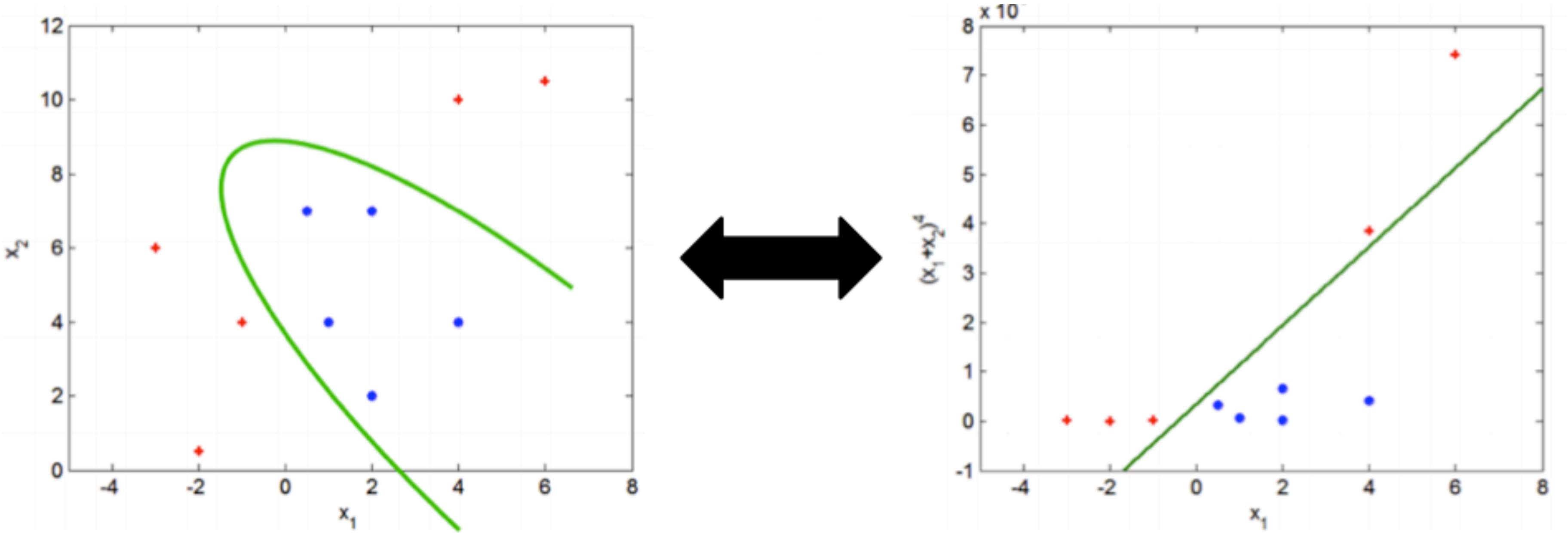
Suppose we need a more complex classifier than a linear decision boundary allows.

One possibility is to add nonlinear combinations of features to the data, and then to create a linear decision boundary in the enhanced (higher-dimensional) feature space.

Suppose we need a more complex classifier than a linear decision boundary allows.

One possibility is to add nonlinear combinations of features to the data, and then to create a linear decision boundary in the enhanced (higher-dimensional) feature space.

This *linear* decision boundary will be mapped to a *nonlinear* decision boundary in the original feature space.




original feature space *K*

higher-dim feature space *K'*

The logic of this approach is sound, but there are a few problems with this version.

The logic of this approach is sound, but there are a few problems with this version.

In particular, this will not scale well, since it requires many high-dimensional calculations.

The logic of this approach is sound, but there are a few problems with this version.

In particular, this will not scale well, since it requires many high-dimensional calculations.

It will likely lead to more complexity (both modeling complexity and computational complexity) than we want.

Let’s hang on to the logic of the previous example, namely:

Let's hang on to the logic of the previous example, namely:

- remap the feature vectors $x_i$ into a higher-dimensional space $K$'
- create a linear decision boundary in $K$'
- back out the nonlinear decision boundary in $K$ from the result

some popular kernels:

linear kernel $k(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle$

polynomial kernel $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}' + 1)^d$

Gaussian kernel $k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma ||\mathbf{x} - \mathbf{x}'||^2)$

some popular kernels:

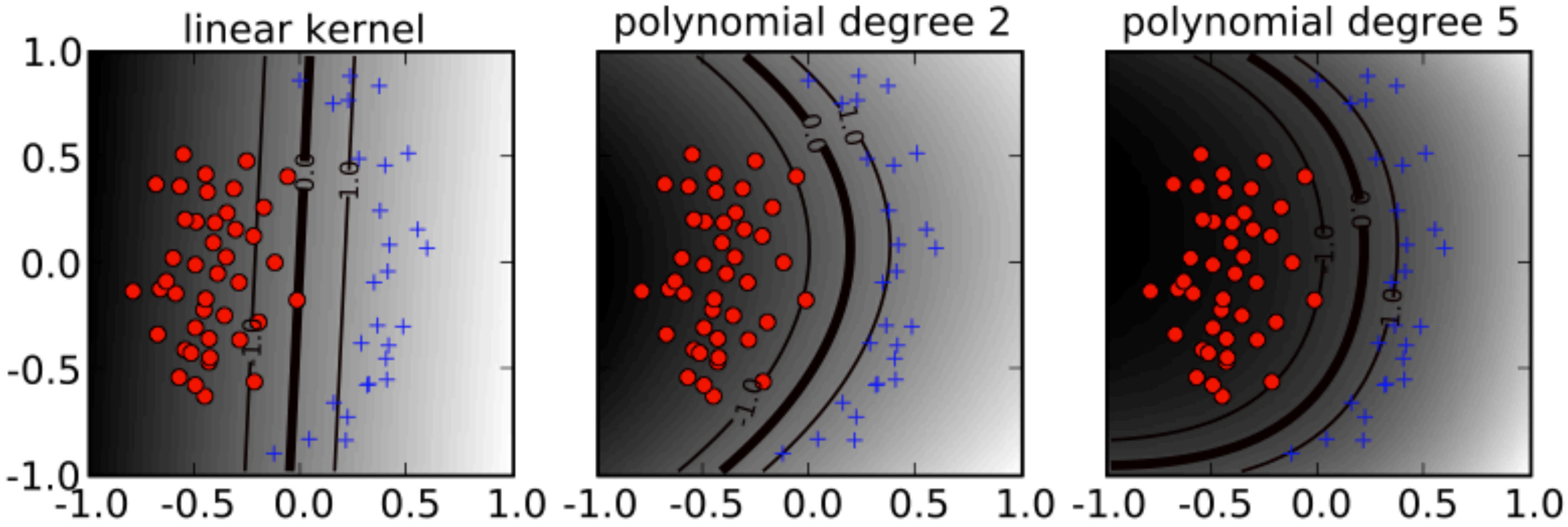
linear kernel $k(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle$

polynomial kernel $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}' + 1)^d$

Gaussian kernel $k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma ||\mathbf{x} - \mathbf{x}'||^2)$

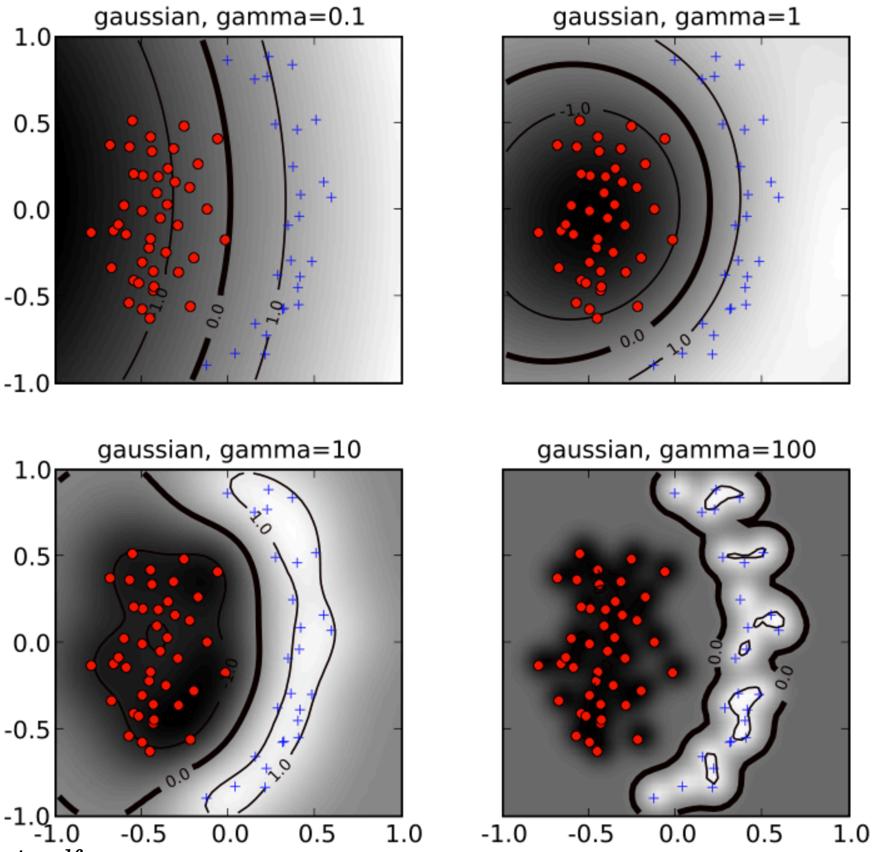
The **hyperparameters** $d$, $\gamma$ affect the flexibility of the decision plane.

# NONLINEAR CLASSIFICATION – POLYNOMIAL KERNEL




source: *http://pyml.sourceforge.net/doc/howto.pdf*

# NONLINEAR CLASSIFICATION – GAUSSIAN KERNEL




source: http://pyml.sourceforge.net/doc/howto.pdf

# III. MAXIMUM MARGIN HYPERPLANES

Q: How is the decision boundary (maximum margin hyperplane) derived?

Q: How is the decision boundary (maximum margin hyperlane) derived?

A: By the **discriminant function**,

$$f(\mathbf{x}) = \mathbf{w}^{\mathsf{T}}\mathbf{x} + b.$$

$$(OR\ f(x) = \beta_1 x_1 + \ldots + \beta_n x_n + b)$$

such that w is the *weight vector* and b is the *bias*.

Q: How is the decision boundary (maximum margin hyperlane) derived?

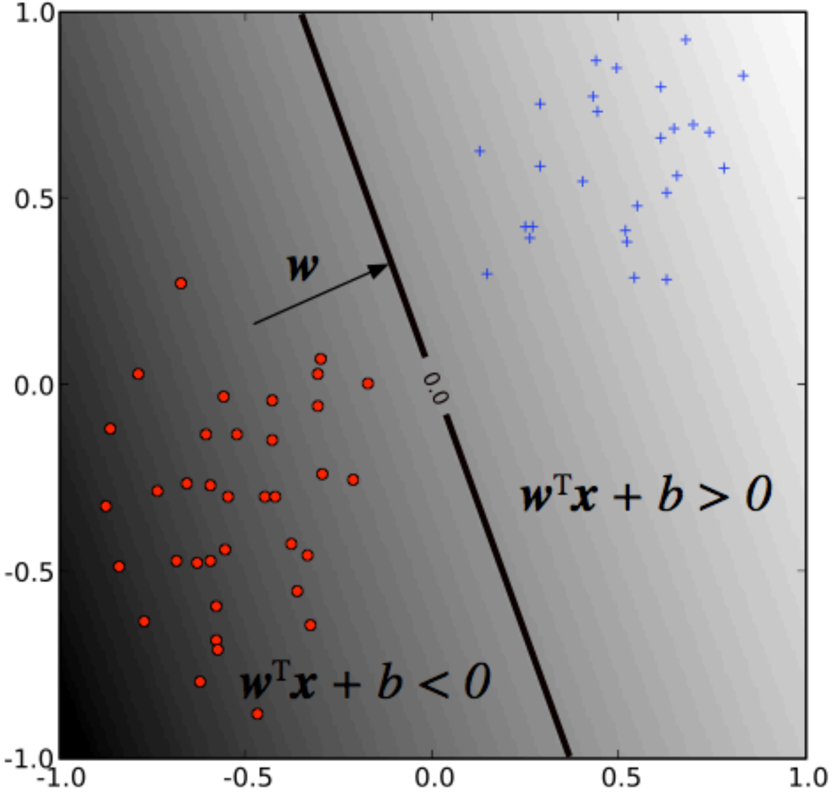
A: By the **discriminant function**,

$$f(\mathbf{x}) = \mathbf{w}^{\mathsf{T}}\mathbf{x} + b.$$

such that w is the *weight vector* and b is the *bias*.

*The sign of f(x) determines the (binary) class label of a record x.*

# MAXIMUM MARGIN HYPERPLANES




*source: http://pyml.sourceforge.net/doc/howto.pdf*

As we said before, SVM solves for the decision boundary that minimizes generalization error, or equivalently, that has the maximum margin.

Q: Why are these the same thing?

As we said before, SVM solves for the decision boundary that minimizes generalization error, or equivalently, that has the maximum margin.

Q: Why are these the same thing?
A: Because using the mmh as the decision boundary minimizes the probability that a small perturbation in the position of a point produces a classification error.

As we said before, SVM solves for the decision boundary that minimizes generalization error, or equivalently, that has the maximum margin.

NOTE

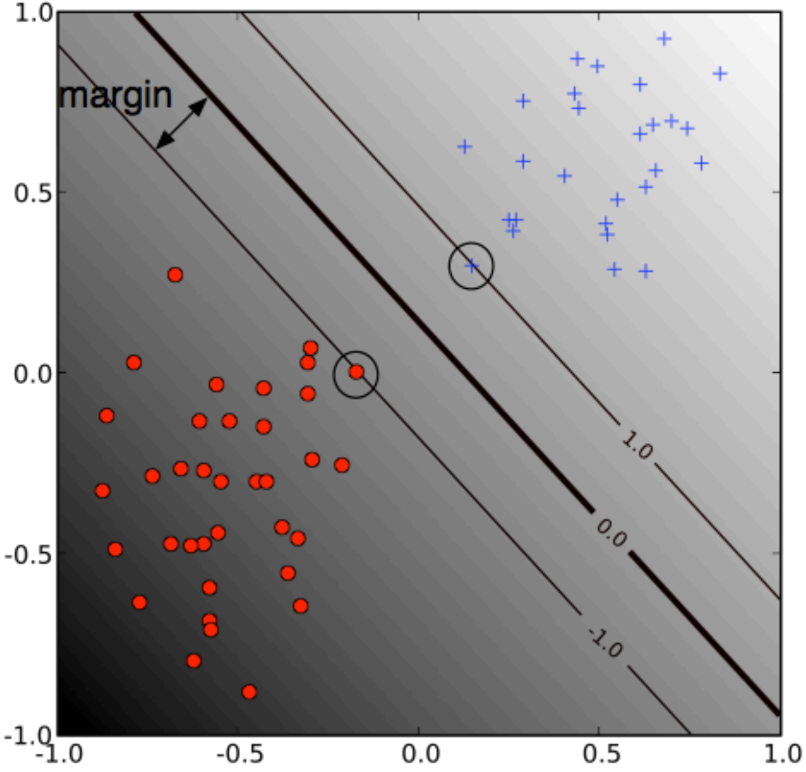
Intuitively, the wider the margin, the clearer the distinction between classes.

Q: Why are these the same thing?

A: Because using the mmh as the decision boundary minimizes the probability that a small perturbation in the position of a point produces a classification error.

Notice that the margin depends only on a *subset* of the training data; namely, those points that are nearest to the decision boundary.

# MAXIMUM MARGIN HYPERPLANES




*source: http://pyml.sourceforge.net/doc/howto.pdf*

Notice that the margin depends only on a *subset* of the training data; namely, those points that are nearest to the decision boundary.

These points are called the **support vectors**.

Notice that the margin depends only on a *subset* of the training data; namely, those points that are nearest to the decision boundary.

These points are called the **support vectors**.

The other points (far from the decision boundary) don't affect the construction of the mmh at all!

All of the decision boundaries we've seen so far have split the data perfectly; eg, the data are **linearly separable**, and therefore the training error is 0.

All of the decision boundaries we've seen so far have split the data perfectly; eg, the data are **linearly separable**, and therefore the training error is 0.

The optimization problem that this SVM solves is:

$$\underset{\mathbf{w},b}{\text{minimize}} \qquad \frac{1}{2}||\mathbf{w}||^2$$

$$\mathbf{w}\cdot\mathbf{x}_i - b \geq 1 \qquad \text{for } \mathbf{x}_i \text{ of the first class}$$

or

$$\mathbf{w}\cdot\mathbf{x}_i - b \leq -1 \qquad \text{for } \mathbf{x}_i \text{ of the second.}$$

All of the decision boundaries we've seen so far have split the data perfectly; eg, the data are **linearly separable**, and therefore the training error is 0.

The optimization problem that this SVM solves is:

$$\begin{aligned} &\underset{\mathbf{w},b}{\text{minimize}} && \tfrac{1}{2}||\mathbf{w}||^2 \\ &\text{subject to:} && y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 \quad i = 1, \ldots, n\,. \end{aligned}$$

# IV. SLACK VARABLES

Recall that in building the hard margin classifier, we assumed that our data was **linearly separable** (eg, that we could perfectly classify each record with a linear decision boundary).

Recall that in building the hard margin classifier, we assumed that our data was **linearly separable** (eg, that we could perfectly classify each record with a linear decision boundary).

Suppose that this was not true, or suppose that we wanted to use a larger margin at the expense of incurring some training error.

Recall that in building the hard margin classifier, we assumed that our data was **linearly separable** (eg, that we could perfectly classify each record with a linear decision boundary).

Suppose that this was not true, or suppose that we wanted to use a larger margin at the expense of incurring some training error.

This can be done using by introducing **slack variables.**

Slack variables $\xi_i$ generalize the optimization problem to permit some misclassified training records (which come at a cost $C$).

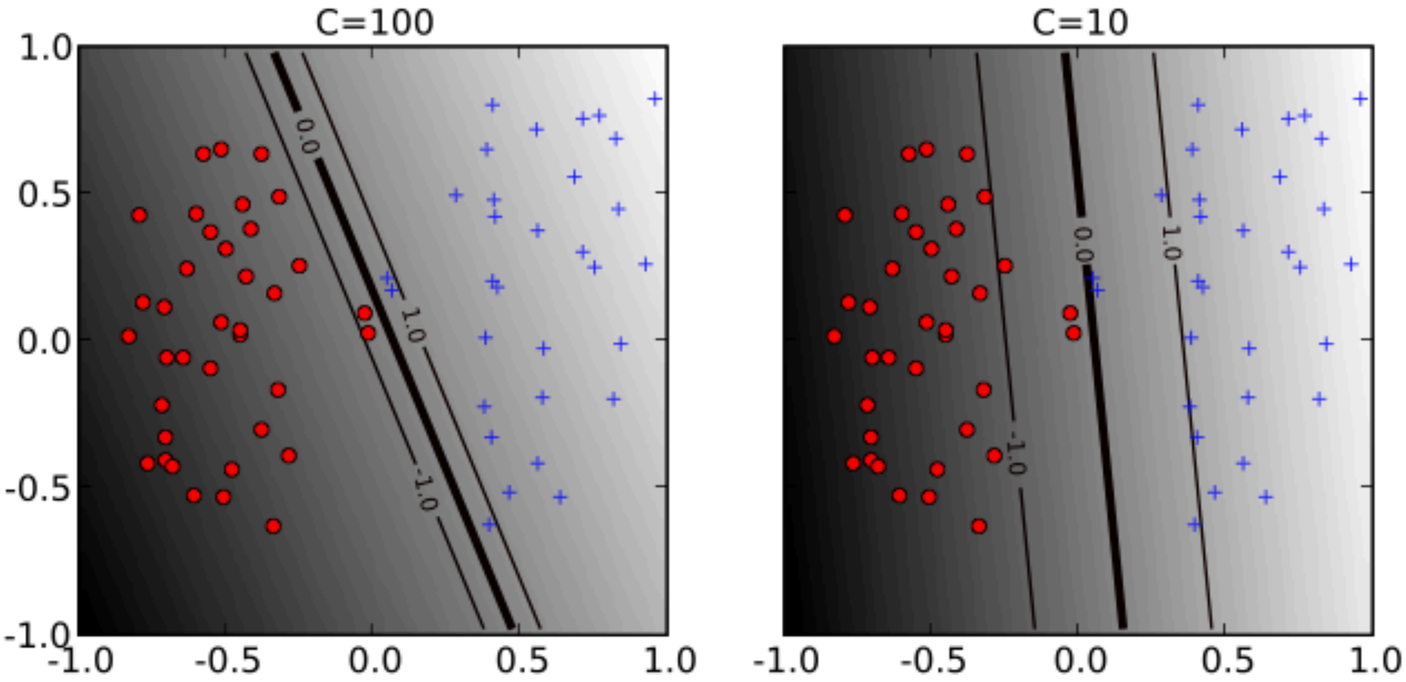
Slack variables $\xi_i$ generalize the optimization problem to permit some misclassified training records (which come at a cost $C$).

The resulting **soft margin classifier** is given by:

$$\underset{\mathbf{w},b}{\text{minimize}} \qquad \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n} \xi_i$$

$$\text{subject to:} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0.$$

# SLACK VARIABLES – SOFT MARGIN CONSTANT




*source: http://pyml.sourceforge.net/doc/howto.pdf*

The resulting **soft margin classifier** is given by:

$$\begin{aligned} &\underset{\mathbf{w},b}{\text{minimize}} && \tfrac{1}{2}||\mathbf{w}||^2 + C\textstyle\sum_{i=1}^{n}\xi_i \\ &\text{subject to:} && y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0. \end{aligned}$$

How do we compute this?

Quadratic programming optimization technique

There is a unique W which gives the best separation

SVMs (and **kernel methods** in general) are versatile, powerful, and popular techniques that can produce accurate results for a wide array of classification problems.

The main disadvantage of SVMs is the lack of intuition they produce. These models are truly black boxes!

Advantages:

Represent non-linear representations

Disadvantages:

Difficult to train –

- Difficult to choose hyperparameter
- Long training times

# EX: SVM IN SCIKIT-LEARN