

# **INTRO to DATA SCIENCE**

## **LECTURE 12: SUPPORT VECTOR MACHINES**

**YUCHEN ZHAO / DAT-14**

## LAST TIME:

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

**I. SUPPORT VECTOR MACHINES**

**II. MAXIMUM MARGIN HYPERPLANES**

**III. SLACK VARIABLES**

**IV. NONLINEAR CLASSIFICATION**

**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).*

**NOTE**

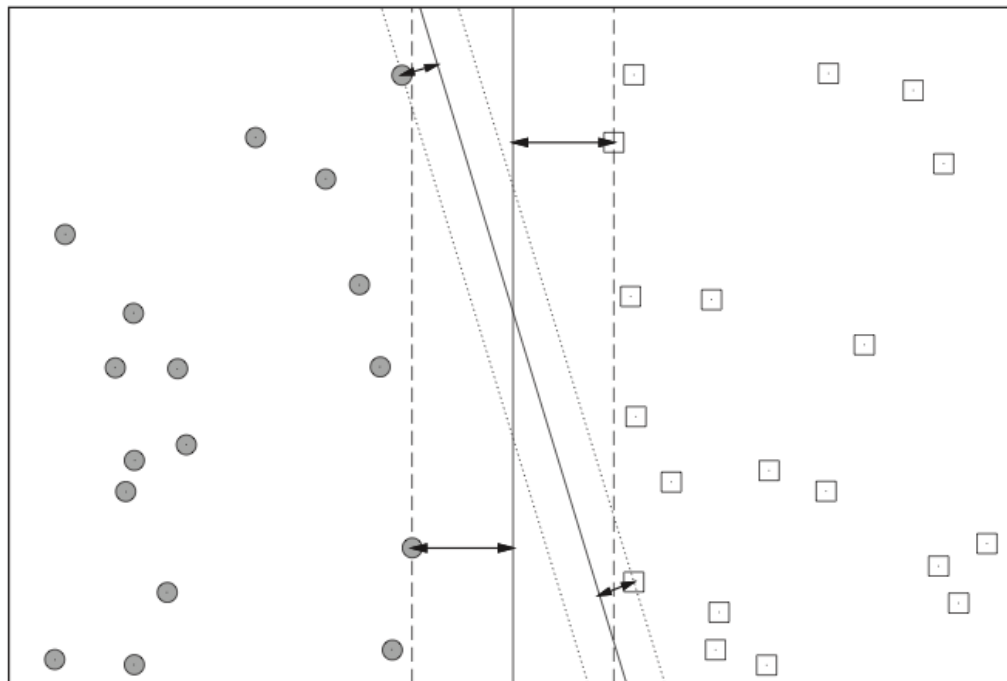
These are two different ways of looking at the same problem.

Familiarity with both leads to deeper understanding!

*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.*

*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 (MMH)**.*

*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  $\mathbb{K}$  into a higher-dimensional feature space  $\mathbb{K}'$ .*

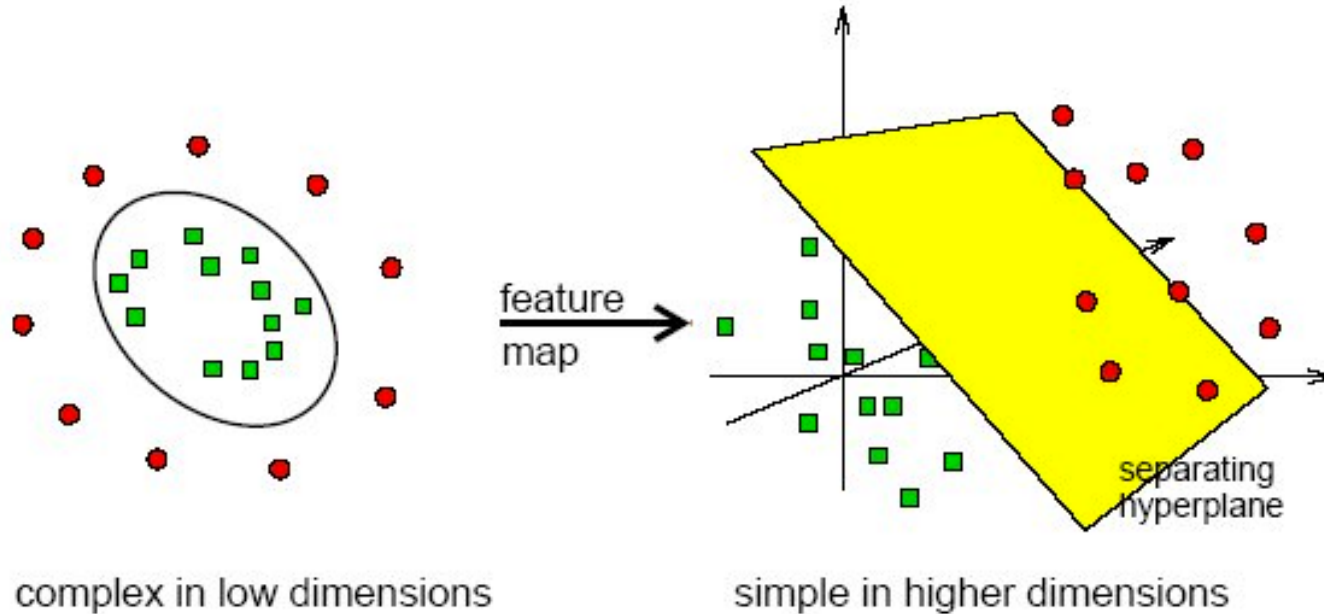
*Nonlinear applications of SVM rely on an implicit (nonlinear) mapping  $\Phi$  that sends vectors from the original feature space  $\mathbb{K}$  into a higher-dimensional feature space  $\mathbb{K}'$ .*

*Nonlinear classification in  $\mathbb{K}$  is then obtained by creating a linear decision boundary in  $\mathbb{K}'$ .*

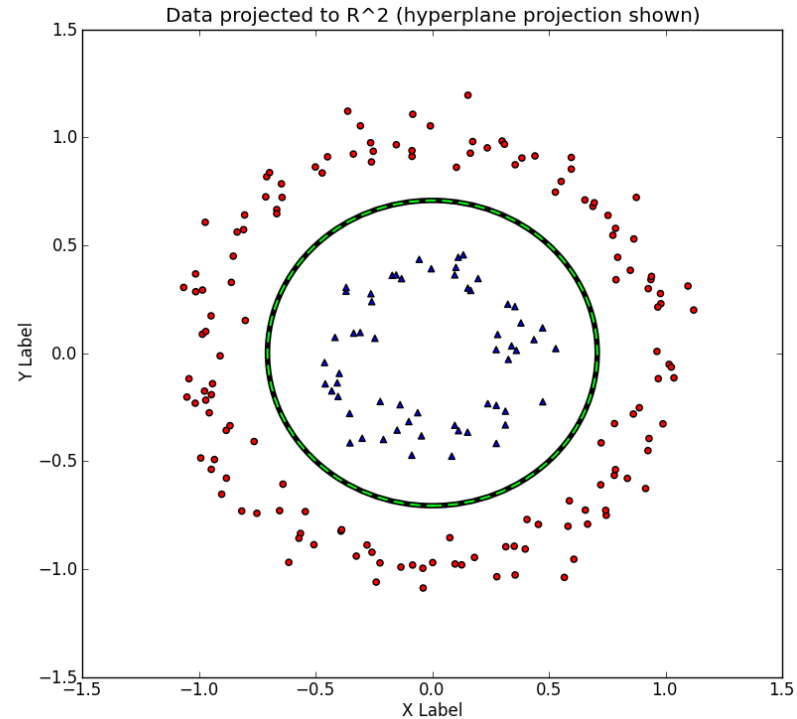
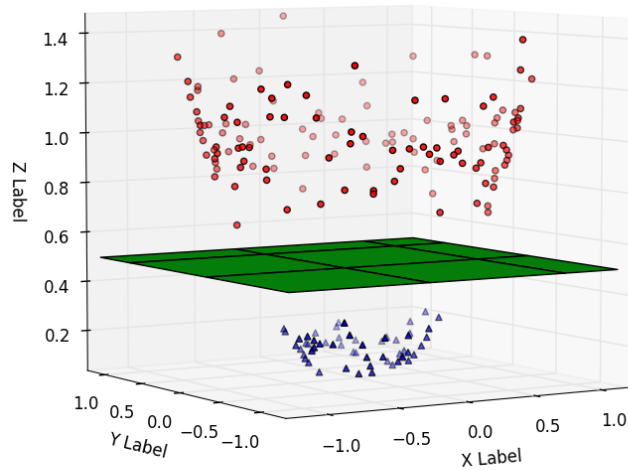
*Nonlinear applications of SVM rely on an implicit (nonlinear) mapping that sends vectors from the original feature space  $\mathbb{K}$  into a higher-dimensional feature space  $\mathbb{K}'$ .*

*Nonlinear classification in  $\mathbb{K}$  is then obtained by creating a linear decision boundary in  $\mathbb{K}'$ .*

*In practice, this involves no computations in the higher dimensional space!*



*If a linear decision boundary cannot be found in the original space, we can map into a higher dimensional space and find the separating surface.*

Data in  $R^3$  (separable w/ hyperplane)

*If a linear decision boundary cannot be found in the original space, we can map into a higher dimensional space and find the separating surface.*

# II. MAXIMUM MARGIN HYPERPLANES

---

## MAXIMUM MARGIN HYPERPLANES

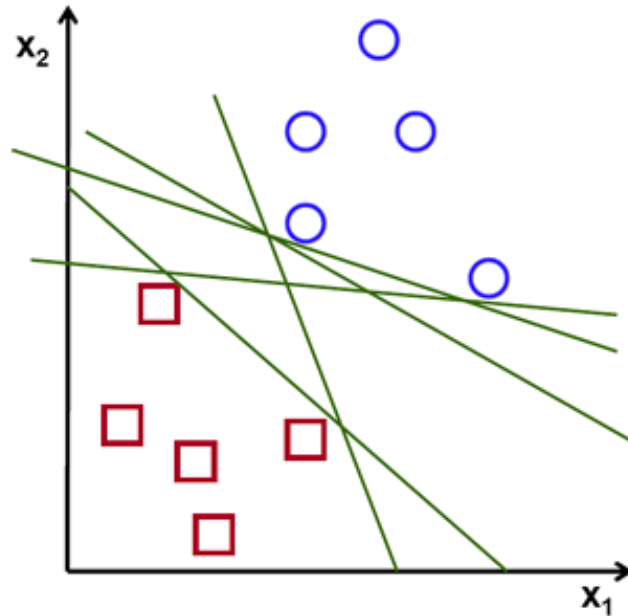
---

*Q: How is the decision boundary (mmh) derived?*

## MAXIMUM MARGIN HYPERPLANES

*Q: How is the decision boundary (mmh) derived?*

so many possible hyperplanes, which one to choose?





---

**MAXIMUM MARGIN HYPERPLANES**

---

*Q: How is the decision boundary (mmh) derived?*

*A: By the **discriminant function**,*

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

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

*Q: How is the decision boundary (mmh) derived?*

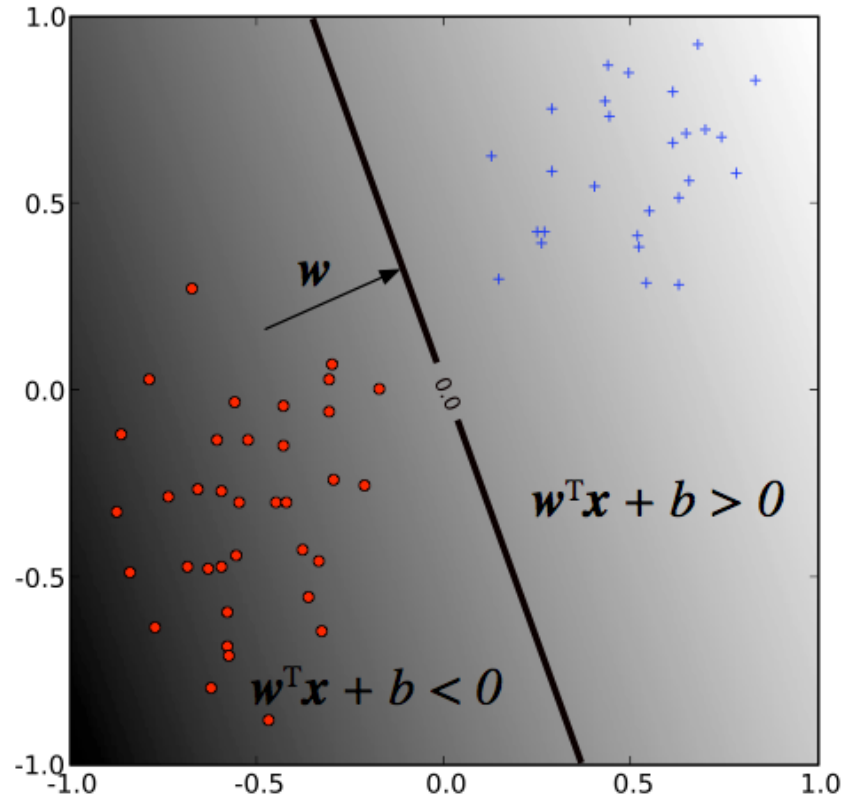
*A: By the discriminant function,*

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

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

*The sign of  $f(\mathbf{x})$  determines the (binary) class label of a record  $\mathbf{x}$ .*

## MAXIMUM MARGIN HYPERPLANES



### NOTE

The weight vector determines the orientation of the decision boundary.

The bias determines its translation from the origin.

## MAXIMUM MARGIN HYPERPLANES

---

*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?*

## MAXIMUM MARGIN HYPERPLANES

---

*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.*

## MAXIMUM MARGIN HYPERPLANES

*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.*

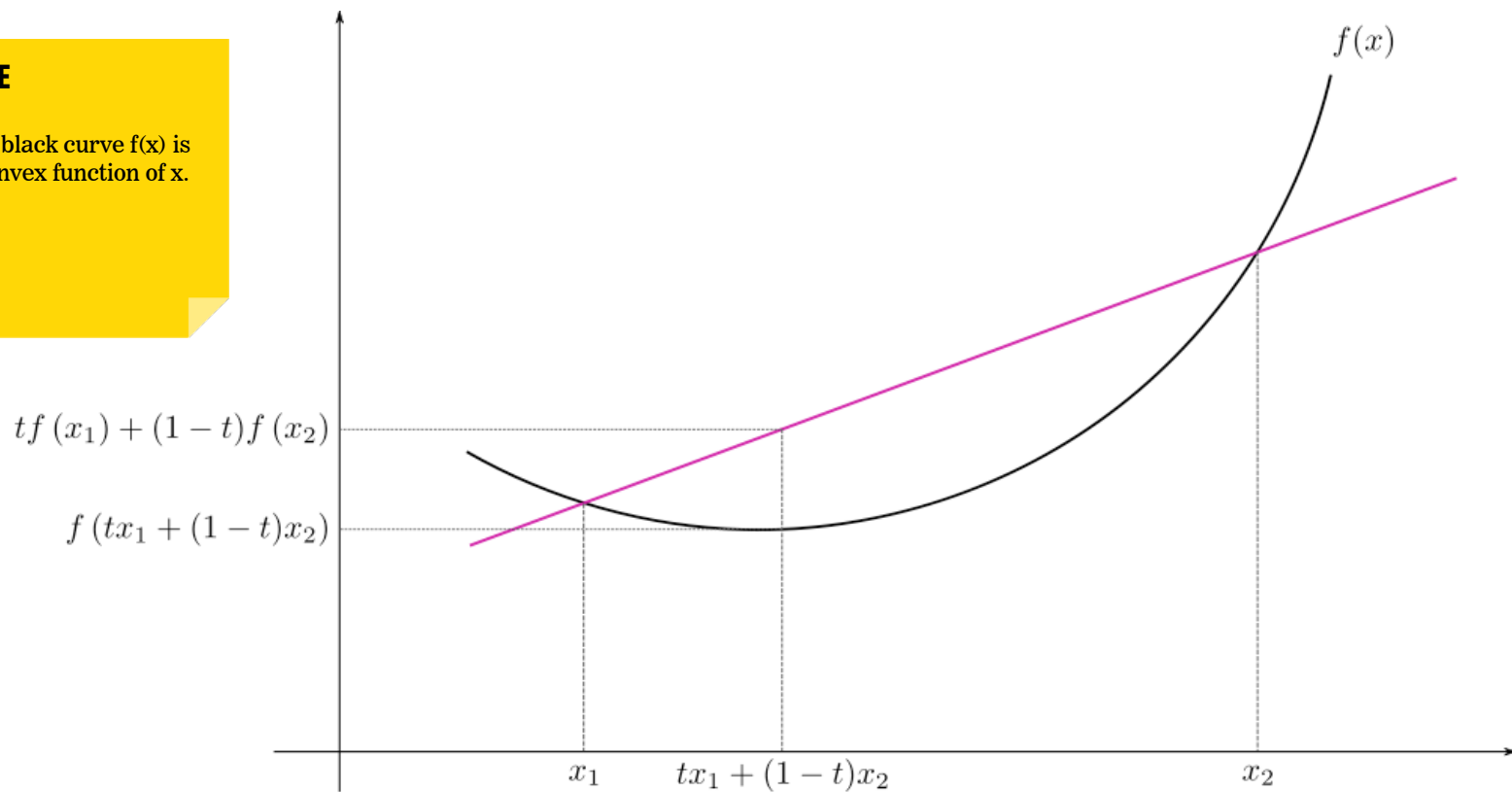
*Selecting the mmh is a straightforward exercise in analytic geometry (we won't go through the details here).*

*In particular, this task reduces to the optimization of a **convex** objective function.*

## CONVEX FUNCTIONS

**NOTE**

The black curve  $f(x)$  is a convex function of  $x$ .





*Selecting the mmh is a straightforward exercise in analytic geometry (we won't go through the details here).*

*In particular, this task reduces to the optimization of a **convex** objective function.*

*This is nice because convex optimization problems are guaranteed to give **global optima** (and they're easy to solve numerically too).*

*Selecting the mmh is a straightforward exercise in analytic geometry (we won't go through the details here).*

*In particular, this task reduces to the optimization of a **convex** objective function.*

### NOTE

The heuristic techniques we've discussed (eg greedy algorithms) are not necessary with convex optimization!

*This is nice because convex optimization problems are guaranteed to give **global optima** (and they're easy to solve numerically too).*

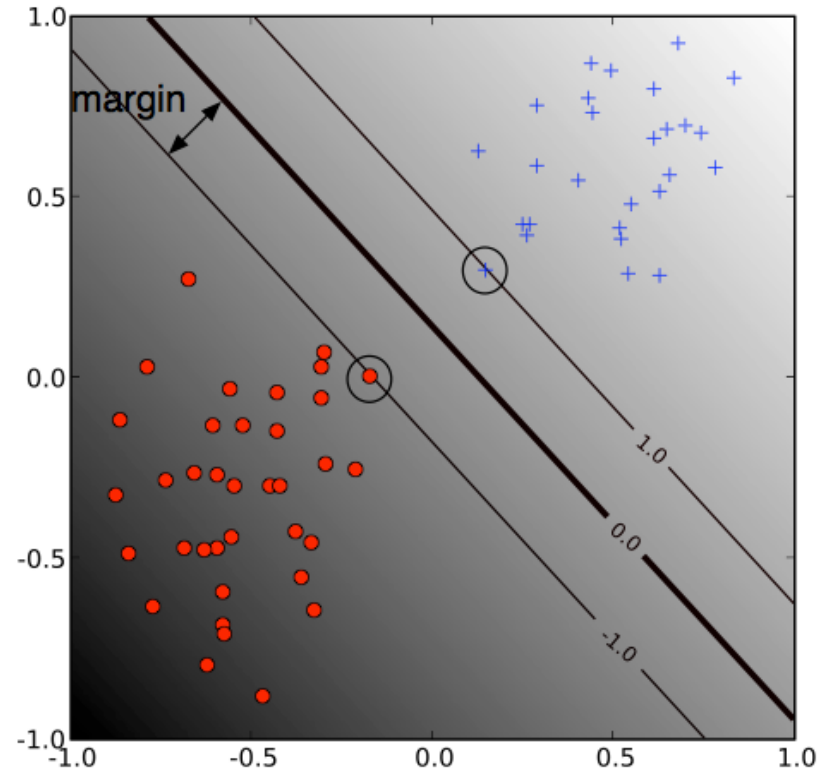
---

**MAXIMUM MARGIN HYPERPLANES**

---

*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



---

## MAXIMUM MARGIN HYPERPLANES

---

*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**.*

## MAXIMUM MARGIN HYPERPLANES

---

*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!*

---

## MAXIMUM MARGIN HYPERPLANES

---

*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:*

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



*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:*

$$\begin{array}{ll} \underset{\mathbf{w}, b}{\text{minimize}} & \frac{1}{2} \|\mathbf{w}\|^2 \\ \text{subject to:} & y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 \quad i = 1, \dots, n. \end{array}$$

**NOTE**

This type of optimization problem is called a quadratic program.

The result of this qp is the hard margin classifier we've been discussing.

# **III. SLACK VARIABLES**

*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:*

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

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

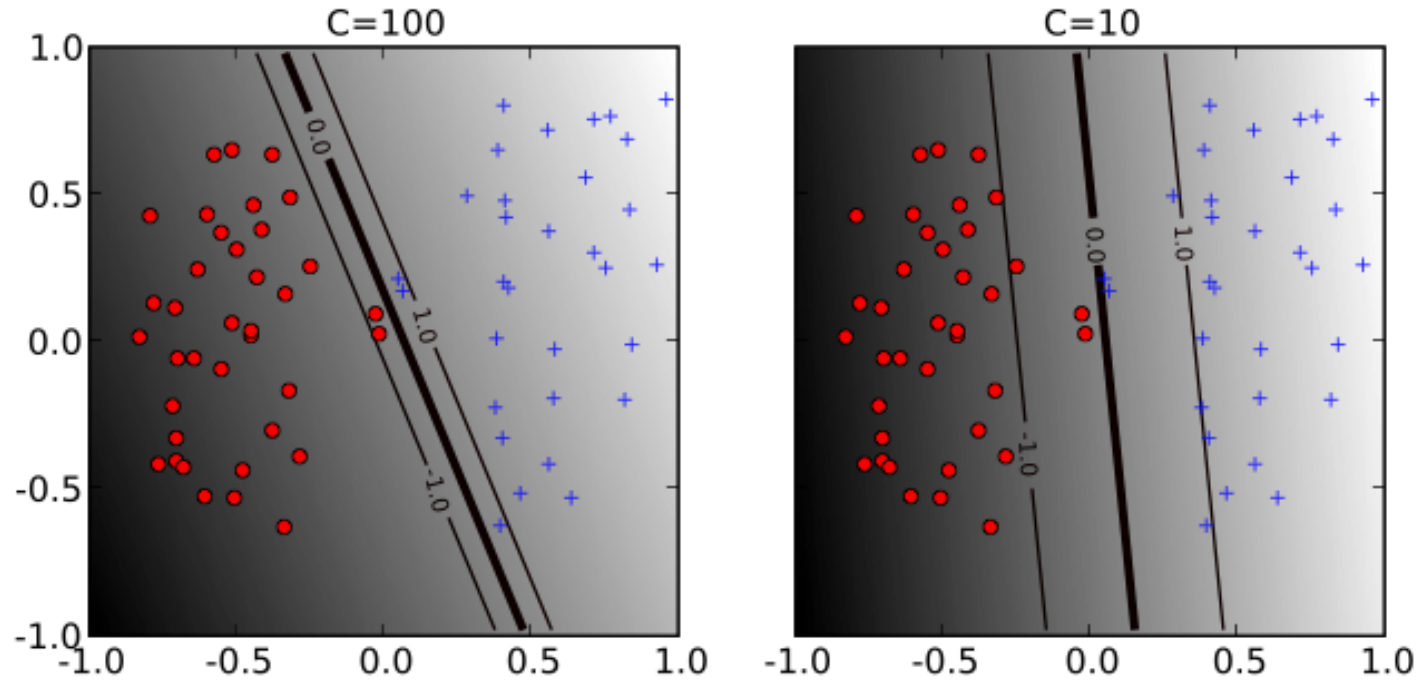
*The resulting **soft margin classifier** is given by:*

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

*This an example of bias-variance tradeoff.*



## SLACK VARIABLES – SOFT MARGIN CONSTANT



*The soft-margin optimization problem can be rewritten as:*

$$\begin{aligned} & \underset{\alpha}{\text{maximize}} && \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n y_i y_j \alpha_i \alpha_j \mathbf{x}_i^T \mathbf{x}_j \\ & \text{subject to:} && \sum_{i=1}^n y_i \alpha_i = 0, \quad 0 \leq \alpha_i \leq C. \end{aligned}$$

*The soft-margin optimization problem can be rewritten as:*

$$\begin{aligned} & \underset{\alpha}{\text{maximize}} && \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n y_i y_j \alpha_i \alpha_j \mathbf{x}_i^T \mathbf{x}_j \\ & \text{subject to:} && \sum_{i=1}^n y_i \alpha_i = 0, \quad 0 \leq \alpha_i \leq C. \end{aligned}$$

**NOTE**

This is called the dual formulation of the optimization problem.

(reached via Lagrange multipliers)

*The soft-margin optimization problem can be rewritten as:*

$$\begin{aligned} & \underset{\alpha}{\text{maximize}} && \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n y_i y_j \alpha_i \alpha_j \mathbf{x}_i^T \mathbf{x}_j \\ & \text{subject to:} && \sum_{i=1}^n y_i \alpha_i = 0, \quad 0 \leq \alpha_i \leq C. \end{aligned}$$

*Notice that this expression depends on the features  $\mathbf{x}_i$  only via the inner product*

$$\langle \mathbf{x}_i, \mathbf{x}_j \rangle = \mathbf{x}_i^T \mathbf{x}_j$$

*The inner product is an operation that takes two vectors and returns a real number.*

*The inner product is an operation that takes two vectors and returns a real number.*

*The fact that we we can rewrite the optimization problem in terms of the inner product means that we don't actually have to do any calculations in the feature space  $\mathbf{K}$ .*

*The inner product is an operation that takes two vectors and returns a real number.*

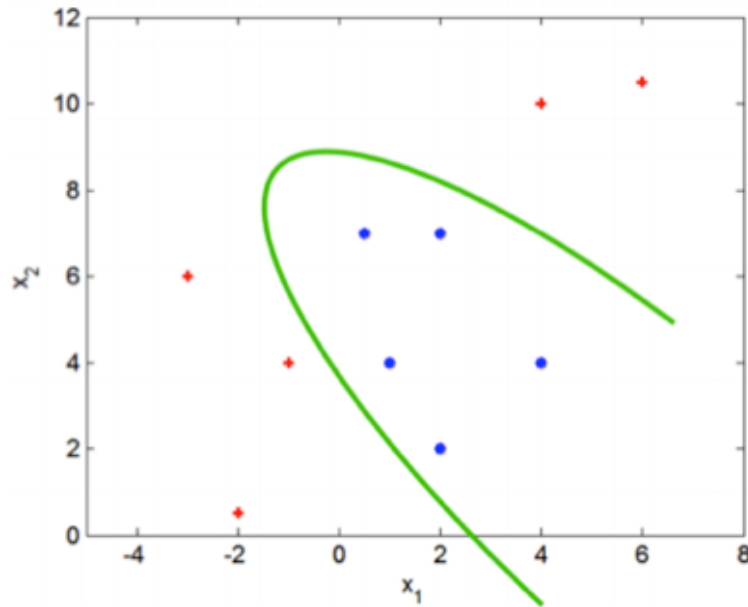
*The fact that we we can rewrite the optimization problem in terms of the inner product means that we don't actually have to do any calculations in the feature space  $\mathbf{K}$ .*

*In particular, we can easily change  $\mathbf{K}$  to be some other space  $\mathbf{K}'$ .*

# **IV. NONLINEAR CLASSIFICATION**



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



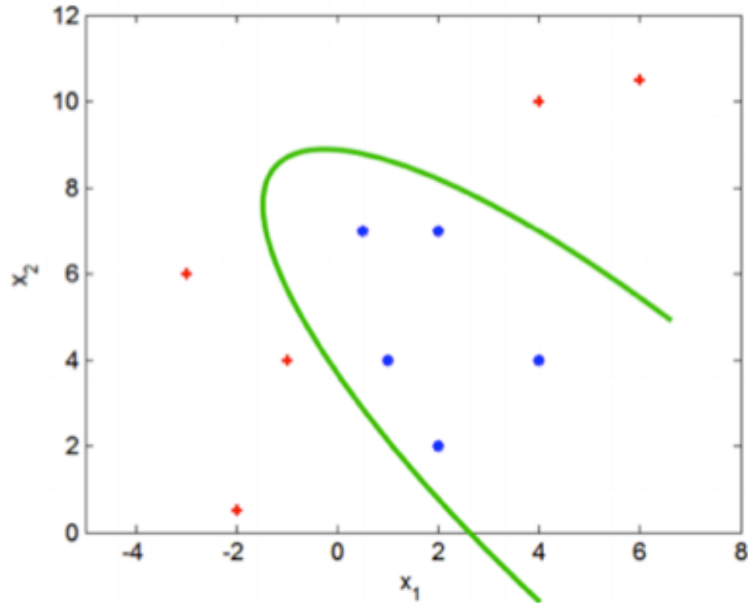
*Suppose we need a more complex classifier than a linear decision boundary allows.*

*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.*

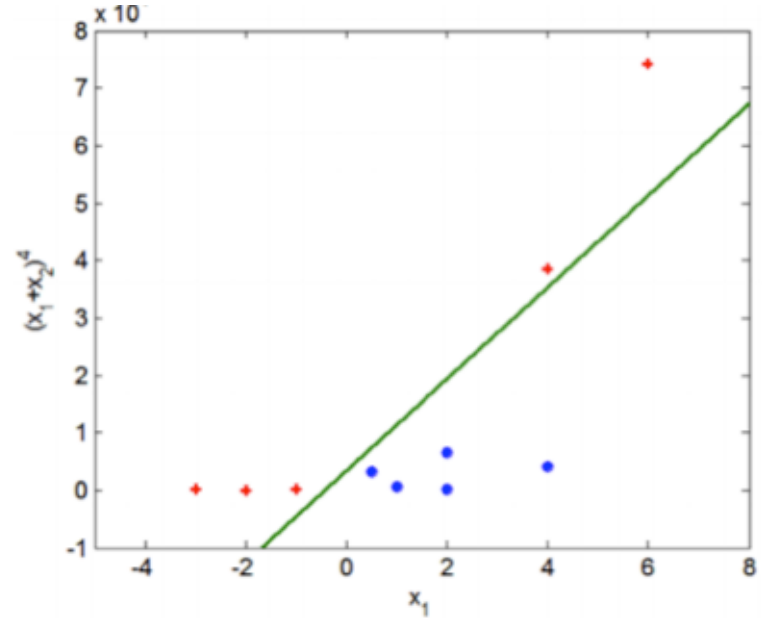
*Suppose we need a more complex classifier than a linear decision boundary allows.*

*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  $\mathbf{x}_i$  into a higher-dimensional space  $\mathbb{K}'$*
- create a linear decision boundary in  $\mathbb{K}'$*
- back out the nonlinear decision boundary in  $\mathbb{K}$  from the result*

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

- remap the feature vectors  $\mathbf{x}_i$  into a higher-dimensional space  $\mathbb{K}'$*
- create a linear decision boundary in  $\mathbb{K}'$*
- back out the nonlinear decision boundary in  $\mathbb{K}$  from the result*

*But we want to save ourselves the trouble of doing a lot of additional high-dimensional calculations. How can we do this?*

*Recall that our optimization problem depends on the features only through the inner product  $\mathbf{x}^T \mathbf{x}$ :*

$$\begin{aligned} & \underset{\alpha}{\text{maximize}} && \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n y_i y_j \alpha_i \alpha_j \mathbf{x}_i^T \mathbf{x}_j \\ & \text{subject to:} && \sum_{i=1}^n y_i \alpha_i = 0, \quad 0 \leq \alpha_i \leq C. \end{aligned}$$

*Recall that our optimization problem depends on the features only through the inner product  $\mathbf{x}^T \mathbf{x}$ :*

$$\begin{aligned} & \underset{\alpha}{\text{maximize}} && \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n y_i y_j \alpha_i \alpha_j \mathbf{x}_i^T \mathbf{x}_j \\ & \text{subject to:} && \sum_{i=1}^n y_i \alpha_i = 0, \quad 0 \leq \alpha_i \leq C. \end{aligned}$$

*We can replace this inner product with a more general function that has the same type of output as the inner product.*

---

*Formally, we can think of the inner product as a map that sends two vectors in the feature space  $\mathbb{K}$  into the real line  $\mathbb{R}$*

*Formally, we can think of the inner product as a map that sends two vectors in the feature space  $\mathbb{K}$  into the real line  $\mathbb{R}$ .*

*We can replace this with a generalization of the inner product called a **kernel function** that maps two vectors in a higher-dimensional feature space  $\mathbb{K}'$  into  $\mathbb{R}$ .*

## NONLINEAR CLASSIFICATION

---

*The upshot is that we can use a kernel function to implicitly train our model in a higher-dimensional feature space, without incurring additional computational complexity!*

*The upshot is that we can use a kernel function to implicitly train our model in a higher-dimensional feature space, without incurring additional computational complexity!*

*As long as the kernel function satisfies certain conditions, our conclusions above regarding the mmh continue to hold.*



*The upshot is that we can use a kernel function to implicitly train our model in a higher-dimensional feature space, without incurring additional computational complexity!*

*As long as the kernel function satisfies certain conditions, our conclusions above regarding the mmh continue to hold.*

### NOTE

These conditions are contained in a result called Mercer's theorem.

*The upshot is that we can use a kernel function to implicitly train our model in a higher-dimensional feature space, without incurring additional computational complexity!*

*As long as the kernel function satisfies certain conditions, our conclusions above regarding the mmh continue to hold.*

*In other words, no algorithmic changes are necessary, and all the benefits of a linear SVM are maintained.*

*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:*

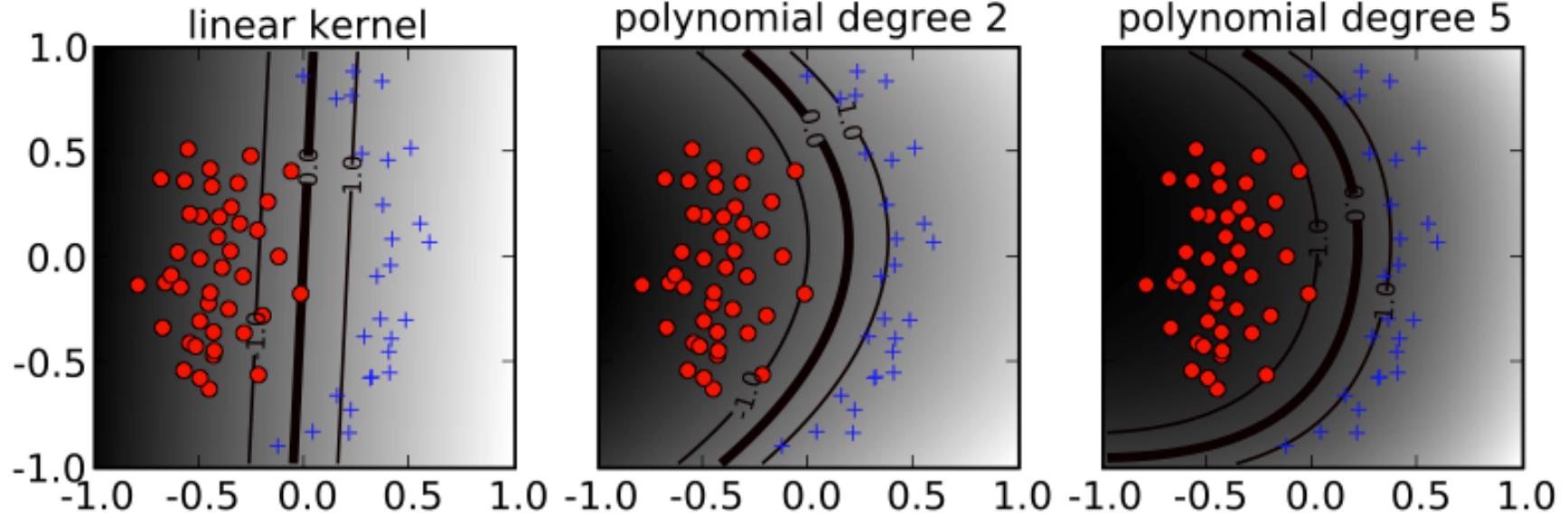
*linear kernel*  $k(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle$

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

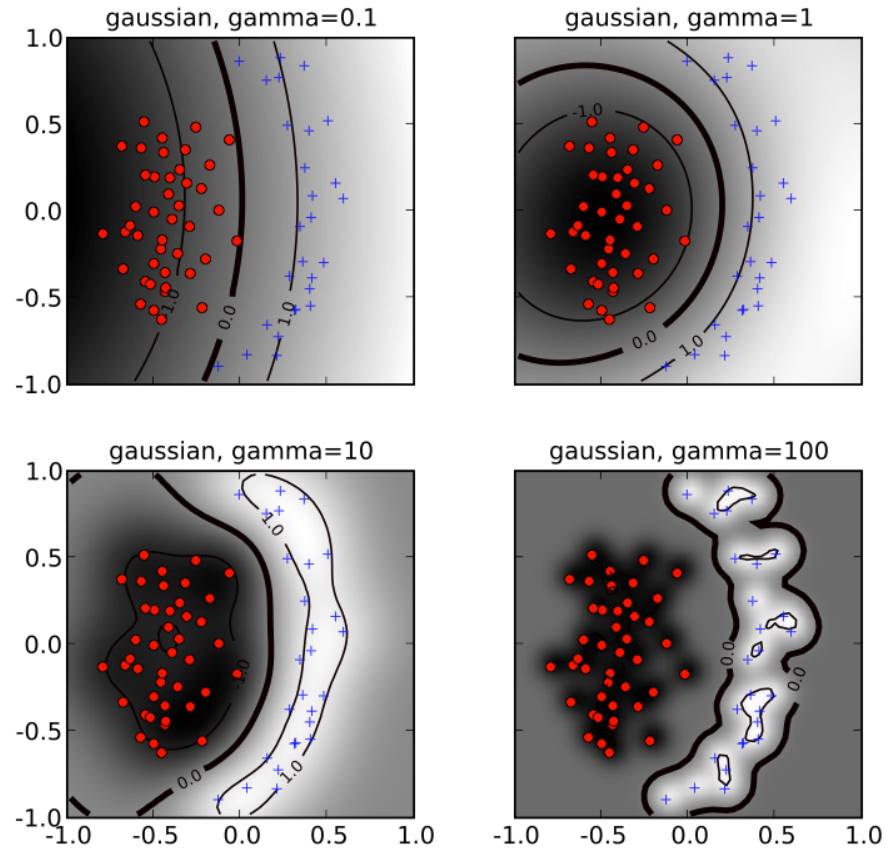
*Gaussian kernel*  $k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$

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

## NONLINEAR CLASSIFICATION – POLYNOMIAL KERNEL



## NONLINEAR CLASSIFICATION – GAUSSIAN KERNEL



*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!*

---

**INTRO TO DATA SCIENCE**

---

**EX: SVM IN SCIKIT-LEARN**