

# **INTRO TO DATA SCIENCE**

## **LECTURE 10: SUPPORT VECTOR MACHINES**

**Paul Burkard**

**11/30/2015**

## **LAST TIME:**

- DECISION TREES**
- BUILDING DECISION TREES**
- OBJECTIVE SPLITTING FUNCTIONS**
- PREVENTING OVERFITTING IN DECISION TREES**

**QUESTIONS?**

**I. SUPPORT VECTOR MACHINES**

**II. SOFT-MARGIN SVM**

**III. NONLINEAR SVM**

**HANDS-ON: SVM**

# **I. SUPPORT VECTOR MACHINES**

**Q:** *What is a support vector machine?*

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

**recall:**

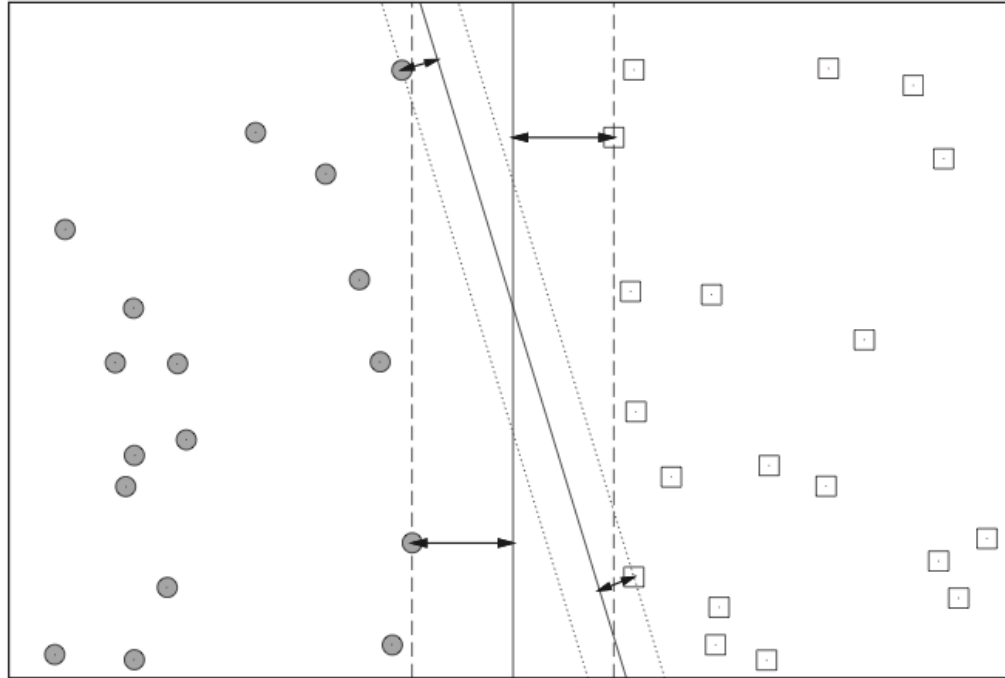
binary classifier — solves two-class problem

linear classifier — creates linear decision boundary (in 2d)

**Q:** *How is the decision boundary derived?*

**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**.*



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

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

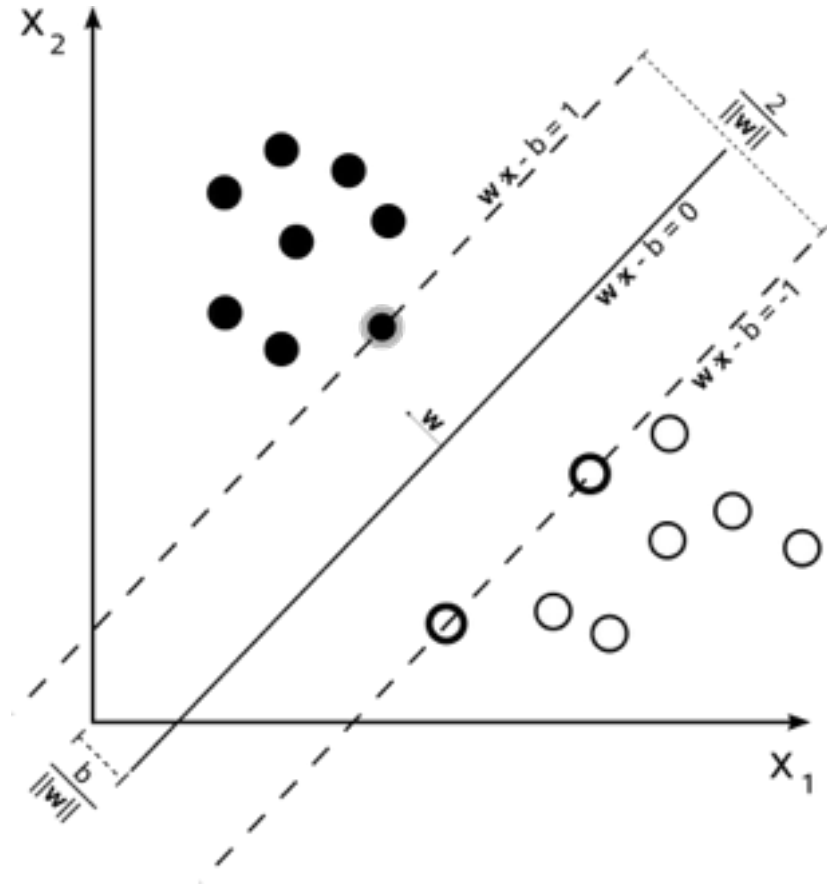
$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b.$$

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

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

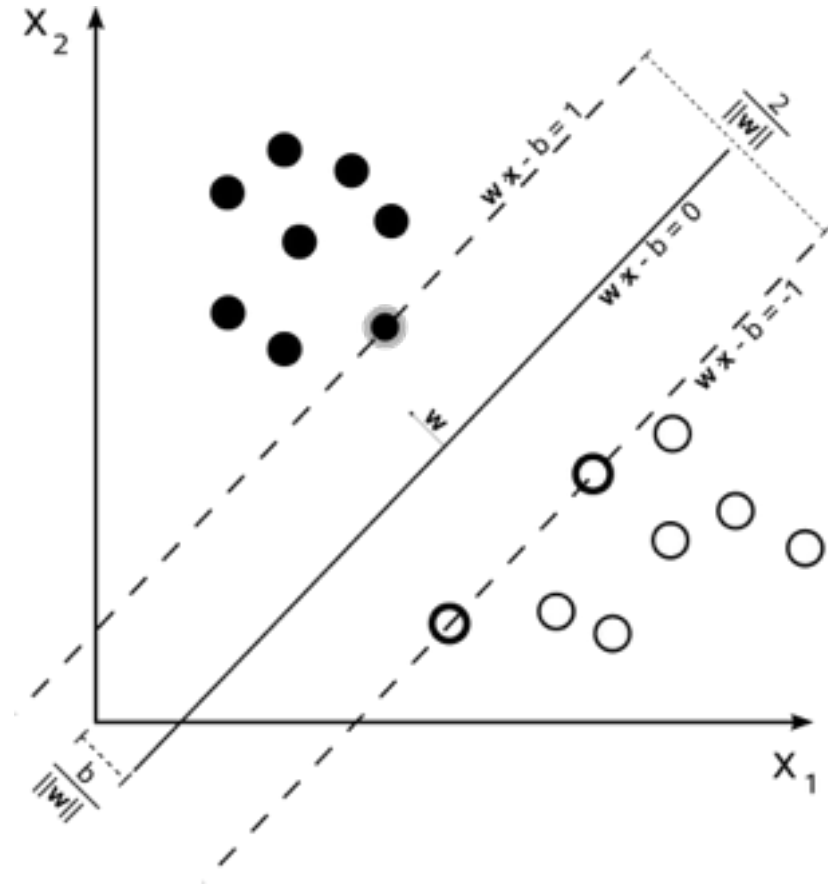
*How is the decision boundary (mmh) derived?*

- Any hyperplane can be written as the set of points where  $\mathbf{w}^T \mathbf{x} - \mathbf{b} = 0$
- $\mathbf{w}$  sets the plane's orientation (it's perpendicular to the plane)
- $\mathbf{b}$  sets the offset from the origin
- Set the margin planes for each class such that  $\mathbf{w}^T \mathbf{x} - \mathbf{b} = +/-1$
- +1 for positive class, -1 for negative class



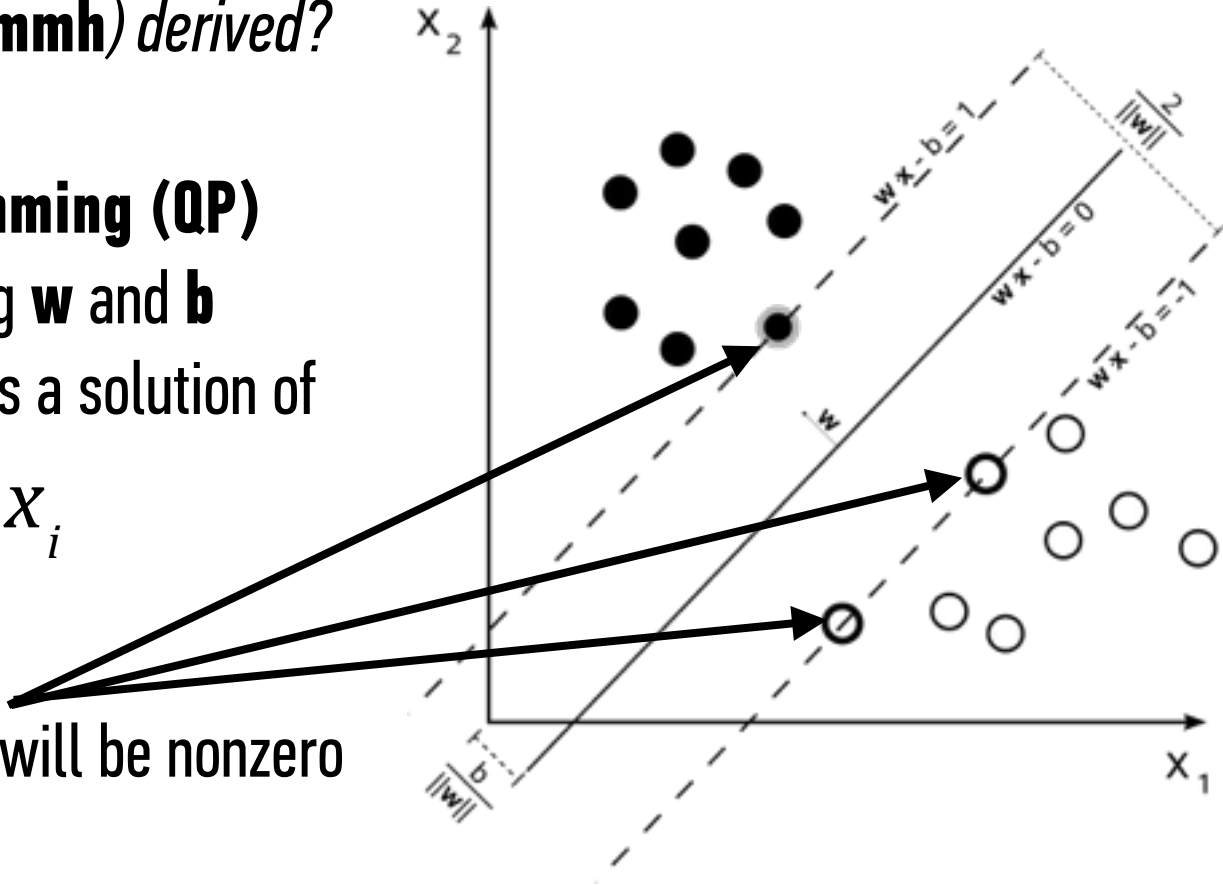
*How is the decision boundary (mmh) derived?*

- Thus, our goal of maximizing the margin width amounts to maximizing  $2/\|w\|$
- To do this, we must minimize  $\|w\|$
- Easier to minimize  $\|w\|^2/2$
- We do this subject to the constraint:  
 $y_i^*(w^T x_i - b) \geq 1$
- This is a **Quadratic Programming (QP)** optimization problem, yielding  $w$  and  $b$



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

- This is a **Quadratic Programming (QP)** optimization problem, yielding **w** and **b**
- Solving this QP problem yields a solution of the form:  $w = \sum_i \alpha_i y_i x_i$
- Most of these alpha are 0
- Only the “**support vectors**” will be nonzero
- Thus only they contribute!



# **II. SOFT-MARGIN SVM**

- *So far we've been dealing with what's called a **hard margin SVM** – what does this mean?*
- *We assume the data is perfectly separable by our hyperplane*
- *No classification instances will fall within the margin*
- *As we've seen, probably not realistic (especially in linear case)*
- *So how do we allow for some error/noise in our model?*

**Q:** *How do we handle potentially inseparable data?*

**A:** *By training a **soft margin SVM** rather than a hard margin*

**soft margin** – basically allows for a fuzzy boundary, where some proportion of elements may be misclassified in order to maintain a simpler boundary

Remember, simpler boundaries are probably more likely to generalize.

**Q:** *How do we train a **soft margin SVM**?*

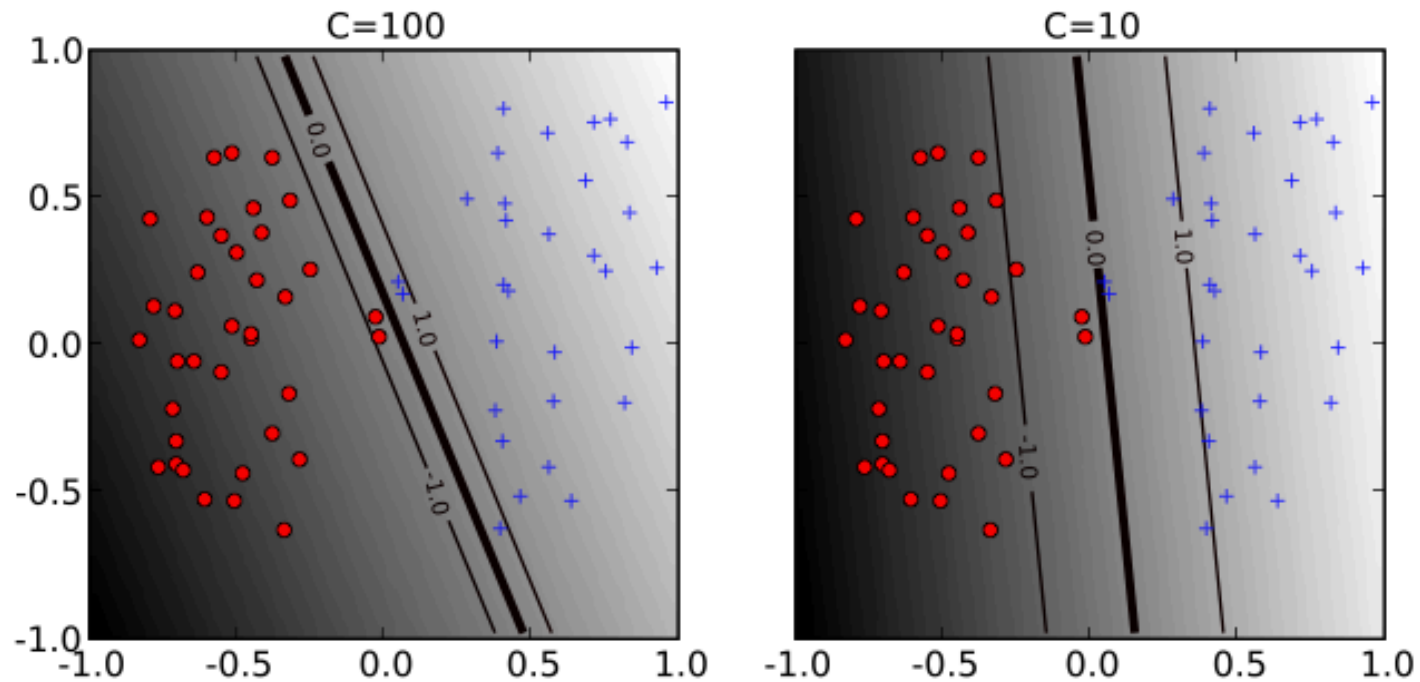
**A:** *By making use of **slack variables** that allow a proportion of instances to be misclassified*

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

The resulting soft margin QP problem is given by:

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





# **III. NONLINEAR SVM**

*Recall that our soft-margin SVM optimization function can be written as follows:*

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

*This can be rewritten in the following form:*

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

## NOTE

This is called the *dual formulation* of the optimization problem.

(reached via Lagrange multipliers)

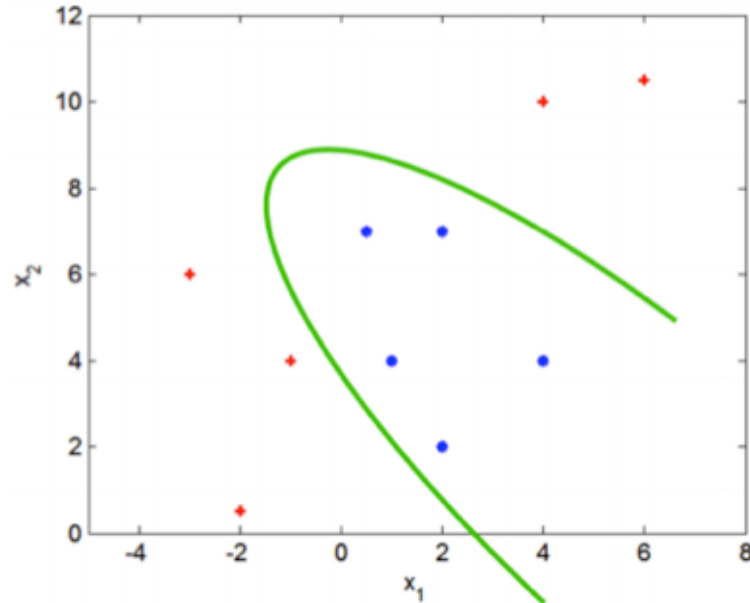
*Notice that this expression depends only on **inner products**  $\mathbf{x}_i^\top \mathbf{x}_j$*

The inner product (dot 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  $K$ , **just dot products**.

In particular, we can easily change  $K$  to be some other space  $K'$ .

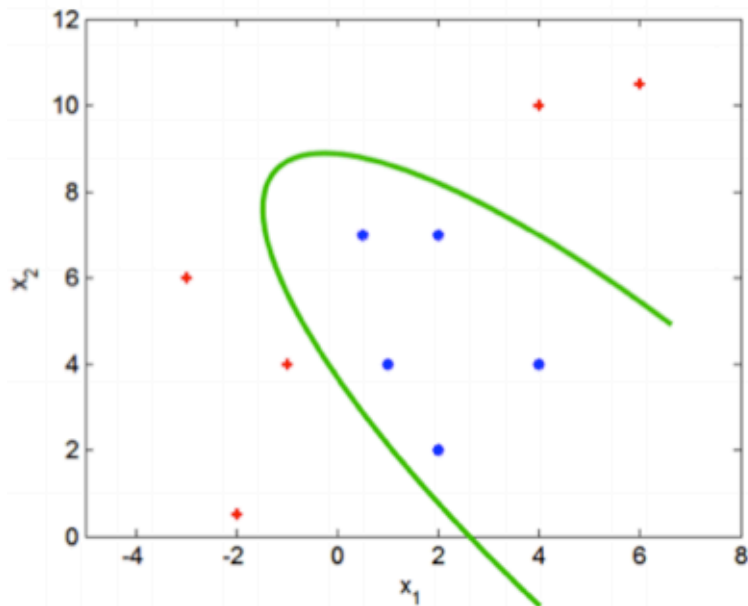
Suppose we need a more complex classifier than a linear decision boundary allows.



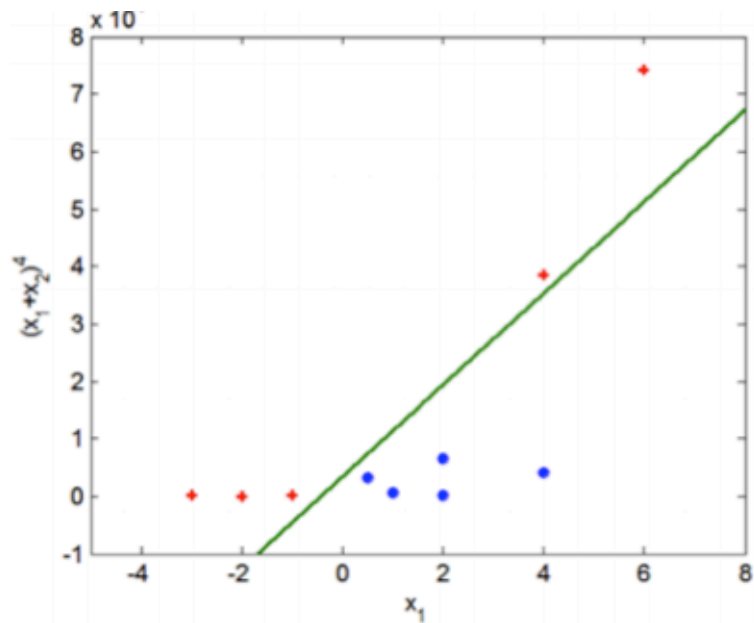
Suppose we need a more complex classifier than a linear decision boundary allows.

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.

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.

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:

- 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

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  $\mathbf{x}_i^T \mathbf{x}_j$ :

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

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

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*.

some popular kernels:

**linear kernel**

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

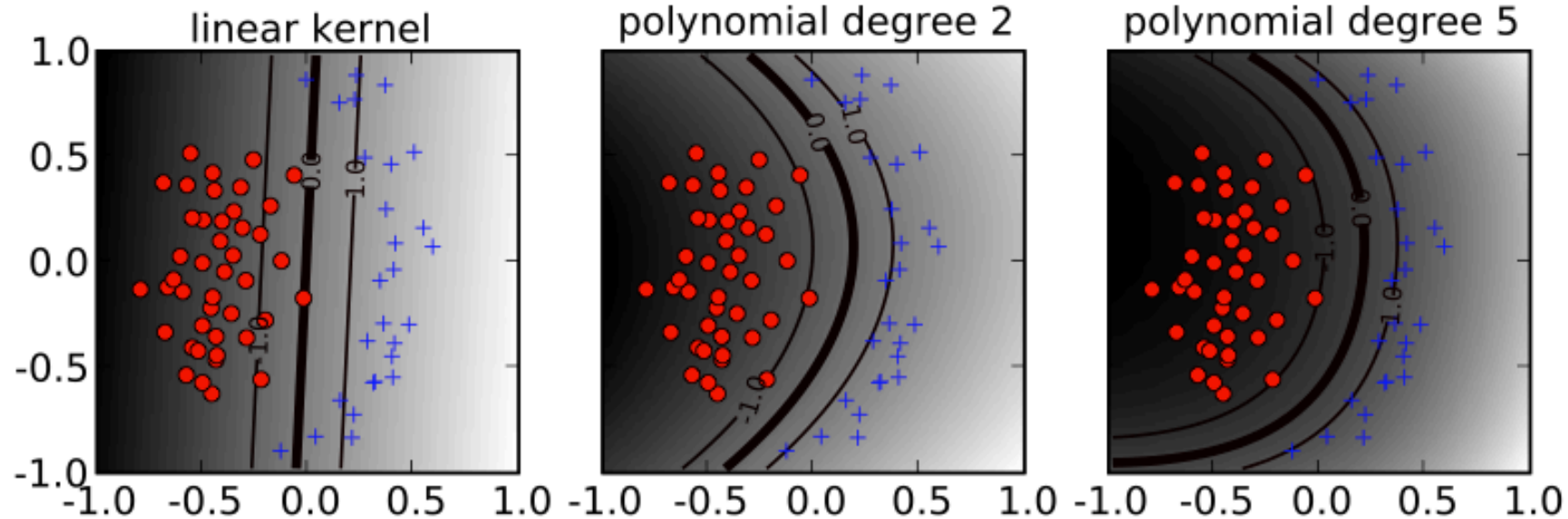
**polynomial kernel**

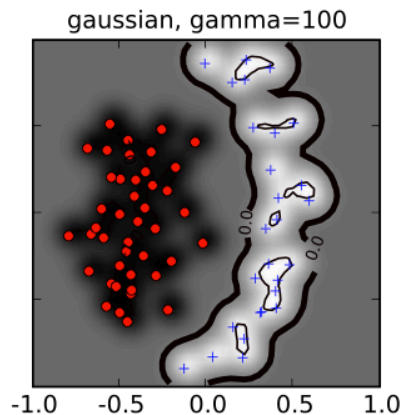
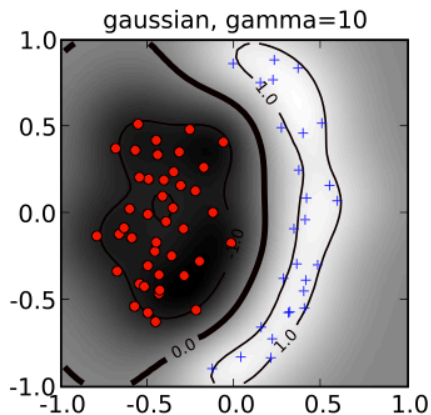
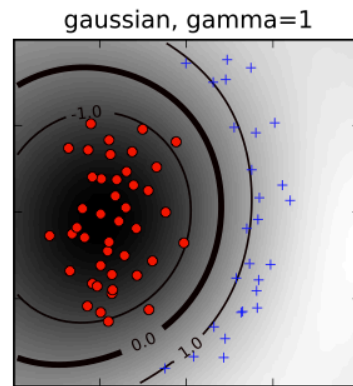
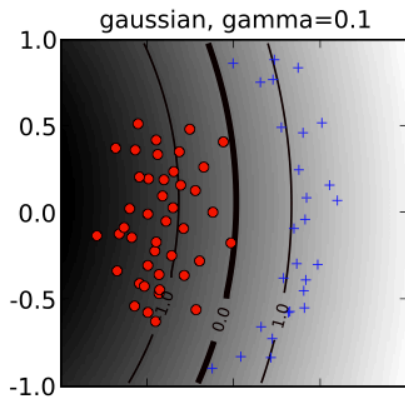
$$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}' + 1)^d$$

**Gaussian (RBF) kernel**

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

The **hyperparameters**  $d$ ,  $\gamma$  affect the flexibility





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**

---

# **HANDS-ON: SVM**