Introduction to Supervised Learning

Dr. Qiwei Gan

Overfitting and Underfitting

- **Generalization** ability refers to an algorithm's ability to give accurate predictions for **new, previously unseen** data.
- **Assumptions**: *Future unseen data (test set) will have the same properties as the current training sets.*
  - *Thus, models that are accurate on the training set are expected to be accurate on the test set.*
  - *But that may not happen if the trained model is tuned too specifically to the training set.*
- Models that are **too complex** for the amount of training data available are said to **overfit** and are not likely to generalize well to new examples.
- Models that are **too simple**, that don't even do well on the training data, are said to **underfit and** also not likely to generalize well.

Trade-off between Complexity and Generalization

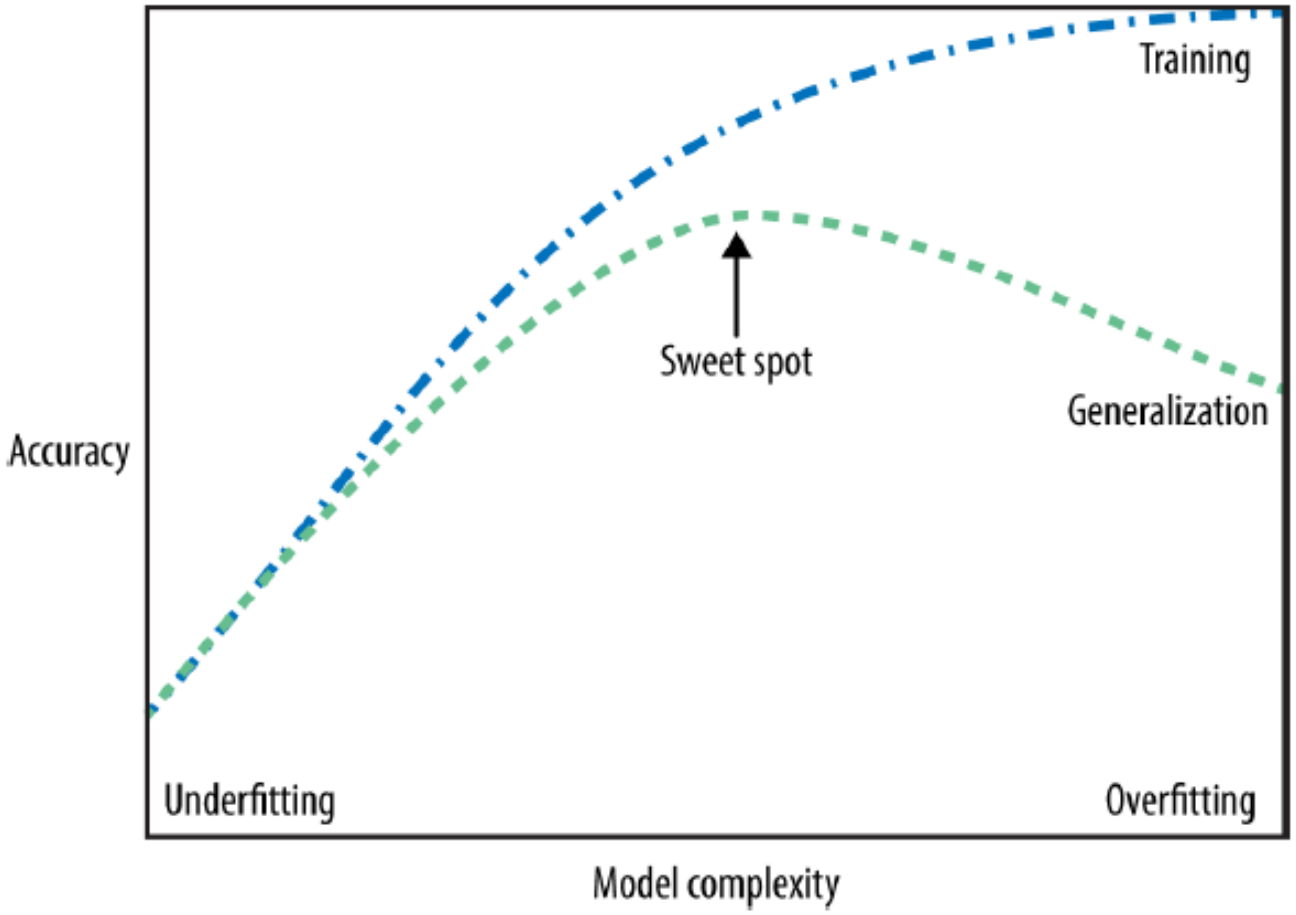


Figure 2-1. Trade-off of model complexity against training and test accuracy

Linear Regression (Ordinary Least Squares)

- **Parameters are estimated from training data.**
- **There are many different ways to estimate *w* and *b*:**
  - *Different methods correspond to different "fit" criteria and goals and ways to control model complexity.*
- **The learning algorithm finds the parameters that optimize an objective function, typically to minimize some kind of loss function of the predicted target values *vs.* actual target values.**

Polynomial Features with Linear Regression

$$\boldsymbol{x} = (x_0, x_1) \Rightarrow \boldsymbol{x}' = (x_0, x_1, x_0^2, x_0 x_1, x_1^2)$$

$$\hat{y} = \hat{w}_0 x_0 + \hat{w}_1 x_1 + \hat{w}_{00}\, x_0^2 + \hat{w}_{01} x_0 x_1 + \hat{w}_{11} x_1^2 + b$$

- **Generate new features consisting of all polynomial combinations of the original two features (x_0, x_1).**
- **The *degree* of the polynomial specifies how many variables participate at a time in each new feature (above example: degree 2)**
- **This is still a weighted linear combination of features, so it's still a linear model, and can use same least-squares estimation method for *w* and *b*.**

Polynomial Features with Linear Regression

Why would we want to transform our data this way?

- *To capture interactions between the original features by adding them as features to the linear model.*
- *To make a classification problem easier (we'll see this later).*

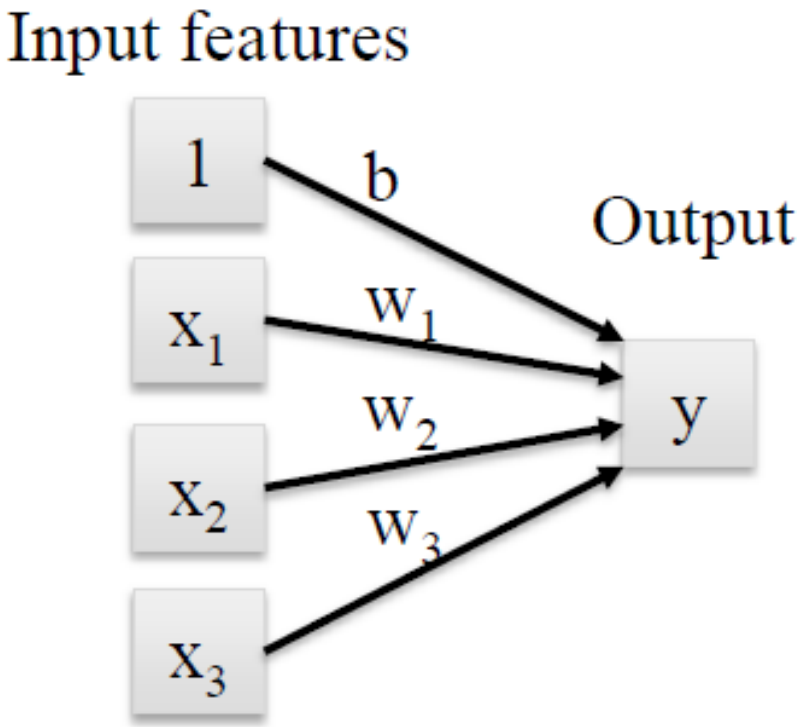
More generally, we can apply other non-linear transformations to create new features

- *(Technically, these are called non-linear basis functions)*

Beware of polynomial feature expansion with high as this can lead to complex models that overfit

- *Thus, polynomial feature expansion is often combined with a regularized learning method like ridge regression.*

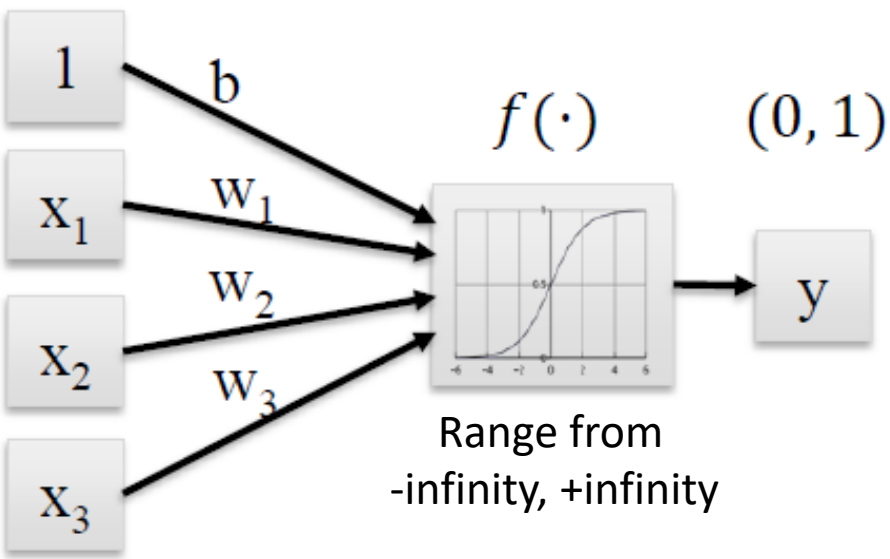
Linear Regression



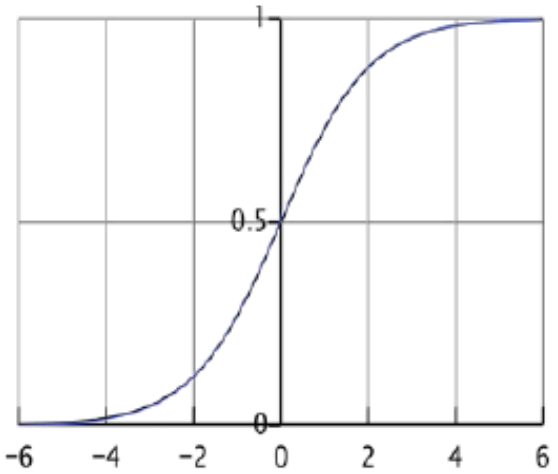
Possible range for X and y?

$$\hat{y} = \hat{b} + \hat{w}_1 \cdot x_1 + \cdots \hat{w}_n \cdot x_n$$

Linear Classifier—Logistic Regression



Range from
-infinity, +infinity



The logistic function transforms real-valued input to an output number y between 0 and 1, interpreted as the probability the input object belongs to the positive class, given its input features $(x_0, x_1, \ldots, x_n)$

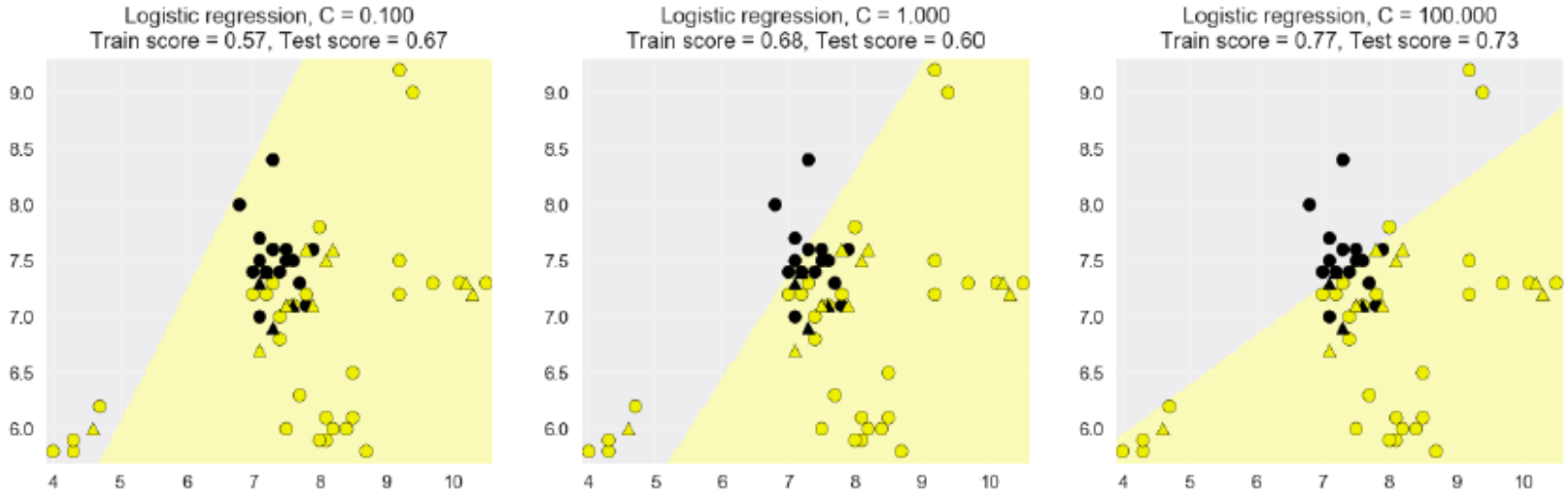
$$\hat{y} = \text{logistic}(\hat{b} + \hat{w}_1 \cdot x_1 + \cdots \hat{w}_n \cdot x_n)$$
$$= \frac{1}{1 + \exp\left[-(\hat{b} + \hat{w}_1 \cdot x_1 + \cdots \hat{w}_n \cdot x_n)\right]}$$

Regularization of Logistic Regression

L2 regularization is 'on' by default (like ridge regression)

Parameter C controls amount of regularization (default 1.0)

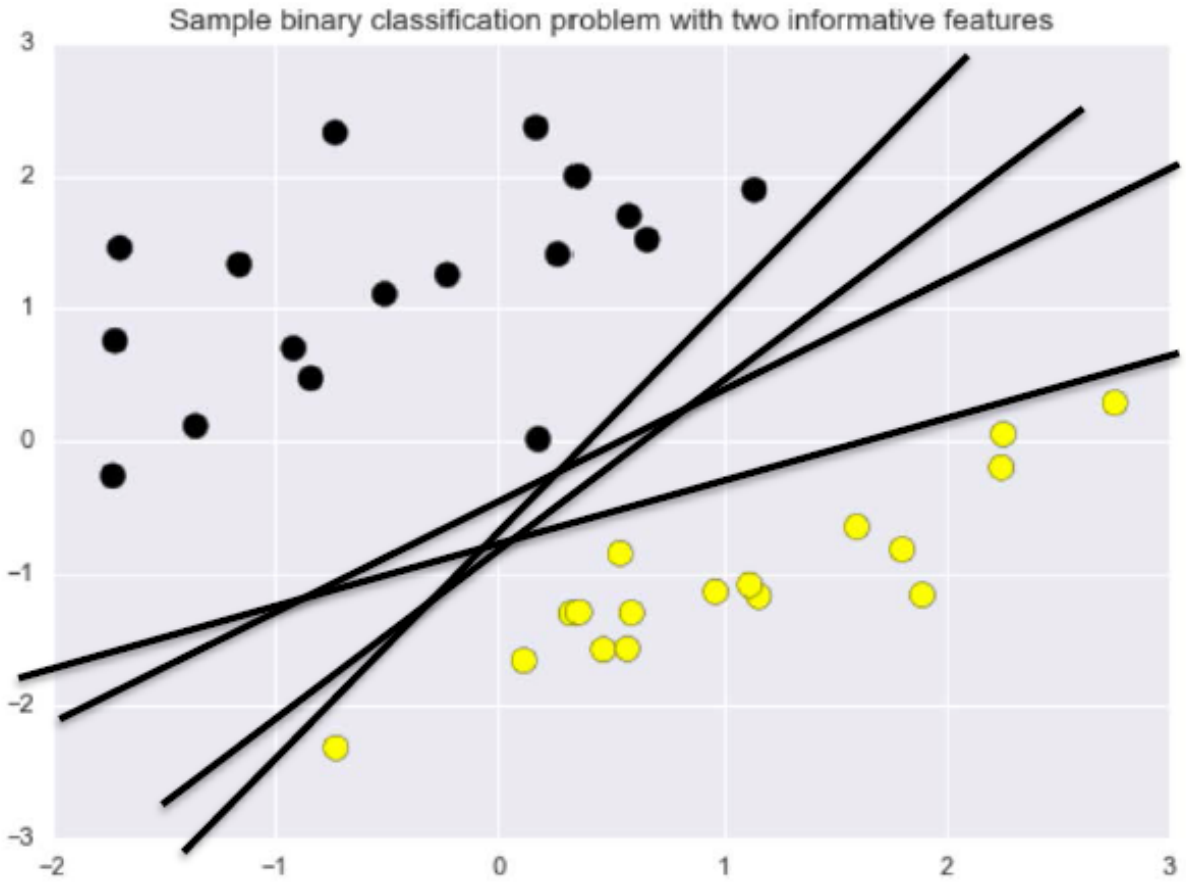
As with regularized linear regression, it can be important to normalize all features so that they are on the same scale.



Linear Classifier

$$f(x, w, b) = sign(w \circ x + b)$$

There are many possible linear classifiers that could separate the two classes.
Which one is best?



Linear Classifier

$$f(x, w, b) = sign(w \circ x + b)$$

Classifier margin

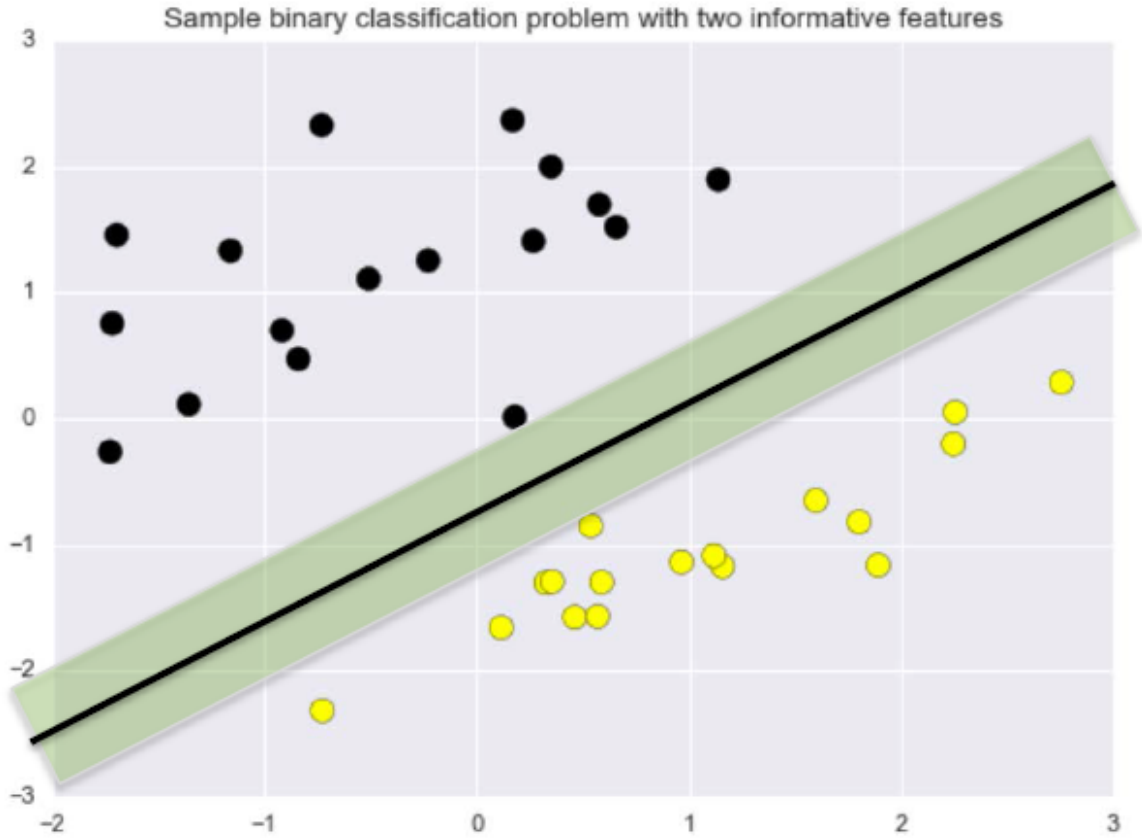
Defined as the maximum width the decision boundary area can be increased before hitting a data point.

Linear Classifier

$$f(x, w, b) = sign(w \circ x + b)$$

Maximum margin classifier

The linear classifier with maximum margin is a linear Support Vector Machine (LSVM).



Linear Support Vector Machines Classifier

The strength of regularization is determined by C

Larger values of C: less regularization

- *Fit the training data as well as possible*
- *Each individual data point is important to classify correctly*

Smaller values of C: more regularization

- *More tolerant of errors on individual data points*

Important Parameters

Model complexity

- **alpha: weight given to the L1 or L2 regularization term in regression models**
  - *default = 1.0*
- **C: regularization weight for LinearSVC and LogisticRegression classification models**
  - *default = 1.0*

The Need for Feature Normalization

- **Important for some machine learning methods that all features are on the same scale (e.g. faster convergence in learning, more uniform or 'fair' influence for all weights)**
  - *e.g. regularized regression, k-NN, support vector machines, neural networks*

Linear Models

Pros:

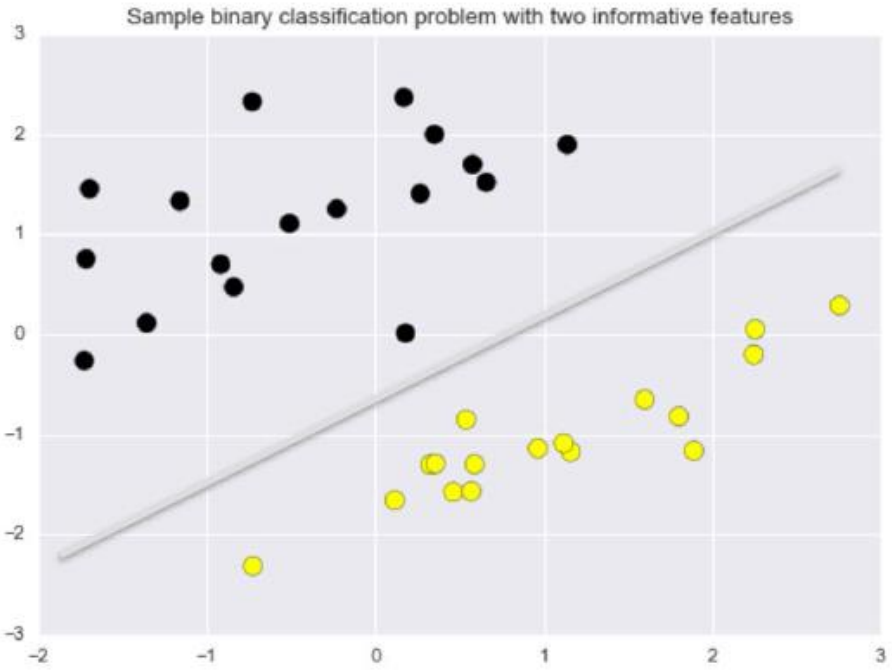
- **Simple and easy to train.**
- **Fast prediction.**
- **Scales well to very large datasets.**
- **Works well with sparse data.**
- **Reasons for prediction are relatively easy to interpret.**

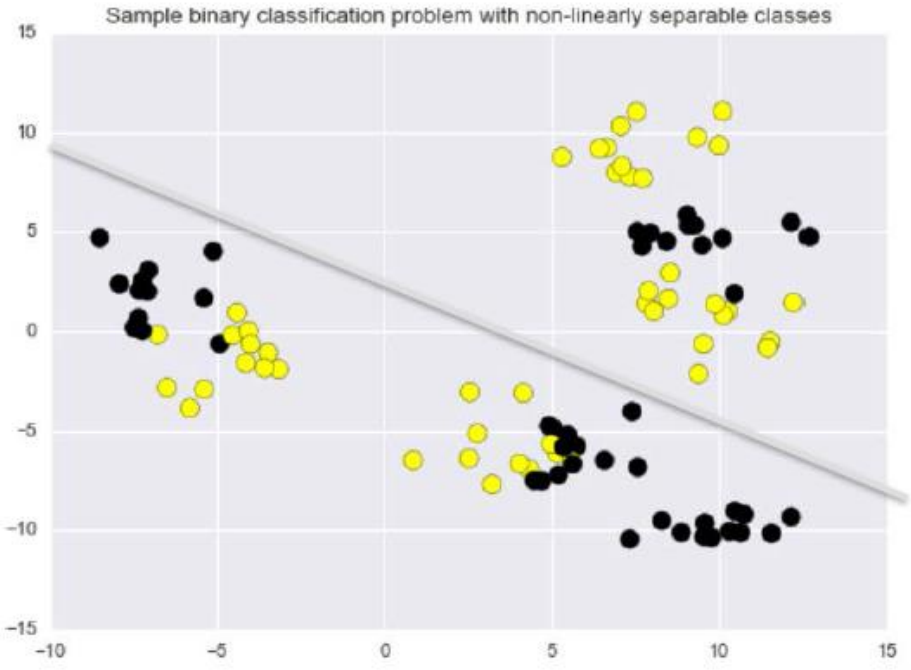
Cons:

- **For lower-dimensional data, other models may have superior generalization performance.**
- **For classification, data may not be linearly separable (more on this in SVMs with non-linear kernels)**

Kernelized SVM

But what about more complex binary classification problems?

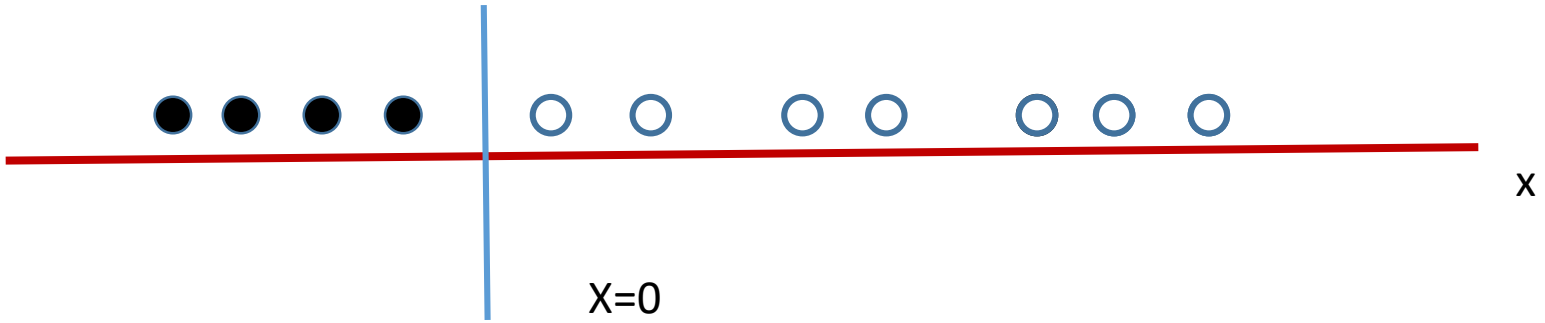




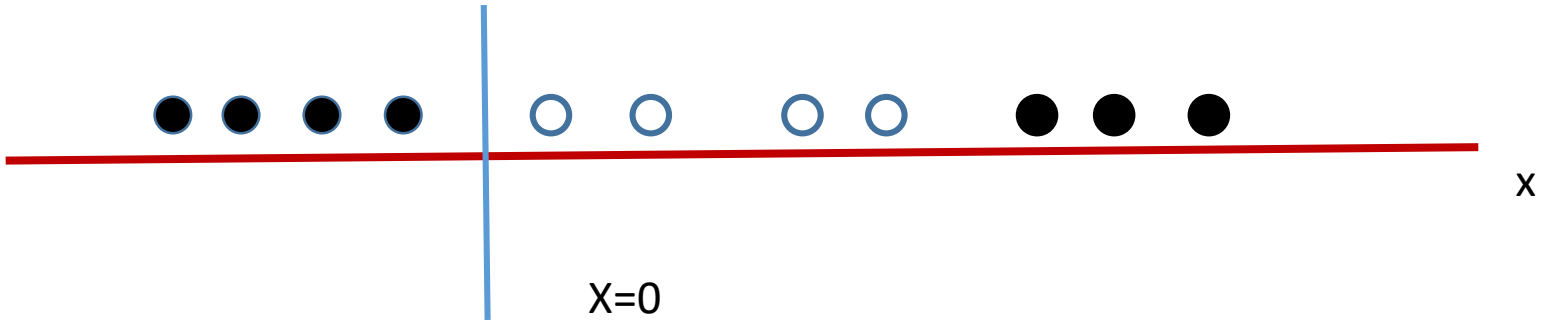
Easy for a linear classifier

Difficult/impossible for a linear classifier

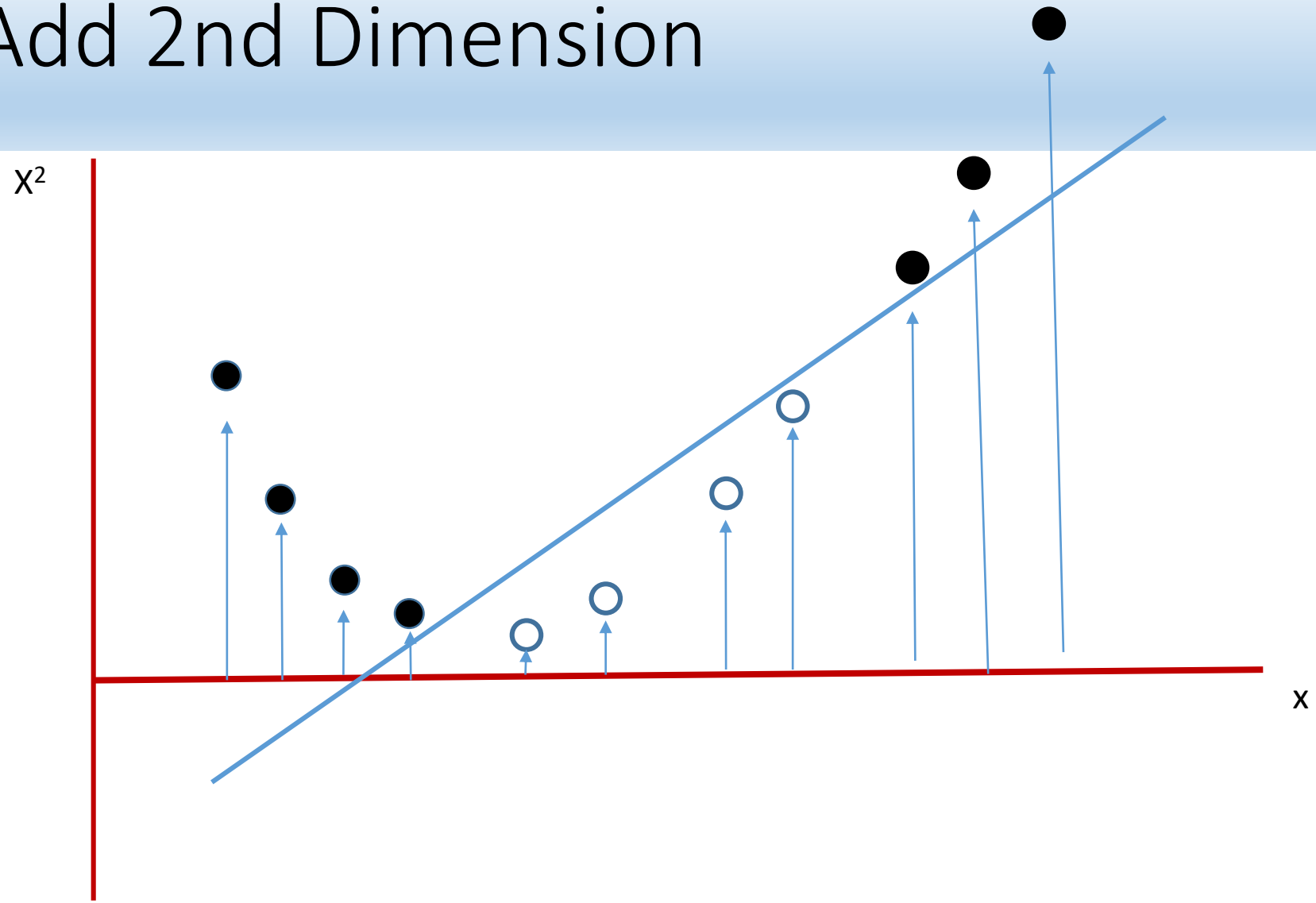
1 Dimensional Classification



1 Dimensional Classification

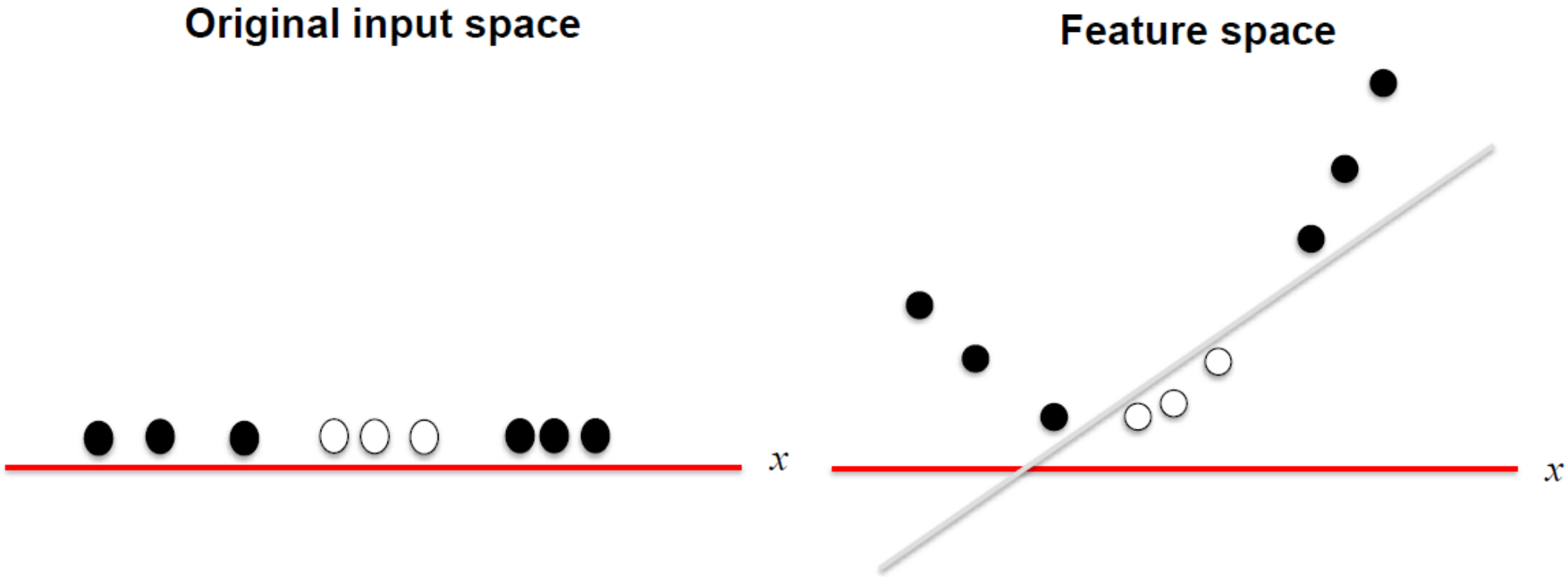


Add 2nd Dimension

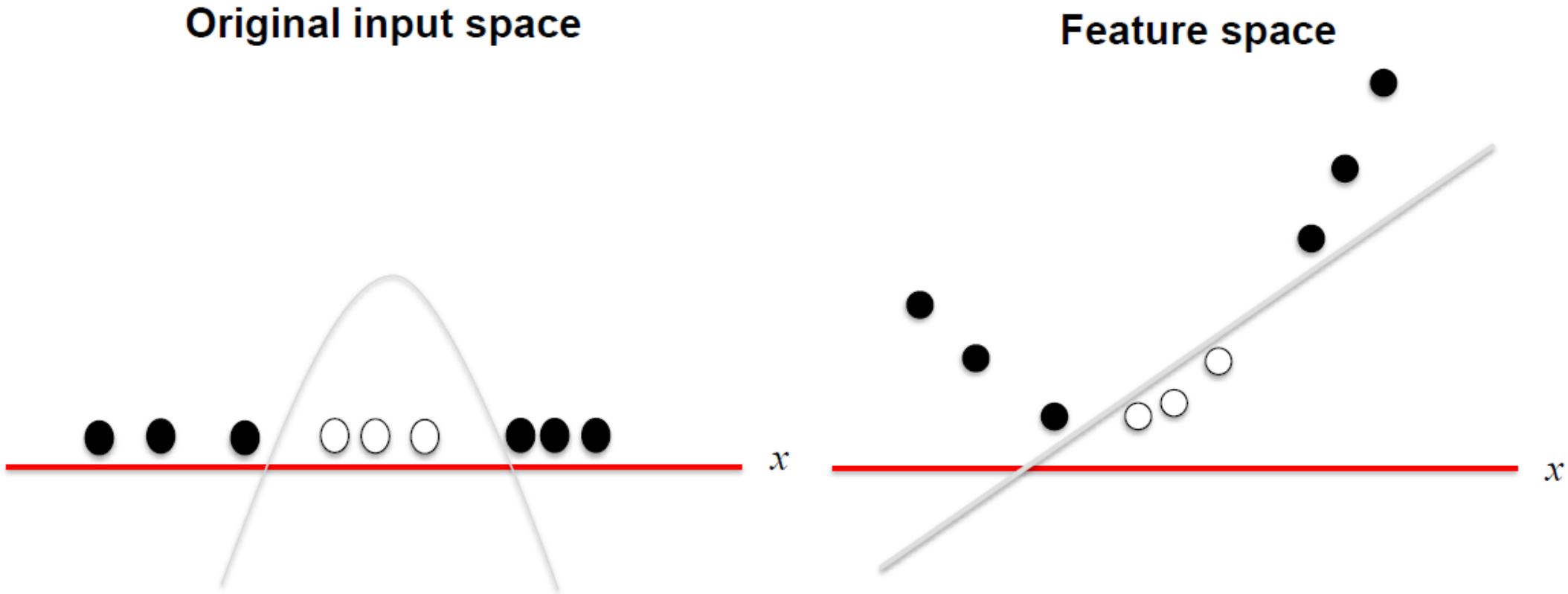


$$v_i = (x_i, x_i^2)$$

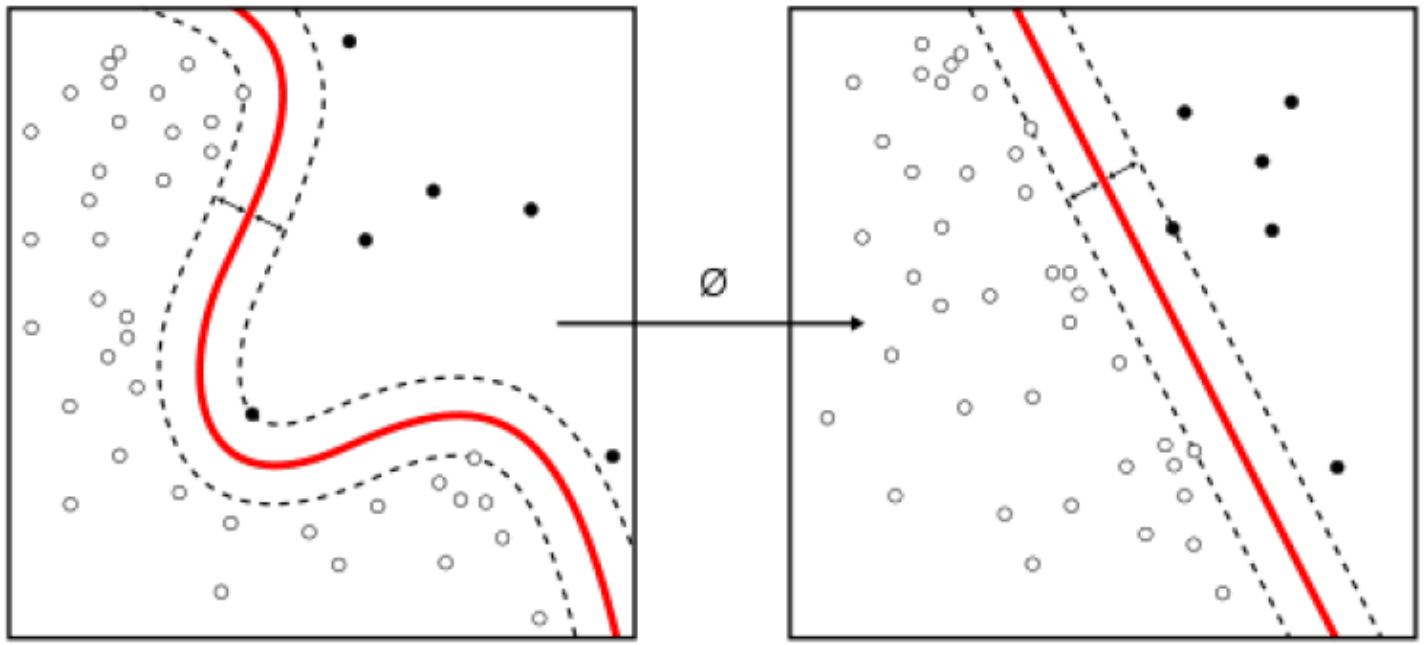
Relationship between Input and Feature Space



Relationship between Input and Feature Space

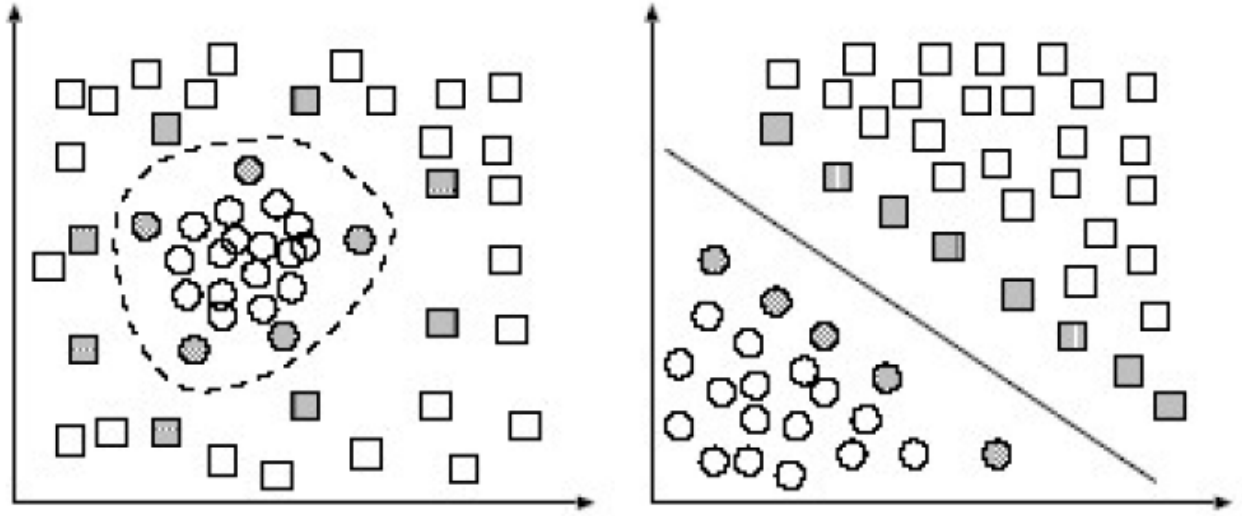


Transforming the Data Can Make It Much Easier for A Linear Classifier



Radial Basis Function Kernel

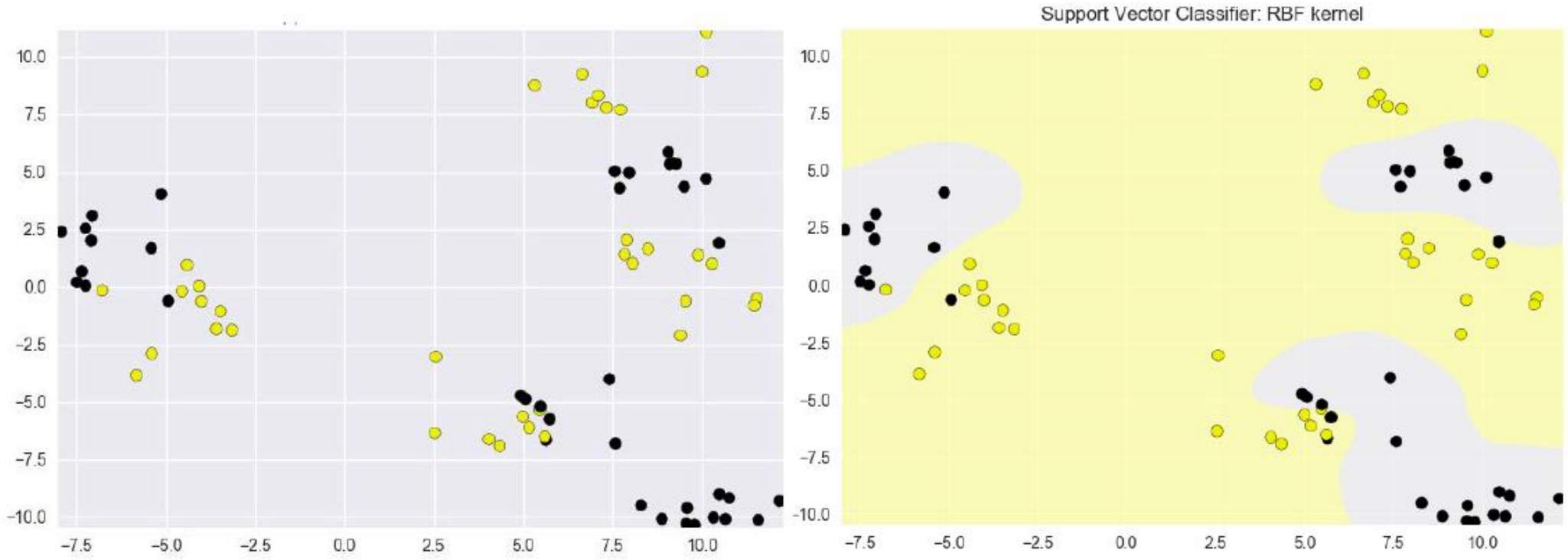
$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp\ [-\gamma \cdot \|\boldsymbol{x} - \boldsymbol{x}'\|^2]$$



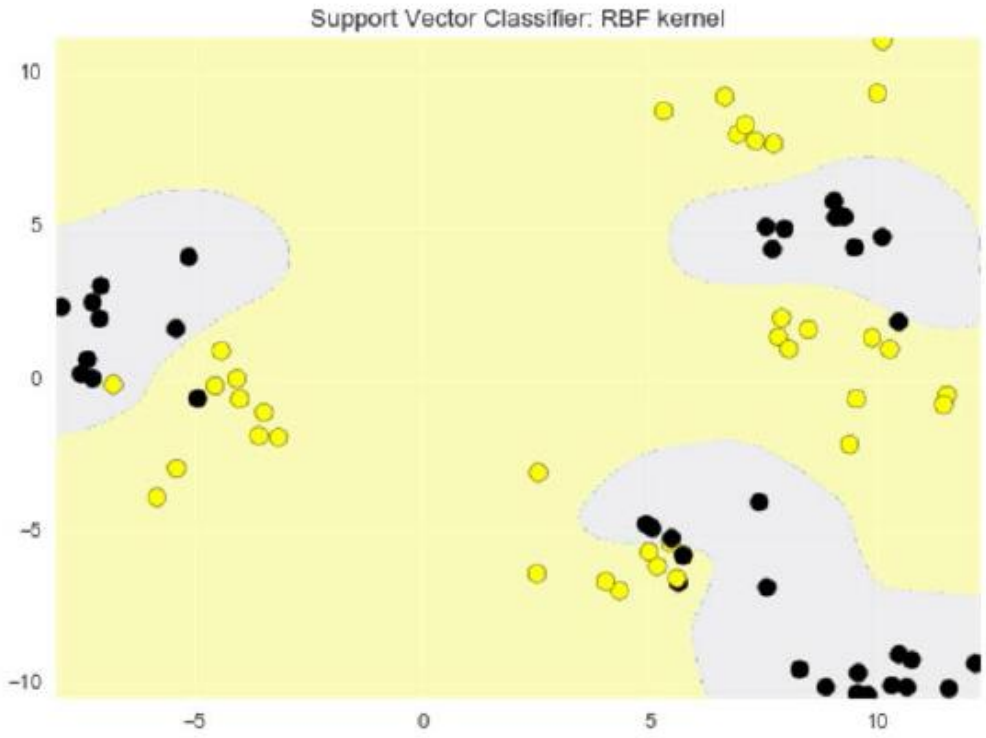
Original input space Feature space

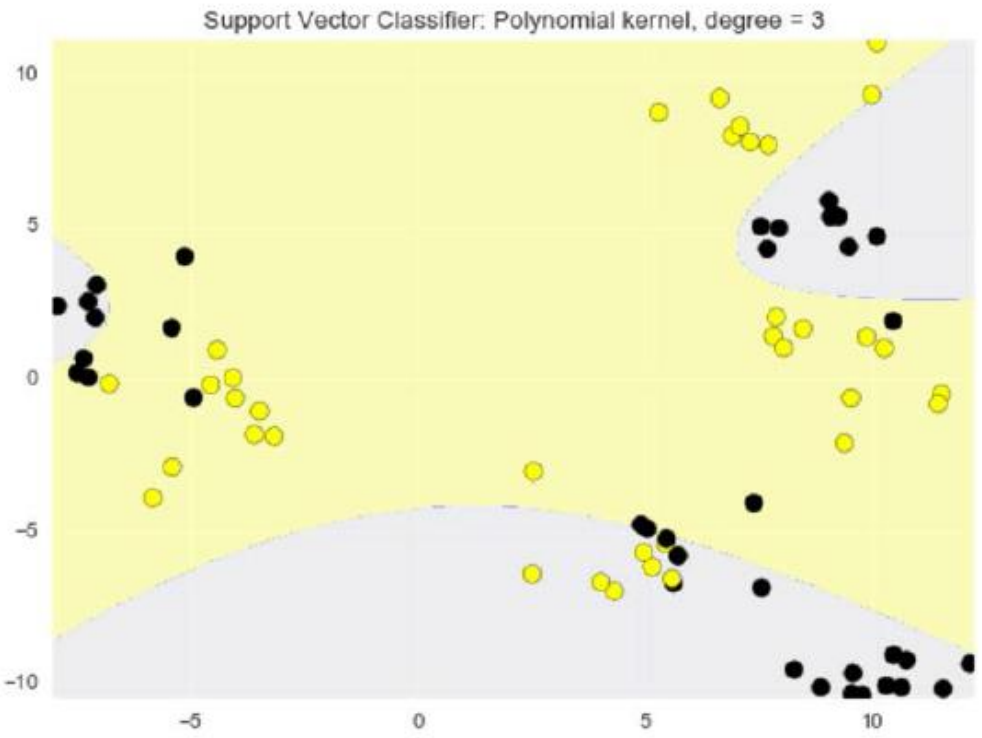
A kernel is a similarity measure (modified dot product) between data points

Applying the SVM with RBF kernel



Radial Basis Kernel vs Polynomial Kernel





Radial Basis Function kernel: Gamma Parameter

$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp\ [-\gamma \cdot \|\boldsymbol{x} - \boldsymbol{x}'\|^2]$$

gamma (γ): kernel width parameter

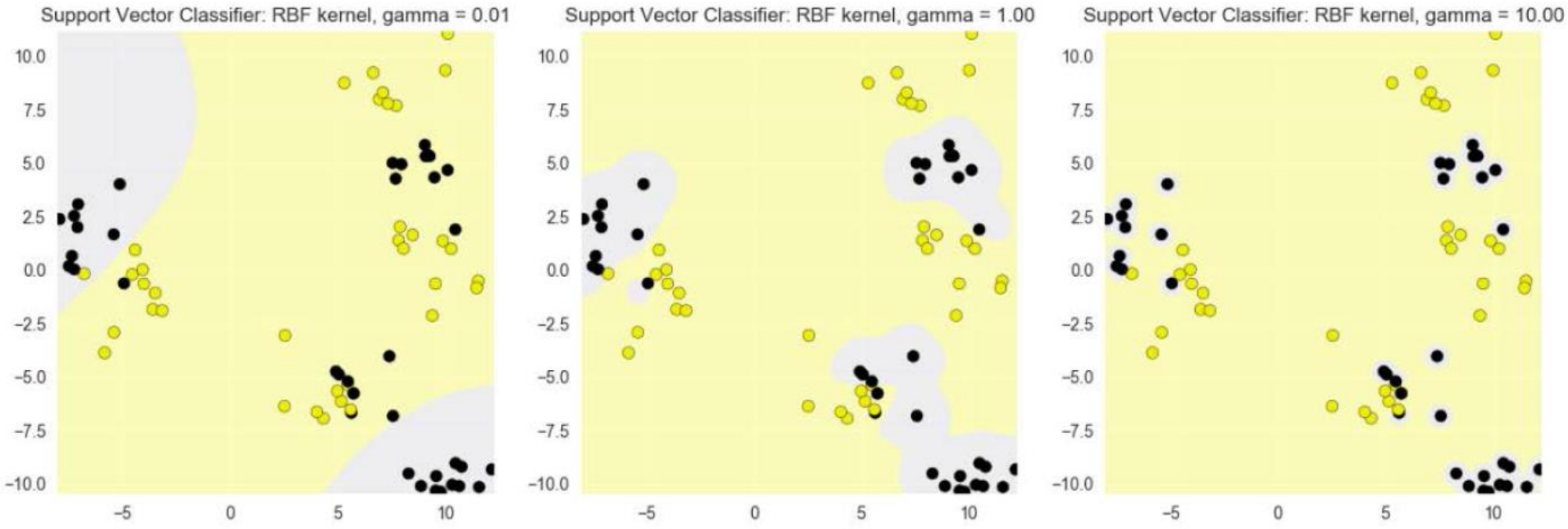
small gamma (0.01)

large gamma (10)

$\|\boldsymbol{x} - \boldsymbol{x}'\|^2$

Squared distance between x and x'

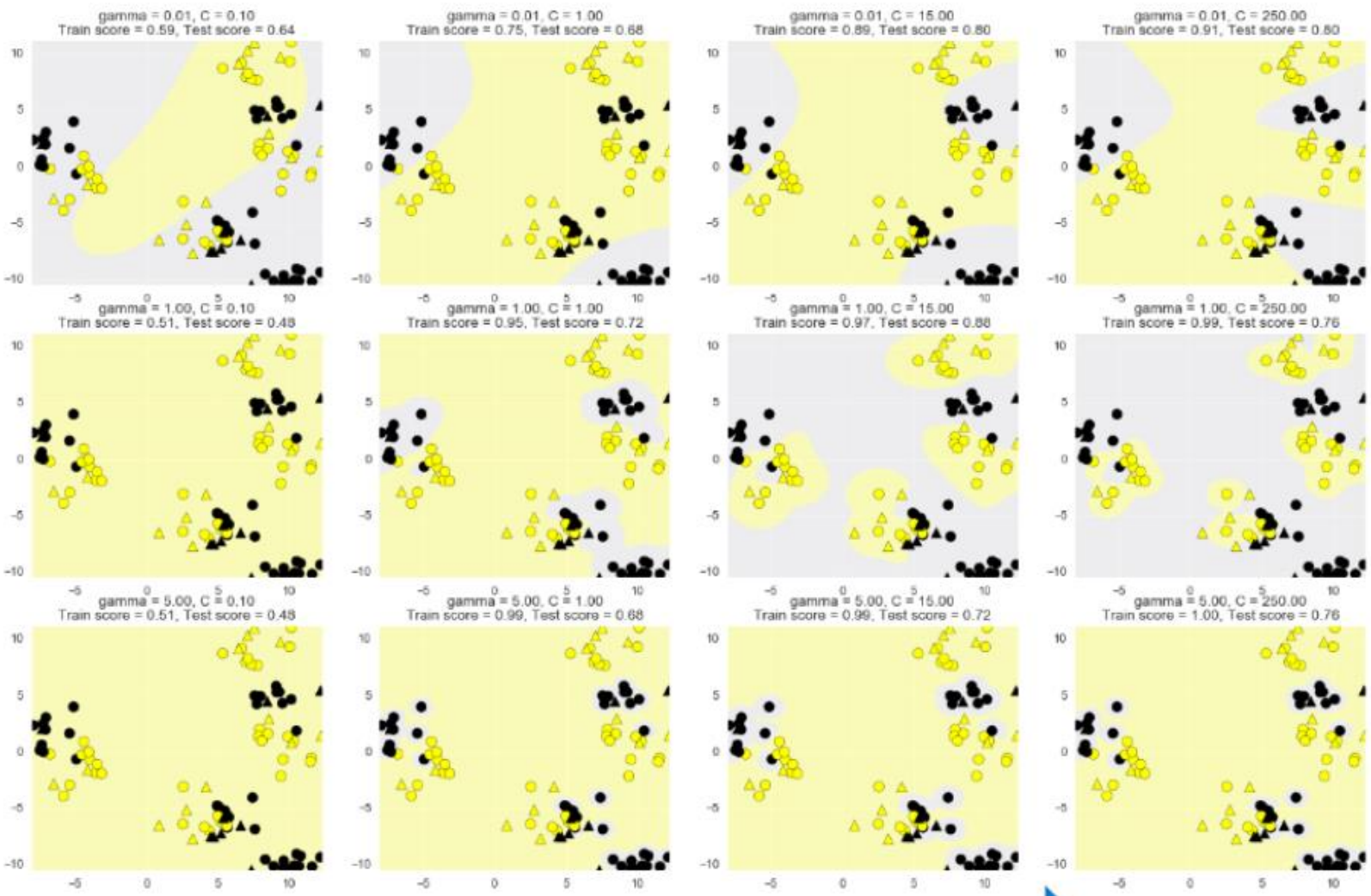
Effect of RBF Gamma on Decision Boundaries



gamma = 0.01

gamma = 1.0

gamma = 10



Increasing gamma

Increasing C

Kernelized SCV: pros and cons

Pros:

- Can perform well on a range of datasets.
- Versatile: different kernel functions can be specified, or custom kernels can be defined for specific data types.
- Works well for both low- and high-dimensional data.

Cons:

- Efficiency (runtime speed and memory usage) decreases as training set size increases (e.g. over 50000 samples).
- Needs careful normalization of input data and parameter tuning.
- Does not provide direct probability estimates (but can be estimated using e.g. Platt scaling).
- Difficult to interpret why a prediction was made.

Kernelized SVC: Important Parameters

Model complexity

- **`kernel`: Type of kernel function to be used**
  - *Default = 'rbf' for radial basis function*
  - *Other types include 'polynomial'*
- **kernel parameters**
  - `gamma` *(γ): RBF kernel width*
- **`C`: regularization parameter**
- **Typically `C` and `gamma` are tuned at the same time.**

Questions?