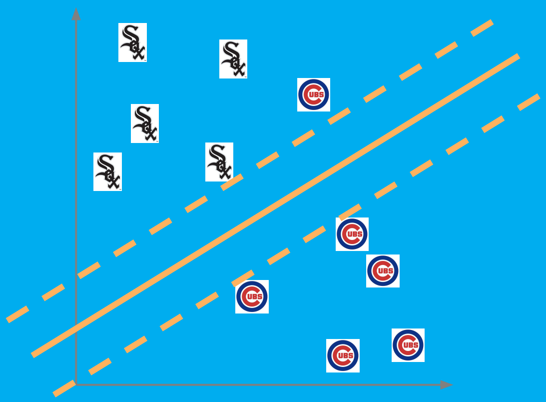
Support Vector Machines

Summer 2018



Hello! Let’s learn some things

Machine learning explained by XKCD

https://xkcd.com/1838/

Last Modified 6/11/2018 12:52 PM Eastern Standard Time

Printed

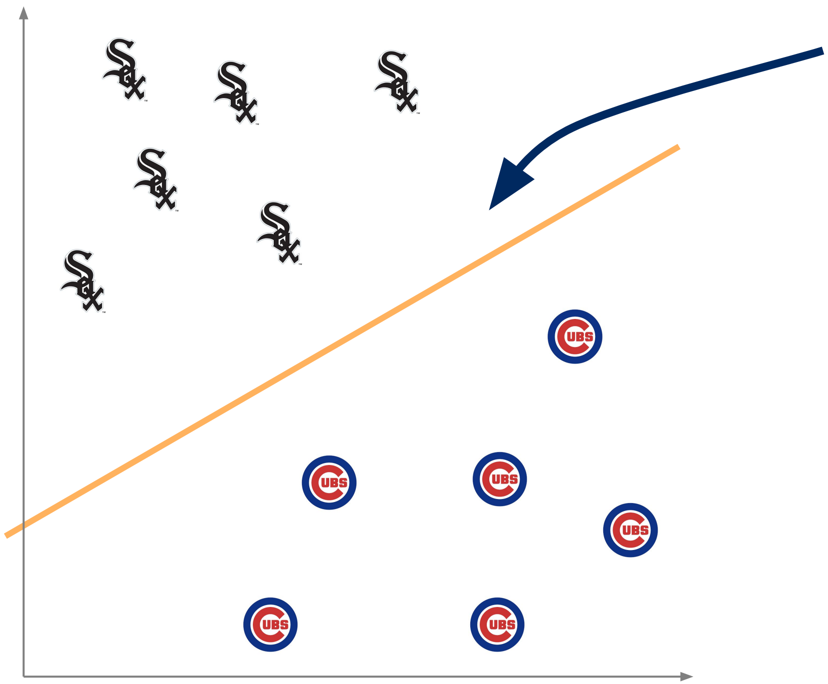
What is a support vector machine?

Supervised, linear classification

You, typeface connoisseur: Is that Comic Sans?!
Me, science genius: I read some paper on the internet once that people remember things written in ugly typefaces better...

SVMs look like a line in 2D space

X2 = fans who know the South Side is the best side



Support vector machine hyperplane

X1 = fans who love to see their team lose

Revisiting our machine learning schema

- SVMs are a type of **supervised learning model**
- Can be used both for **classification and regression** (covering only classification today)
- They are a **binary classifier** (can be extended to multiple classes, but gets complicated)
- It is a **linear classifier**, meaning it uses a linear combination of the inputs to make its classification prediction
- Other ways of doing supervised classification include: logistic regression, k-nearest neighbors, decision trees, neural networks

Reasons to care about support vector machines

1. Someone will ask you about it in an interview
2. It’s easy to interpret when you need a binary classifier
3. It’s memory efficient
4. It’s robust to a whole bunch of issues, including sparse data and high dimensions

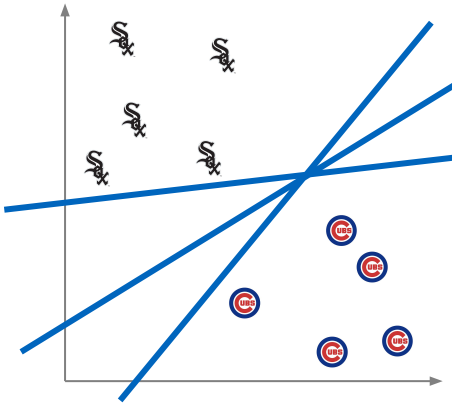
In the beginning, there was the perceptron

Perceptron

In the beginning, there was the perceptron

Perceptron

- Invented in late 1950s
- Separates observations into classes using **a hyperplane as the decision boundary**
- Requires that the classes can be separated by a line (**linear separability**)
- Rigid and sensitive to outliers

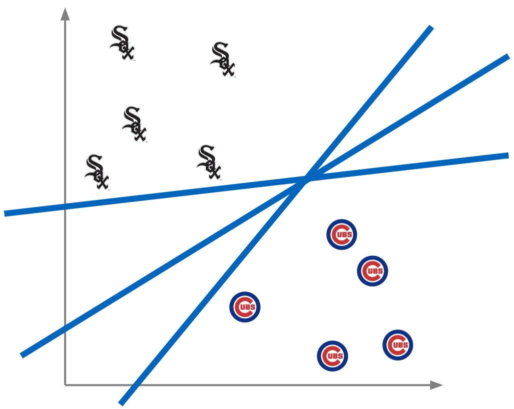


$$f(x) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

In the beginning, there was the perceptron

Perceptron

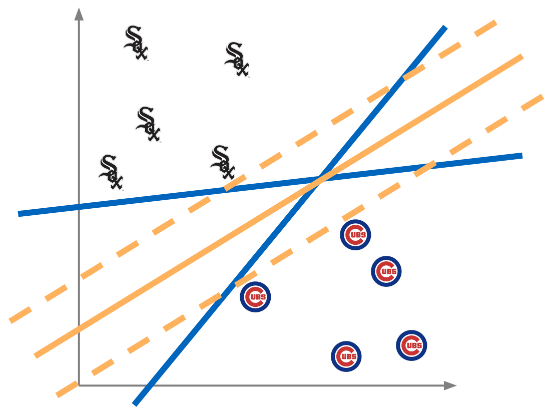
- Invented in late 1950s
- Separates observations into classes using **a hyperplane as the decision boundary**
- Requires that the classes can be separated by a line (**linear separability**)
- Rigid and sensitive to outliers



$$f(x) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

Maximal margin classifier

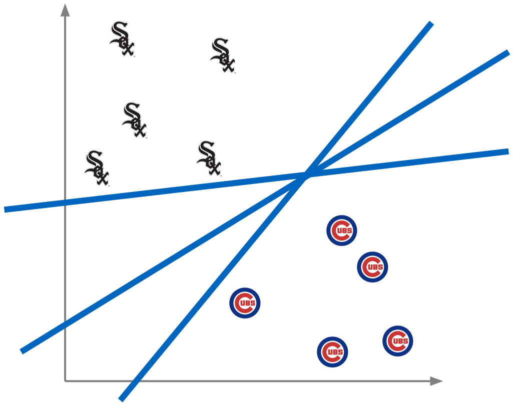
- Makes statistically concrete the **intuition that some lines are better** than others
- Takes perceptron and adds **optimal stability** (widest margin)
- Fails completely if data not linearly separable



In the beginning, there was the perceptron

Perceptron

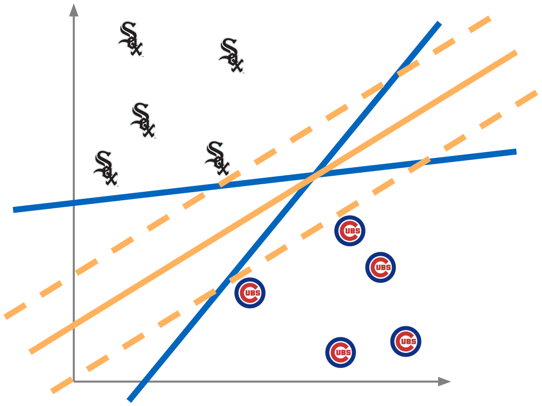
- Invented in late 1950s
- Separates observations into classes using **a hyperplane as the decision boundary**
- Requires that the classes can be separated by a line (**linear separability**)
- Rigid and sensitive to outliers



$$f(x) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

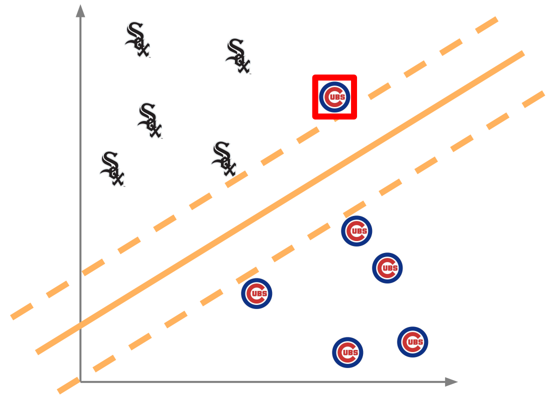
Maximal margin classifier

- Makes statistically concrete the **intuition that some lines are better** than others
- Takes perceptron and adds **optimal stability** (widest margin)
- Fails completely if data not linearly separable



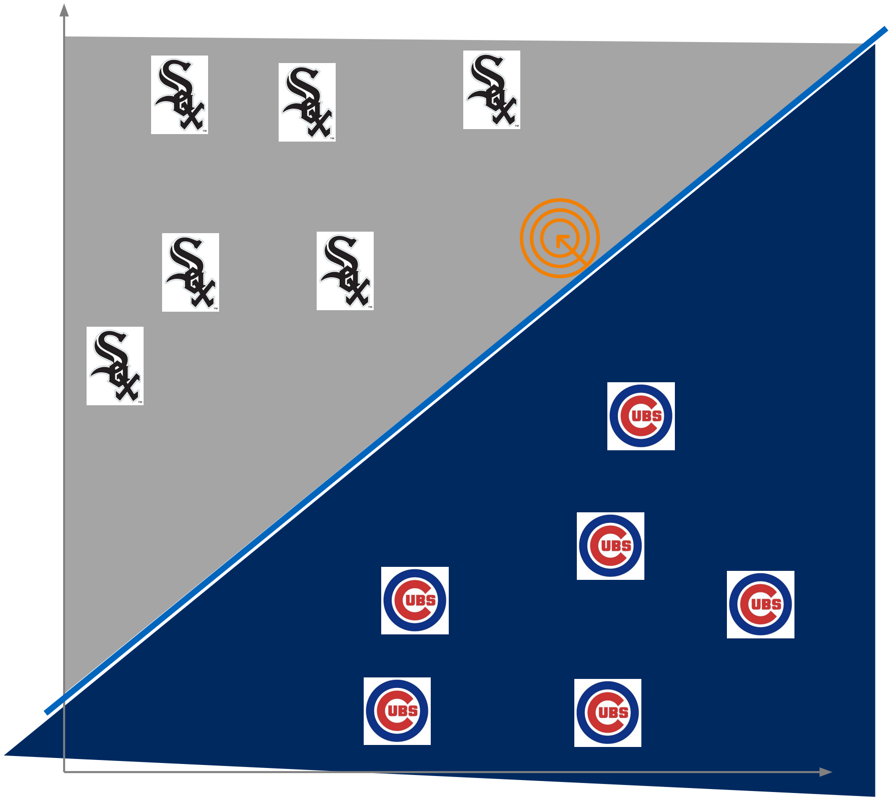
Support vector machine

- Developed early 1990s
- SVM adds two generalizations:
  1. **Soft margin** for outliers
  2. **Kernel trick** for data that's not **linearly separable**: gives us complex feature space with minimal computational complexity



SVMs can be further extended to multi-class data.

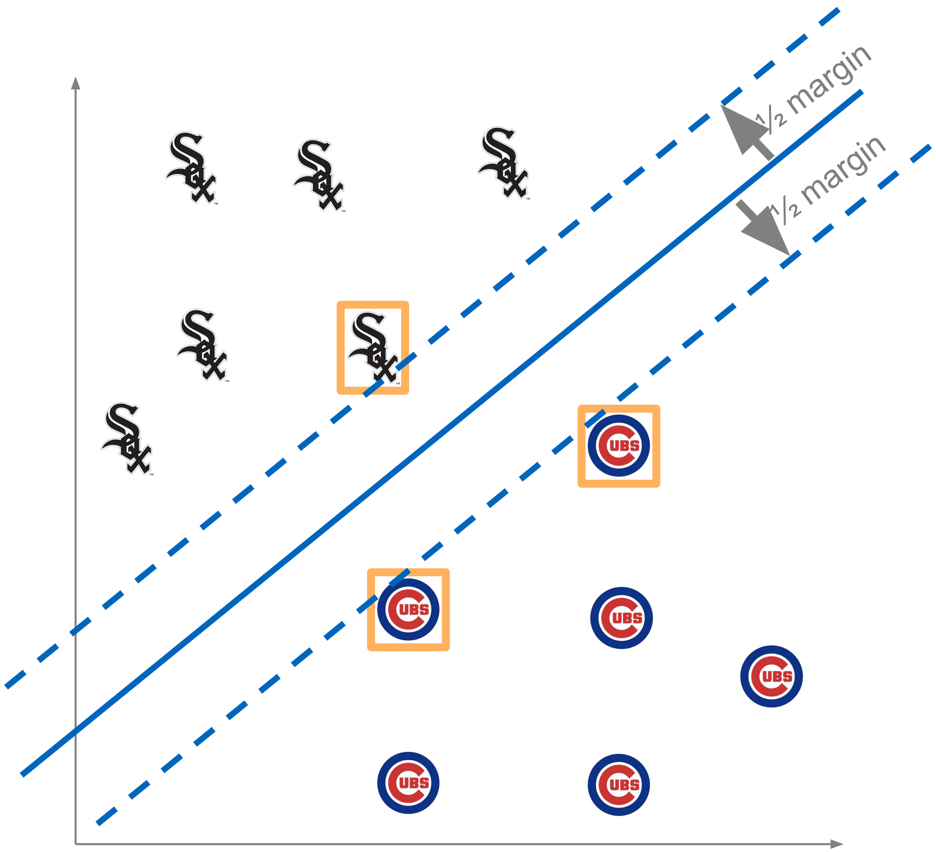
What's the use of SVM? Splitting the feature space to enable prediction



- Why are we interested in finding the best line (hyperplane) that separates our classes?
- In order to predict new instances or observations, we need to separate our feature space

- A better split gives us **higher confidence** that our predictions of new data are correct

Defining the decision boundary: the support vectors of SVM



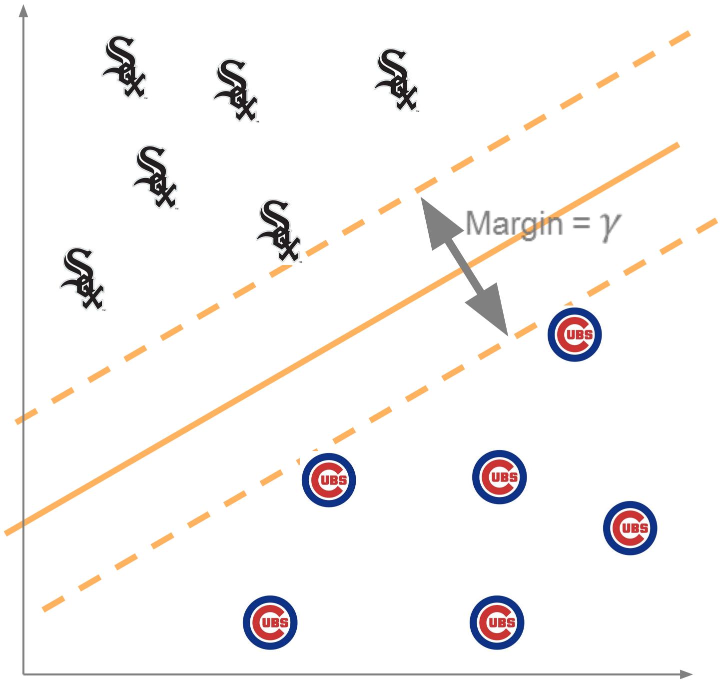
- **Each observation is represented by a vector of values**

- Why do we define the margin by the distance to the closest points (and not, e.g., by the average distance)?
- SVM is insensitive to points far away from the decision boundary

- The separating hyperplane is defined by the **support vectors** (highlighted in orange)

- If there are no degeneracies (if the data is linearly separable), then we **need d+1 support vectors** (where d is the # of dimensions)

Creating the largest margin between classes (hard margin)



- Why do we want the largest margin?
  - **Intuition**: the larger the margin, the more confidence we have that we have the correct classification of our test data
  - **Statistics**: the VC dimension, or how complex a model is and how likely it is to capture too many wrong points
  - E.g. a curved line may fit the test data better but is more likely to make mistakes in prediction

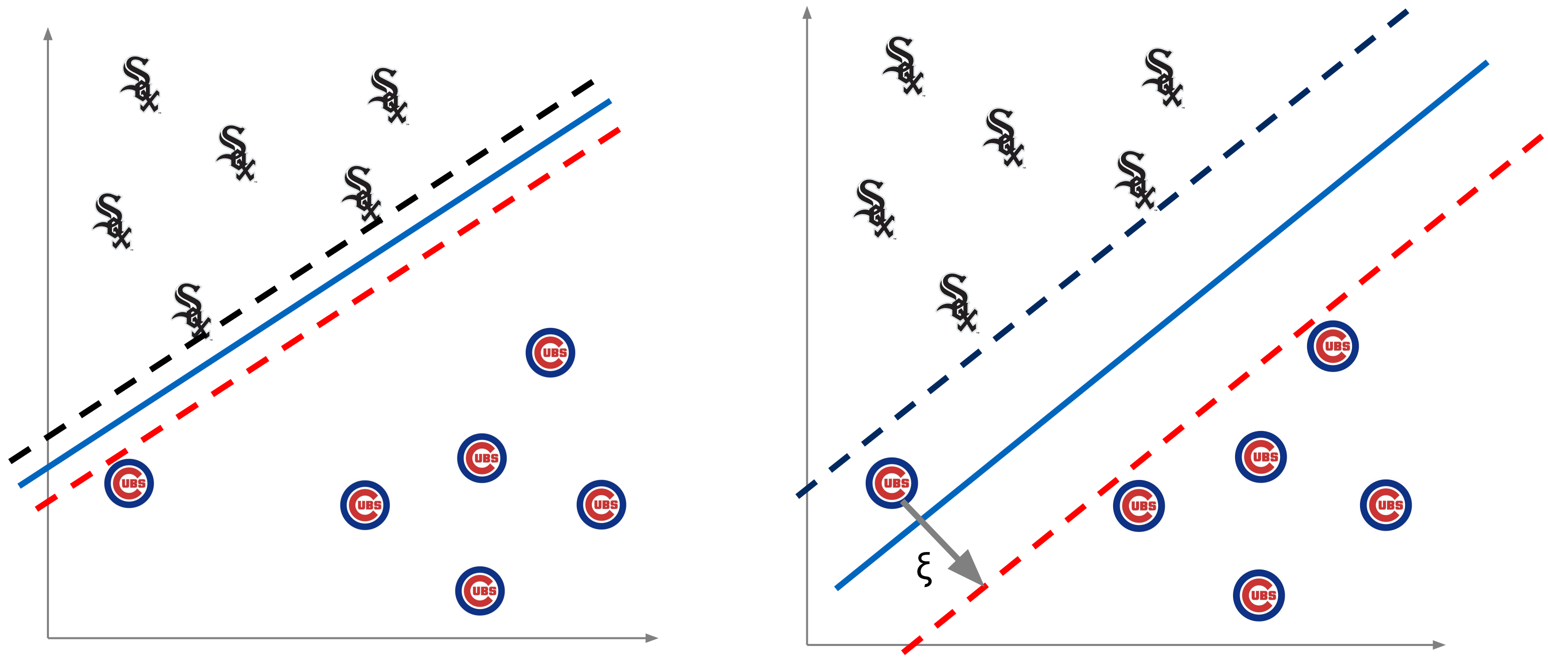
Solving for the largest margin

$$\max_{w,\gamma} \gamma \quad s.t. \forall i, y_i(w \cdot x_i + b) \geq \gamma \quad \Rightarrow \quad \min_w \tfrac{1}{2} \|w\|^2 \quad s.t. \forall i, y_i(w \cdot x_i + b) \geq 1$$

- Solving for the margin is a **constrained optimization** problem
- If we maximize γ, we can do so by increasing w (weights) as much as we want
- Instead, rewrite to maximize the unit w
- SVM can be used for **online learning**, where the model parameters are slightly modified with the introduction of a new data point and where data comes in as a stream

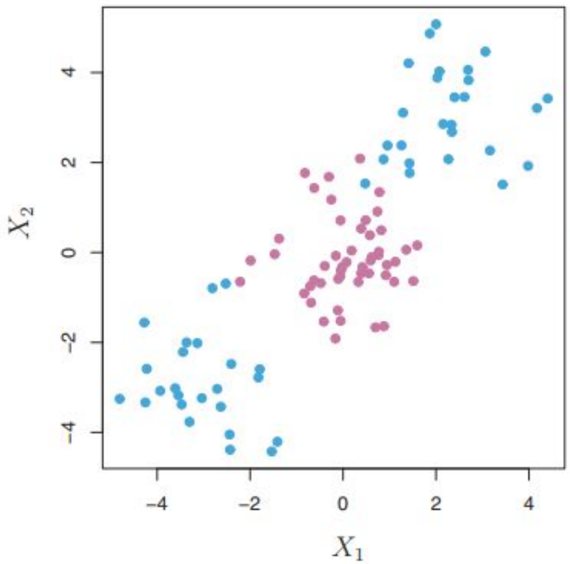
From maximal margin classifier to SVM: ξ and kernel trick

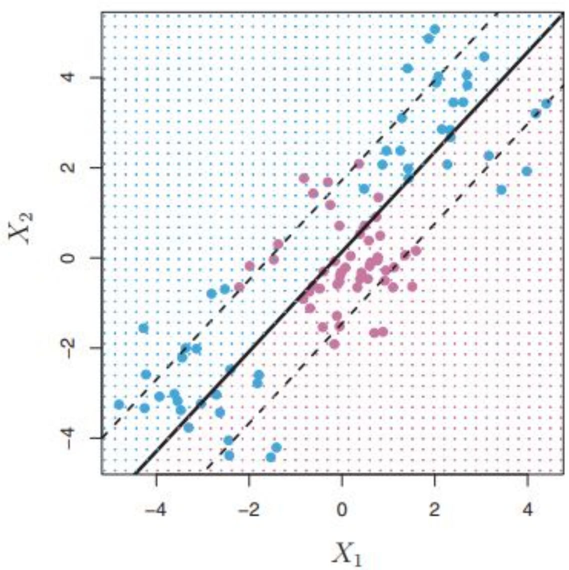
SVM: Soft margin classification



- A **hard margin** (1) works only if the data is linearly separable, and (2) is sensitive to outliers as it will work to completely capture the data on the correct side of the line
- We create a **soft margin** by adding in **slack to our cost function**
- We manually set **C**, a tuning parameter, to tell the algorithm how much slack it has to misclassify observations
    - Large C = misclassify more

SVM: Using the kernel trick when the data isn't linearly separable





- When the classes aren't linearly separable, we can't use a linear separator
- In the regression case (e.g. OLS), we dealt with this by adding to the dimension of our feature space by transforming the regressors (e.g. x^2)
- The kernel trick is a similar approach

- **Mathematical aside**: turns out, to solve our SVM, we need only the dot product of the observations (instead of the observations themselves)

- **In the kernel trick, replace every instance of the dot product with a new function** (e.g. polynomial, like X^2) that's equivalent to the dot product

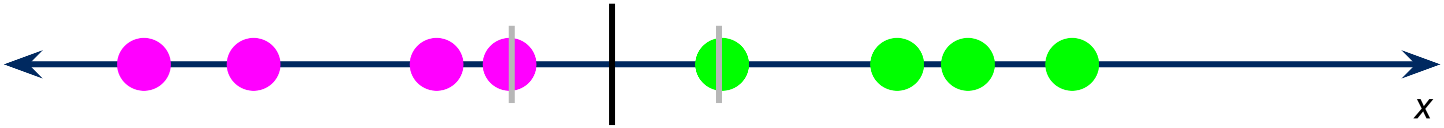
Last Modified 6/11/2018 12:52 PM Eastern Standard Time

Printed

Charts from http://www-bcf.usc.edu/~gareth/ISL/ISLR%20First%20Printing.pdf

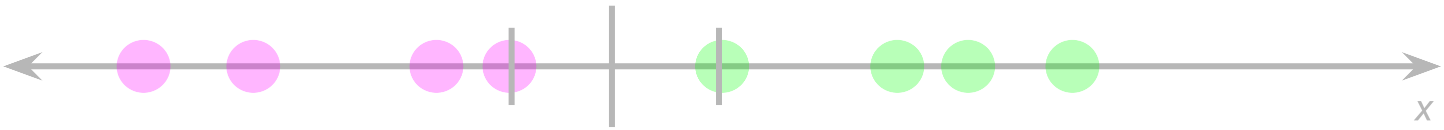
Visualizing the kernel trick

Best case scenario is when our data is linearly separable: it's easy to draw our separating line in this case:

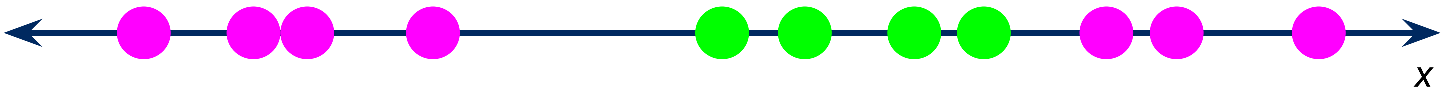


Visualizing the kernel trick

Best case scenario is when our data is linearly separable: it's easy to draw our separating line in this case:

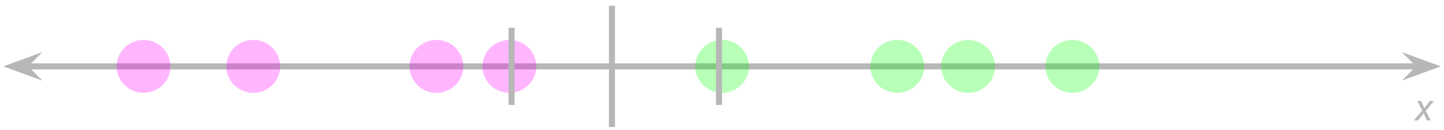


Are we out of luck if our data isn't perfectly separable?

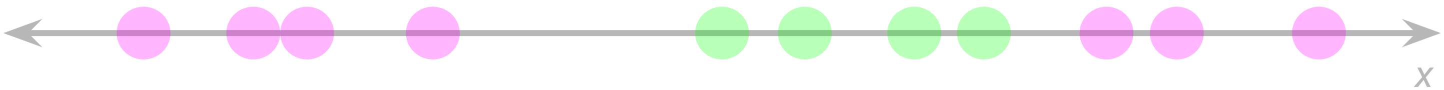


Visualizing the kernel trick

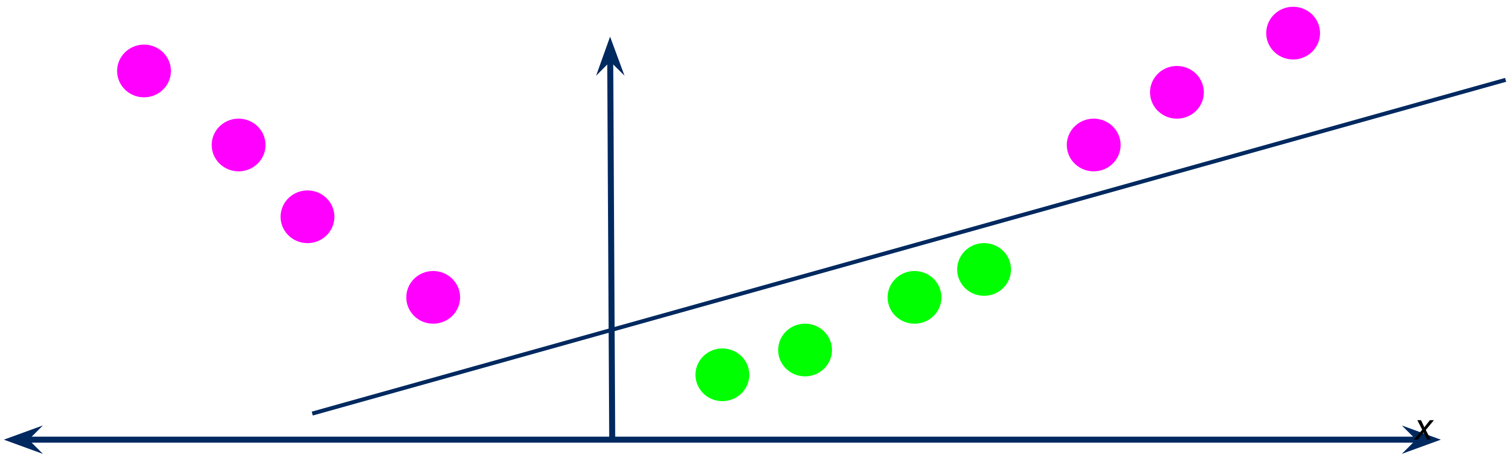
Best case scenario is when our data is linearly separable: it's easy to draw our separating line in this case:



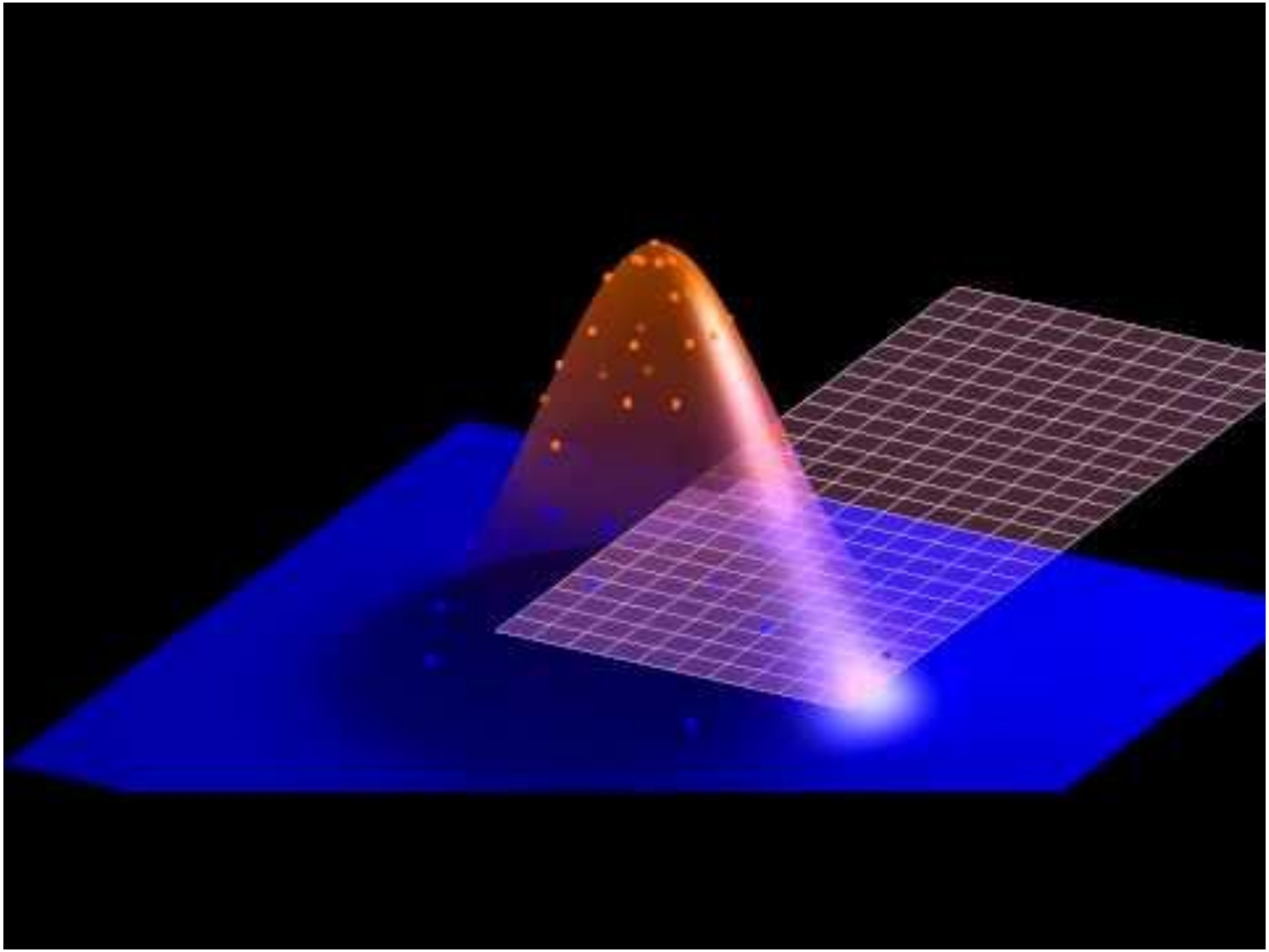
Are we out of luck if our data isn't perfectly separable?



We can still use a linear separator, if we shift our data into a higher-dimensional space!

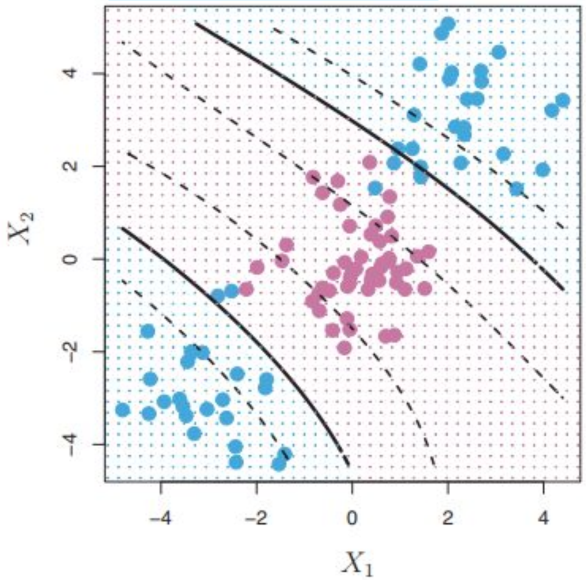


Visualizing the kernel trick

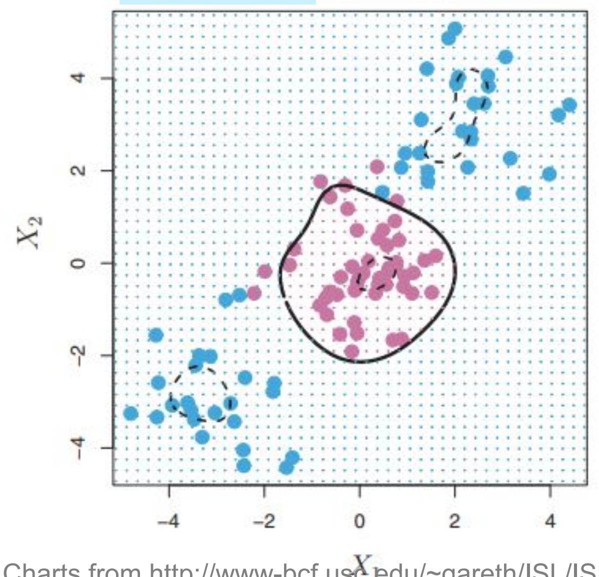


SVM: Using the kernel trick when the data isn't linearly separable

Polynomial kernel (=3)



Radial kernel

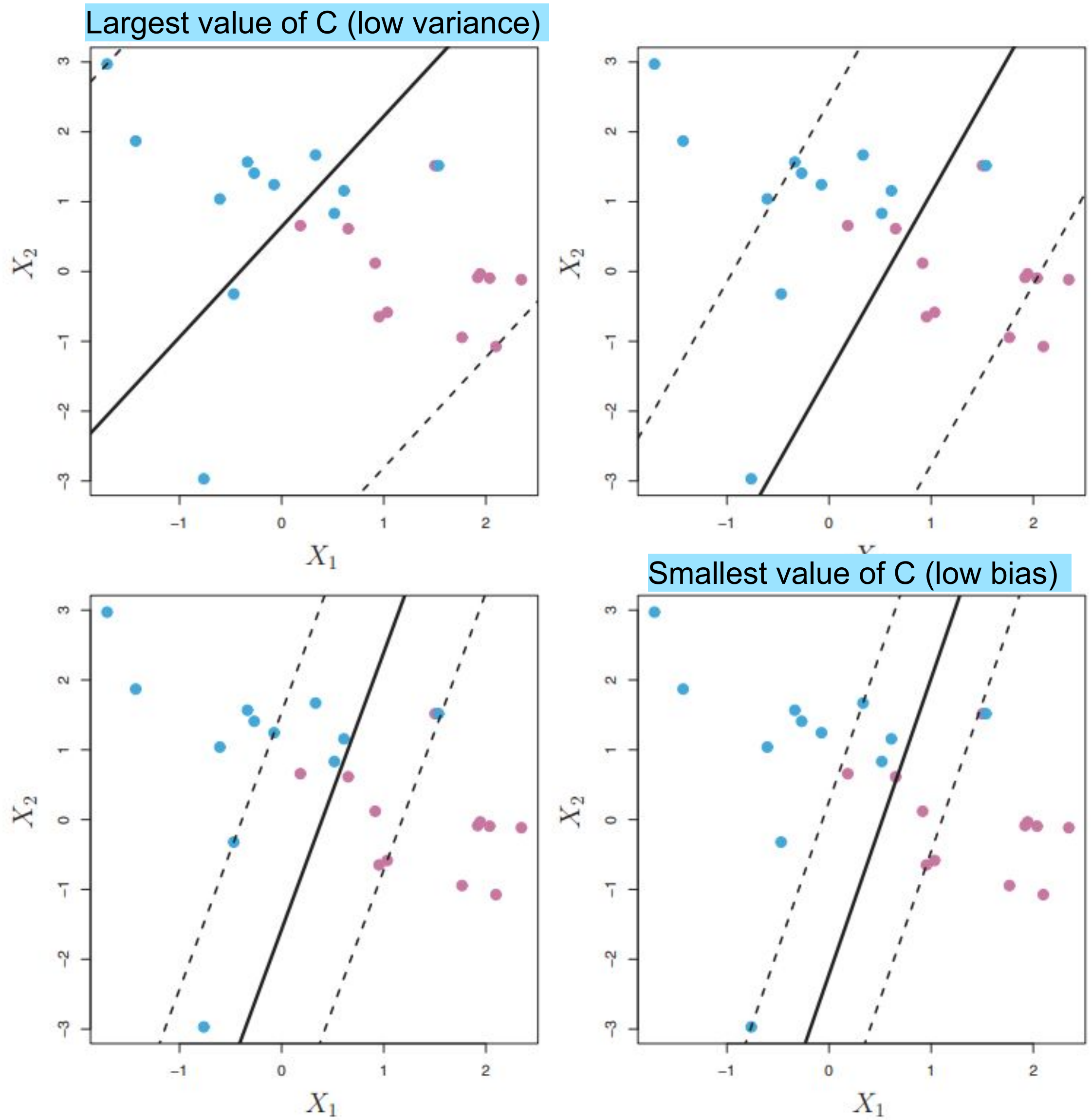


- **Mathematical aside**: turns out, to solve our SVM, we need only the inner product of the observations (instead of the observations themselves)
- In the kernel trick, replace every instance of the inner product with a new function (e.g. polynomial, like X^2)
- But, **why use kernels instead of just enlarging the feature space directly**?
  - By using the kernel, we need to compute only the dot product of all observations, not the actual transformations
  - This **saves computation**
  - Some expansions are infinite so we couldn't solve them without the kernel anyway

Charts from http://www-bcf.usc.edu/~gareth/ISL/ISLR%20First%20Printing.pdf

Tuning SVM hyperparameters: C and γ

Tuning the hyperparameters of an SVM: soft margin through “C”

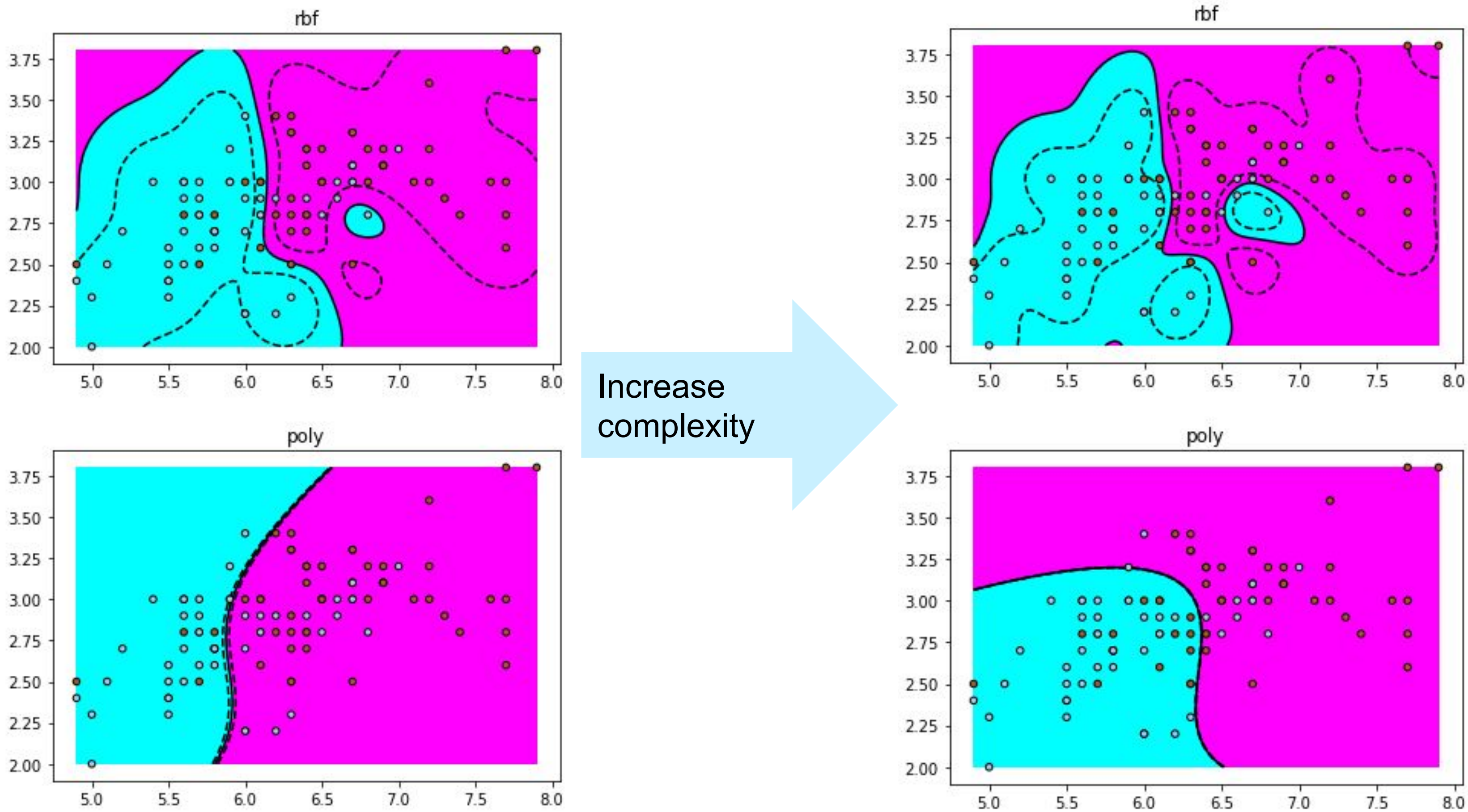


- The level of **C** tells the algorithm **how much slack it has to misclassify** some observations
- Finding the **optimal value** of C requires manual tuning and in practice is **often done via grid search and cross-validation**

Charts from http://www-bcf.usc.edu/~gareth/ISL/ISLR%20First%20Printing.pdf

Tuning the hyperparameters of an SVM: complexity through γ

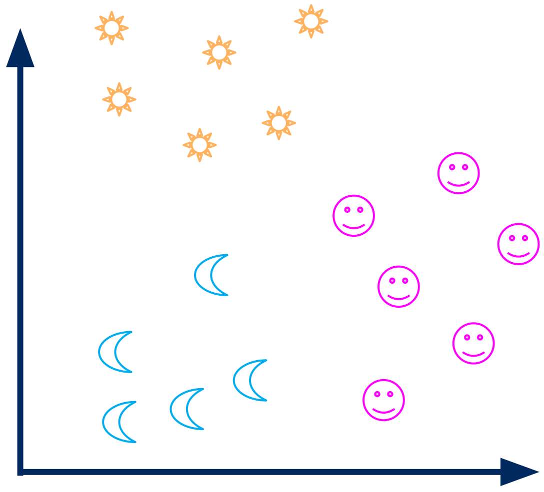
γ=10 γ=20 (more complex)



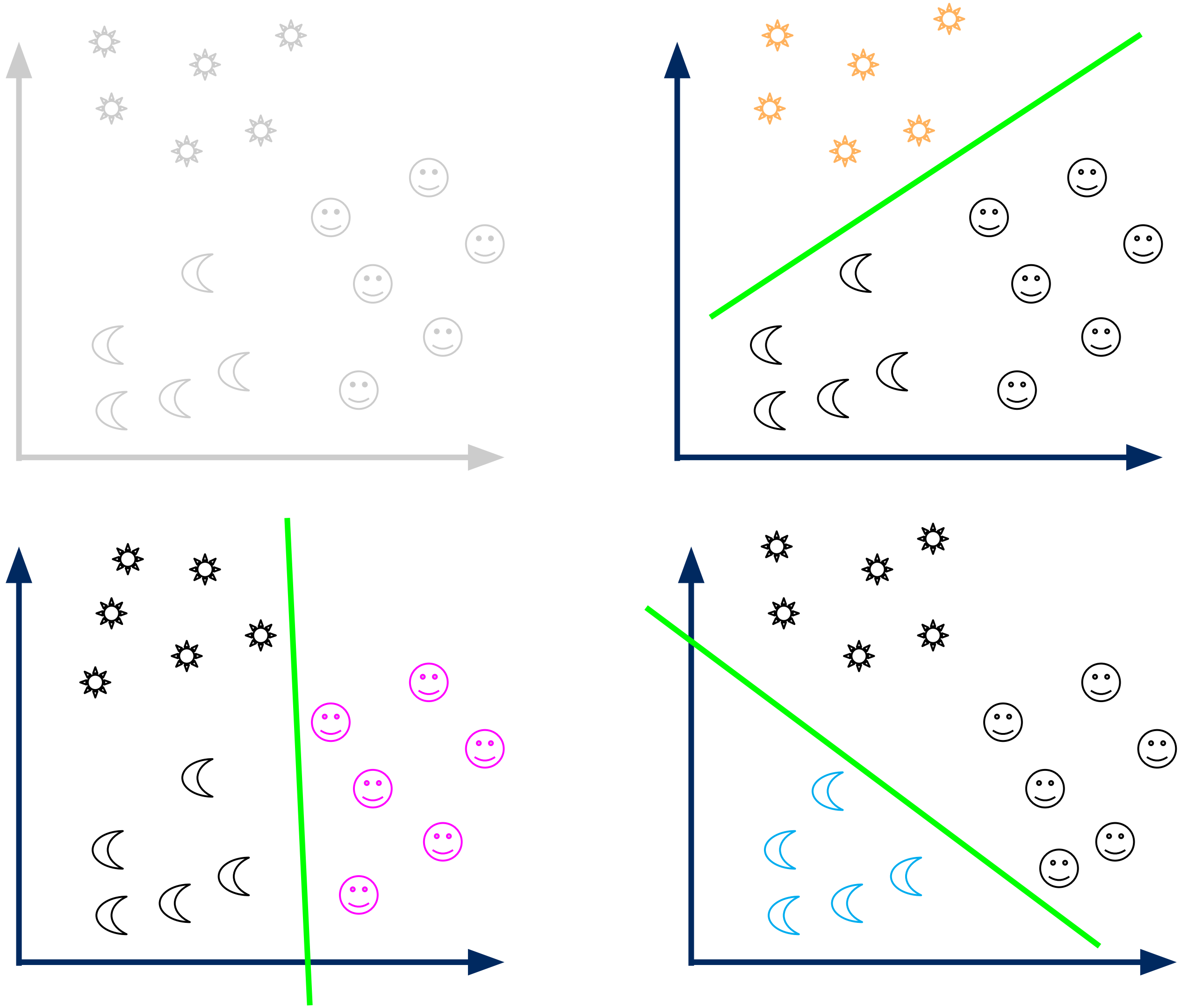
- The hyperparameter gamma is **used only with non-linear kernels** (e.g. polynomial, radial basis function (RBF))
- **Gamma** tells the model the kernel coefficient, which **affects how each observation influences the support vectors**

Extending to multiclass

Extension to multiclass estimating: one vs. rest



Extension to multiclass estimating: one vs. rest

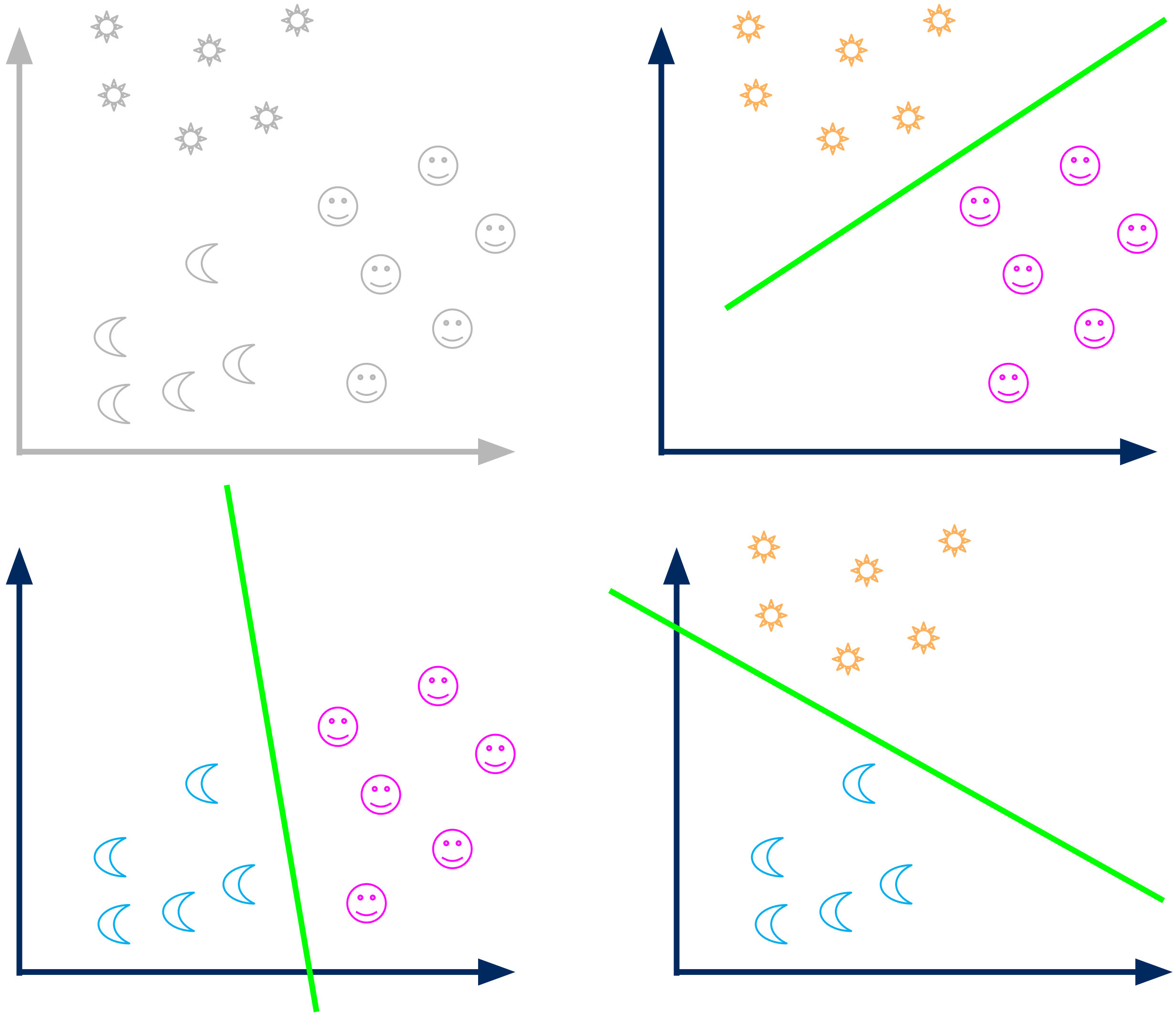


Three models are fit, and prediction is done using the model that would give the widest margin.

Last Modified 6/11/2018 12:52 PM Eastern Standard Time

Printed

Extension to multiclass estimating: one vs. one

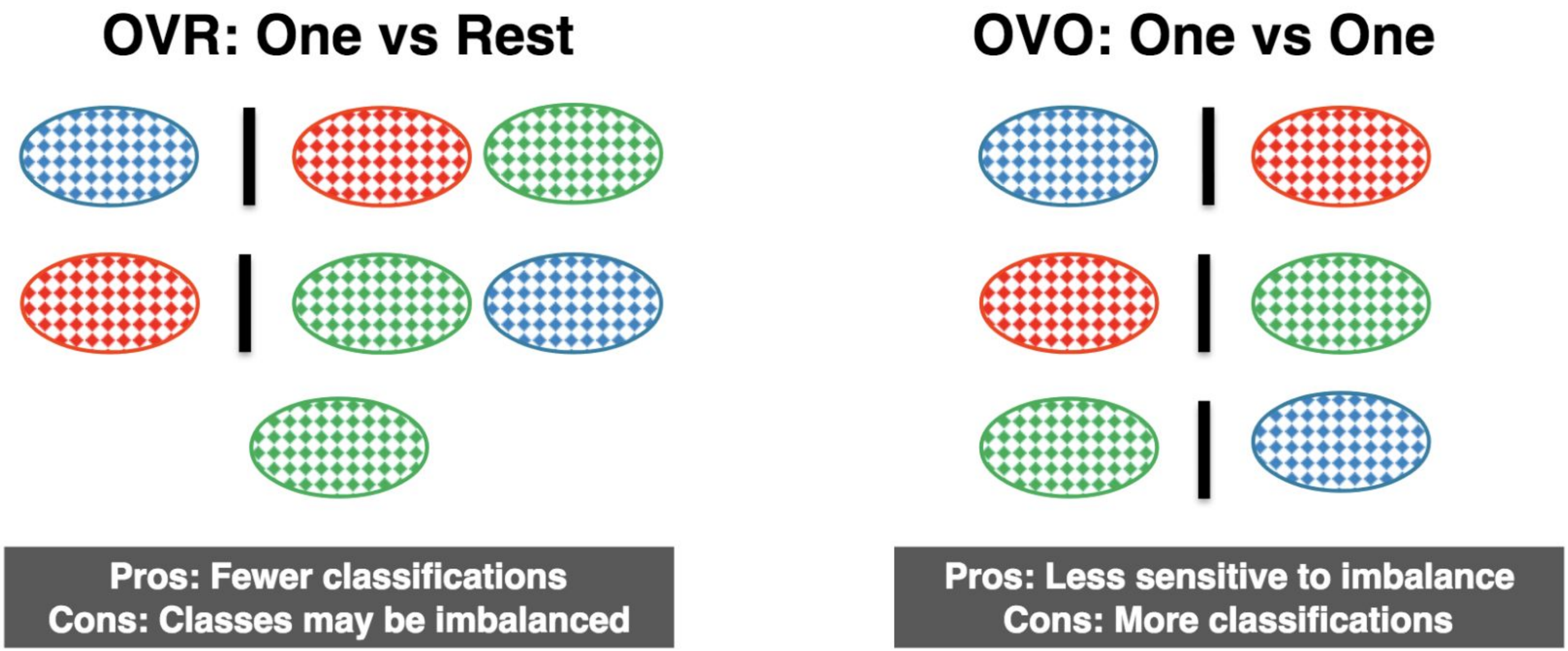


Pairwise models are fit, and prediction is done using majority voting.

Last Modified 6/11/2018 12:52 PM Eastern Standard Time

Printed

Expanding to multiple classes



Scaling data before using SVM

- Inputs should be scaled prior to running SVM
- Most common scaling: subtract the mean and divide by the SD
  - Remember to scale the testing dataset with the mean and SD of the training dataset
- Why scale?
  - Avoid data with large ranges dominating columns with smaller ranges
  - Avoid numerical issues from kernel computation

When should we use an SVM?

Pros	Cons
▪ Easy to interpret (output is a class) ▪ Can be used with high dimensional data, even if the # dimensions > # observations ▪ Can be used with sparse data ▪ Robust to outliers that are far away from the decision hyperplane (which is affected only by the support vectors) ▪ Defined by low number of support vectors so memory efficient	▪ Does not compute probabilities ▪ To compute probabilities requires cross-validation, which is computationally expensive ▪ Perform poorly with unbalanced classes ▪ Performs poorly with overlapping classes ▪ Hard to generalize beyond two classes (though possible)

Last Modified 6/11/2018 12:52 PM Eastern Standard Time

Printed

Implementing SVM in Python with scikit-learn

Fitting a model

class `sklearn.svm.` **SVC** (*C=1.0, kernel='rbf', degree=3, gamma='auto', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', random_state=None*)

Implementing SVM in Python with scikit-learn

Fitting a model and predicting a new observation

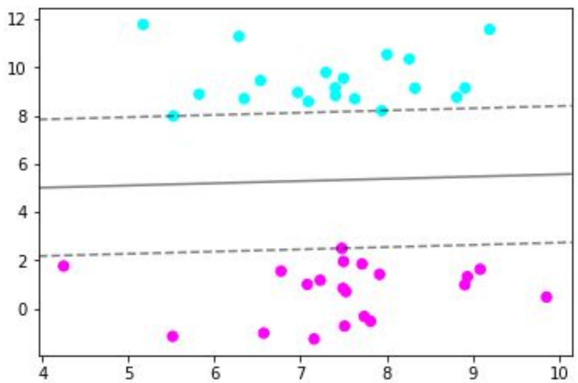
In [2]:
```
# create 40 separable points in two classes
X, y = make_blobs(n_samples=40, centers=2, random_state=6)

# fit the model in two steps

# 1) Set the hyperparameters
model = svm.SVC(kernel='linear', C=1000)

#2) Fit the model
model.fit(X, y)
```

Out[2]:
```
SVC(C=1000, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='linear',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```



To predict a new data point, we test (6,11) and (5,0):

```
model.predict(np.array([6,11]).reshape(1,-1))
```

```
array([0])
```

```
model.predict(np.array([5,0]).reshape(1,-1))
```

```
array([1])
```

Last Modified 6/11/2018 12:52 PM Eastern Standard Time

Printed