

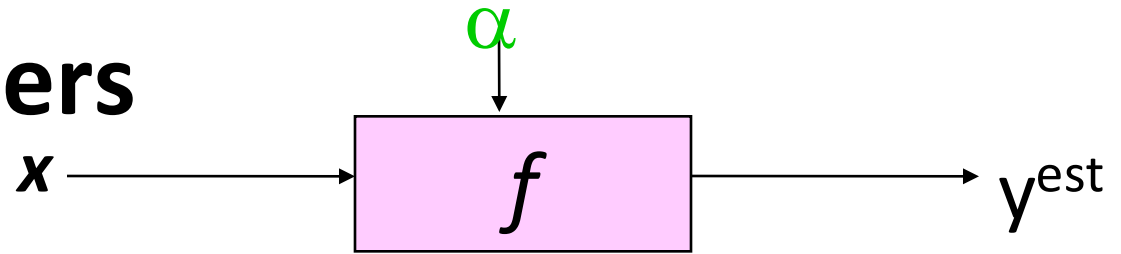
Support Vector Machines

Some slides borrowed from Andrew Moore's [slides on SVMs](#).

Support Vector Machines

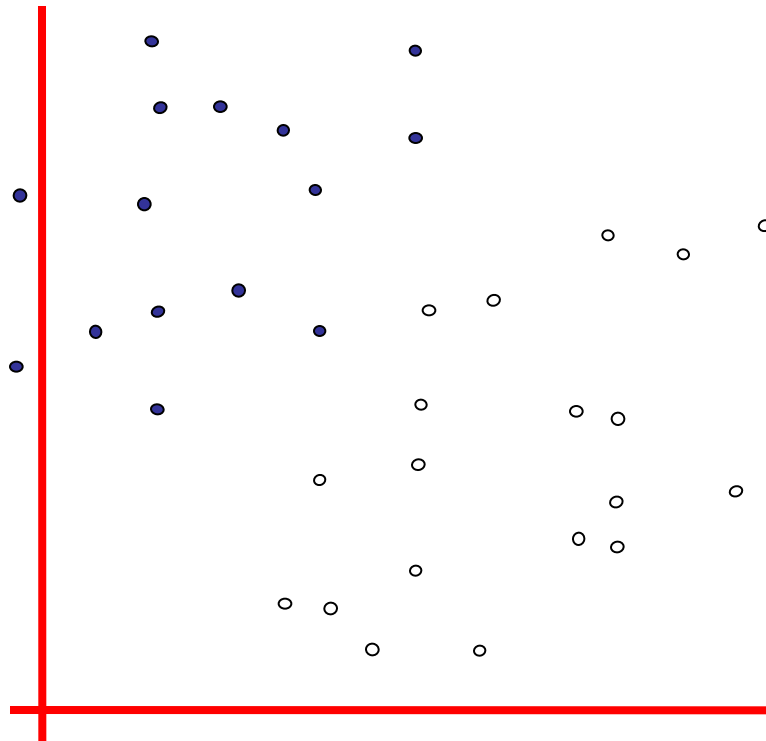
- Popular ML technique for **classification tasks**
 - Became popular in late 90s (Vapnik 1995; 1998) but Invented in the late 70s (Vapnik, 1979)
- Controls complexity and overfitting, so works well on a wide range of practical problems
- Can handle high dimensional vector spaces, which makes feature selection less critical
- Fast and memory efficient implementations, e.g., [svm_light](#)
- Not always best solution, especially for problems with small vector spaces

Linear Classifiers



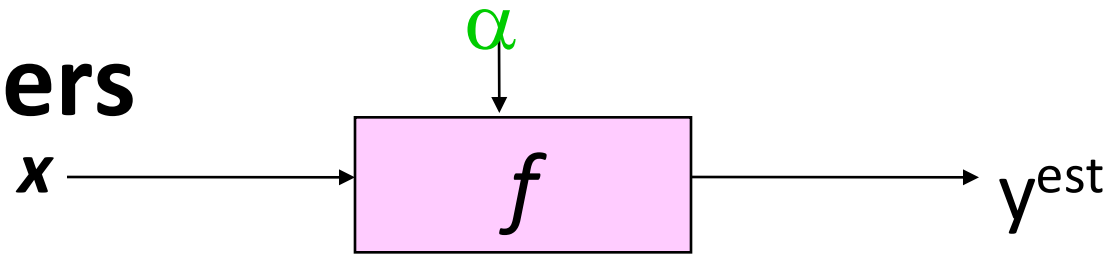
- denotes +1
- denotes -1

$$f(x, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} - b)$$

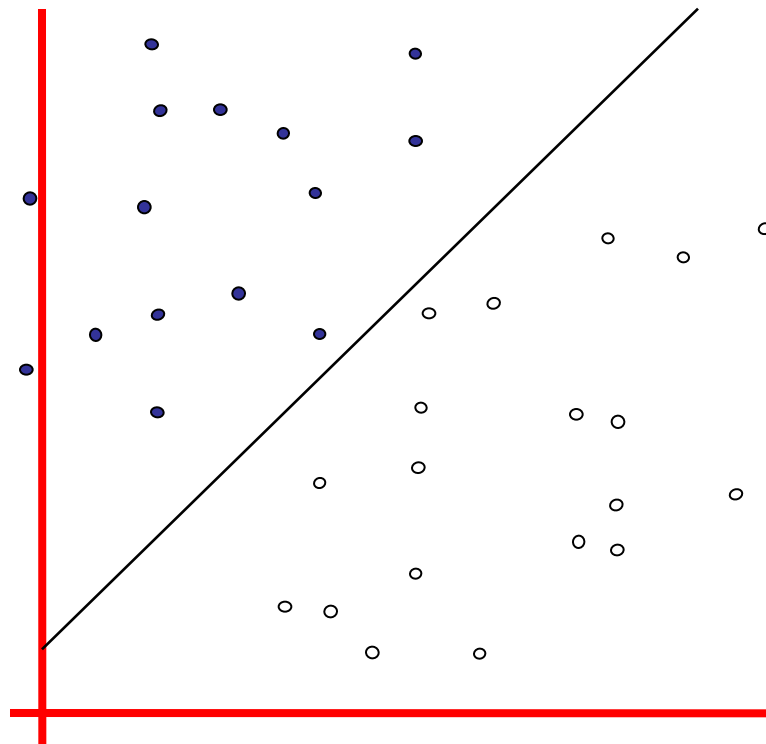


How would you classify this data?

Linear Classifiers



- denotes +1
- denotes -1

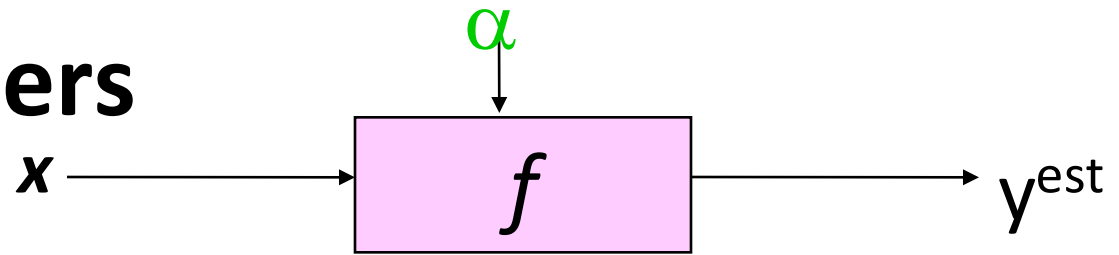


$$f(x, w, b) = \text{sign}(w \cdot x - b)$$

How would you classify this data?

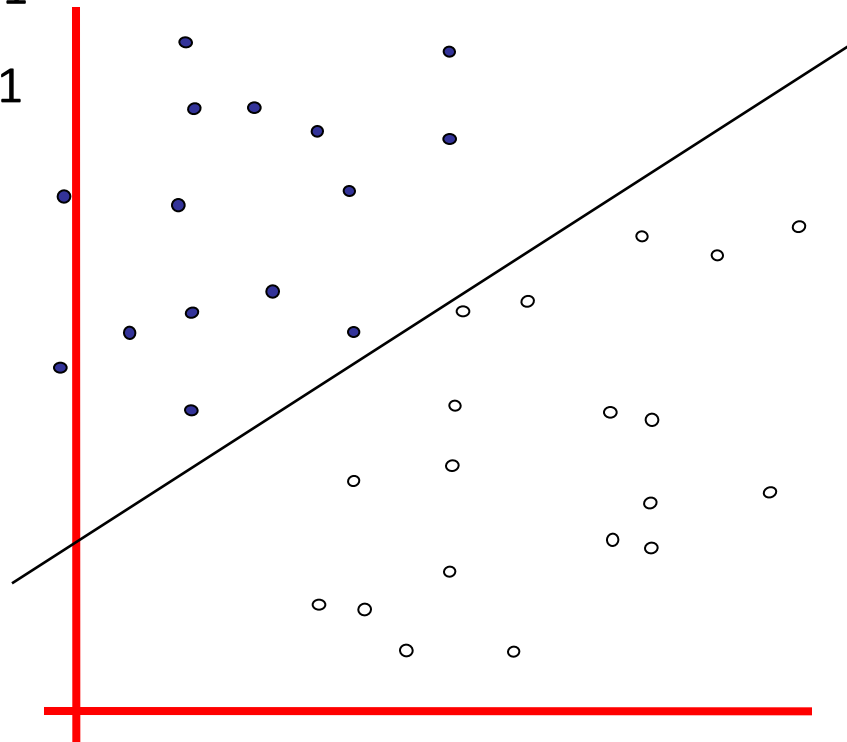
- Draw a line separating the positive and negative examples
- Use that to classify new points

Linear Classifiers



$$f(x, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} - b)$$

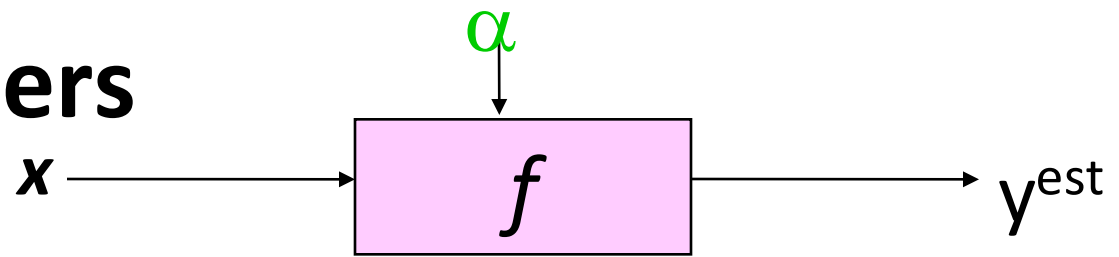
- denotes +1
- denotes -1



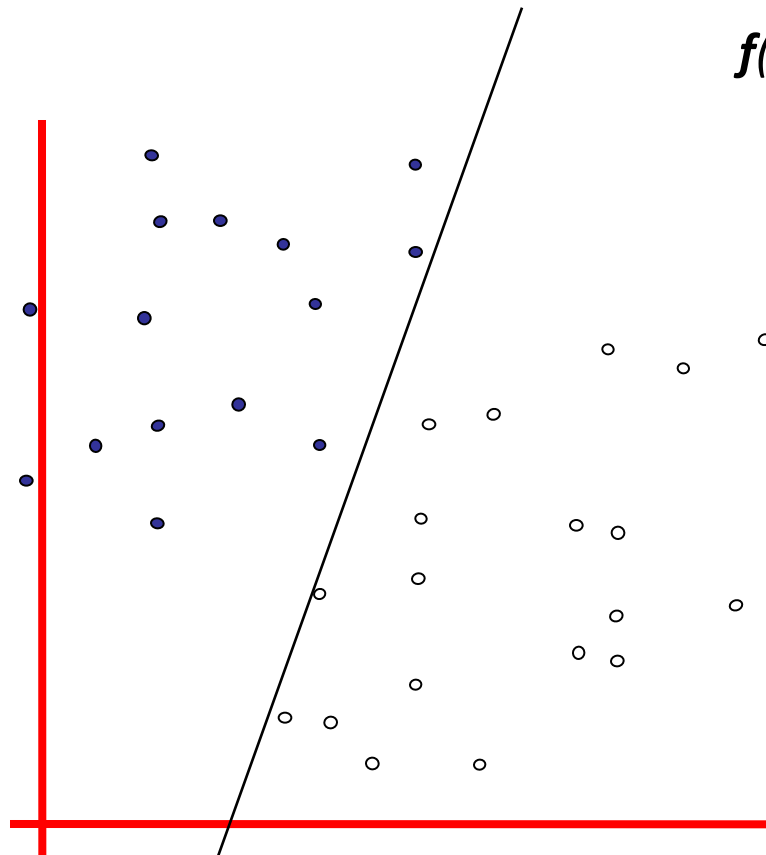
How would you classify this data?

- Here's another line separating positive and negative examples
- Maybe we should use this one?

Linear Classifiers



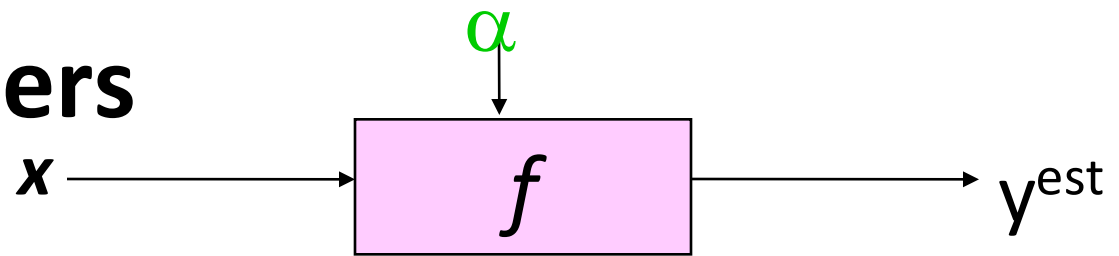
- denotes +1
- denotes -1



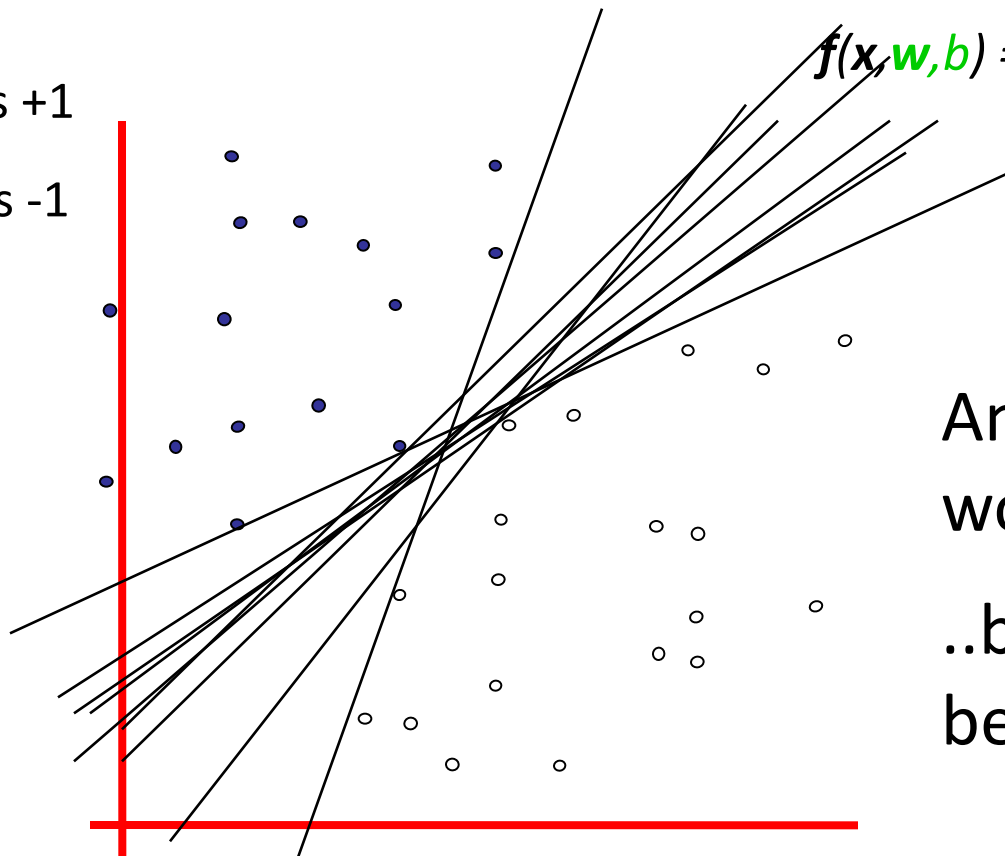
How would you classify this data?

- Yet another line separating positive and negative examples
- Maybe we should use this one?

Linear Classifiers

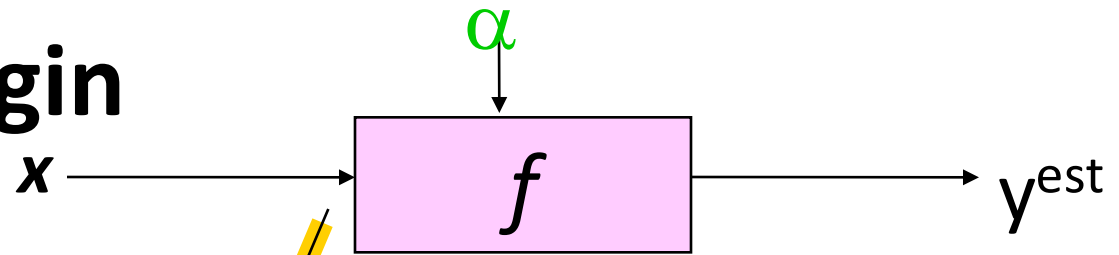


- denotes +1
- denotes -1



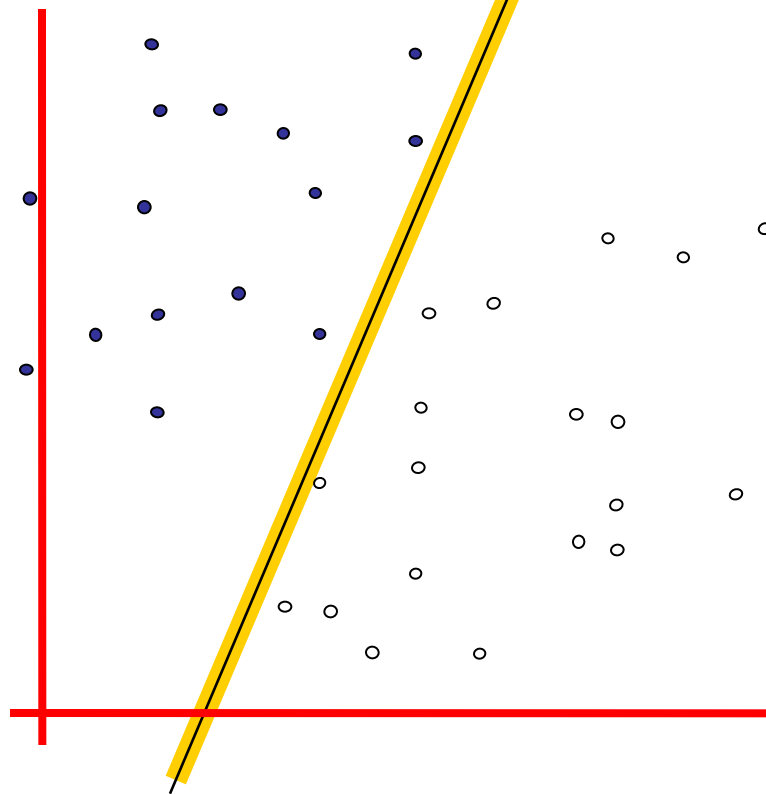
Any of these
would be fine..
..but which is
best?

Classifier Margin



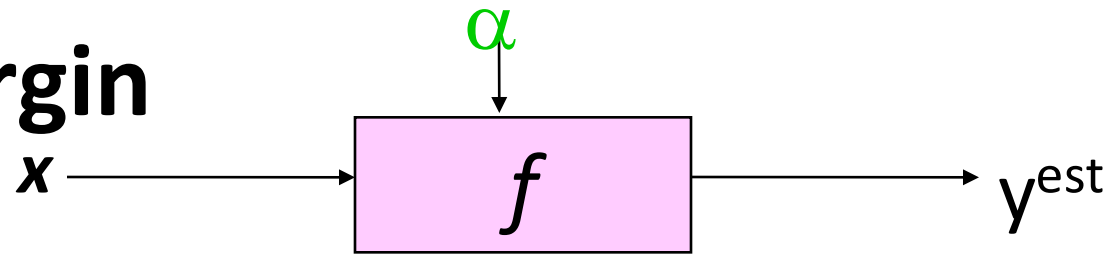
$$f(x, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} - b)$$

- denotes +1
- denotes -1

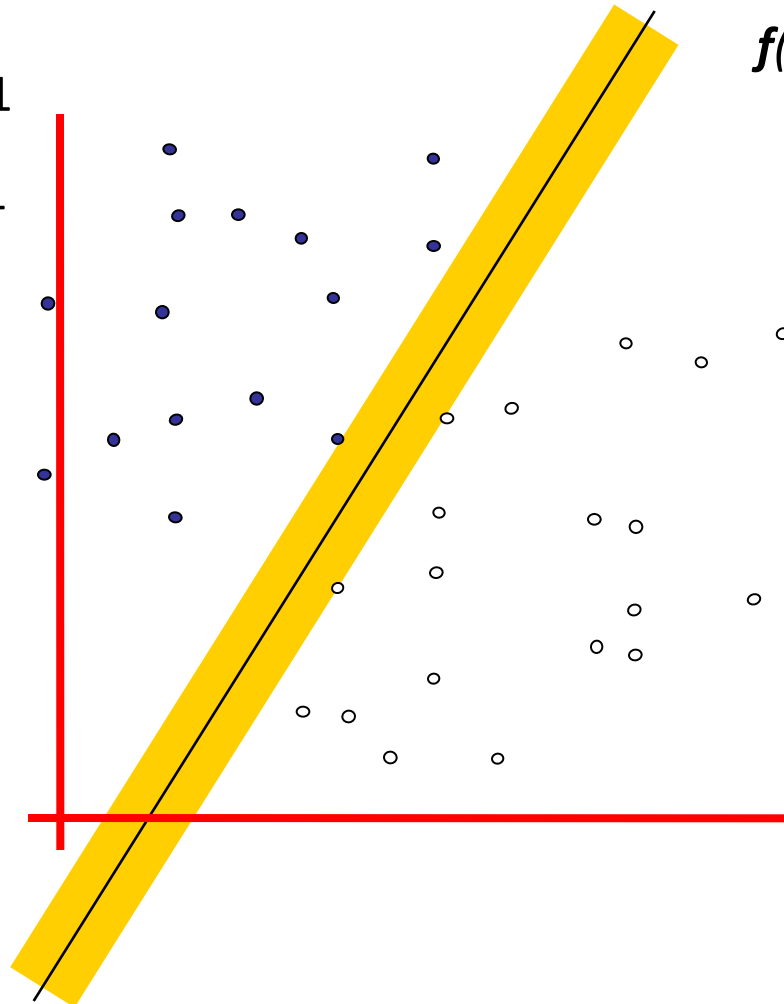


A linear classifier's **margin** is width that boundary could be increased by before hitting a datapoint

Maximum Margin



- denotes +1
- denotes -1



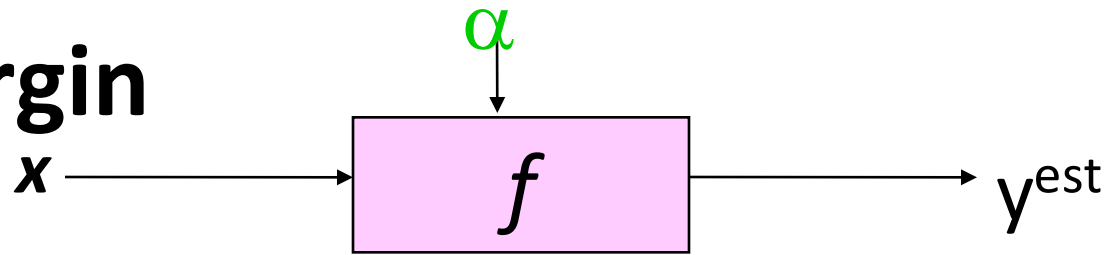
$$f(x, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} - b)$$

Maximum margin linear classifier is the linear classifier with the largest margin

The simplest kind of SVM, called an LSVM

Linear SVM

Maximum Margin



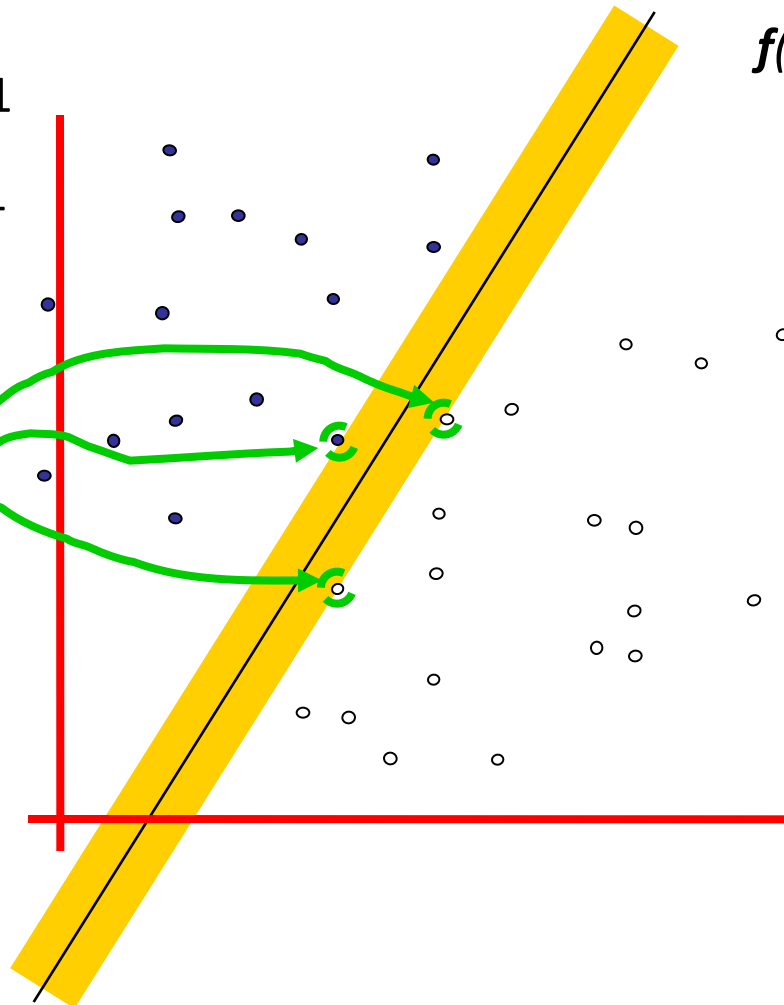
- denotes +1
- denotes -1

$$f(x, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} - b)$$

Maximum margin linear classifier is the linear classifier with the largest margin

The simplest kind of SVM, called an **LSVM**

Linear SVM



Support Vectors

- The data points that the margin pushes up against
- In general, they're vectors, depending on how many dimensions there are

Why Maximum Margin?

- denotes +1
- denotes -1

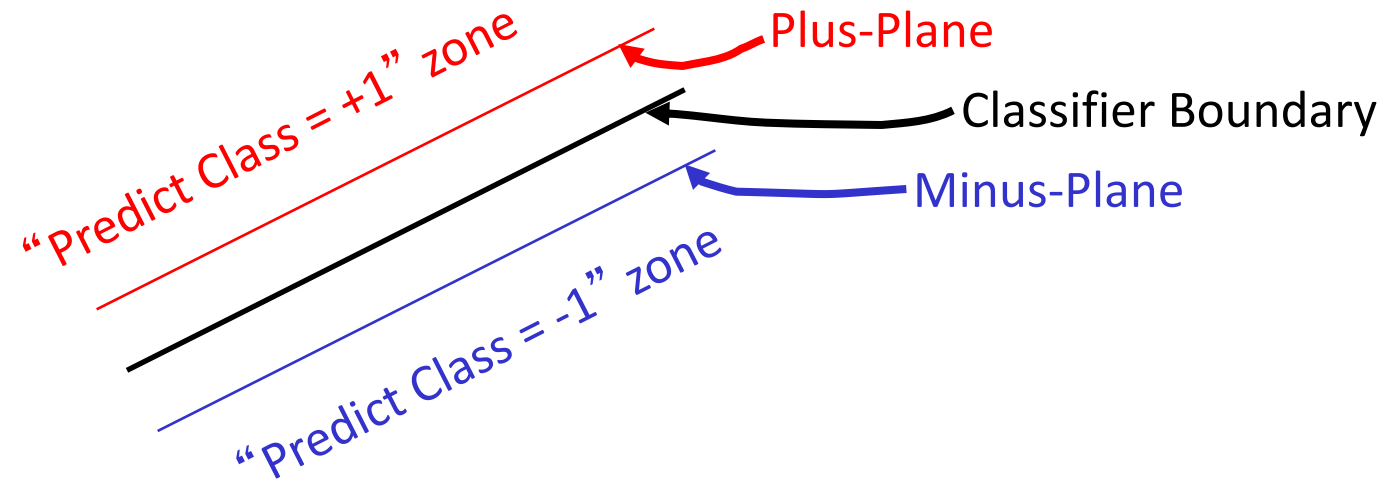
Support Vectors
are those
datapoints that
the margin pushes
up against

1. Intuitively this feels safest
2. Small errors in boundary location unlikely to cause misclassification
3. LOOCV* is easy since model is immune to removal of non-support-vector datapoints
4. Empirically it works very very well margin.

This is the simplest kind of SVM (Called an LSVM)

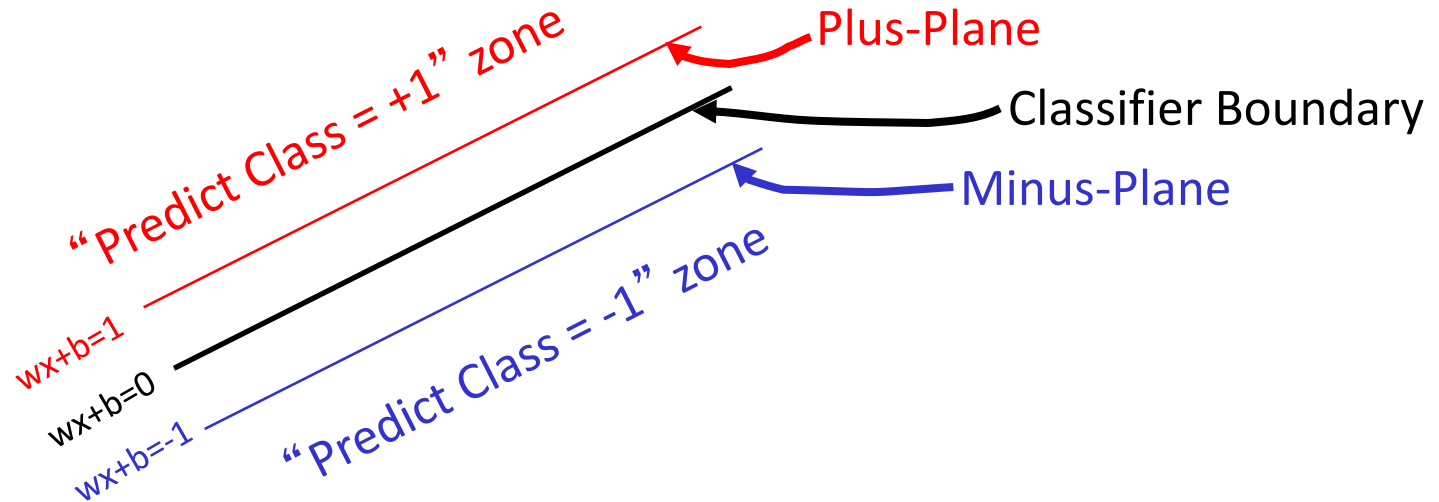
*LOOCV = leave one out cross validation

Specifying a line and margin



- How do we represent this mathematically?
- ...in m input dimensions?

Specifying a line and margin



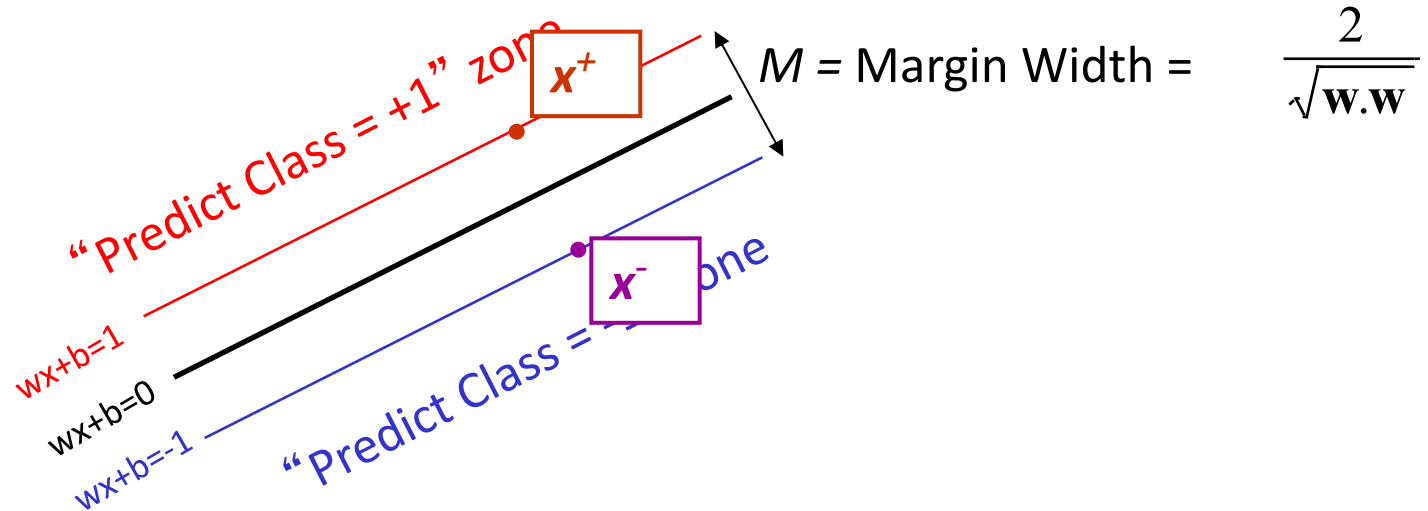
- Plus-plane = $\{ \mathbf{x} : \mathbf{w} \cdot \mathbf{x} + b = +1 \}$
- Minus-plane = $\{ \mathbf{x} : \mathbf{w} \cdot \mathbf{x} + b = -1 \}$

Classify as.. **+1** if **$\mathbf{w} \cdot \mathbf{x} + b \geq 1$**

-1 if **$\mathbf{w} \cdot \mathbf{x} + b \leq -1$**

Universe if **$-1 < \mathbf{w} \cdot \mathbf{x} + b < 1$**
explodes

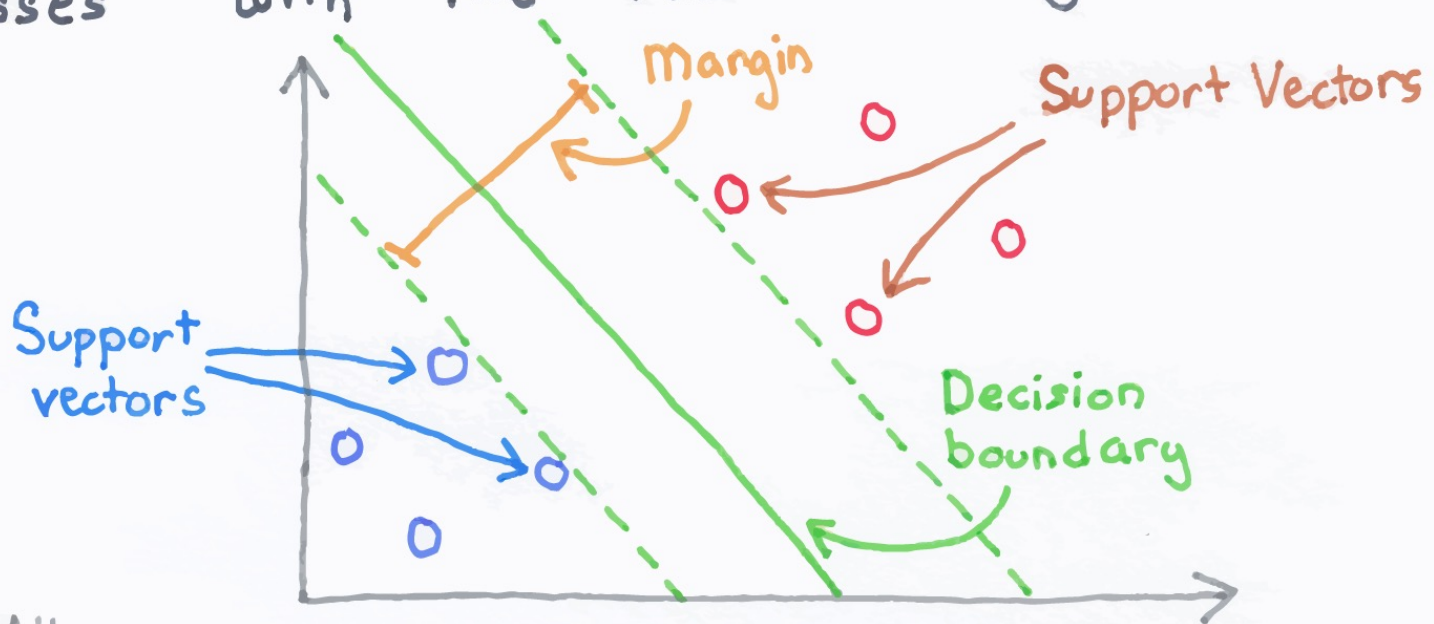
Learning the Maximum Margin Classifier



- Given a guess of \mathbf{w} and b we can
 - Compute whether all data points in the correct half-planes
 - Compute the width of the margin
- Write a program to search the space of \mathbf{w} s and b s to find widest margin matching all the datapoints.
- *How?* -- Gradient descent? Simulated Annealing? Matrix Inversion? EM? Newton's Method?

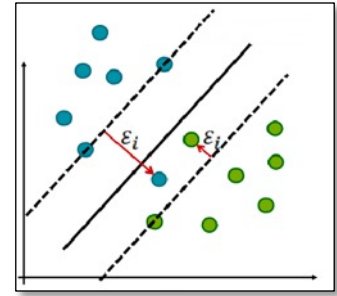
SVM

Finds the linear hyperplane that separates classes with the Maximum Margin.



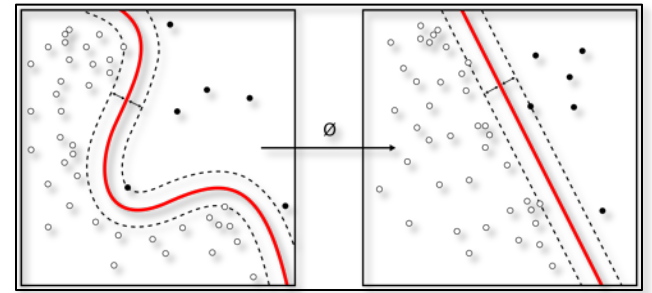
Chris Albon

Soft margin classification



- What if data from two classes not linearly separable?
- Allow a fat decision margin to make a few mistakes
- Some points, **outliers** or noisy examples, are inside or on wrong side of the margin
- Each outlier incurs a cost based on distance to hyperplane

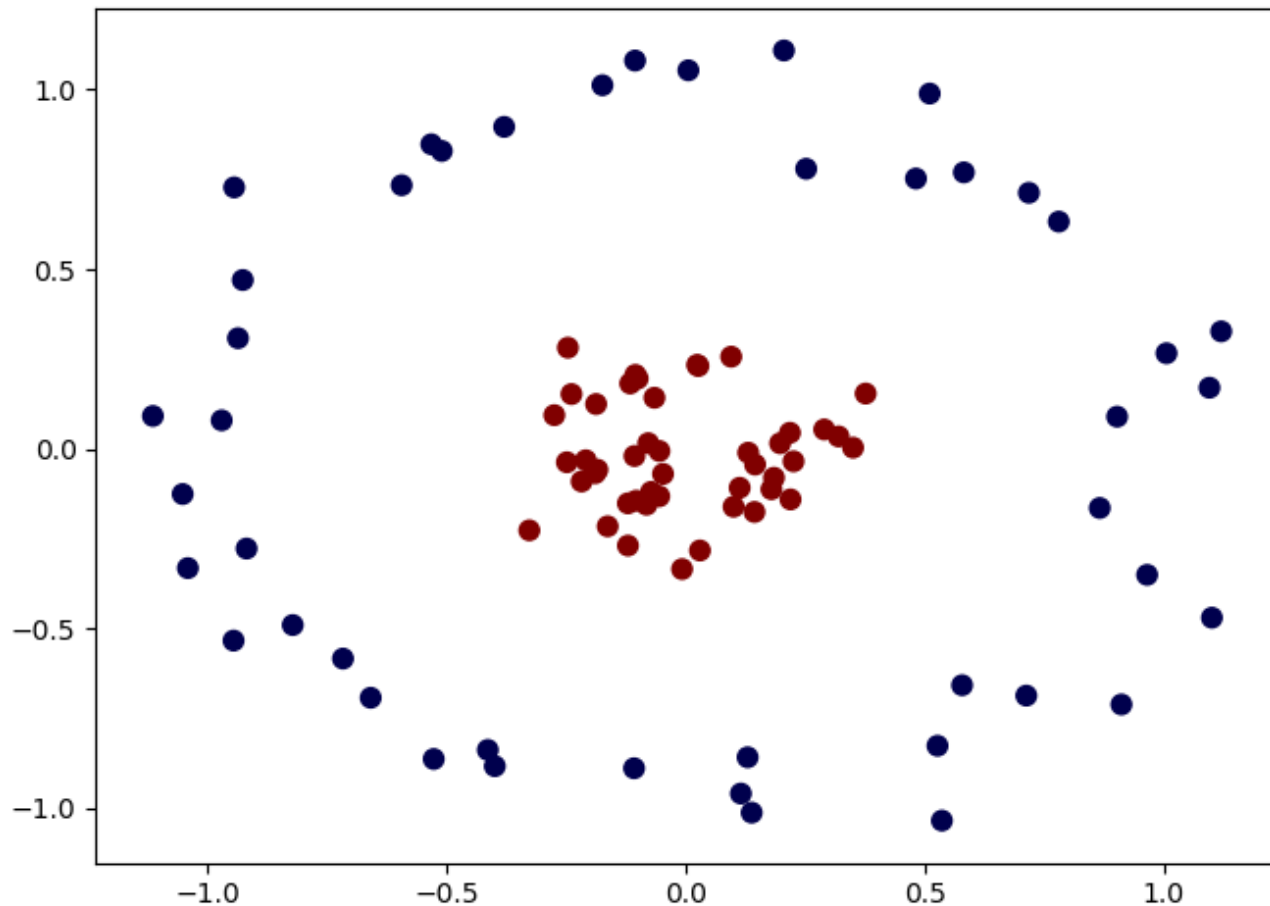
Kernel trick



- What if data from two classes not linearly separable?
- Project data onto a higher dimensional space where it becomes linearly separable
- Many SVMs can take an argument, a **kernel**, that does the transformation of the data
- Deciding what **kernel function** to use is done through experimentation

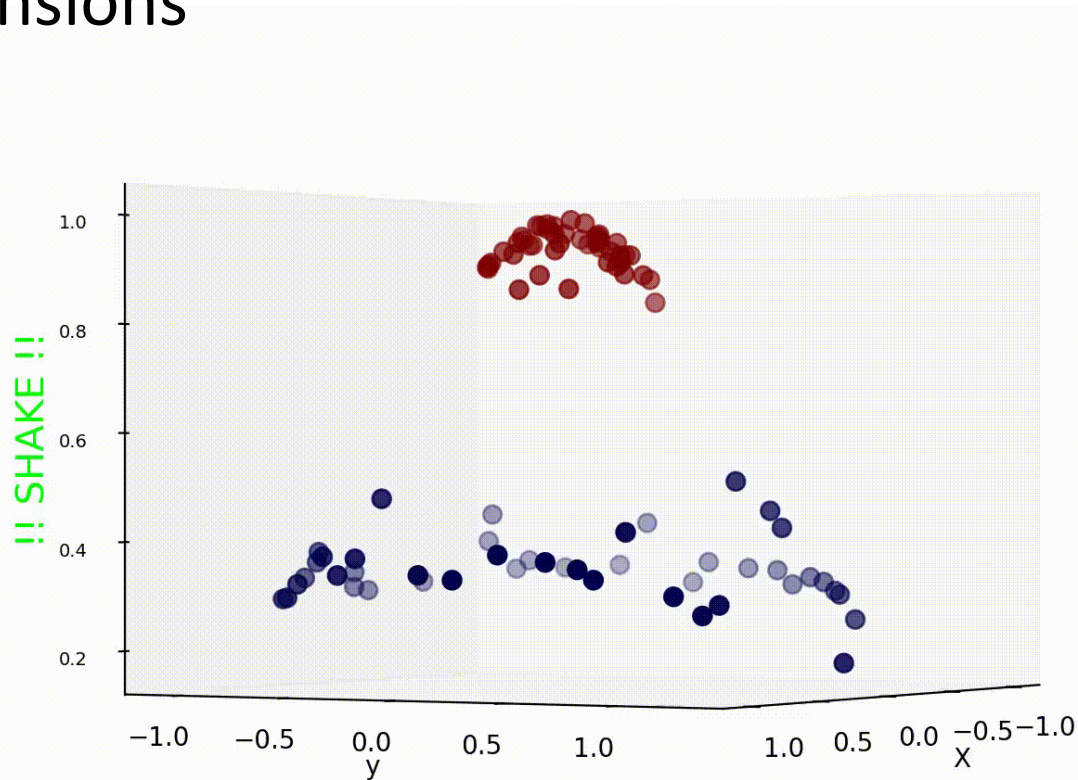
Kernel Trick example

Can't separate the blue & red points with a line



Use a different kernel

- Applying a kernel can transform data to make it more nearly linearly separable
- E.g., use polar coordinates or map to three dimensions



SVM Performance

- SVMs can handle very large features spaces (e.g., 100K features!)
- Relatively fast
- Anecdotally they work very well indeed
- Example: They are among the best-known classifier on a well-studied hand-written-character recognition benchmark

100k features?

- SVM used for simple binary text classification, e.g., classify a text document as
 - expressing sentiment (+, -) or political leaning (rep, demo)
- Typically use each word in fixed vocabulary of 25k-100k words as a feature/dimension
 - Map words \leftrightarrow integers for efficiency and numpy/scipy use
 - value often [TF-IDF](#) number: frequency of word in document relative to all documents in large corpus
- Product reviews might have ~200 unique words, social media posts only < 20 unique words
- SVMs typically use sparse matrices for efficiency

Binary vs. multi classification

- SVMs only do **binary** classification 😞
 - E.g.: can't classify an iris into one of three species
- A common constraint for many ML classifiers
- Two approaches to multiclass classification: OVA and OVO
- Consider Zoo dataset, which classifies animals into one of 7 classes based on 17 attributes
 - **Classes:** mammal, bird, reptile, fish, amphibian, insect, invertebrate
 - **Attributes:** hair, feathers, eggs, milk, aquatic, toothed, fins, ...

OVA or one-vs-all classification

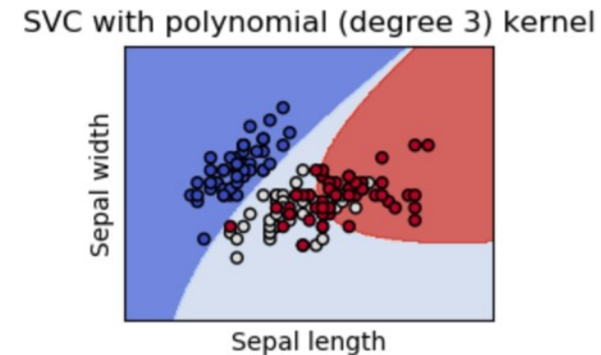
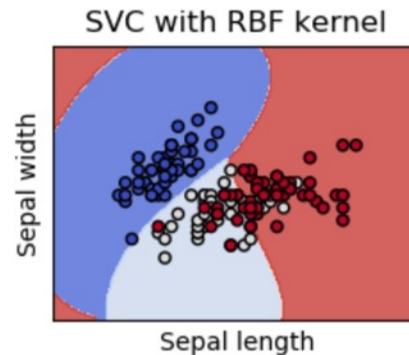
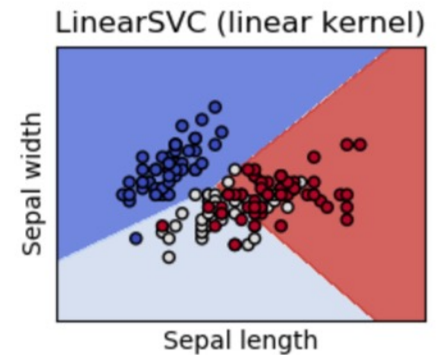
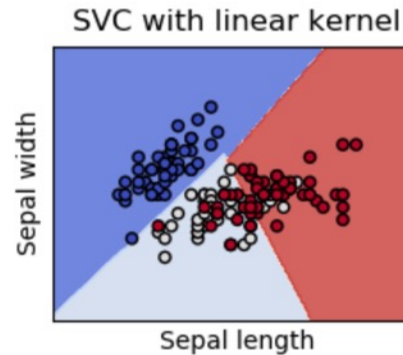
- **OVA** or one-vs-all: turn n-way classification into n binary classification tasks
 - Also know as one-vs-rest
- For zoo problem with 7 categories, train and run 7 binary classifiers:
 - mammal vs. not-mammal
 - fish vs. not-fish
 - bird vs. not-bird, ...
- Pick the one that gives the highest score
 - For an SVM this could be measured the one with the **widest margin**

OVO or one vs one classification

- OVO or one vs one: turn n-way classes into $N*(N-1)/2$ one-vs-one classifiers
- For the zoo dataset we have 21 classifiers
 - mammal vs. bird, mammal vs. reptile ...
 - bird vs. reptile, bird vs. fish, ...
 - fish vs. amphibian, fish vs. insect, ...
 - ...
- Use resulting scores to choose the classification that wins the most 1x1 pairings

SVMs in scikit-learn

- Scikit-learn has three [SVM classifiers](#): SVC, NuSVC, and LinearSVC
- Data can be either in **dense numpy** arrays or **sparse scipy** arrays
- All directly support multi-way classification, SVC and NuSVC using OvO and LinearSVC using OvA



This [colab jupyter notebook](#) gets an accuracy of 97% using OvO with [scikit.svm.SVC](#)

SVM Summary

- SVM is a good classification technique for problems with a large feature space
- Relatively fast to train and apply the model
- The kernel trick can help make some problems more-nearly linearly separable
- Their binary nature makes them a poorer fit for multi-way classification