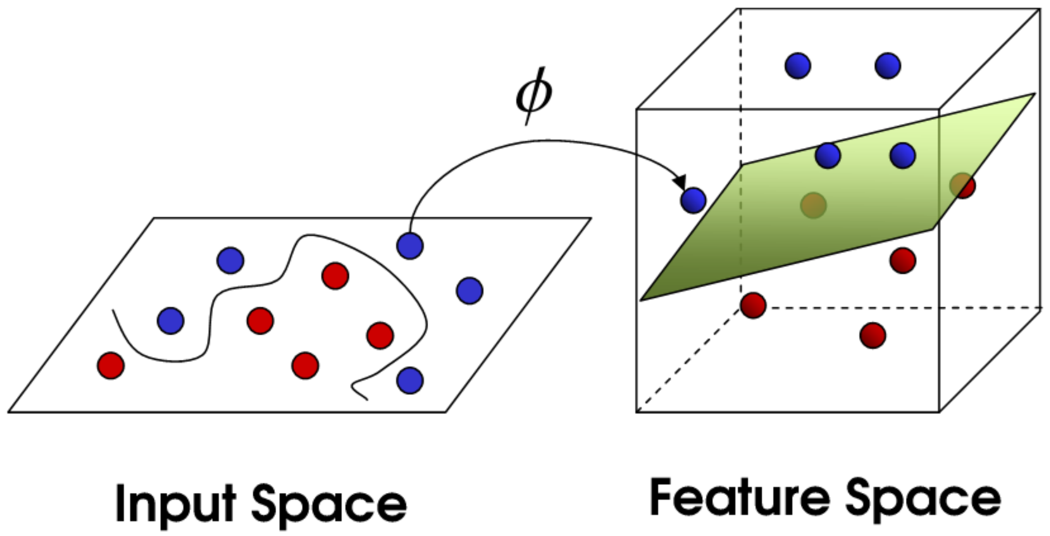
# CS109 – Data Science
# SVM, Performance evaluation

Joe Blitzstein, Hanspeter Pfister, Verena Kaynig-Fittkau



http://i.stack.imgur.com/1gvce.png

## Announcements

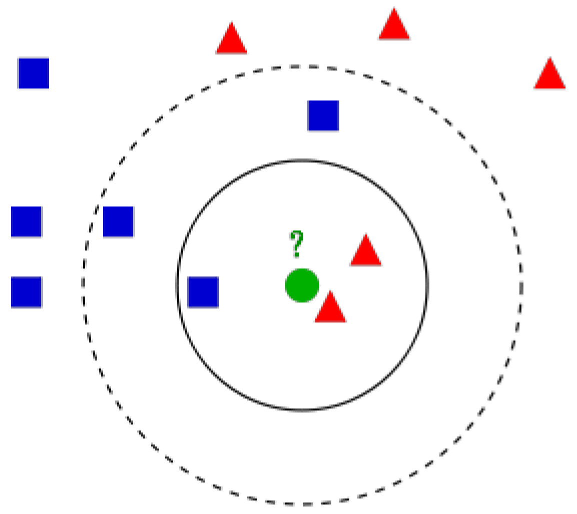
- HW1 grades went out yesterday
- They are looking really good, well done everyone!

- HW2 is due this Thursday!

- You should submit an executed notebook
- But please without pages of test output

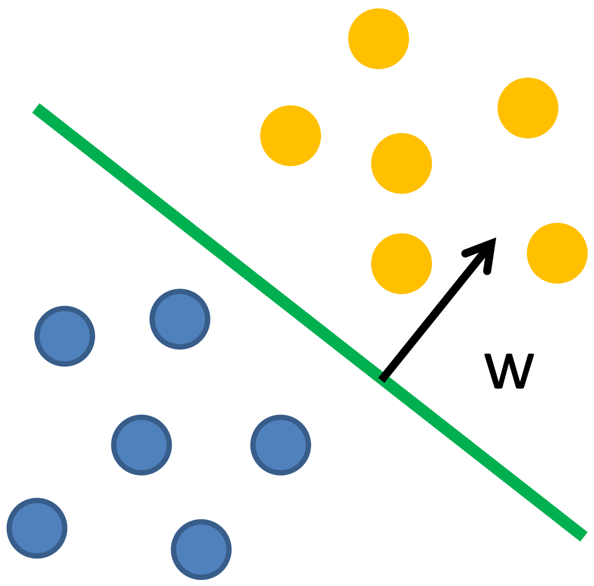
# Recap K-NN

- Keeps all training data
- Training is fast
- Prediction is slow



# Separating Hyperplane

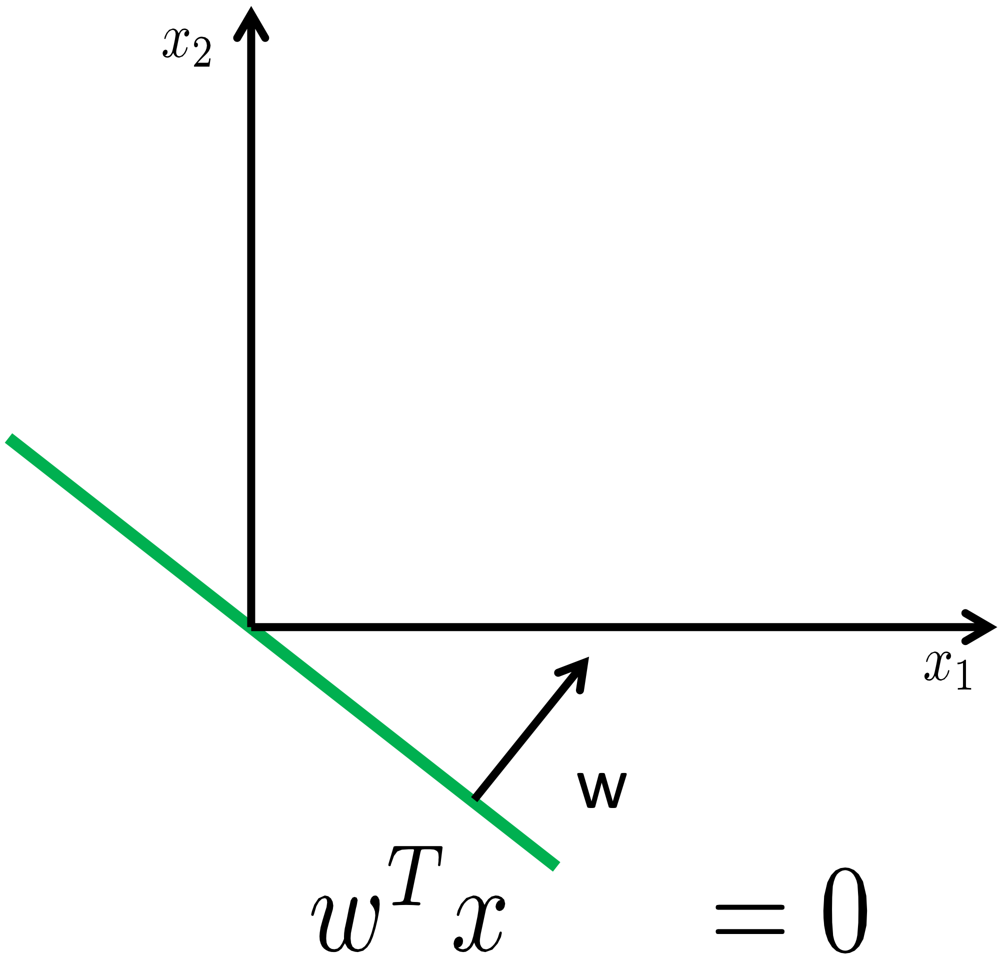
- x: data point
- y: label $\in \{-1, +1\}$
- w: weight vector



$$w^T x \qquad = 0$$

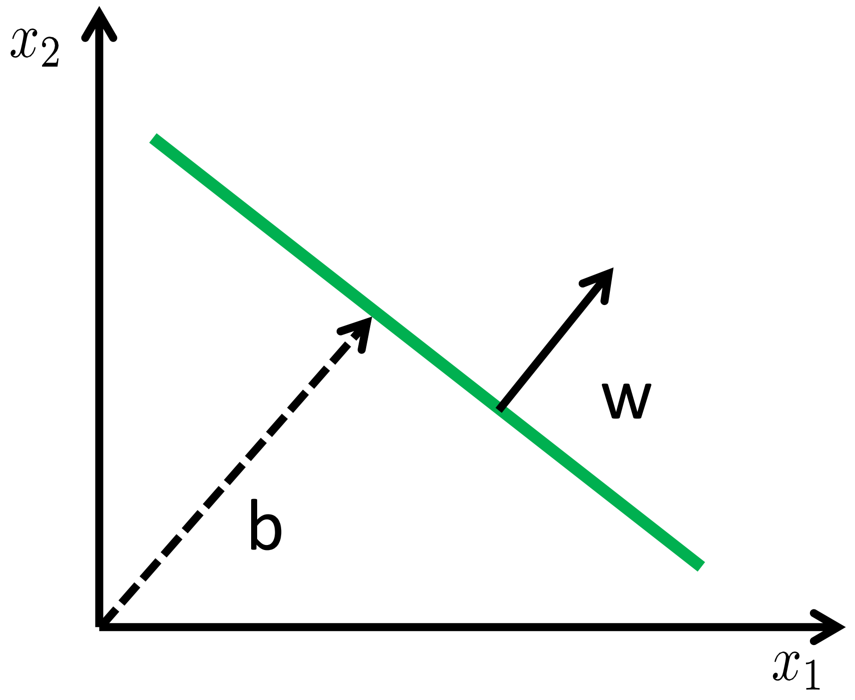
# Separating Hyperplane

- x: data point
- y: label $\in \{-1, +1\}$
- w: weight vector



# Separating Hyperplane

- x: data point
- y: label $\in \{-1, +1\}$
- w: weight vector
- b: bias



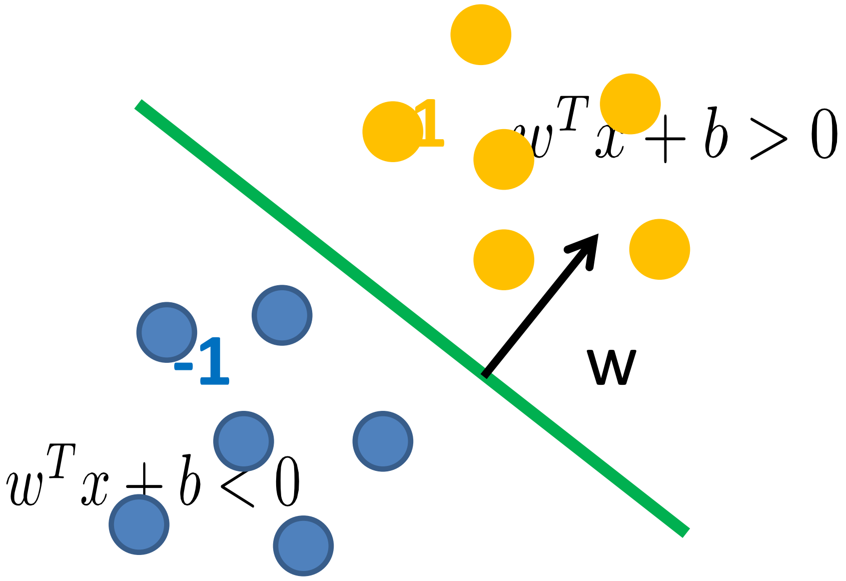
Need 2 things to specify hyperplane: w and b

$$w^T x + b = 0$$

# Separating Hyperplane

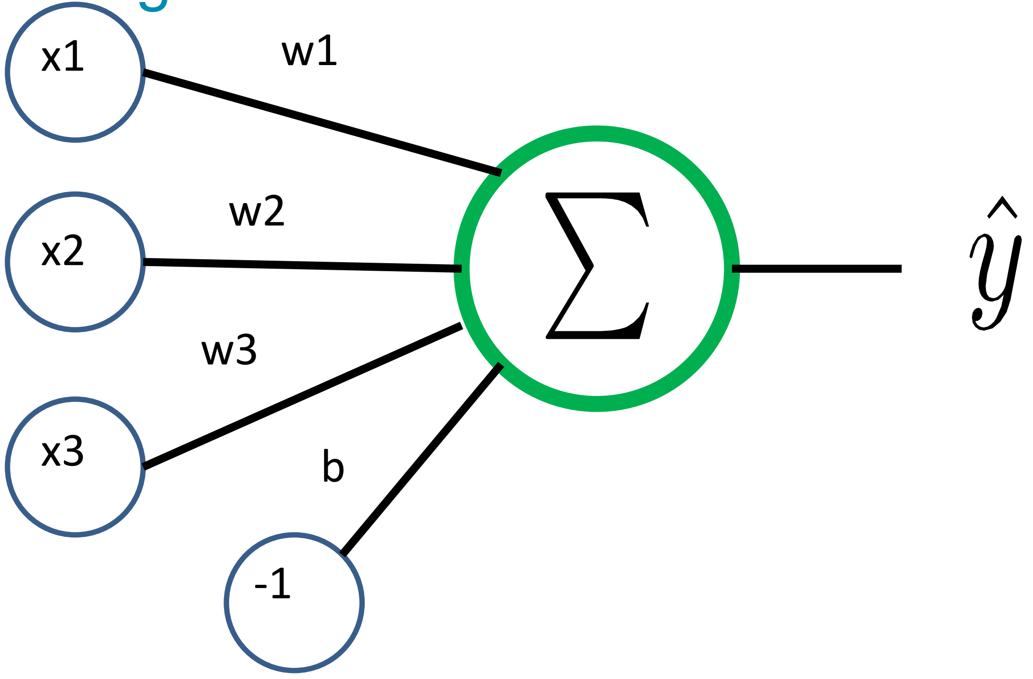
- x: data point
- y: label $\in \{-1, +1\}$
- w: weight vector
- b: bias

+ storage very small
+ prediction very fast
but restricted: only a line



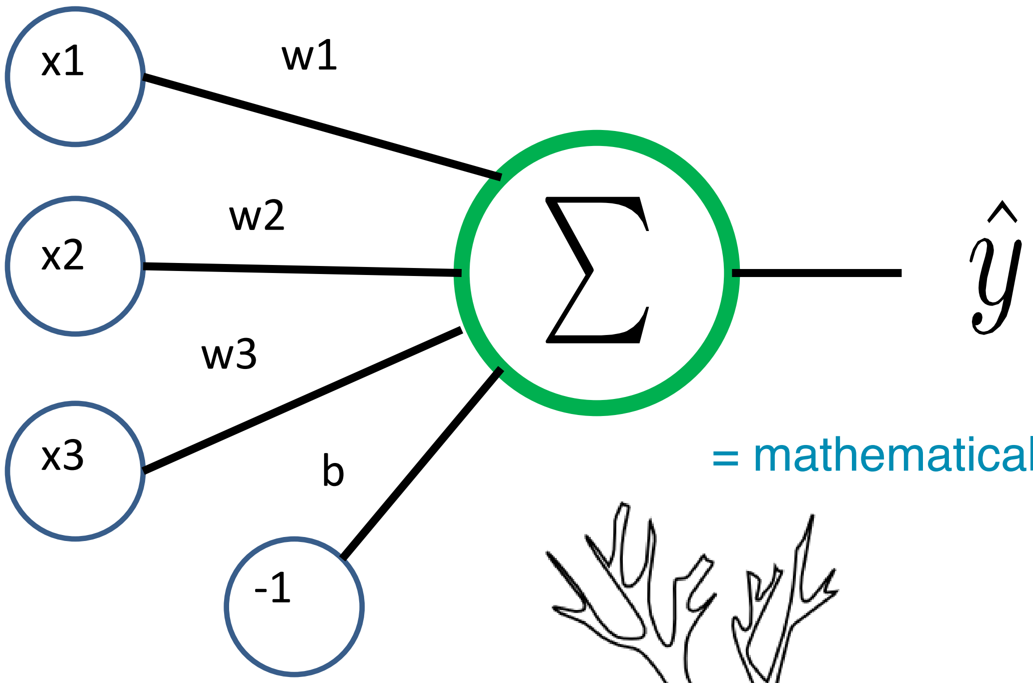
# Perceptron

x described by 3 features
+ augment -1

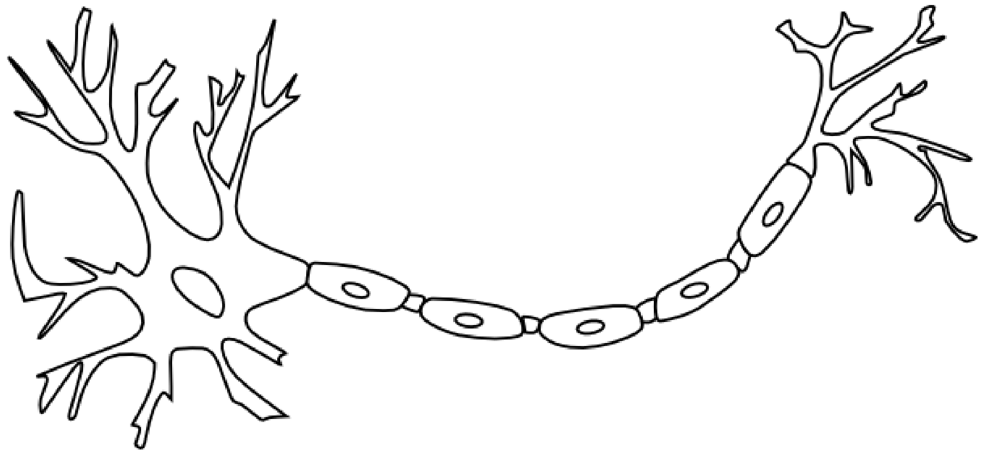


$$w^T x + b = 0$$

# Perceptron



= mathematical model of a neuron



# Perceptron History

- invented 1957
- by Frank Rosenblatt

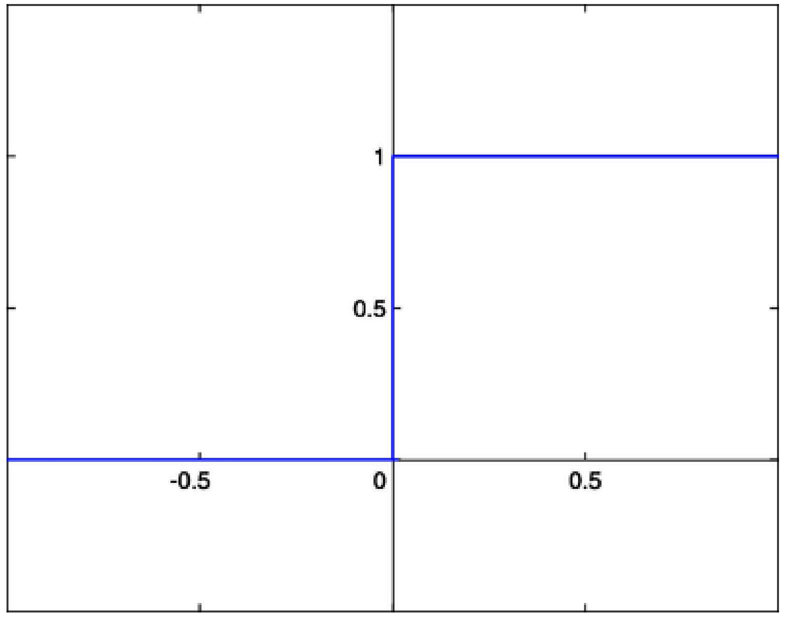
- the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence. (NYT 1958)

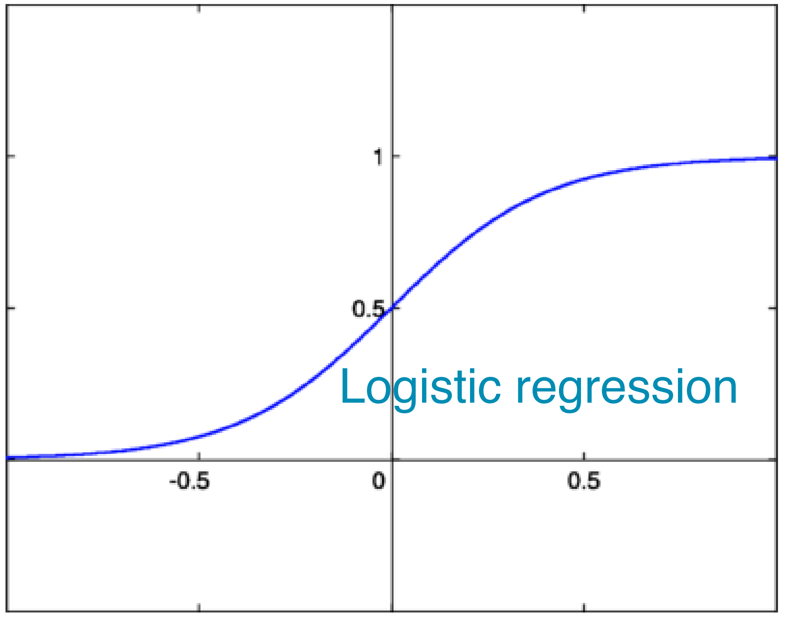
(http://en.wikipedia.org/wiki/Perceptron

Perceptron.mp4

https://www.youtube.com/watch?v=cNxadbrN_aI&list=PLdVOMWcqwwIlaygvb9ZteZ1r4Br6kRuBO

# Side Note: Step vs Sigmoid Activation





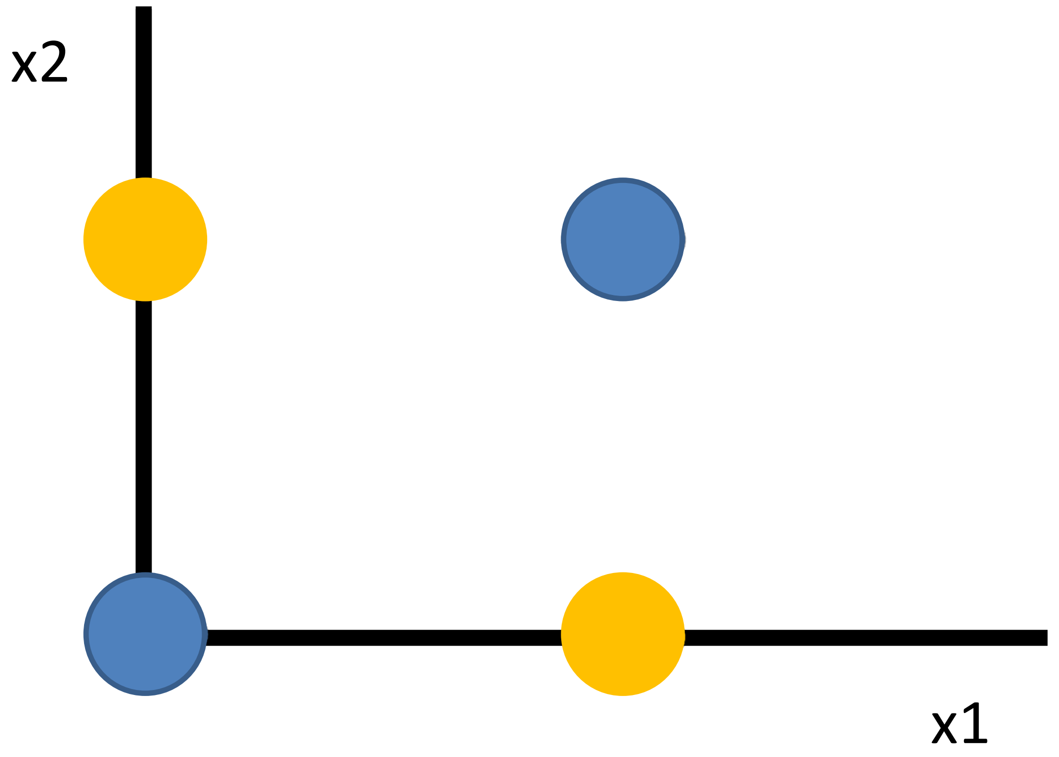
$$s(x) = \frac{1}{1 + e^{-cx}}$$

## The Critics

- 1969: Minsky and Papert publish their book "Perceptrons"
- Very controversial book, some blame the book for causing the whole research area to stagnate.

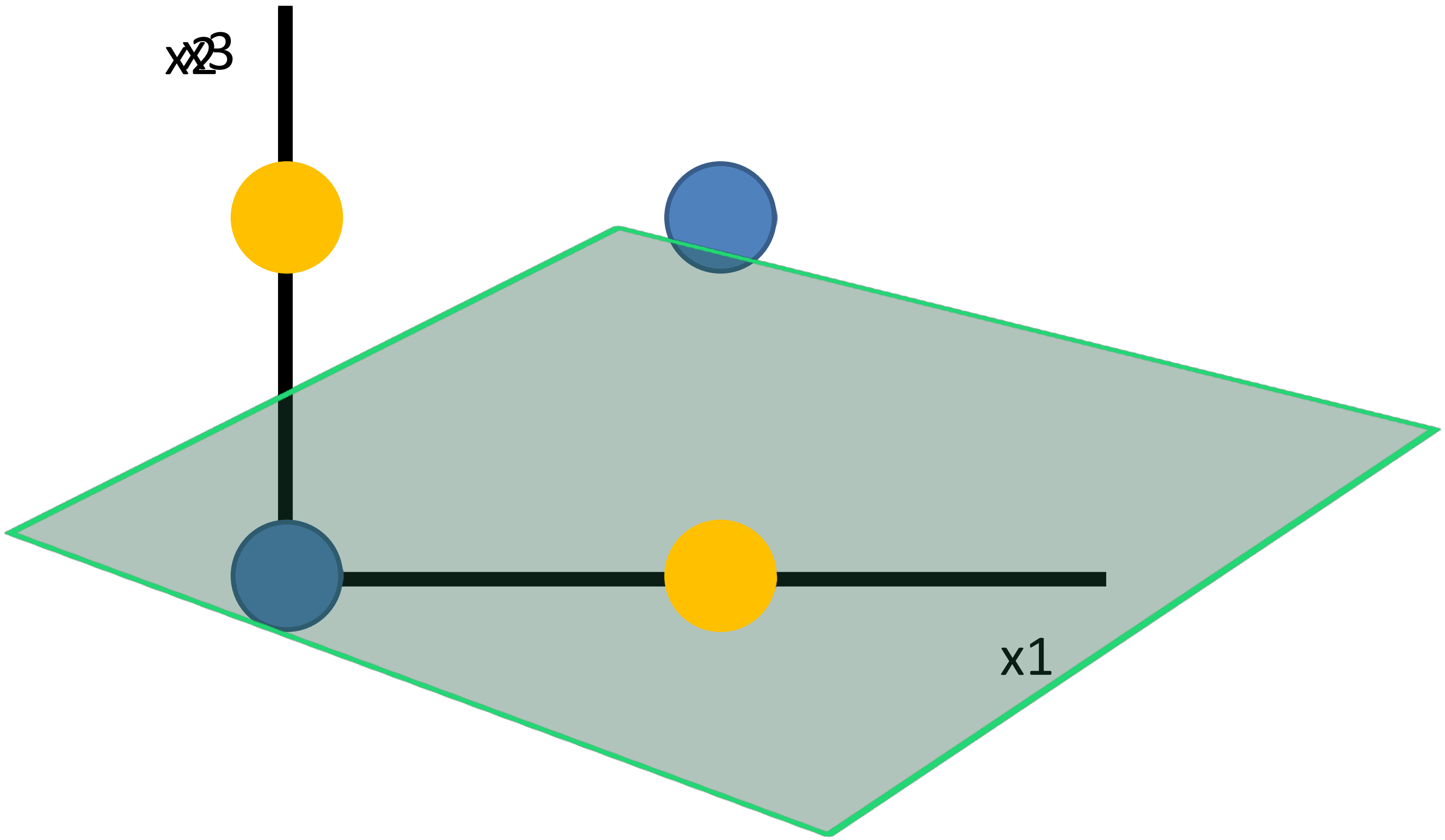
https://en.wikipedia.org/wiki/Perceptrons_(book)

# The XOR Problem



= limitation of simple separating hyperplane

# The XOR Problem



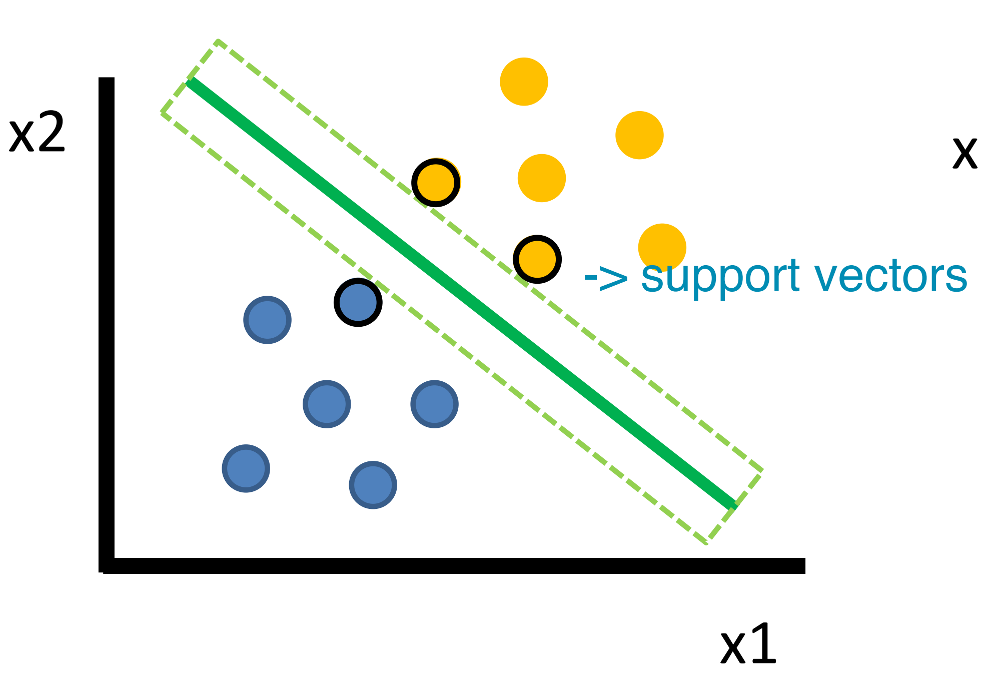
## Support Vector Machine

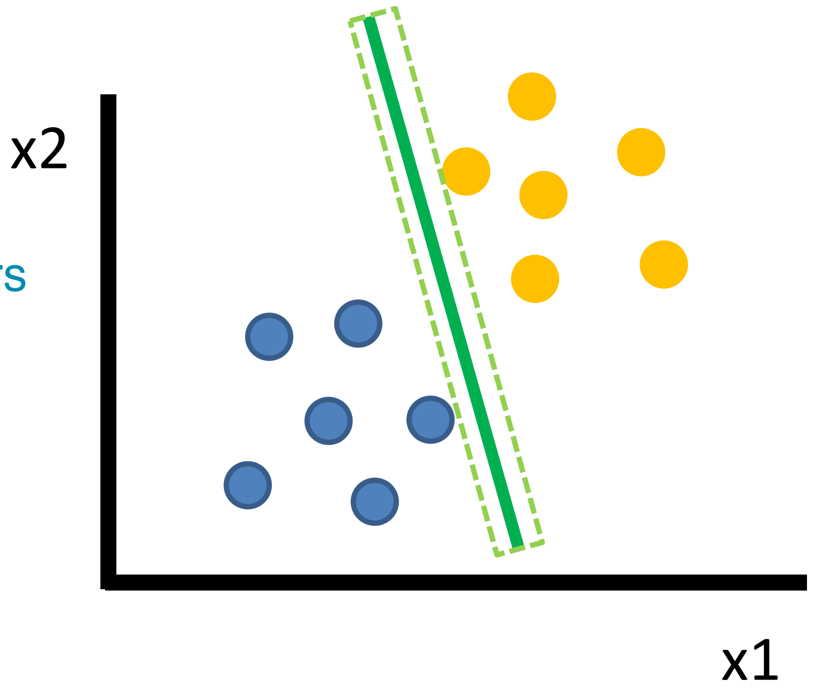
- Widely used for all sorts of classification problems

- Some people say it is the best of the shelf classifier out there

# Maximum Margin Classification

equally correct for the perceptron

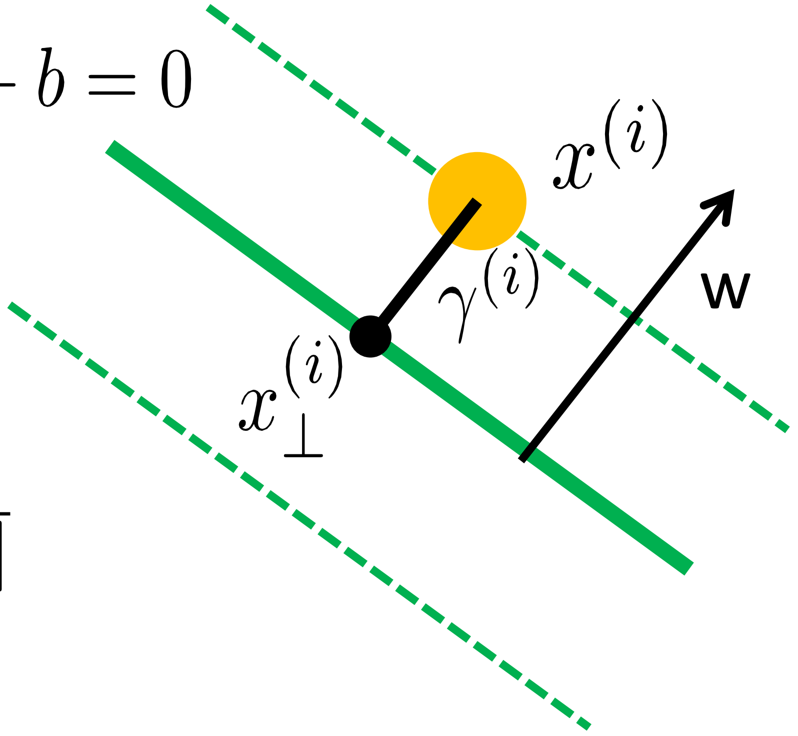




Solution depends only on the support vectors!

but we still need sufficient training vectors because the support vectors are on the boundary -> they are the rare points

# Maximum Margin Classification



$$w^T x + b = 0$$

margin:

$$x_\perp^{(i)} = x^{(i)} - \gamma^{(i)} \cdot \frac{w}{||w||}$$

$$w^T x_\perp^{(i)} + b = 0$$

$$\Rightarrow \quad \gamma^{(i)} = \quad \left(\frac{w^T x^{(i)} + b}{||w||}\right)$$

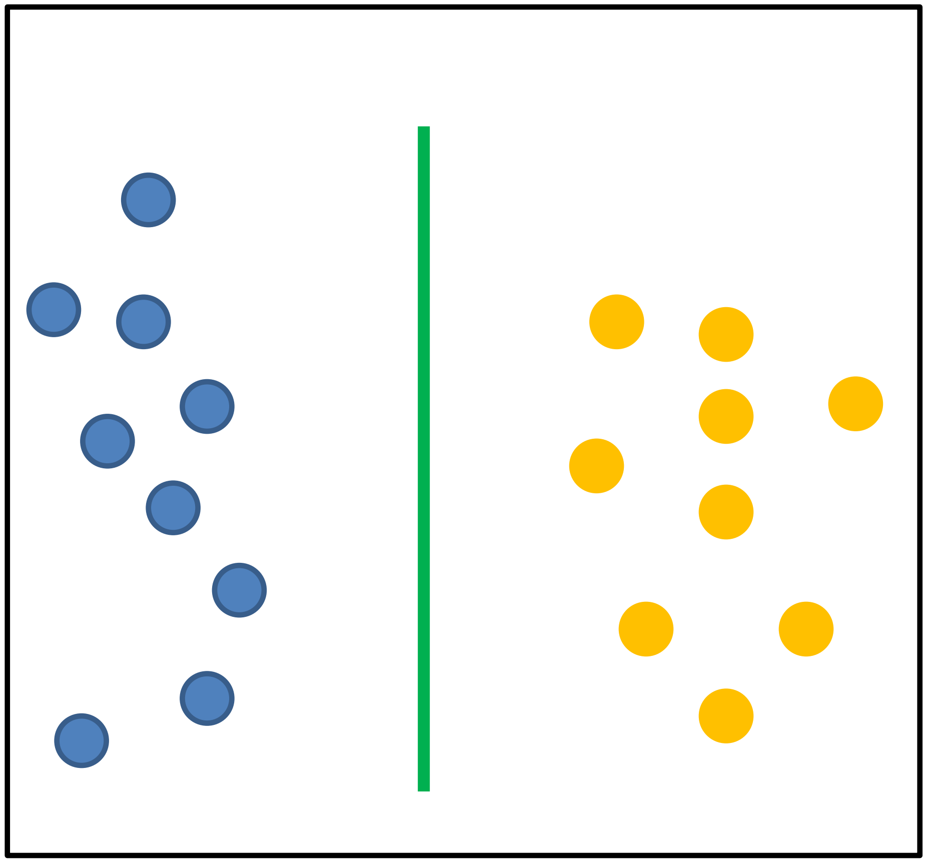
# Maximum Margin Classification

$$\gamma^{(i)} = y^{(i)}(w^T x + b)$$

$$\begin{aligned} \max_{\gamma,w,b} \quad & \gamma \\ \text{s.t.} \quad & y^{(i)}(w^T x^{(i)} + b) \geq \gamma, \quad i = 1, \ldots, m \\ & ||w|| = 1. \end{aligned}$$

non-convex

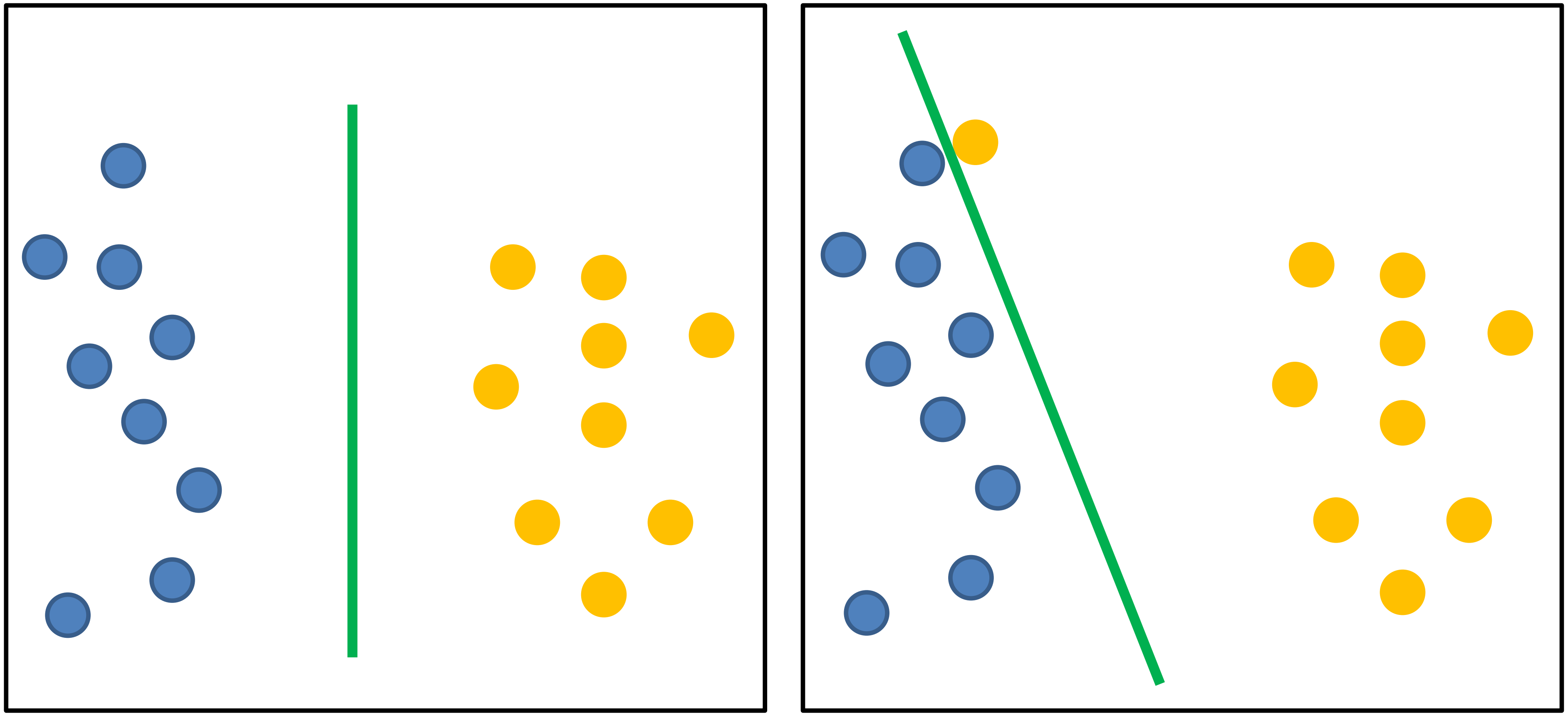
# This Is Kind of Odd



- Which data points do we care the most about? rare points
- What would those samples look like?

The SVM doesn't care about the ideal sample

# Two Very Similar Problems



# What about outliers?

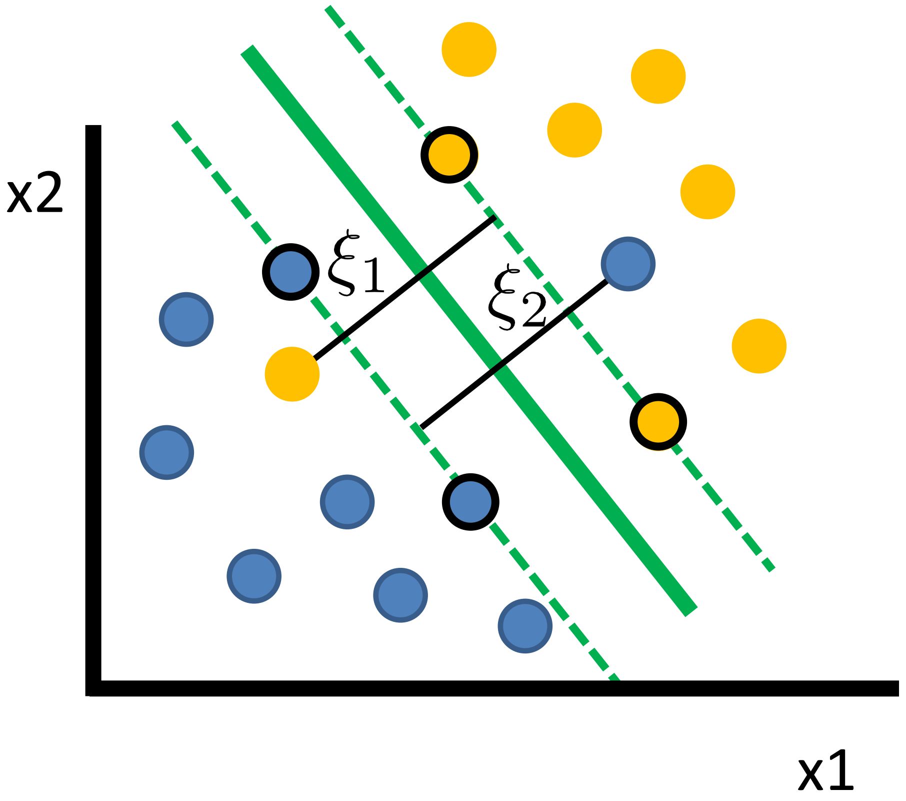
$\xi_i$: slack variables

$$\min_{w,b,\xi} \frac{1}{2}||w||^2$$

subject to:

$$y^{(i)}(w^T x^{(i)} + b) \geq 1$$

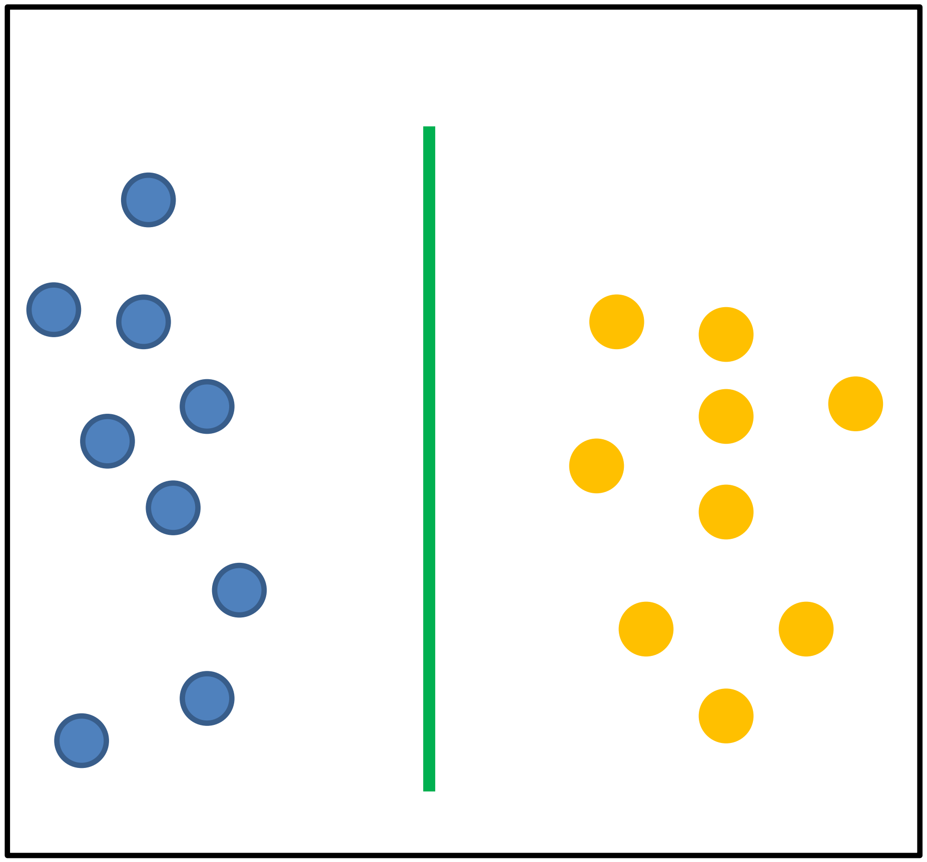
$$(i = 1, \ldots, n)$$

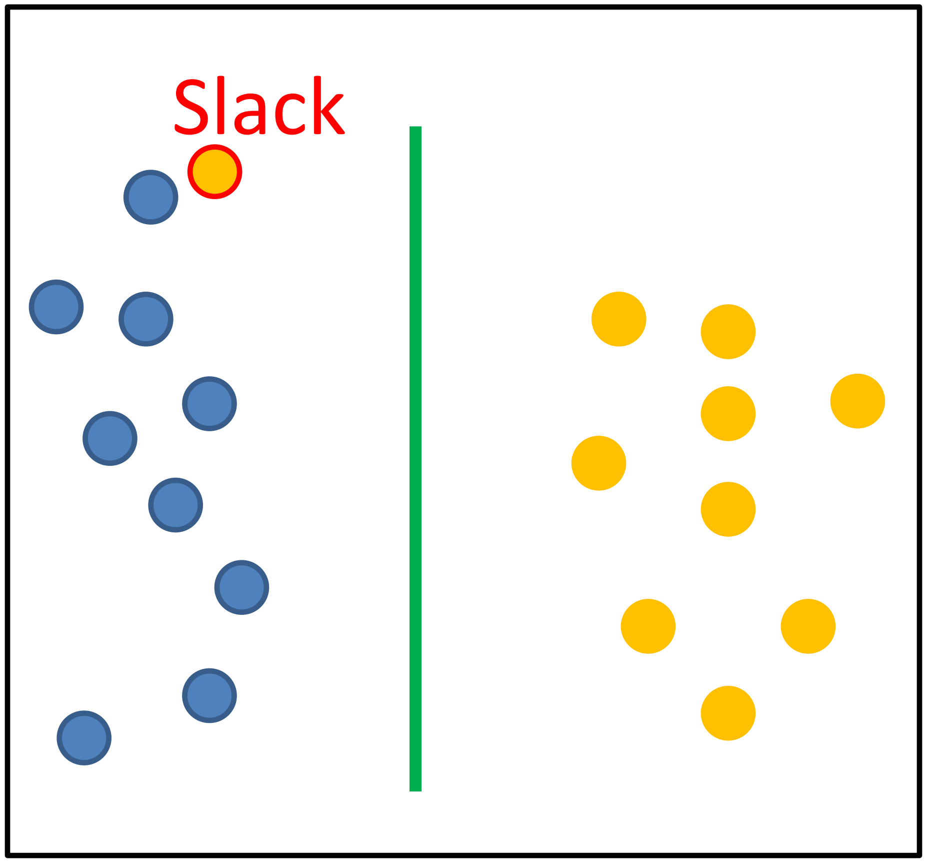


If C large: no slack allowed -> equiv to previous method

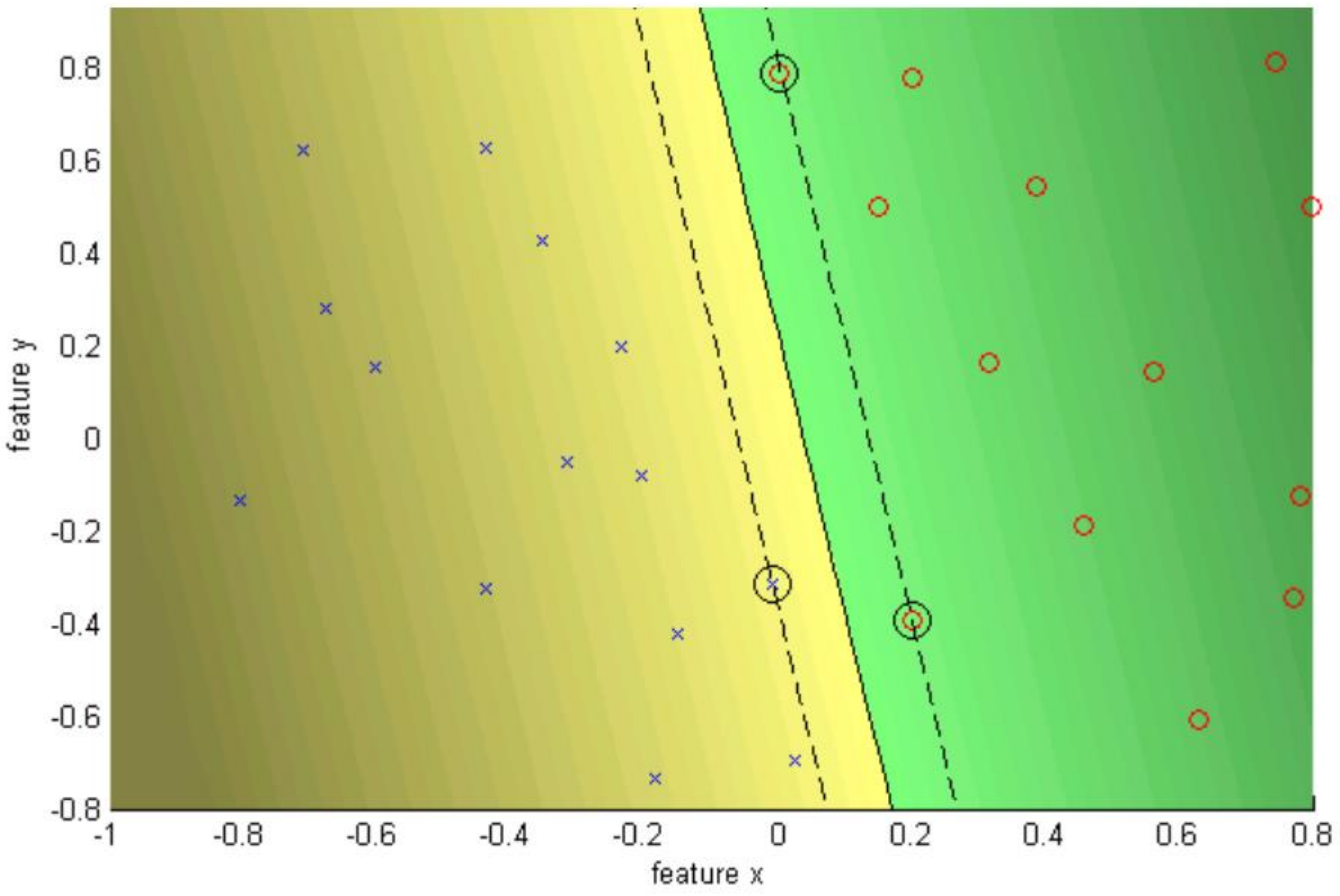
if C smaller: slack allowed

# Two Very Similar Problems



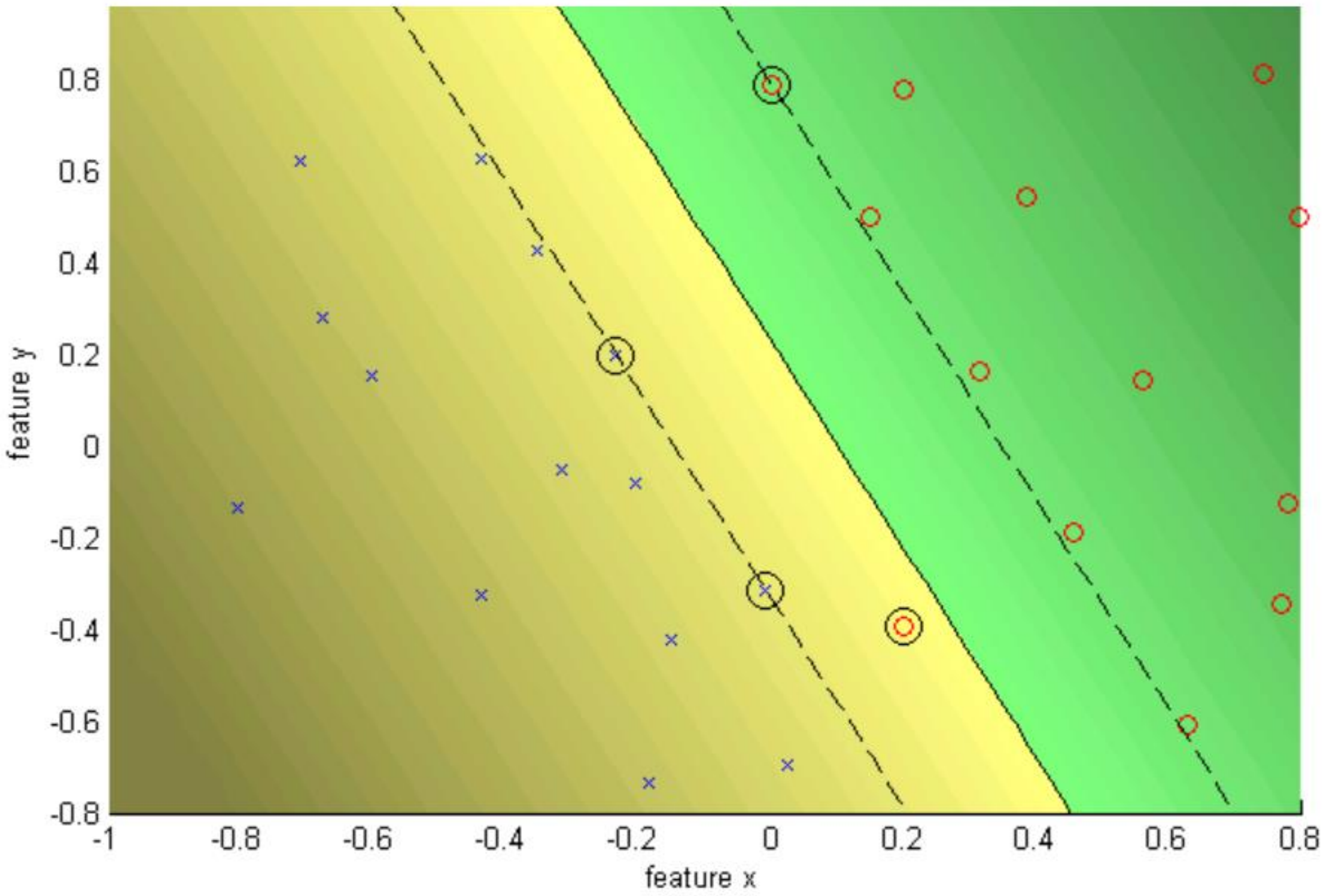


# Hard Margin (C = Infinity)



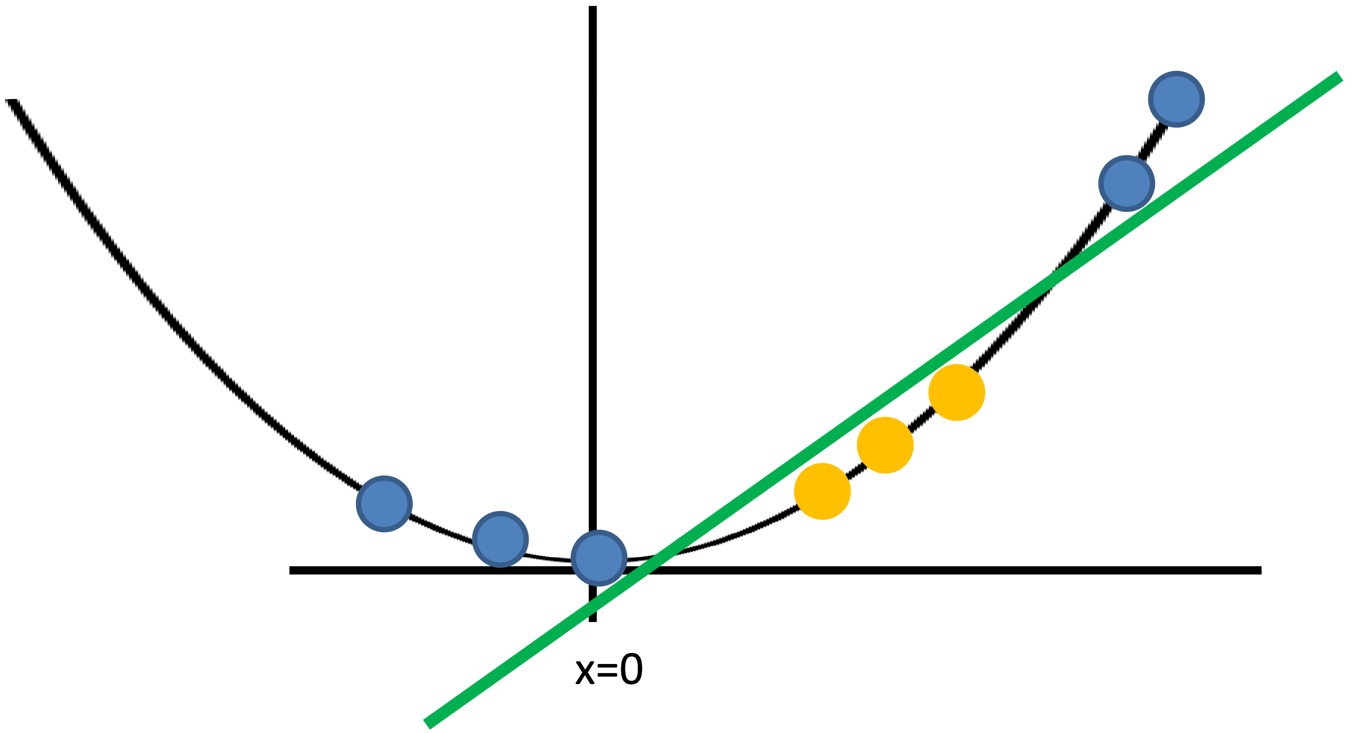
http://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

# Soft Margin (C = 10)



http://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

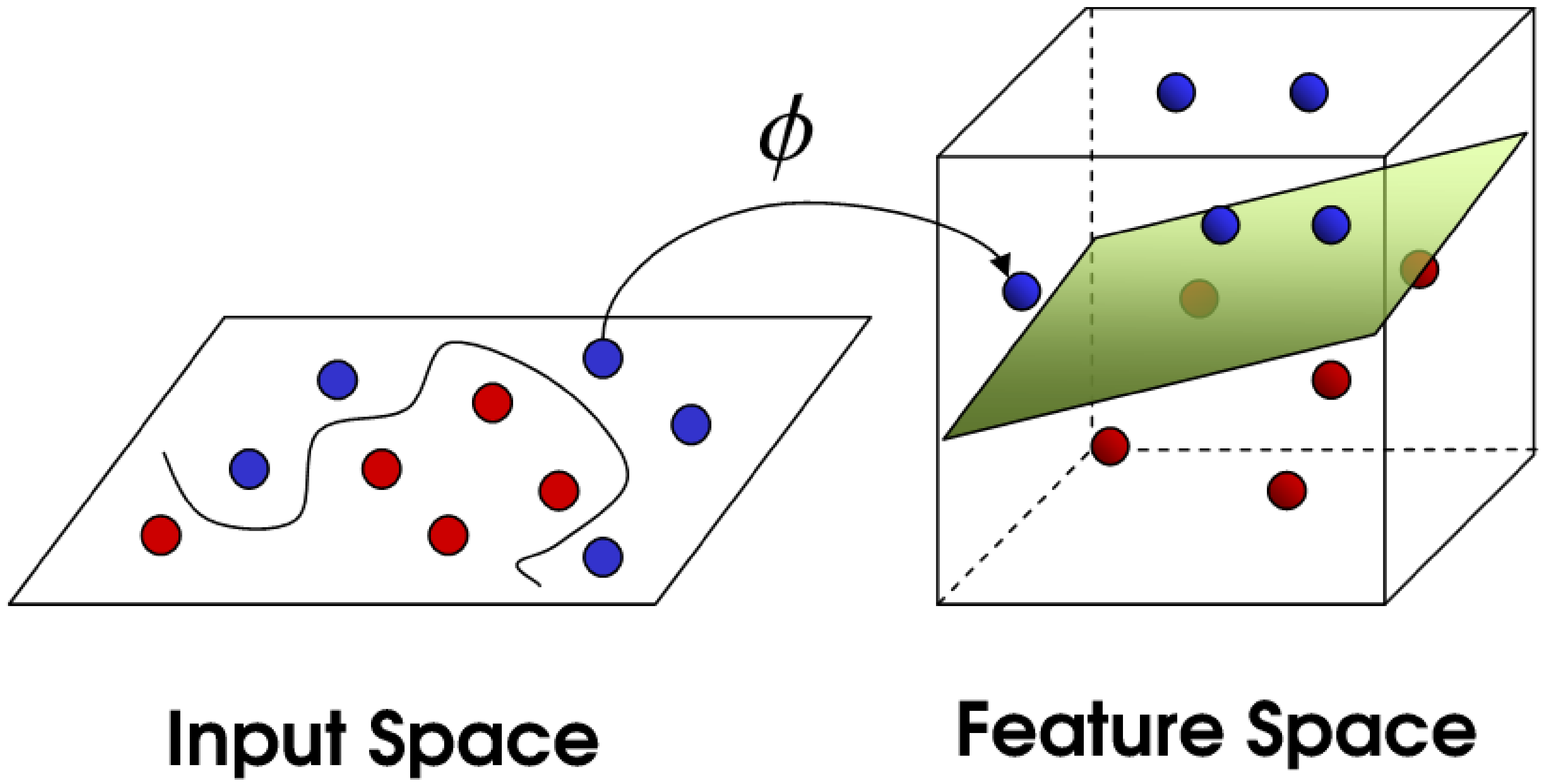
# XOR problem revised



Did we add information to make the problem seperable?

no extra measurements, no extra data

# Non-Linear Decision Boundary



## SVM with a polynomial Kernel visualization

Created by:
Udi Aharoni

# Quadratic Kernel

$$x = (x_1, x_2)$$

$$\Phi(x) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

$$\Phi(x) \cdot \Phi(z) = 1 + 2\sum_{i=1}^{d} x_i z_i$$

$$+ \sum_{i=1}^{d} x_i^2 z_i^2 + 2\sum_{i=1}^{d}\sum_{j=i+1}^{d} x_i x_j z_i z_j$$

$$\boxed{= (1 + x \cdot z)^2}$$

you don't have to compute phi(x)!

# Kernel Functions

$$K(x,z) = \Phi(x) \cdot \Phi(z)$$

- Polynomial:

$$K(x,z) = (1 + x \cdot z)^s$$

- Radial basis function (RBF):

$$K(x,z) = \exp(-\gamma(x-z)^2)$$

## So what is the excitement?

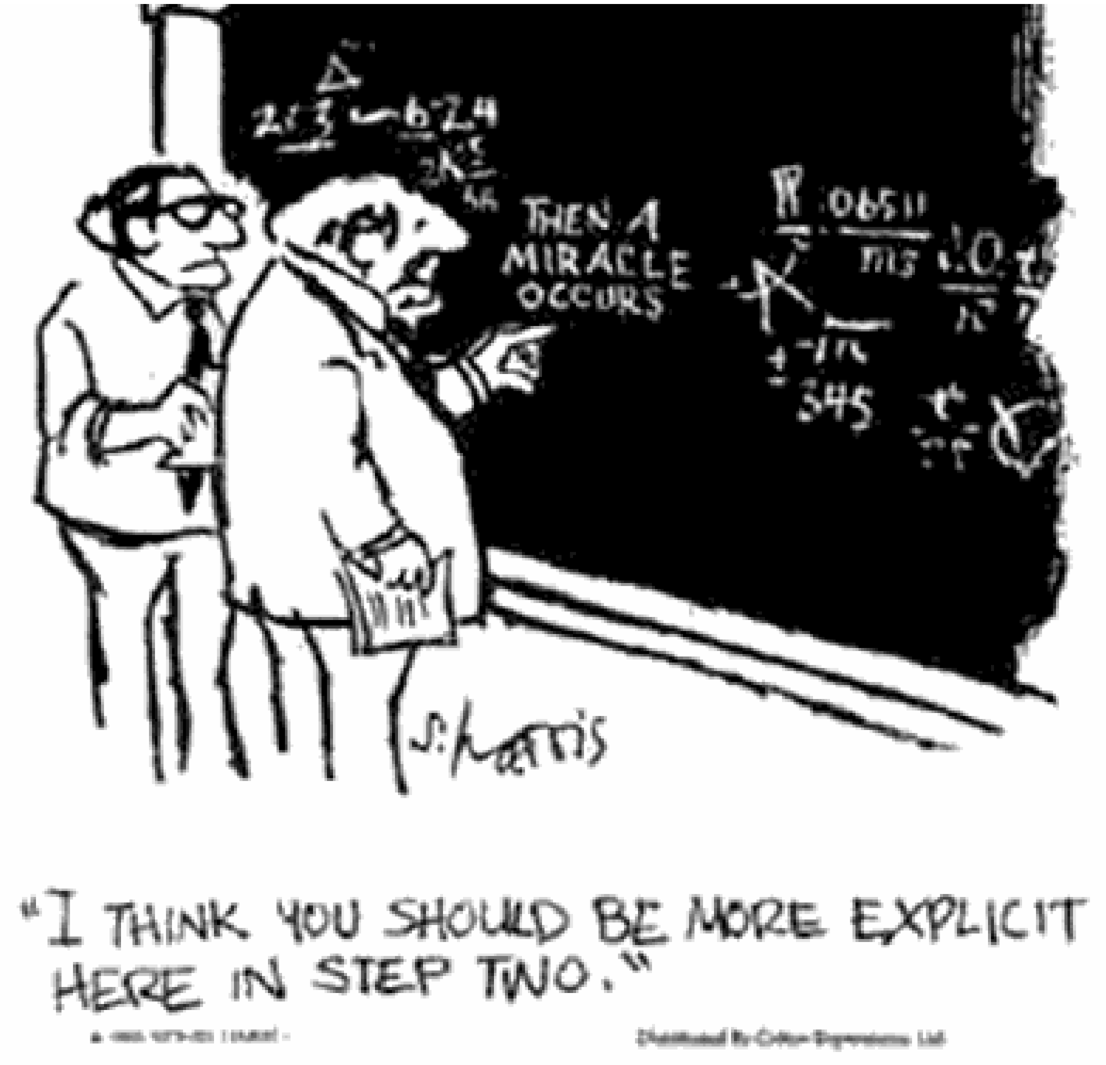
$\max_\alpha \sum$ $^{(i)^T} x^{(j)}$

s.t. $\alpha_i$

$\sum$

arg r

s.t. $y$



## So what is the excitement?

$$\max_\alpha \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \boxed{{x^{(i)}}^T x^{(j)}}$$

$$\text{s.t. } \alpha_i \geq 0,\; i = 1, \ldots, m$$

$$\sum_{i=1}^{m} \alpha_i y^{(i)} = 0$$

$$\Uparrow$$

$$\boxed{K(x^{(i)}, x^{(j)})}$$

$$\Uparrow$$

$$\arg\min_{w,b} \frac{1}{2}||w||^2$$

$$\text{s.t. } y^{(i)}(w^T x^{(i)} + b) \geq 1$$

## Prediction

$$w^T x + b = \sum_{i=1}^{m} \alpha_i y^{(i)} \langle x^{(i)}, x \rangle + b.$$

- Again we can use the kernel trick!
- Prediction speed depends on number of support vectors

## The Miracle Explained

- Andrew Ng does this really well

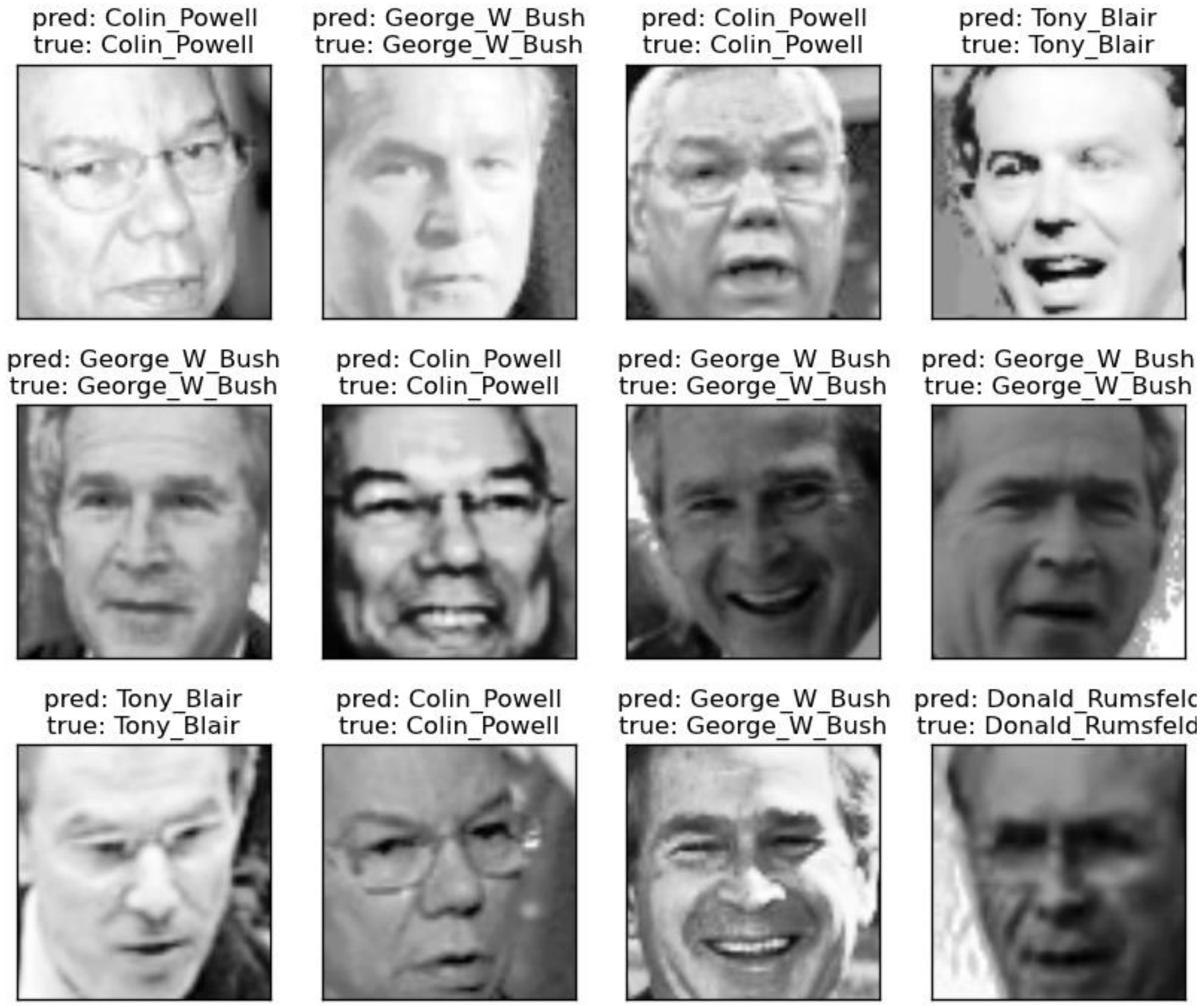
- http://cs229.stanford.edu/notes/cs229-notes3.pdf
- Course is also on Youtube, ItunesU, etc.

## Kernel Trick for SVMs

- Arbitrary many dimensions
- Little computational cost
- Maximal margin helps with curse of dimensionality

Helps avoid overfitting

# Face Recognition



## Face Recognition

- Load image data
- Put your test data aside
- Extract Eigenfaces
- Train SVM
- Evaluate performance

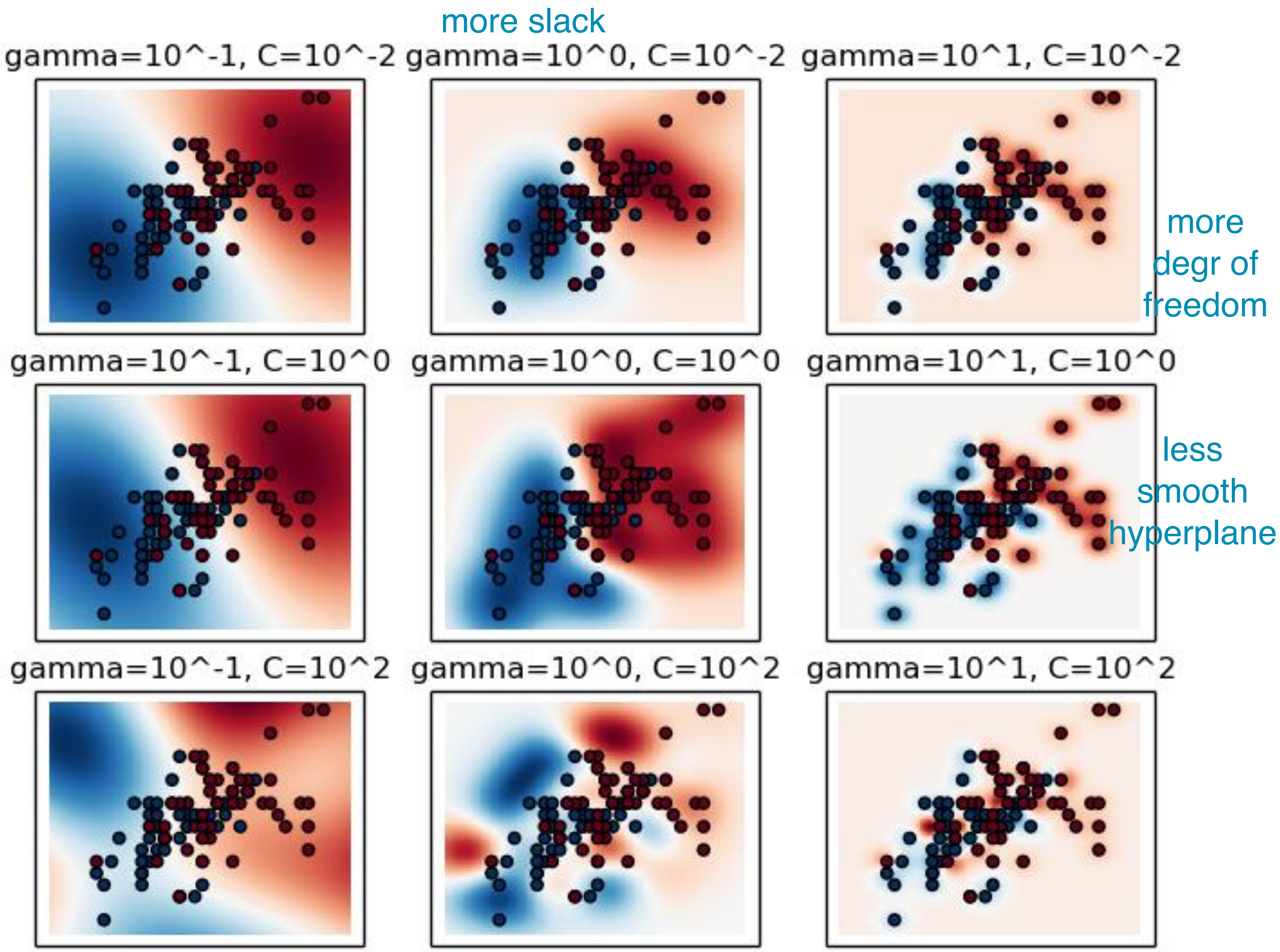
- Red are cross validation steps

http://scikit-learn.org/stable/auto_examples/applications/face_recognition.html#example-applications-face-recognition-py

SVM_sign_language.mp4

Jhon Gonzalez

https://www.youtube.com/watch?v=cxHMgl2_5zg



http://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html

## Tips and Tricks

- SVMs are not scale invariant
- Check if your library normalizes by default
- Normalize your data
  - mean: 0 , std: 1
  - map to [0,1] or [-1,1]
- Normalize test set in same way!

## Tips and Tricks

- RBF kernel is a good default
- For parameters try exponential sequences
- Read:

  Chih-Wei Hsu et al., "A Practical Guide to Support Vector Classification", Bioinformatics (2010)

# SVM vs KNN

- What are the main key differences?

SVM: no need to keep all training data, just support vectors

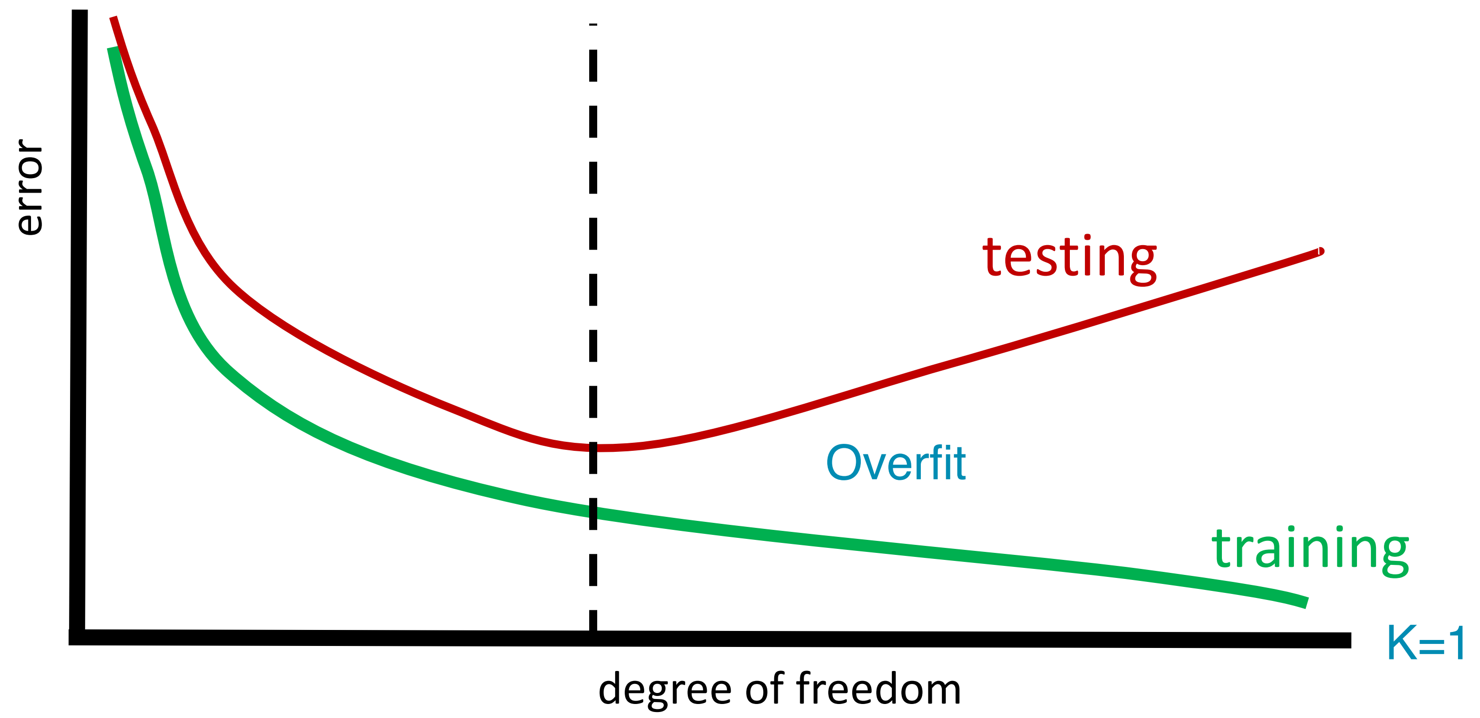
## Parameter Tuning

- Given a classification task

- Which kernel ?
- Which kernel parameter values?
- Which value for C?

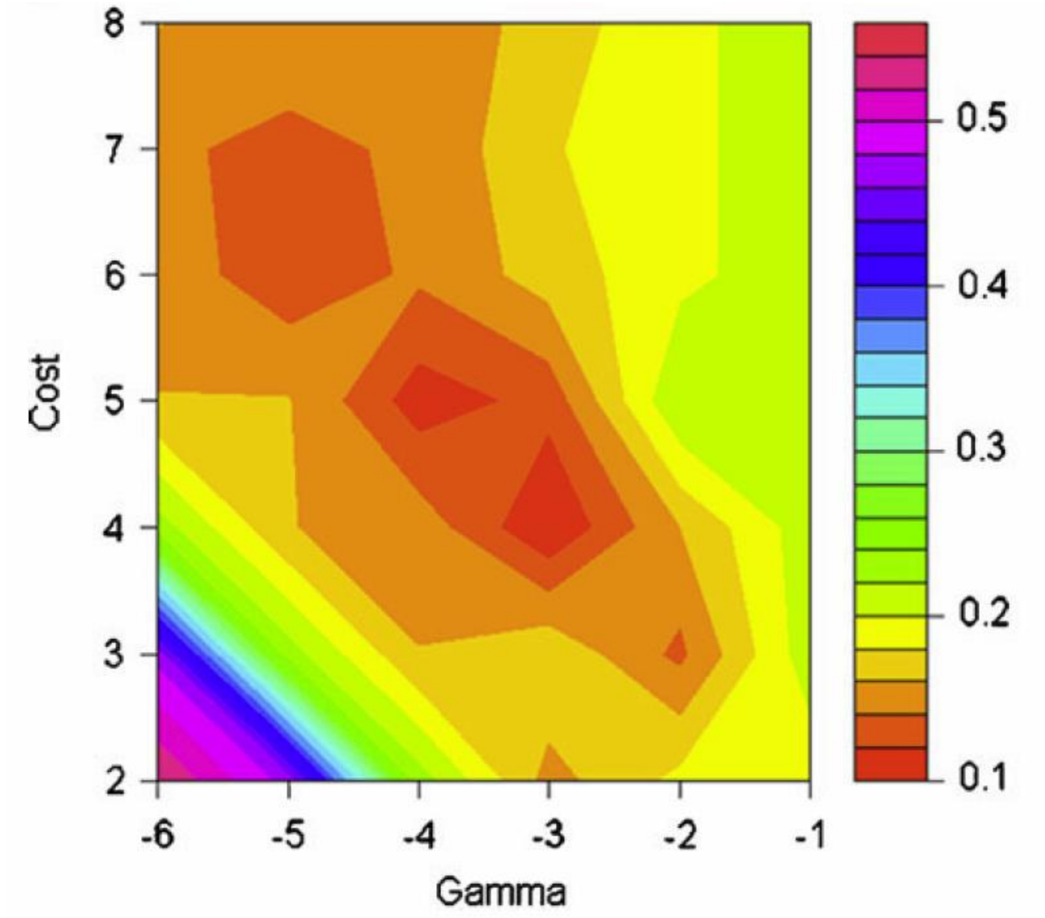
Try different combinations
and take the best.

# Train vs. Test Error



Where is KNN on this graph for K=1, or for K=Inf?

# Grid Search



Zang et al., “Identification of heparin samples that contain impurities or contaminants by chemometric pattern recognition analysis of proton NMR spectral data”, Anal Bioanal Chem (2011)

## Error Measures

- True positive (tp)
- True negative (tn)
- False positive (fp)
- False negative (fn)

| | | predicted | |
|---|---|---|---|
| | | 1 | -1 |
| true | 1 | tp | fn |
| | -1 | fp | tn |

# TPR and FPR

- True Positive Rate:

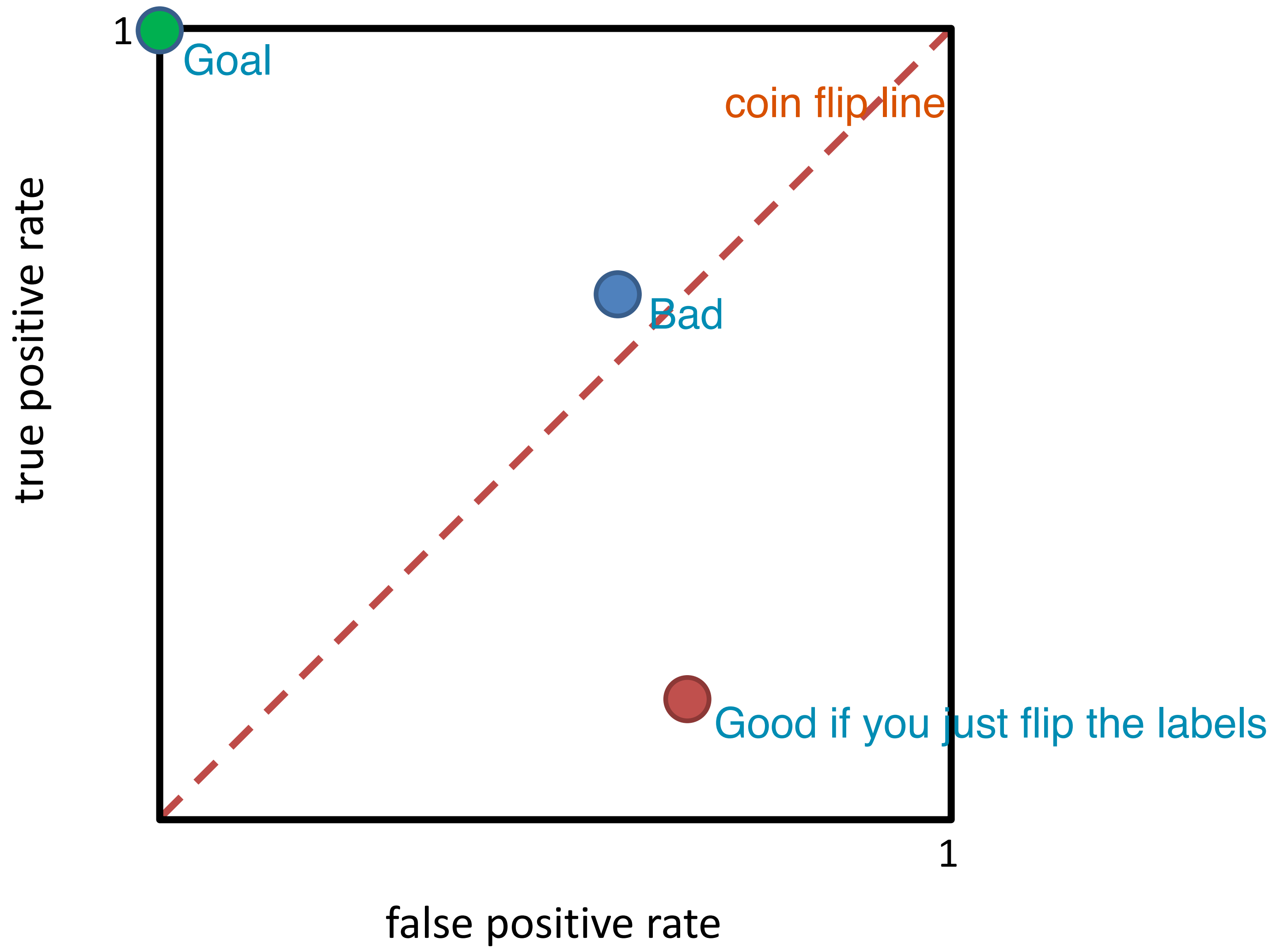
$$\frac{tp}{tp+fn}$$

- False Positive Rate:

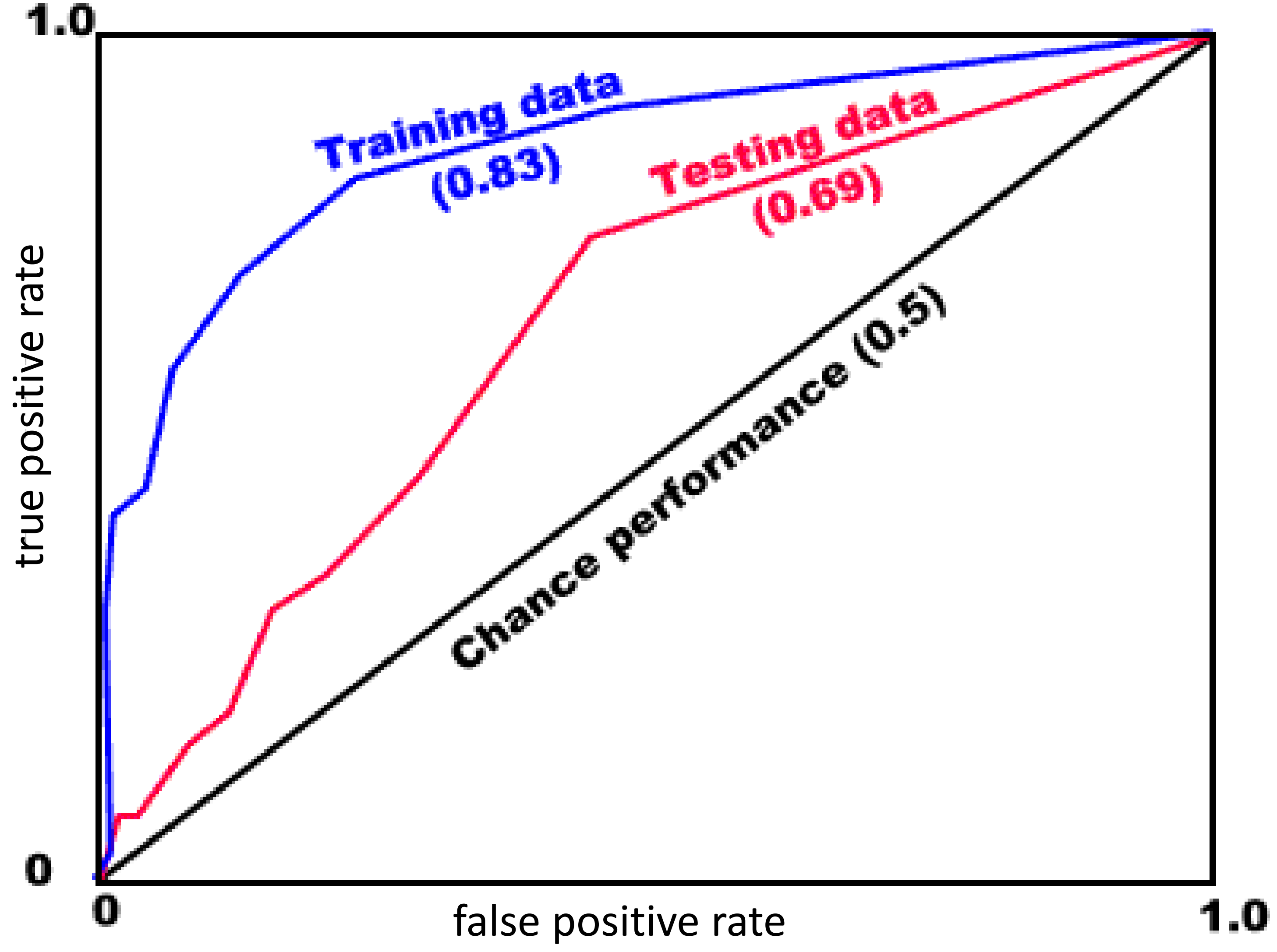
$$\frac{fp}{fp+tn}$$

| | | predicted | |
|---|---|---|---|
| | | 1 | -1 |
| true | 1 | tp | fn |
| | -1 | fp | tn |

# Reciever Operating Characteristic



# ROC Example



https://inclass.kaggle.com/c/ca-2015/details/evaluation

# Precision Recall

- Recall: $\frac{tp}{tp+fn}$

- Precision: $\frac{tp}{tp+fp}$

|  |  | predicted |  |
|---|---|---|---|
|  |  | 1 | -1 |
| true | 1 | tp | fn |
|  | -1 | fp | tn |

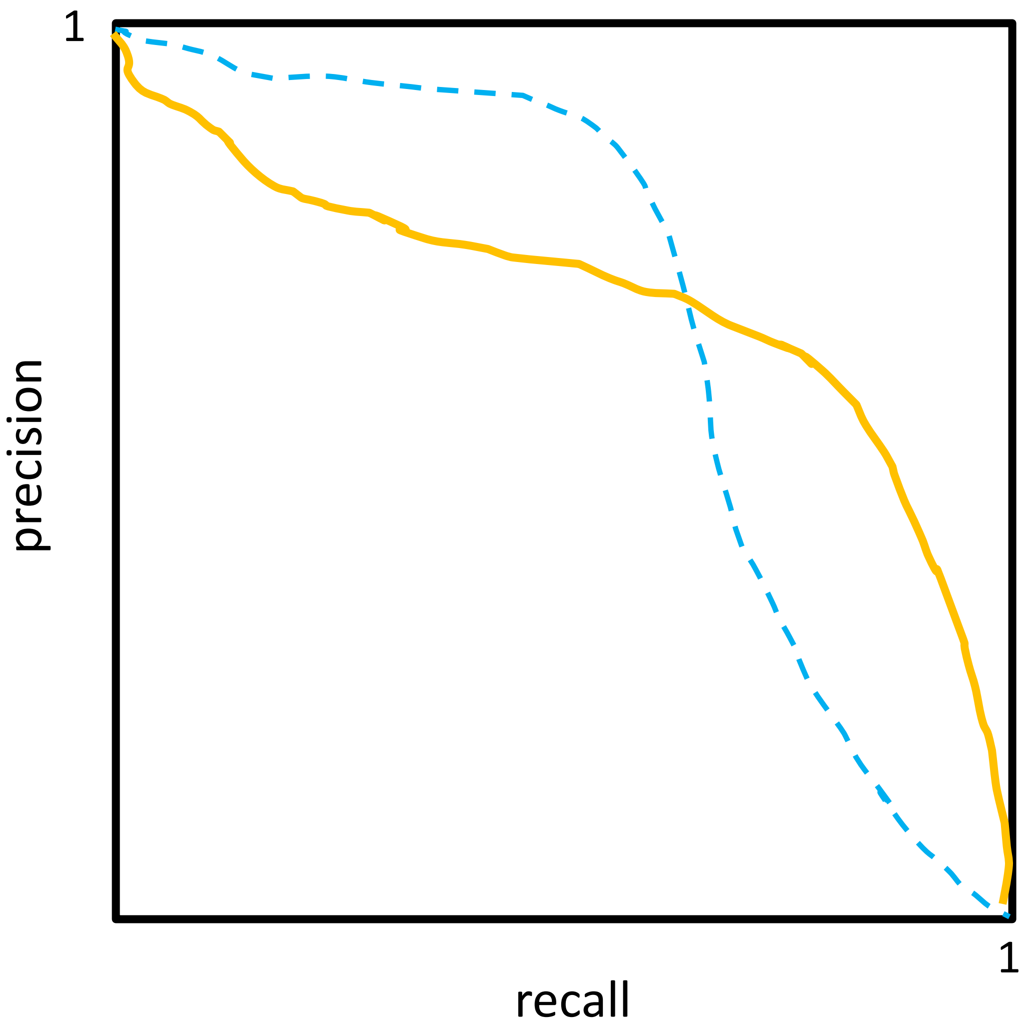
## Precision Recall

- **Recall**: If I pick a random positive example, what is the probability of making the right prediction?

Extreme case: classify everything +1 -> recall = 1, precision = 1/2

- **Precision**: If I take a positive prediction example, what is the probability that it is indeed a positive example?

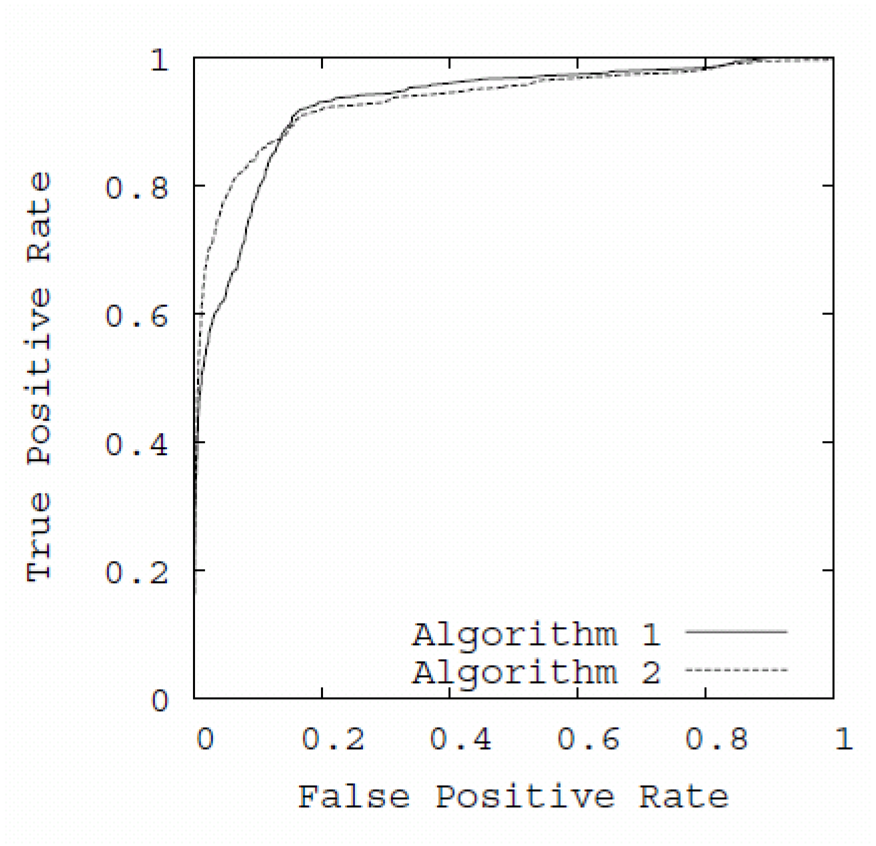
# Precision Recall Curve

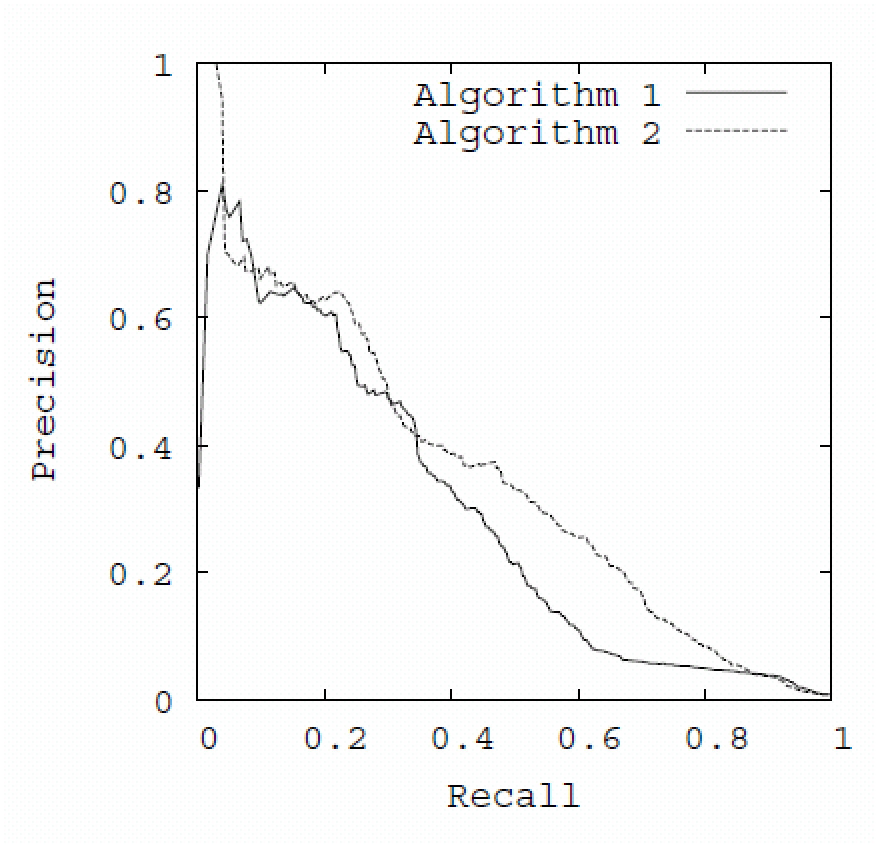


What is better?
depends on application

ex: screening at airport:
recall more important

# Comparison





Algo 2 is better

J. Davis & M. Goadrich,
“The Relationship Between Precision-Recall and ROC Curves.”,
*ICML (2006)*

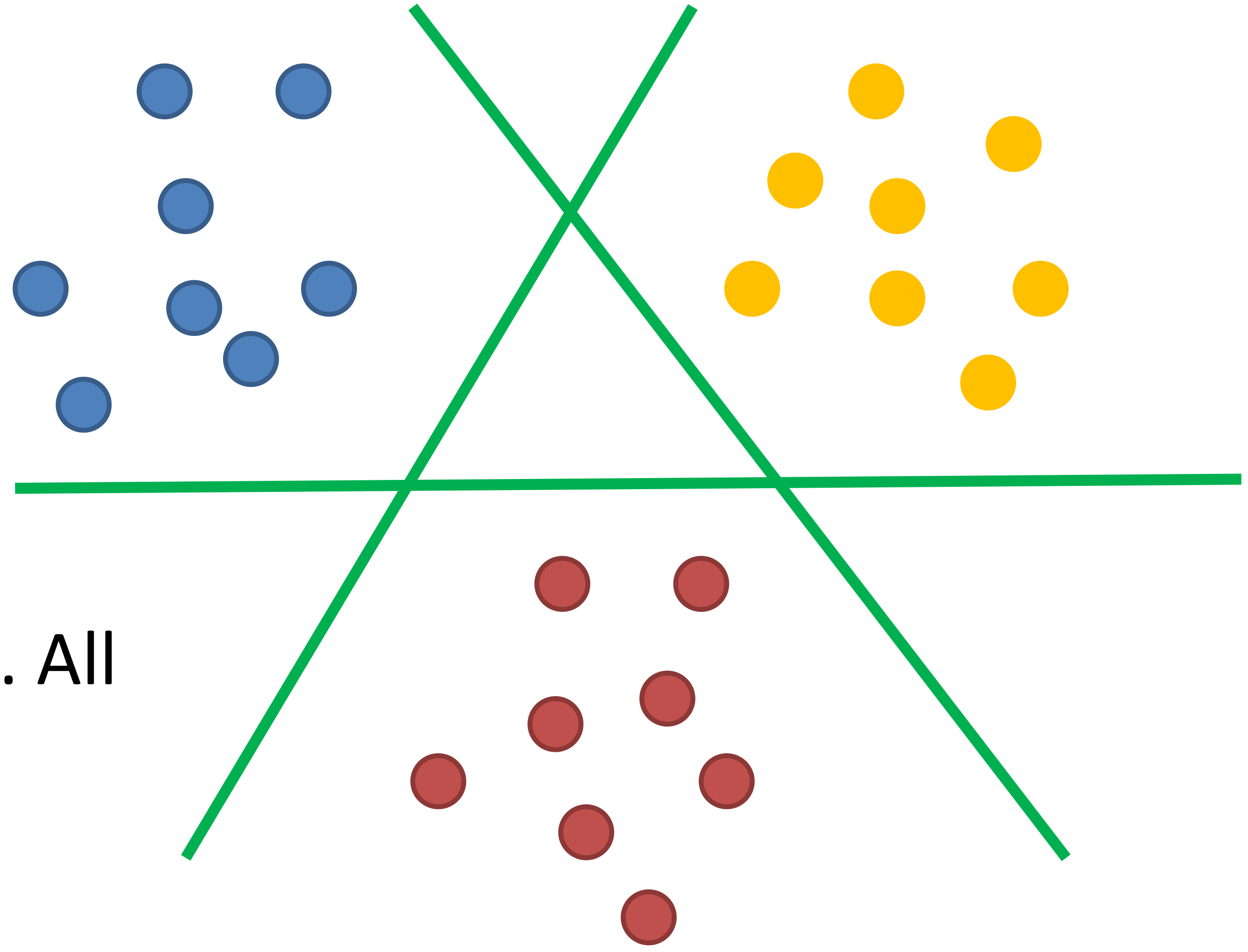
## F-measure

- Weighted average of precision and recall

$$F_\beta = \frac{(\beta^2+1)\cdot P\cdot R}{\beta^2\cdot P+R}$$

- Usual case: $\beta = 1$
- Increasing $\beta$ allocates weight to recall

# Multi Class

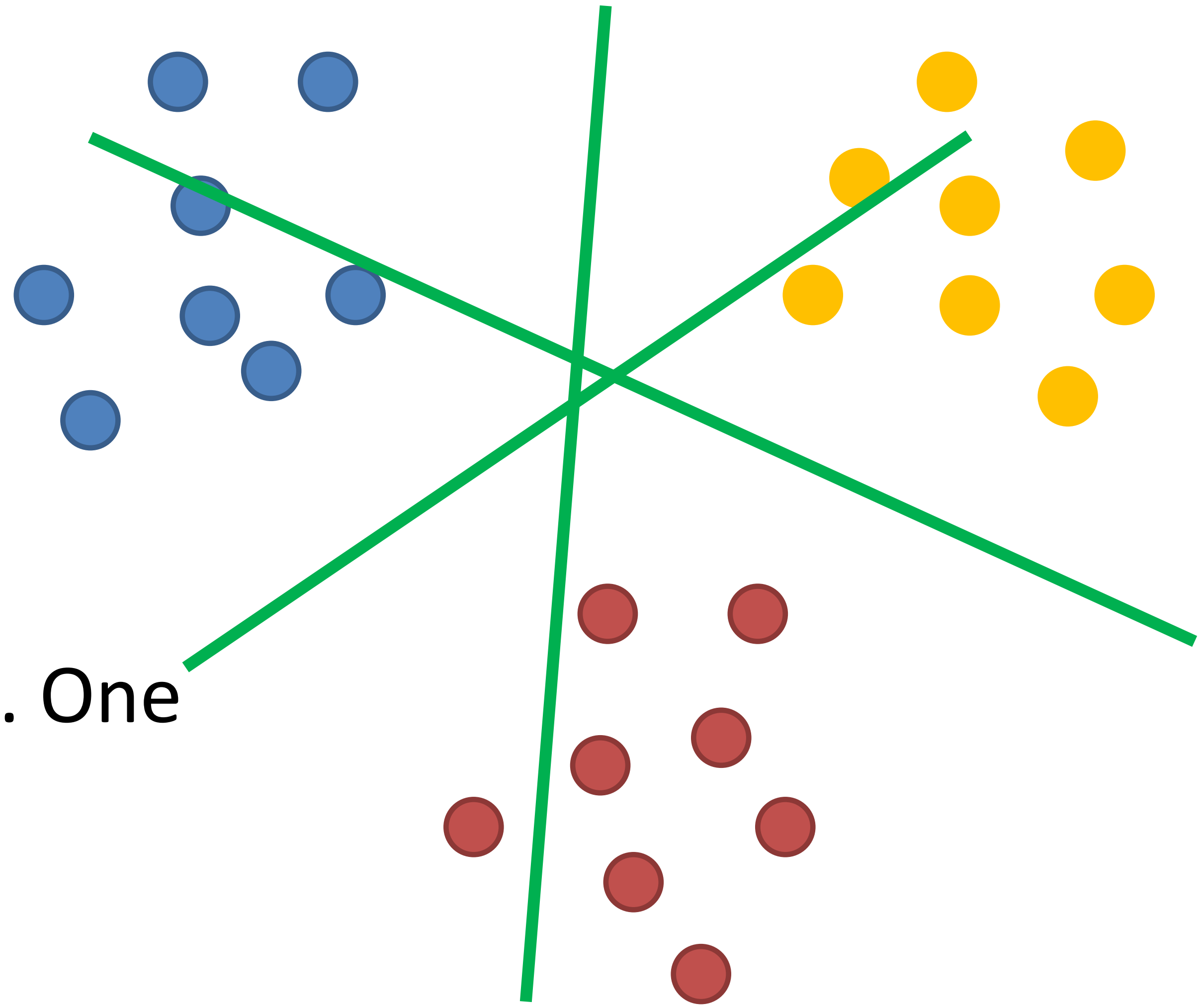


One vs. All

## One vs All

- Train n classifier for n classes
- Take classification with greatest margin
- Slow training

# Multi Class

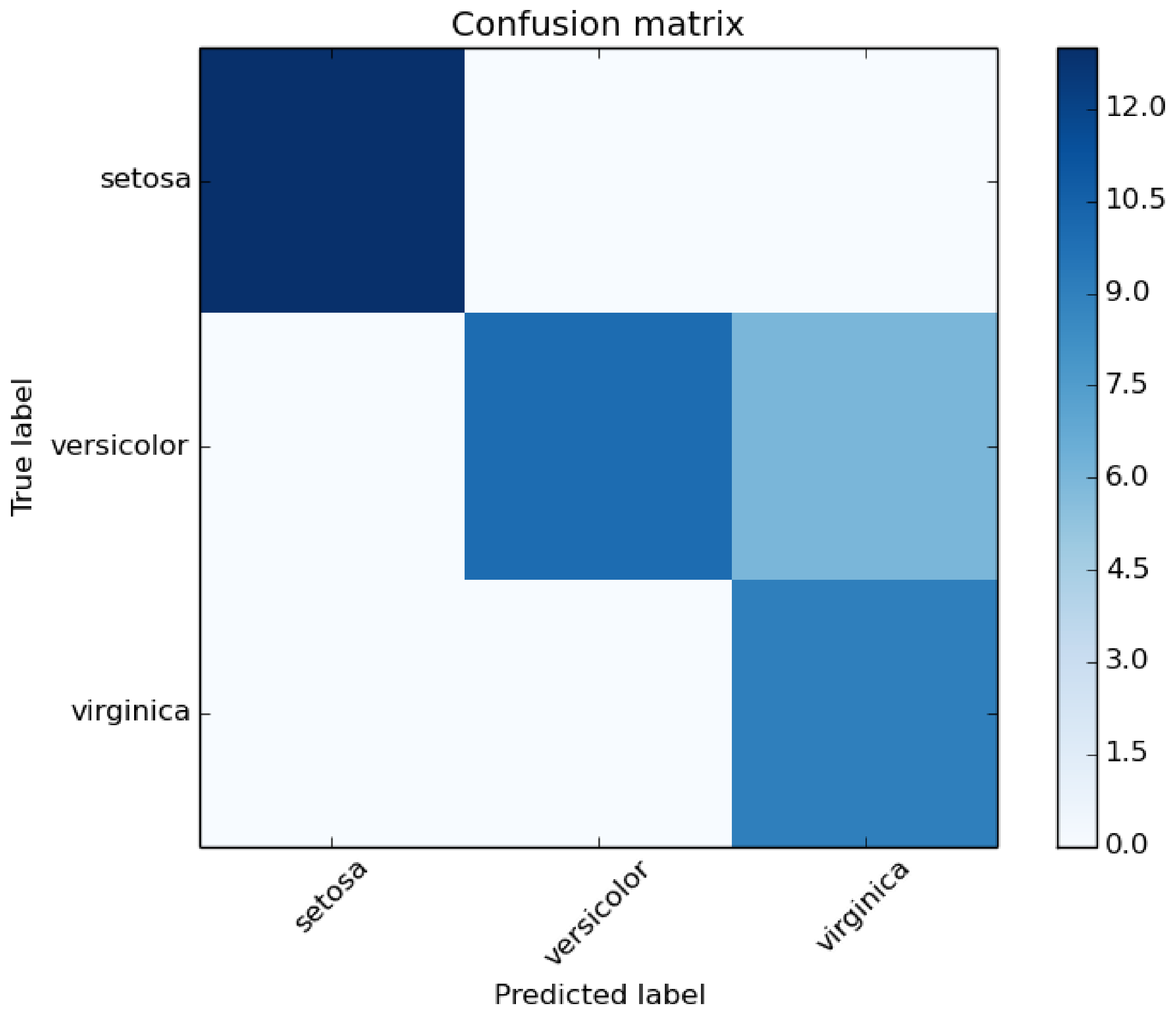


One vs. One

## One vs One

- Train n(n-1)/2 classifiers
- Take majority vote
- Fast training because a lot less points

# Confusion Matrix



http://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html

## Recap

- Perceptrons are great
- But really just a separating hyperplane
- So is SVM
- Kernels are neat
- Evaluation metrics are important