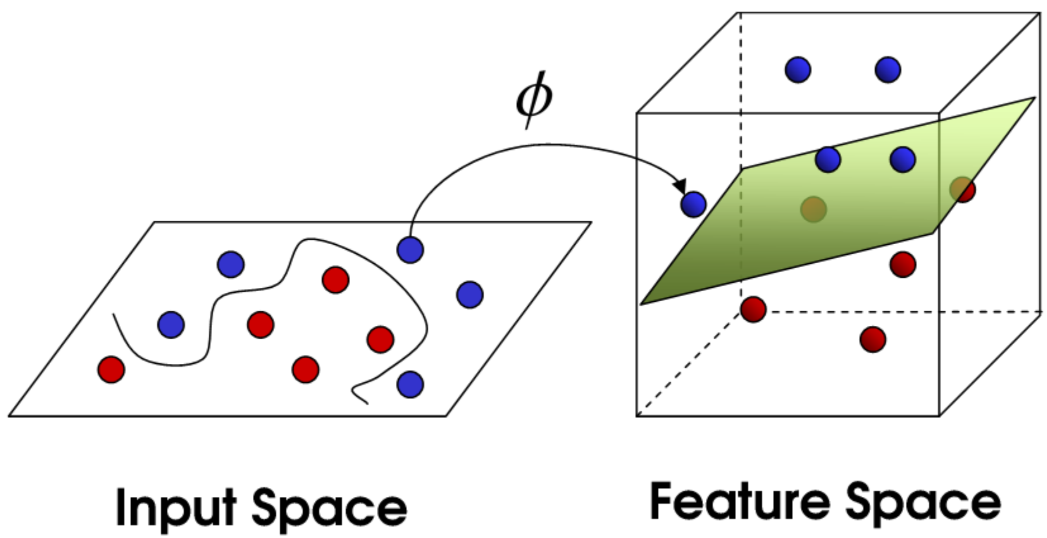
CS109 – Data Science
SVM, Performance evaluation

Joe Blitzstein, Hanspeter Pfister, Verena Kaynig-Fittkau



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

Announcements

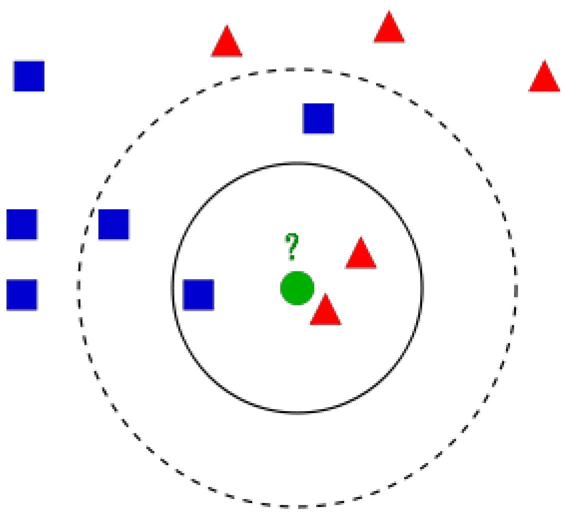
- HW1 grades went out yesterday
- They are looking really good, well done everyone!

- 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

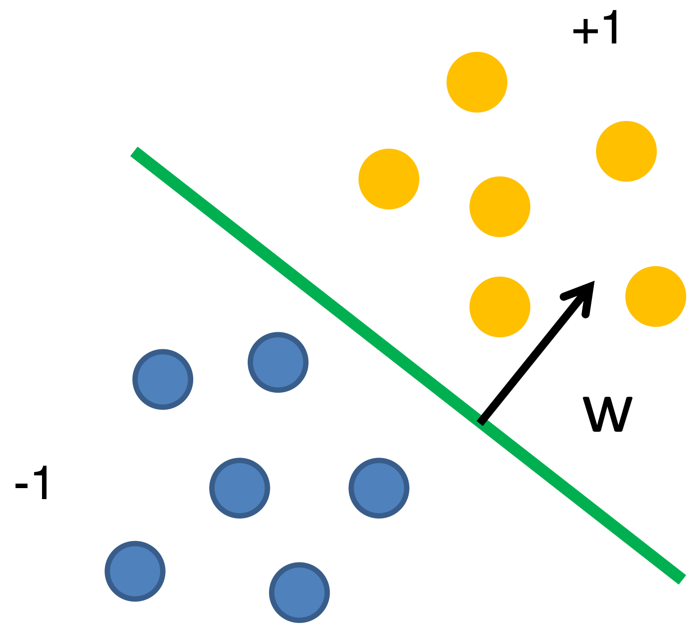


Have to keep all training data stored.
Prediction slow: need to go through all k data points each time.

Separating Hyperplane

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

+/- 1 common labels b/c make support vector machine math easy —> probs default option.

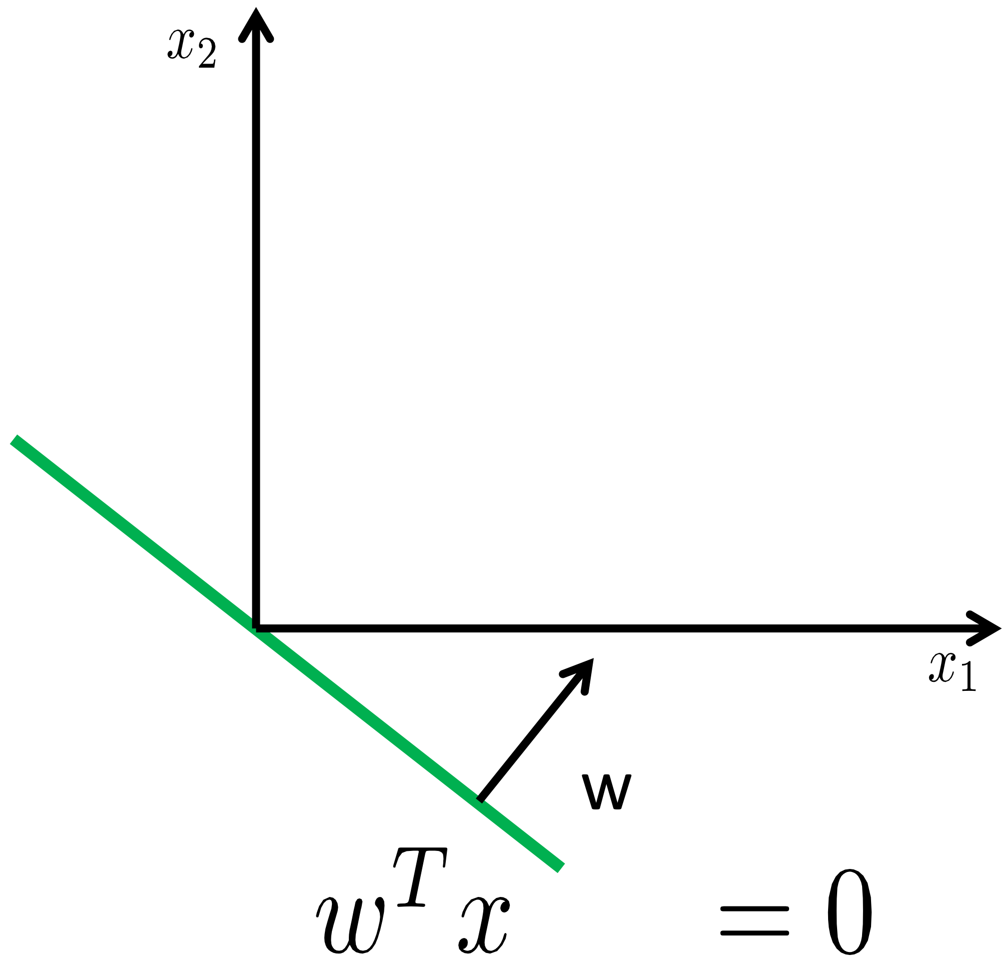


w orthogonal to hyperplane: changing x changes the plane.

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

Separating Hyperplane

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

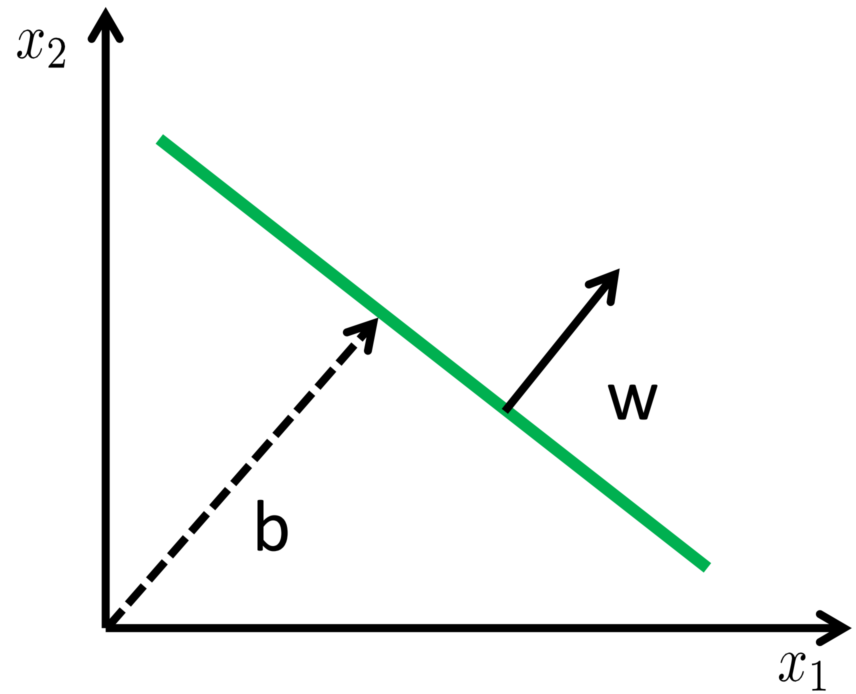


Separating Hyperplane

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

not restricted to origin with bias:

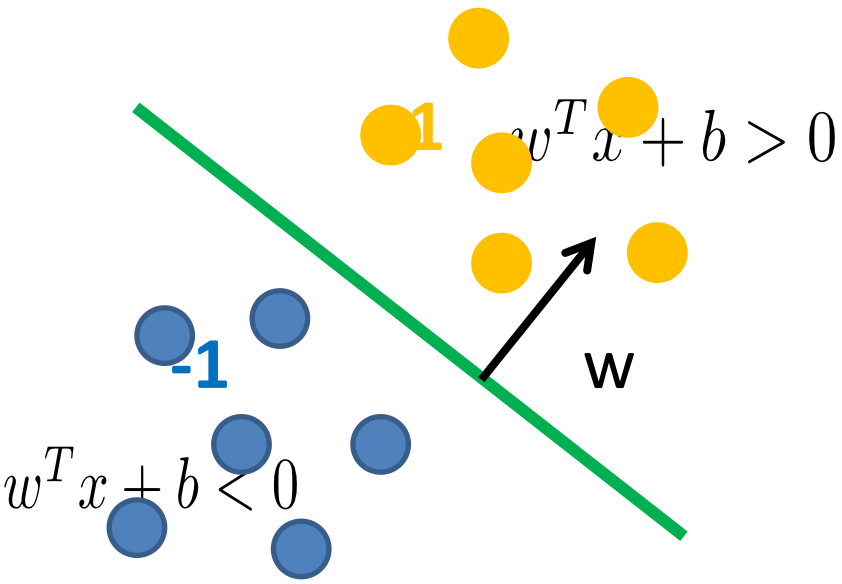
Bias: shift
Weight: orientation of hyperplane



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

Separating Hyperplane

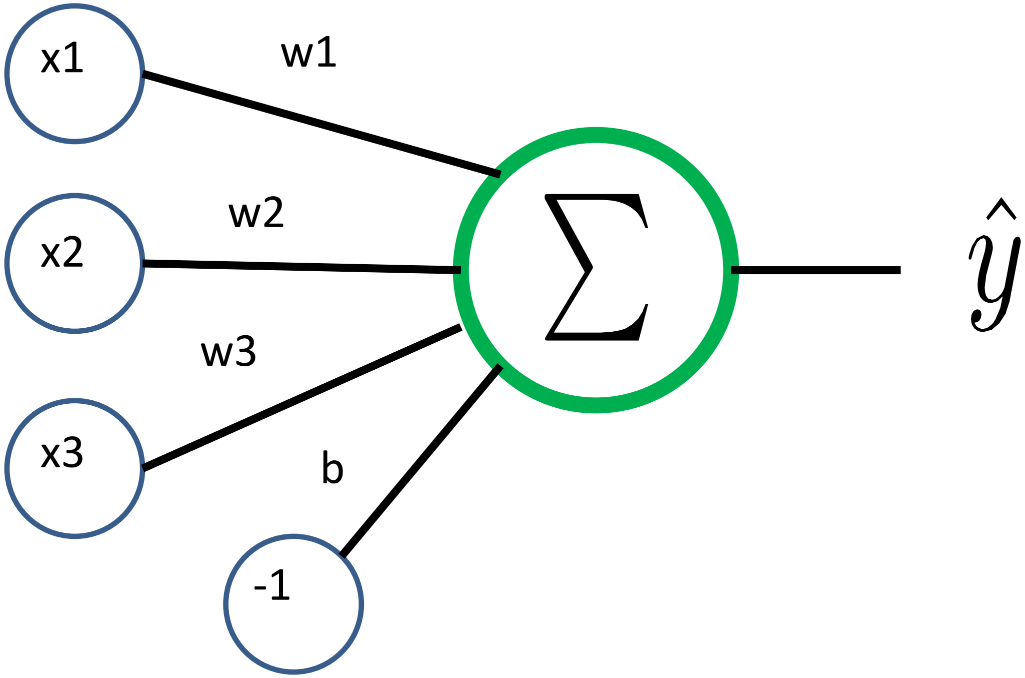
- x: data point
- y: label $\in \{-1, +1\}$
- w: weight vector
- b: bias



what is result of plugging in x:
(+) or (-)
—> forget training data: only
need parameters (weight, bias)

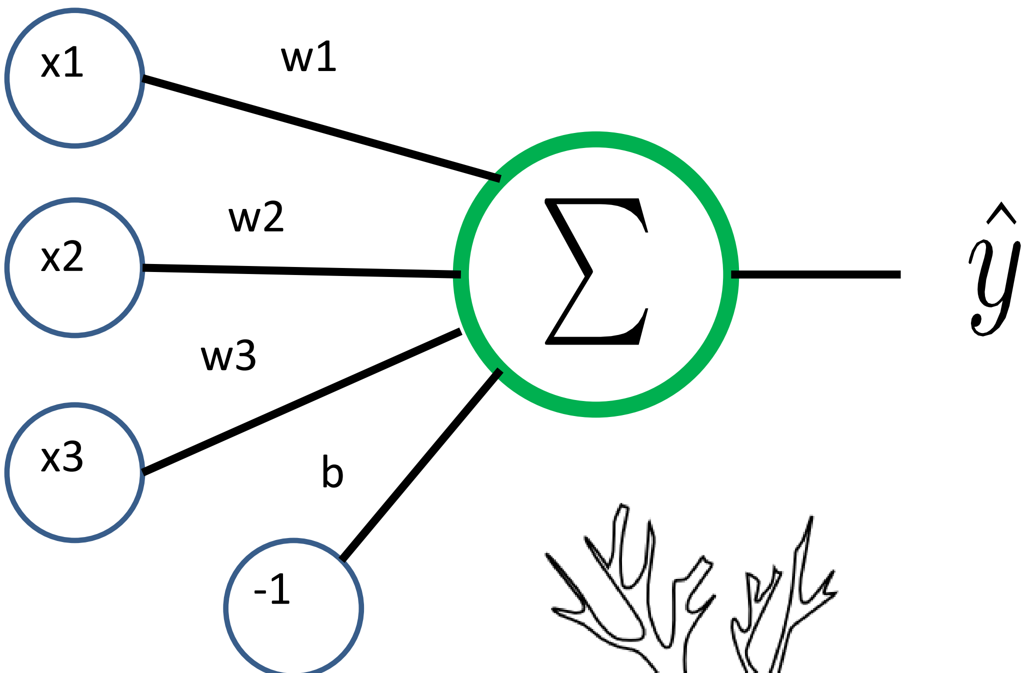
trade-off: now we only have a line: limiting.

Perceptron

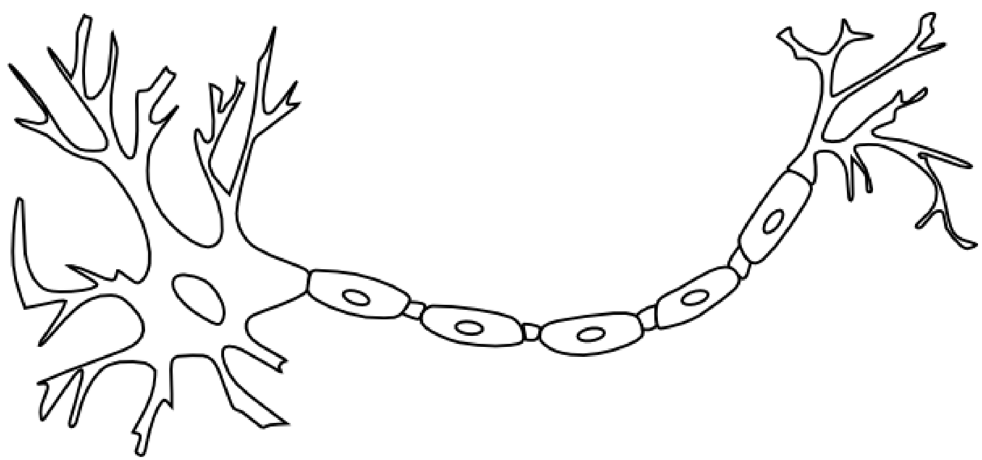


$$w^T x + b = 0$$

Perceptron



mathematical neuron.



Perceptron History

- invented 1957
- by Frank Rosenblatt

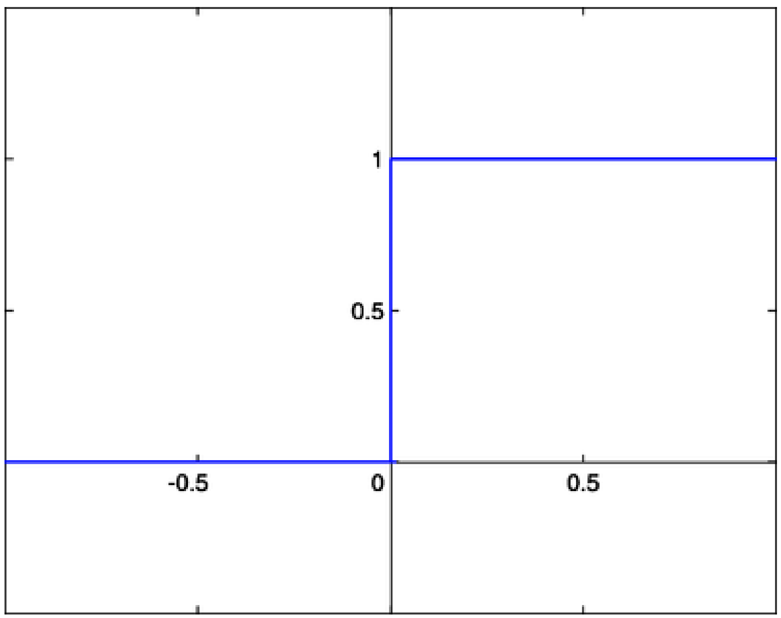
- the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence. (NYT 1958)

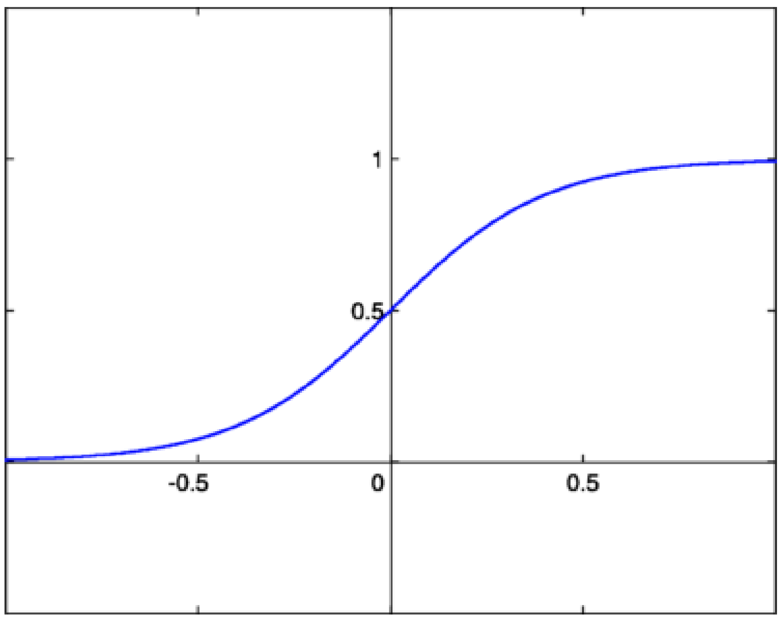
  (http://en.wikipedia.org/wiki/Perceptron basis of deep learning.

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

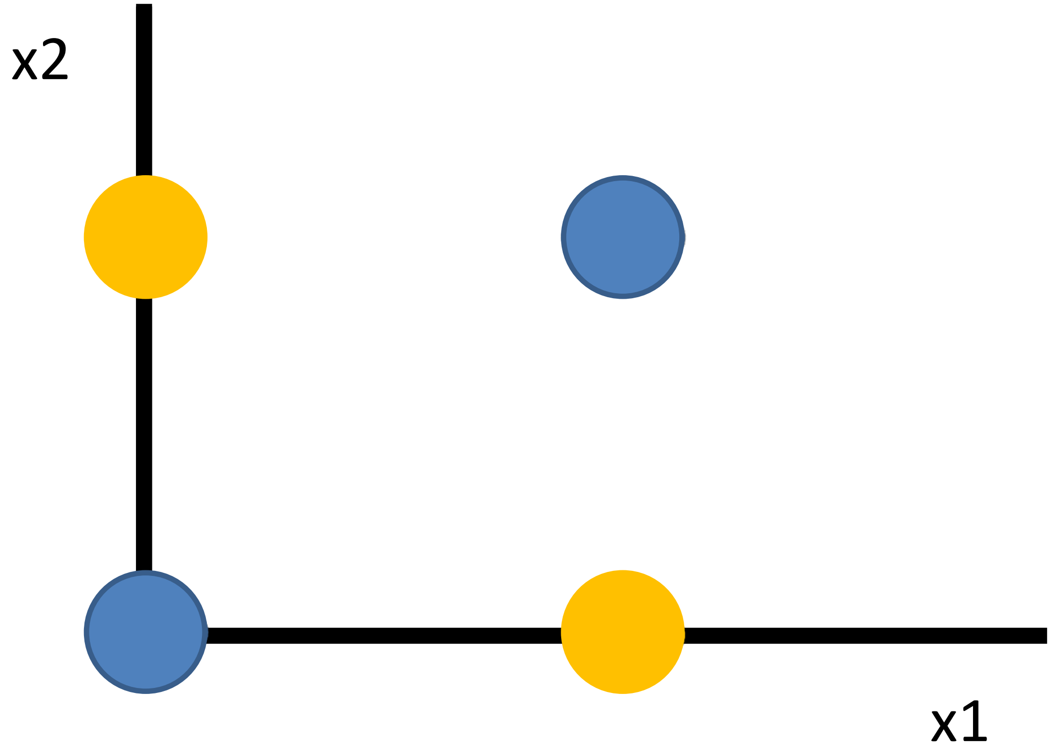
- 1969: Minsky and Papert publish their book "Perceptrons"
  Minsky pointed out weakness in perceptrons

- Very controversial book, some blame the book for causing the whole research area to stagnate.

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

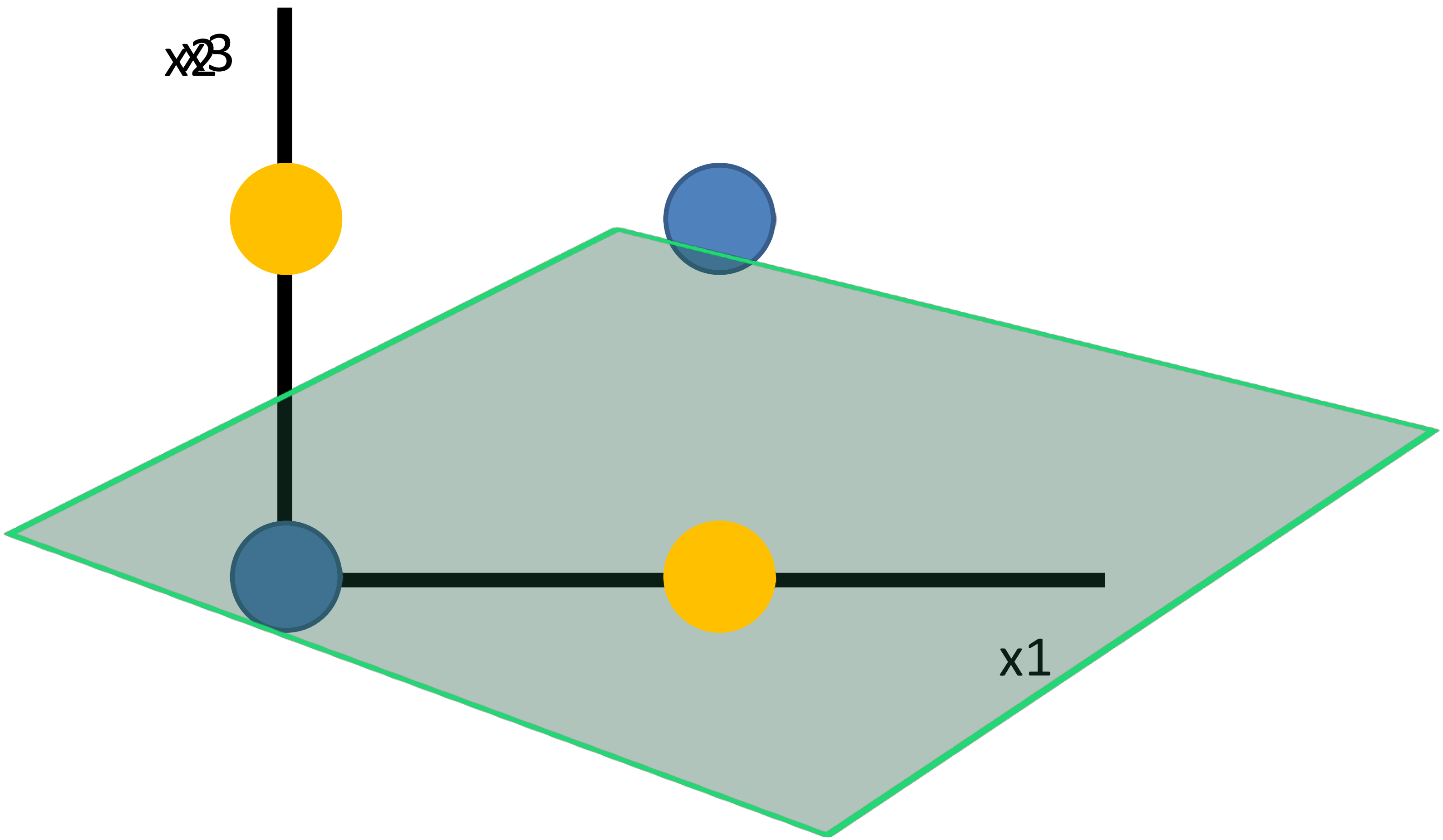
The XOR Problem

can’t separate with a single hyperplane.



The XOR Problem

Trick: transform into a 3D problem: now can separate with a 2D hyperplane.
But, don’t add too many dimensions: hit curse of dimensionality
Add as few dimensions as possible to still get seperability.



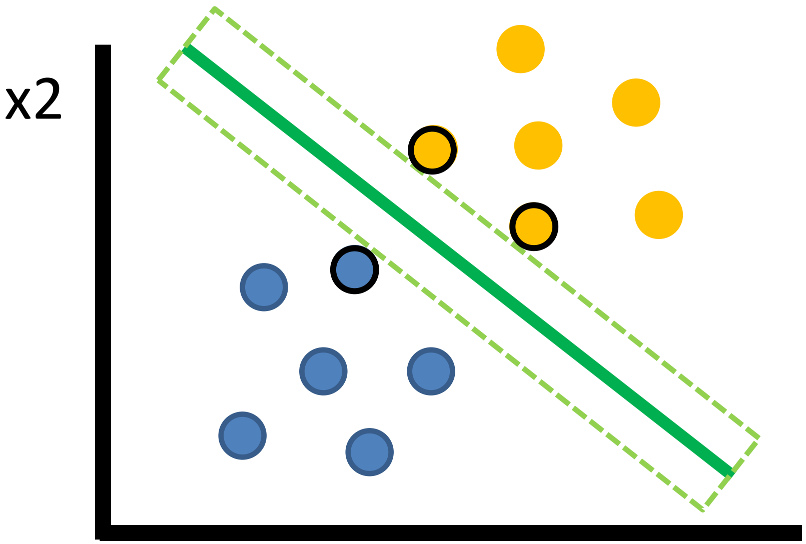
Support Vector Machine

- Widely used for all sorts of classification problems
- Some people say it is the best of the shelf classifier out there

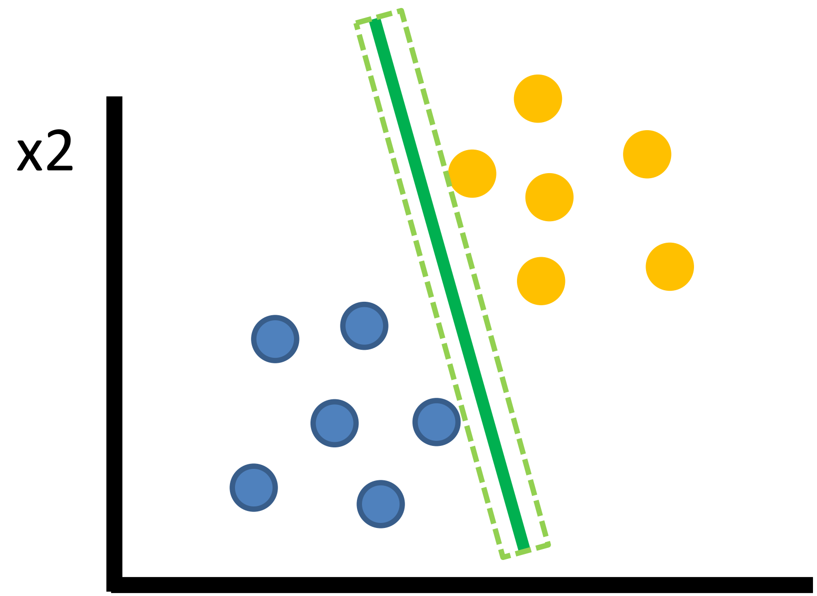
Maximum Margin Classification

Both equally valid to perceptron.

supporting vectors (dashed): they define margin (tangent to closest points)
points defining support vectors can be away from mean: want many pts regardless.



Focus on maximizing distance between hyper plane and closest points —> more generalizable.

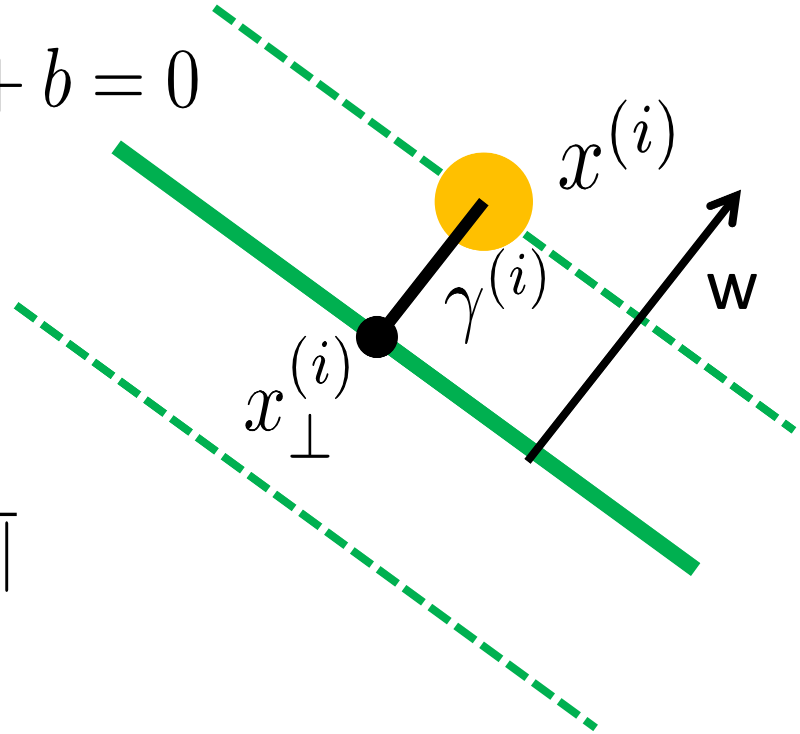


There is too little distance between hyper plane and point. Less generalizable.

Solution depends only on the support vectors!

Maximum Margin Classification

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



margin:

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

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

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

Find w and b that make gamma as large as possible

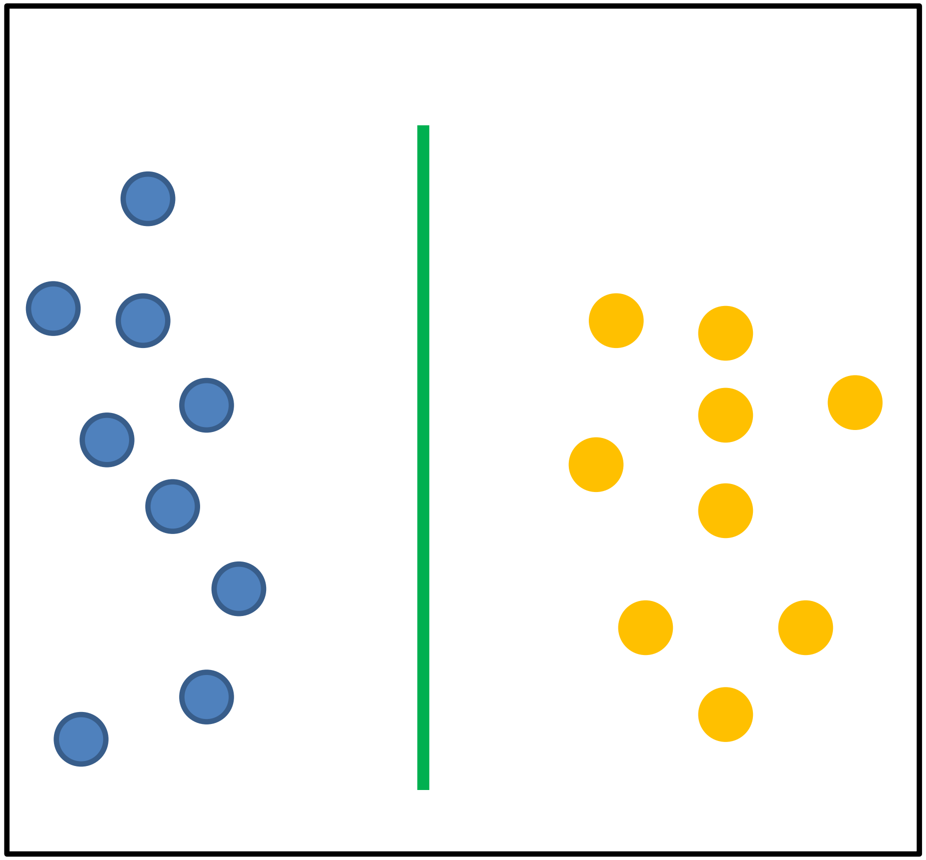
Maximum Margin Classification

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

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

non-convex

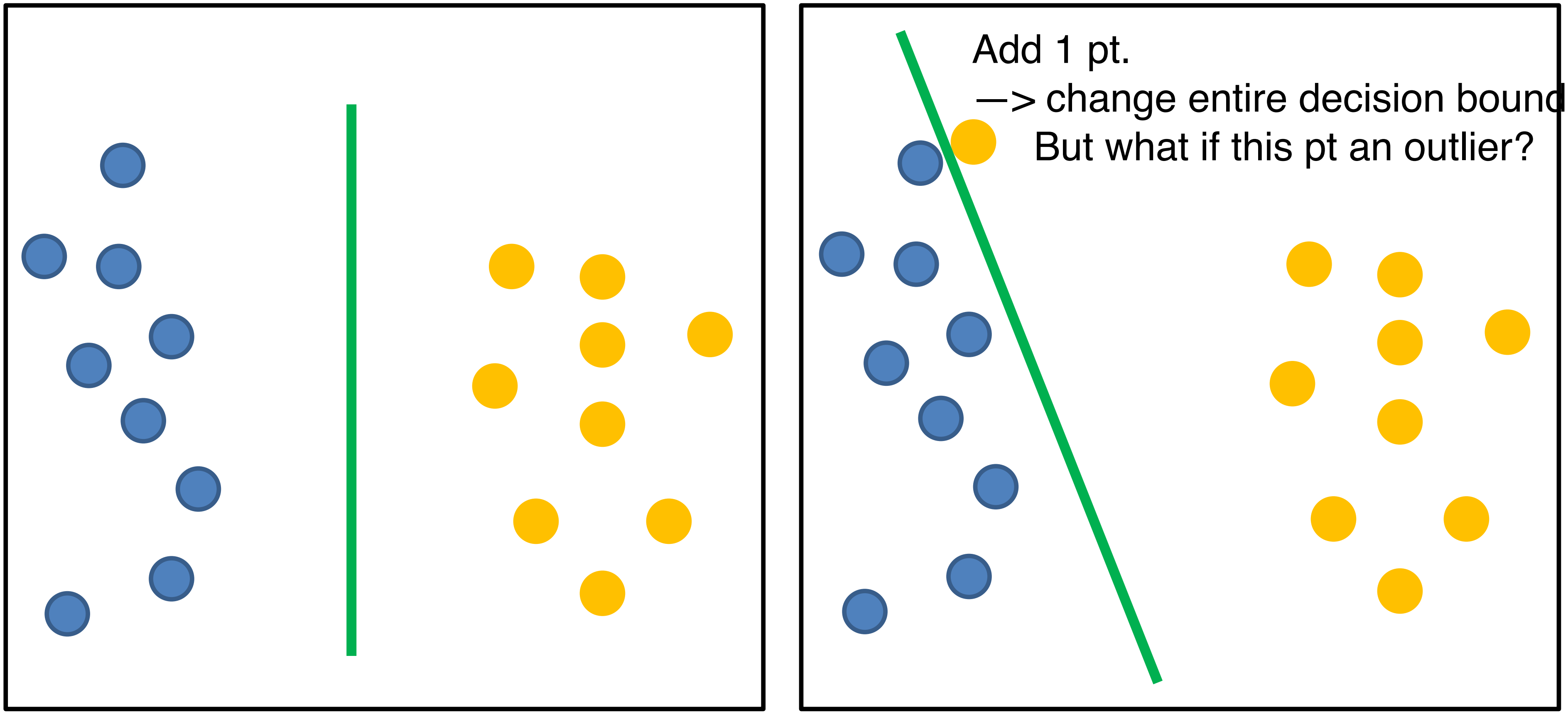
This Is Kind of Odd



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

SVM only cares about borderline cases (those closest to boundary, which are more likely to be outliers.

Two Very Similar Problems



What about outliers?

min distance to margin they should be on.

ξ_i: slack variables

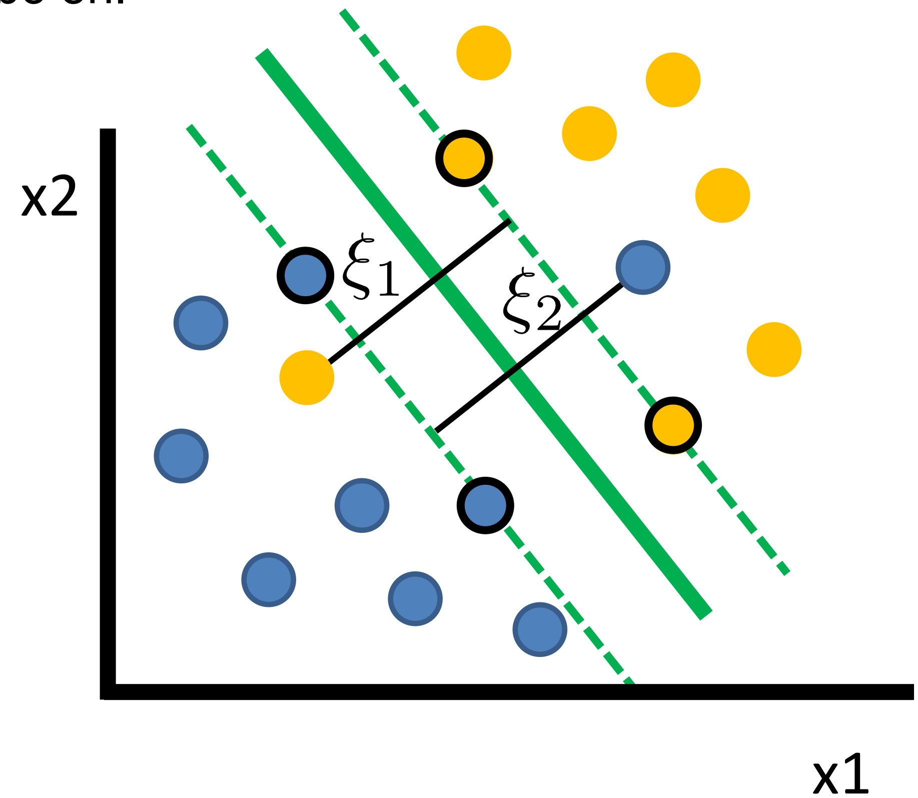
$$\min_{w,b,\xi} \frac{1}{2}||w||^2 + C \text{ sum(slack)}$$

subject to:

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

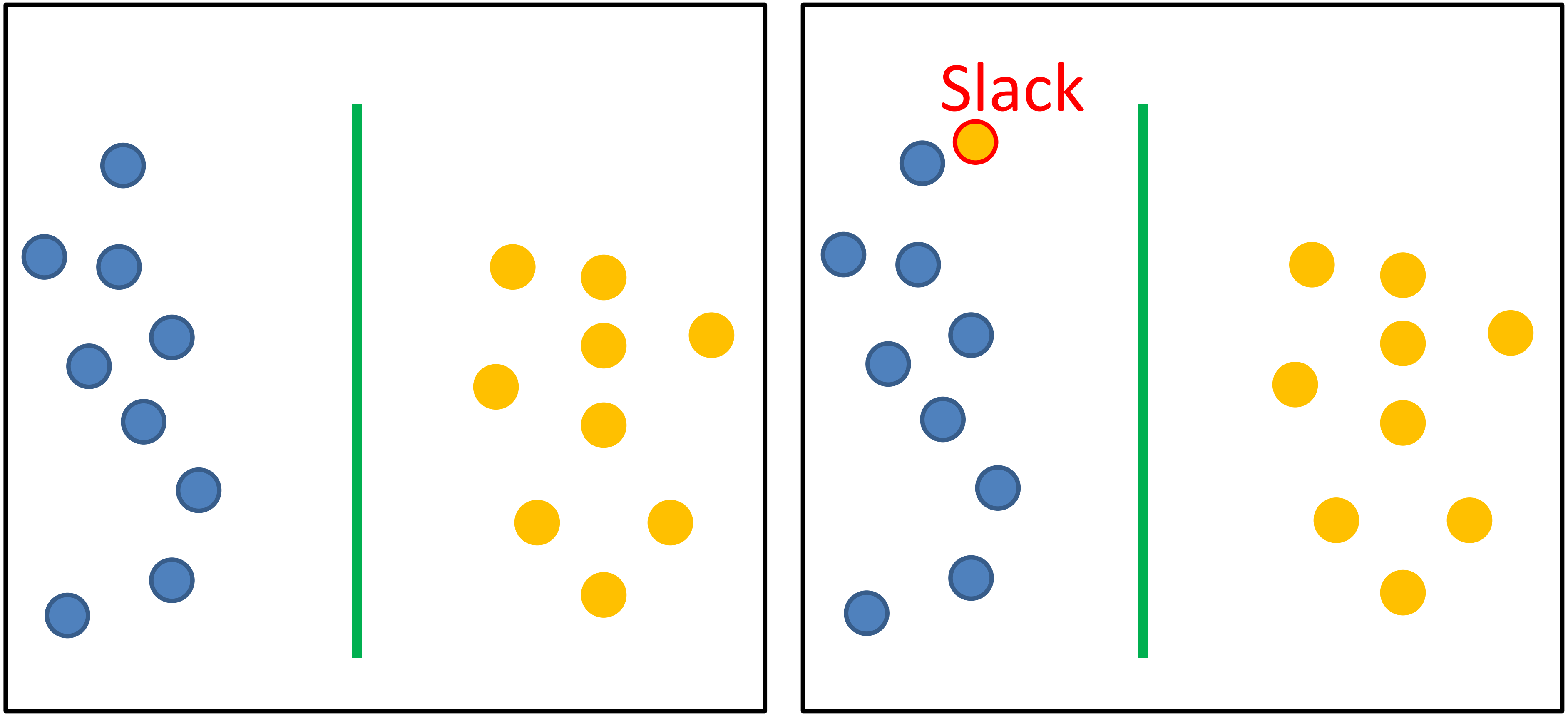
$$(i = 1, \ldots, n)$$

Large C: focus on minimizing slack
Small C: allow for slack.

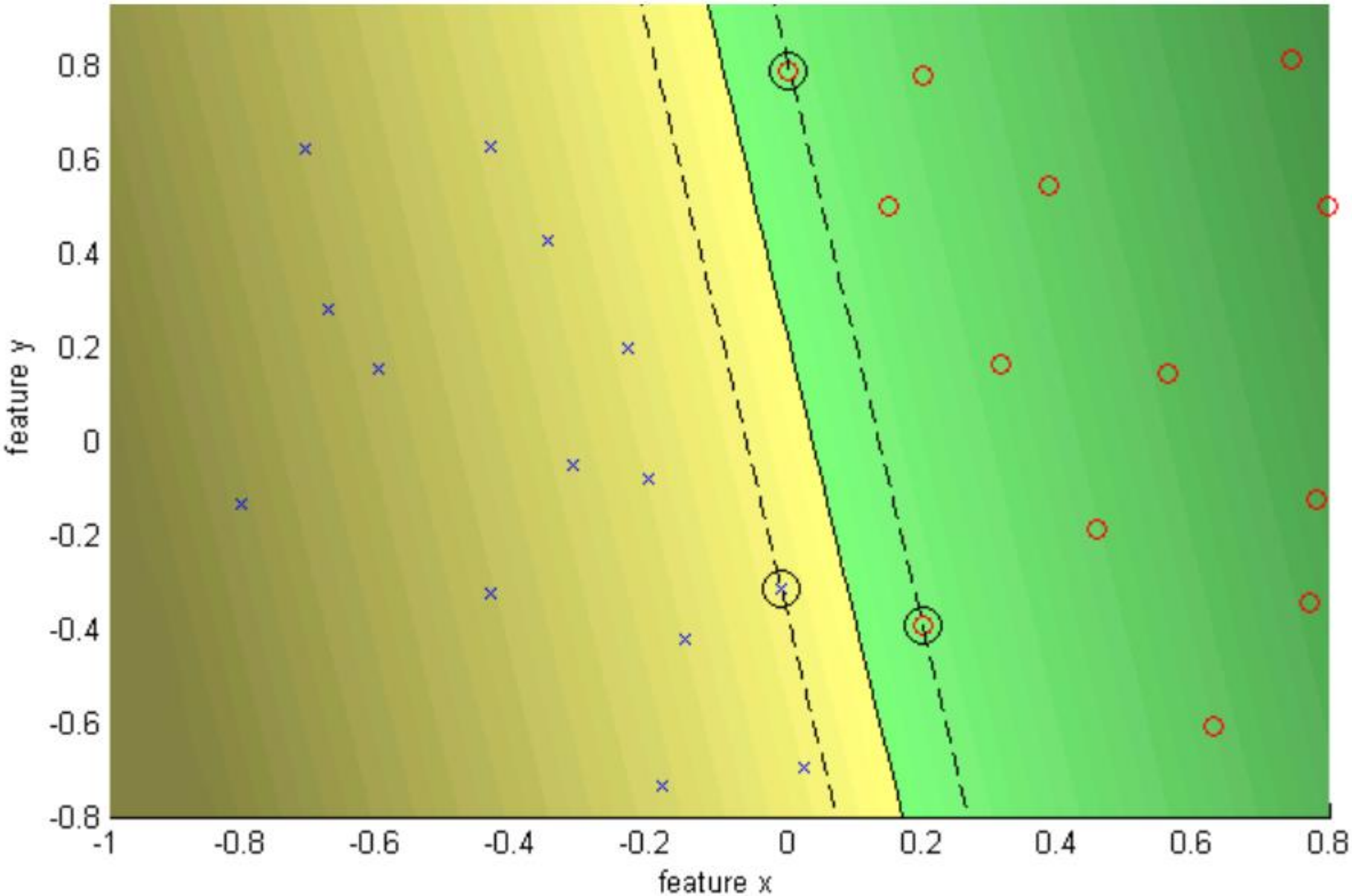


Do not necessarily want complete correct characterization

Two Very Similar Problems



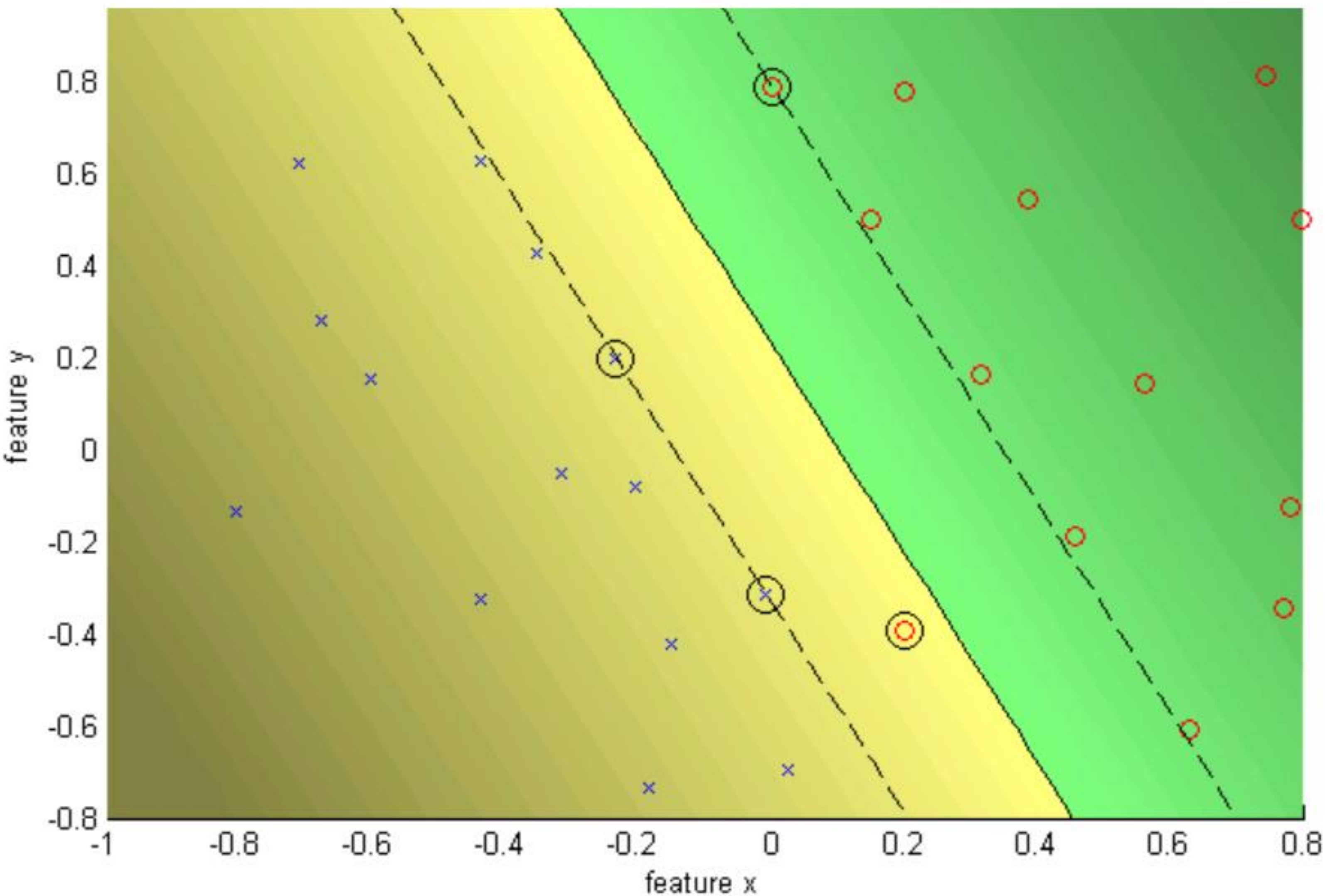
Hard Margin (C = Infinity)



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

Wider margin, we have some misclassifications (potential outlier) but a more general hyperplane.

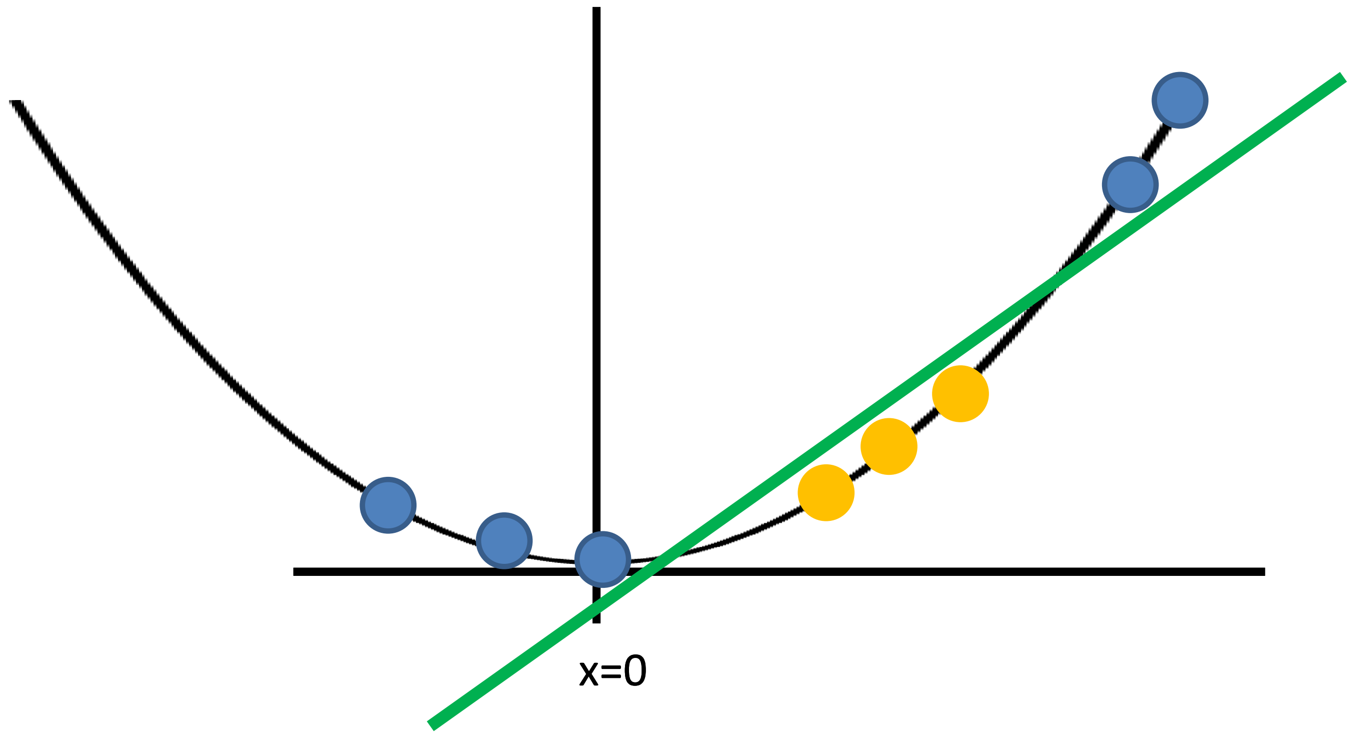
Soft Margin (C = 10)



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

XOR problem revised

increase dimension: x^2 vs x.

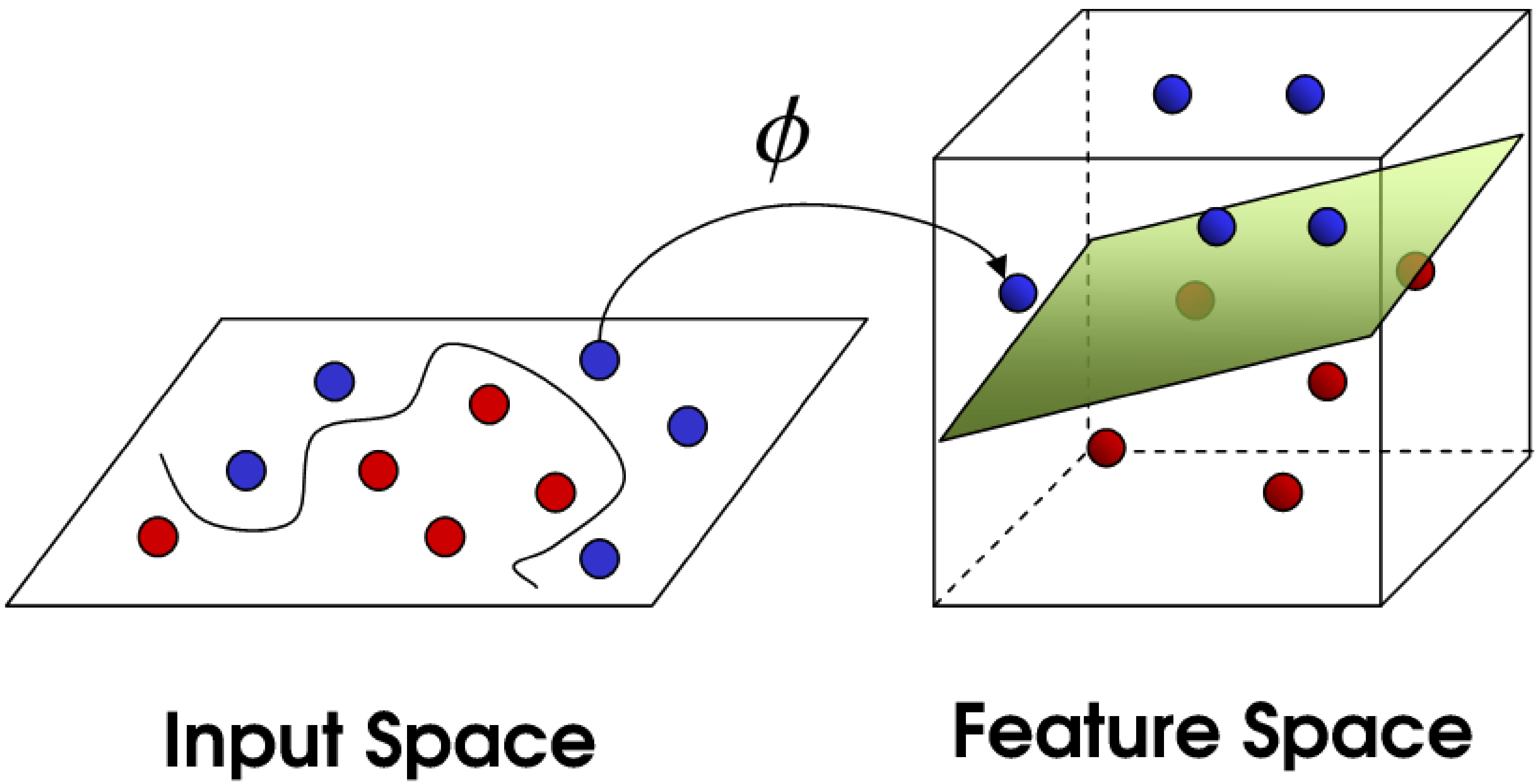


Did we add information to make the problem seperable?

No info added.

How many dimensions up? unlimited, but then curse of dimensionality …
Do cross validation to find number of dimensions you care about.

Non-Linear Decision Boundary



Can get non-linear decision boundary by projecting linearly computed linear hyperplane down to original space.

SVM with a polynomial Kernel visualization

Created by:
Udi Aharoni

Quadratic Kernel

$$x = (x_1, x_2)$$ 2D point

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

dot product: pay with computational cost @ higher dimensions

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

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

Need to tune degree of the kernel.

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

Kernel trick: no phi(x)
The result is this formula here.

Kernel Functions

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

- Polynomial:

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

s is degree of kernel.

- Radial basis function (RBF):

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

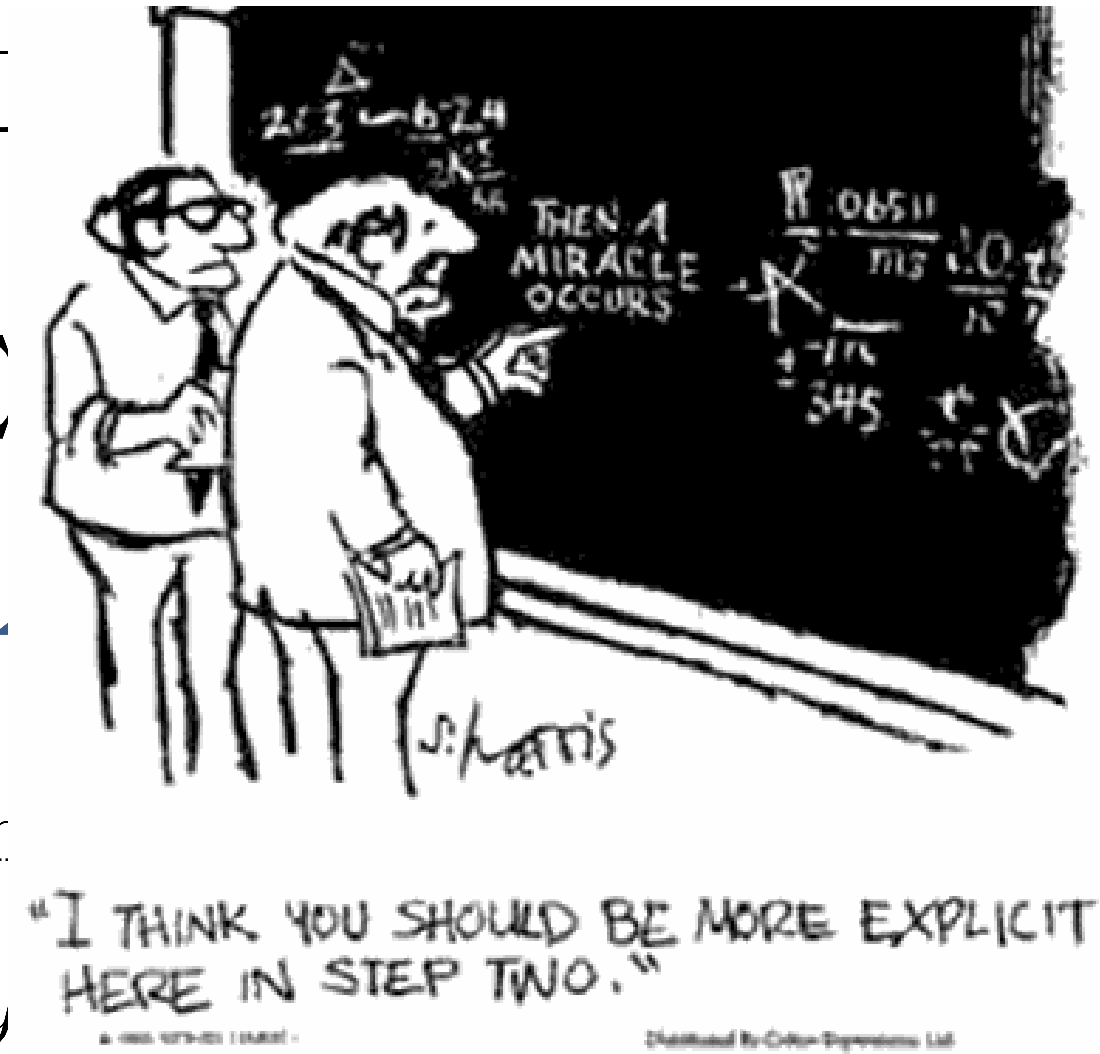
Need to tune gamma for SVM

So what is the excitement?

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

s.t. α_i

$\sum$



$\arg\mathrm{r}$

s.t. y

So what is the excitement?

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

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

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

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

Computer whole SVM in high dimensions with only Kernel —> no computational cost.

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

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

Prediction

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

only need support vectors dot product

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

The Miracle Explained

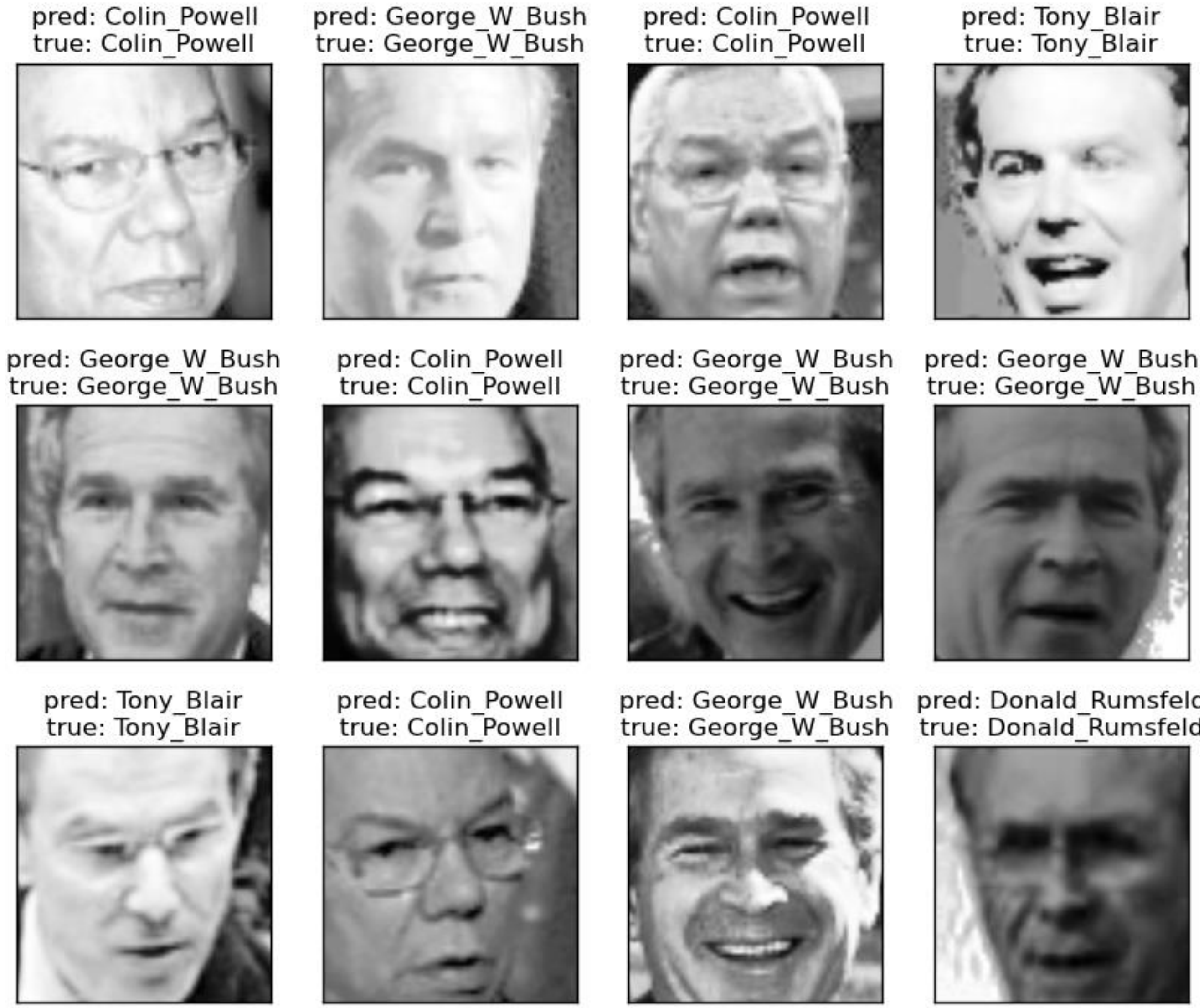
- Andrew Ng does this really well

- 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

Face Recognition



Face Recognition

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

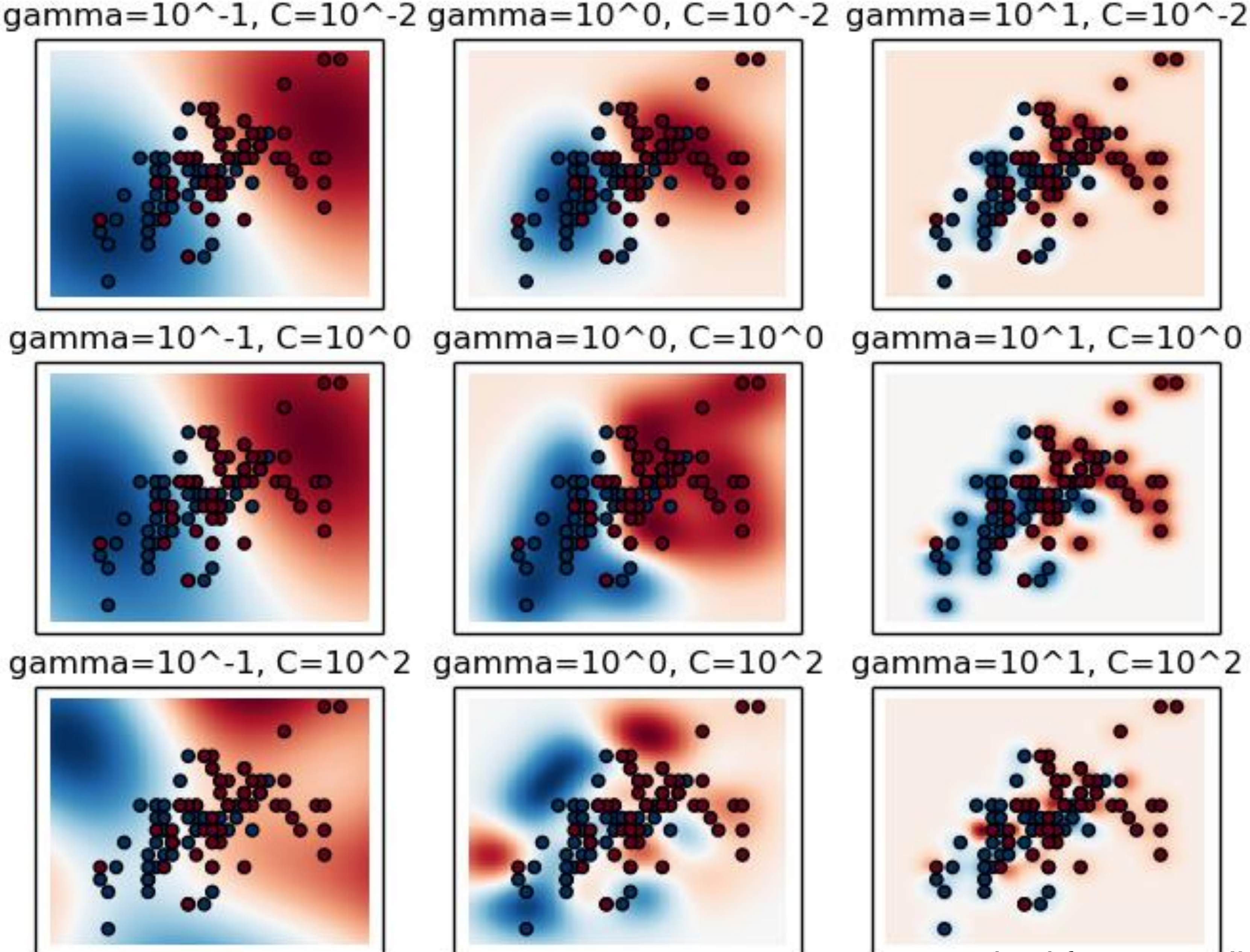
- Red are cross validation steps

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)

https://www.youtube.com/watch?v=cxHMgl2_5zg larger gamma: larger variation in decision boundary.



bad for generalization

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

Tips and Tricks

- SVMs are not scale invariant Normalize your data.
- 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!

Need to normalize test set. Should be able to run entire program w/o test data —> TEST COMES AT END.

Tips and Tricks

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

  10^-2,10ˆ-1,10,10^2
- Read:

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

SVM vs KNN

- What are the main key differences?

Keep all training data with KNN, keep ONLY support vectors with SVM
KNN only tune k, SVM tune C and gamma.

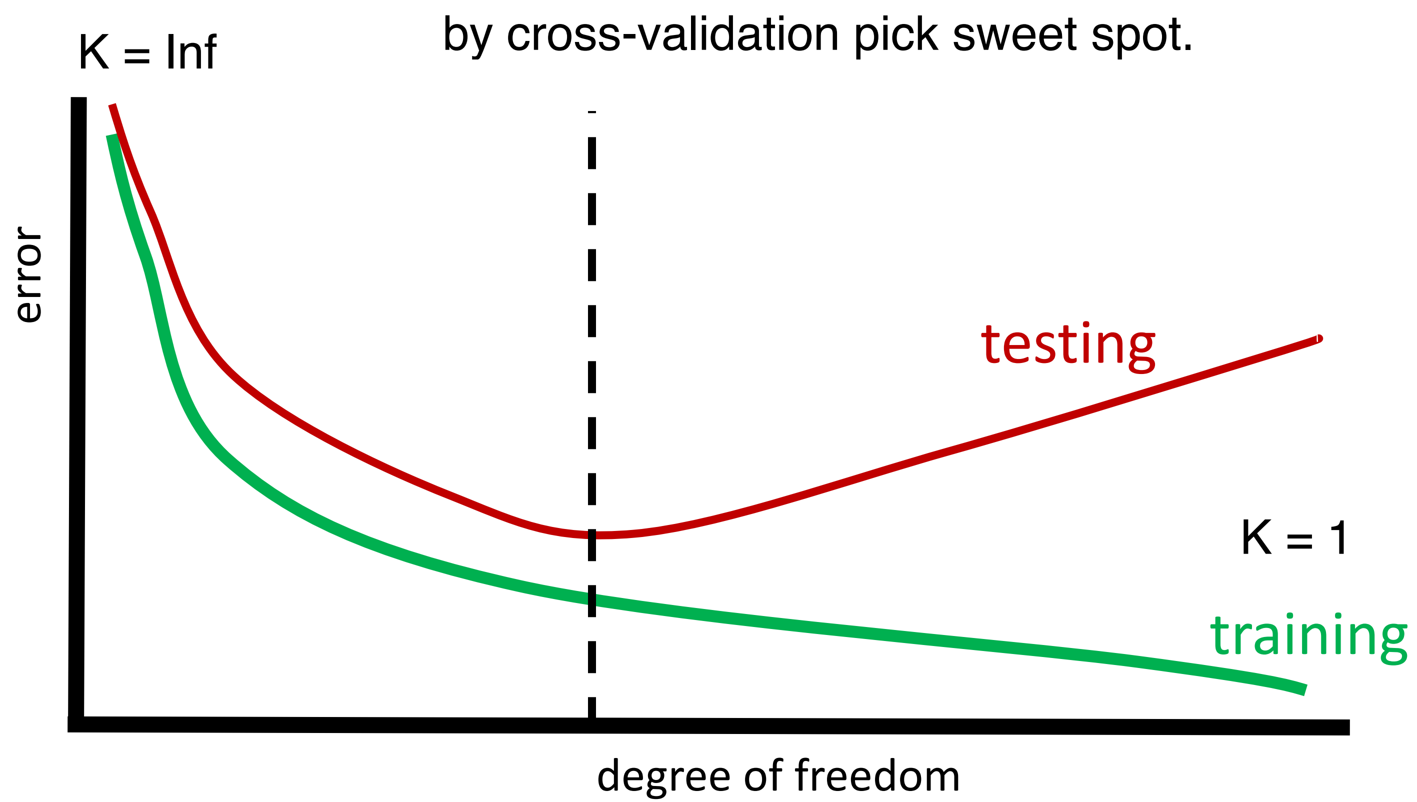
Parameter Tuning

- 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



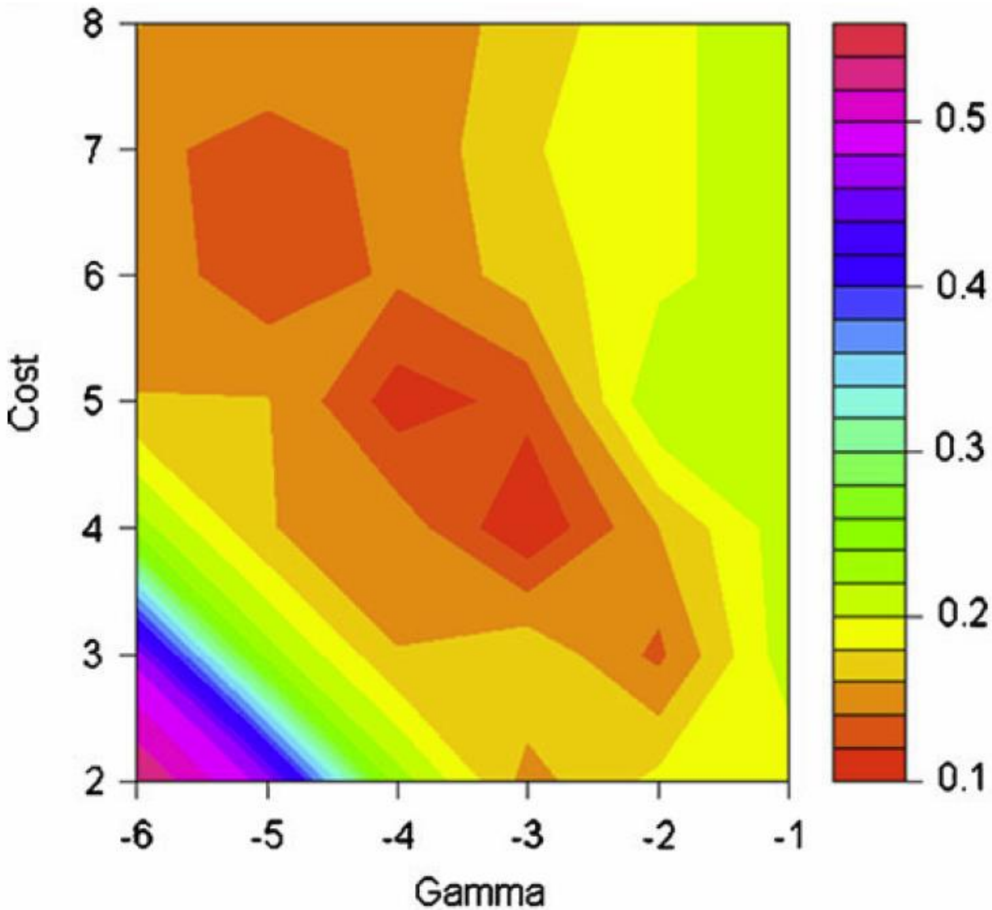
More degrees of freedom: risk overfitting and not generalizable.

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

Compute score (in 3D).

Grid Search

C = Cost



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:

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

everything labeled positive.

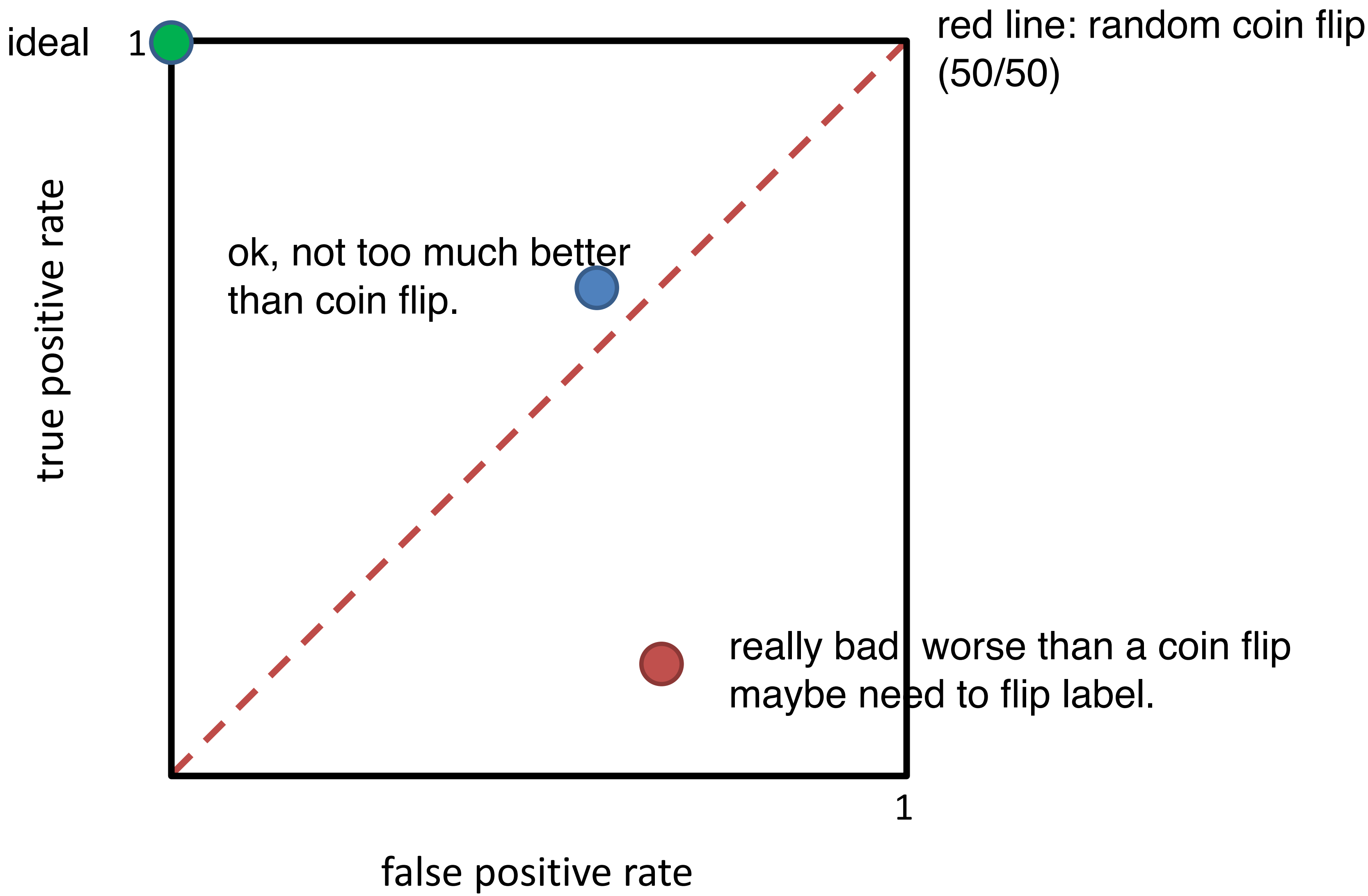
- False Positive Rate:

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

everything labeled negative.

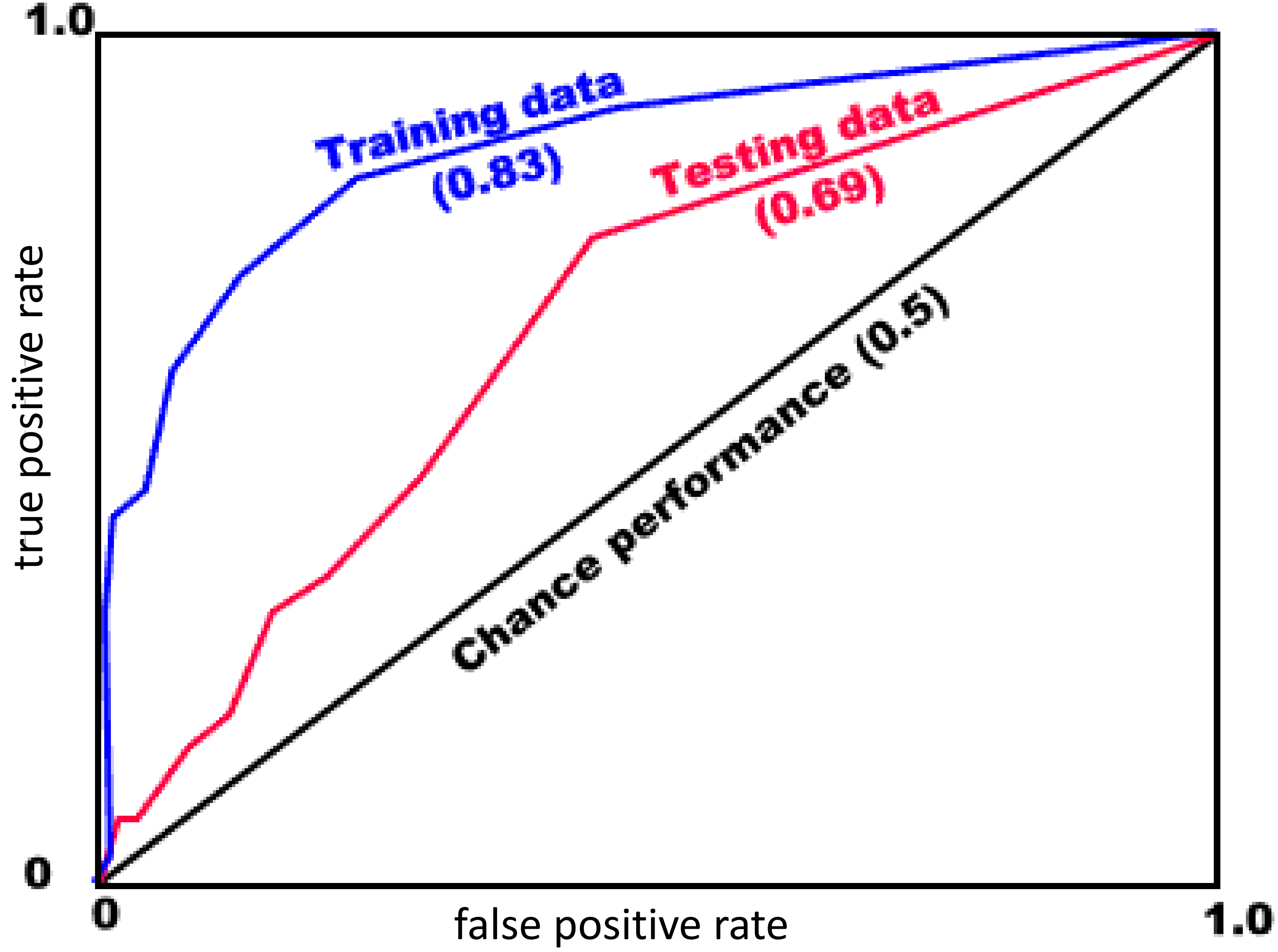
		predicted	
		1	-1
true	1	tp	fn
	-1	fp	tn

Reciever Operating Characteristic



Signal numbers: one number for grid search -> area under curve

ROC Example



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

Precision Recall

- Recall: $\dfrac{tp}{tp+fn}$ equivalent of true positive rate.

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

predicted

true		1	-1
	1	tp	fn
	-1	fp	tn

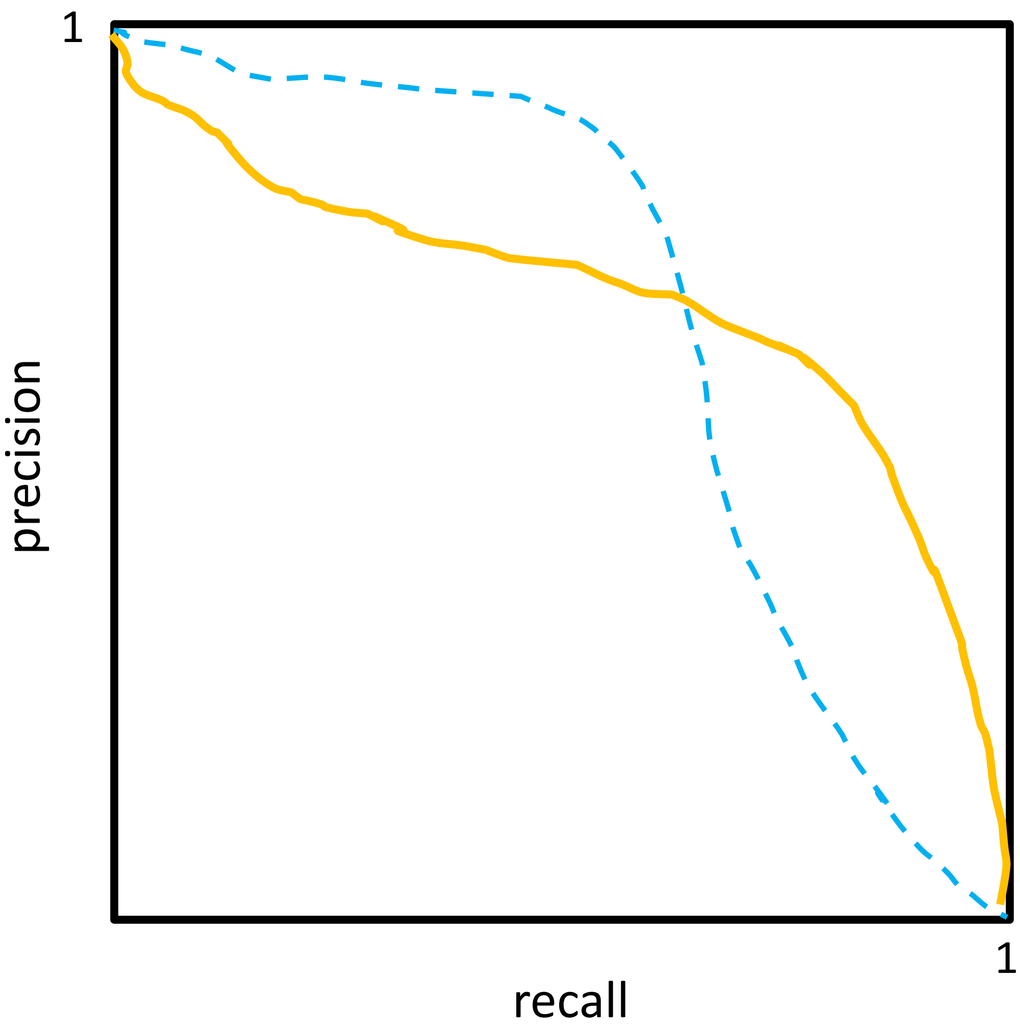
imbalanced data set: ie positive small % of the sample.
w/ ROX, get good values because you have a lot of false negative (background)

Precision Recall

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

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

Precision Recall Curve



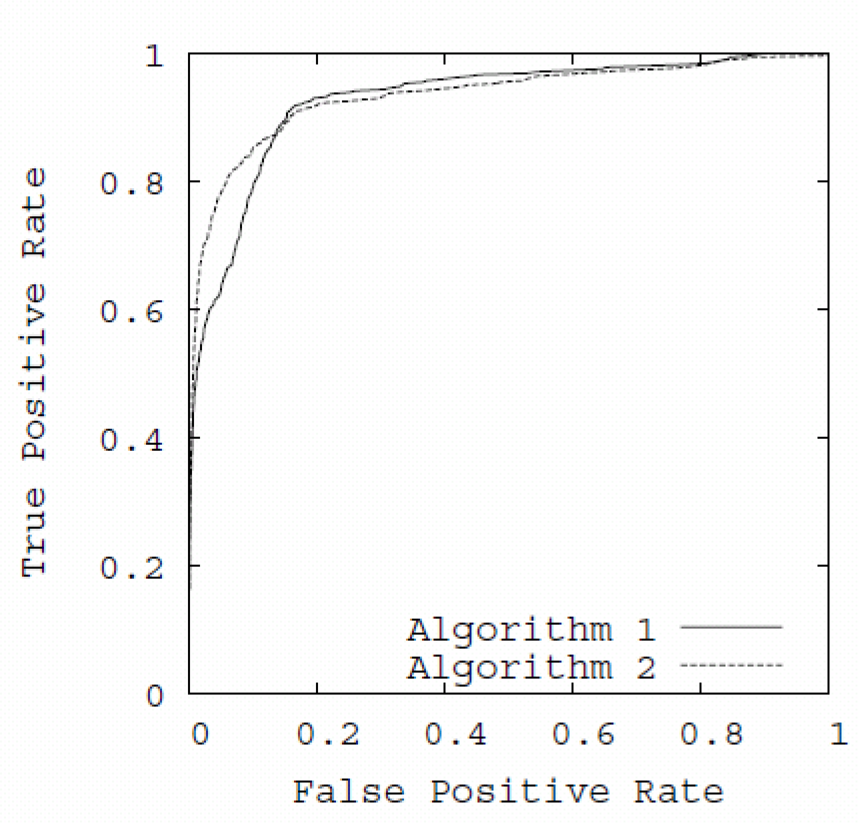
want to be in upper right
want both precession and recall to be 1.

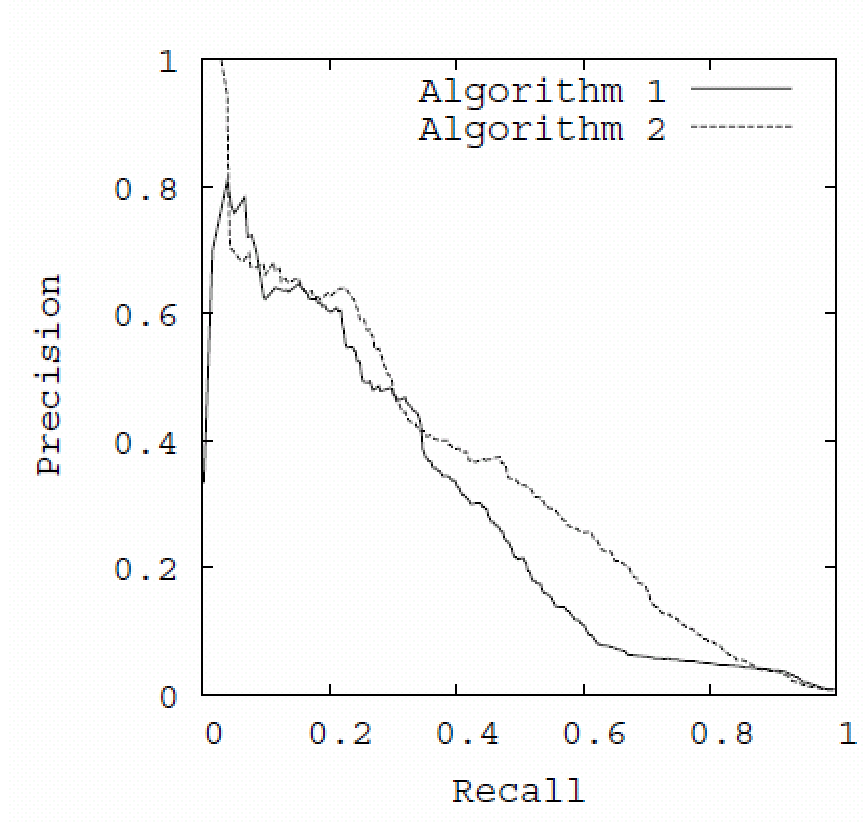
Choose based on situation
do we want more preceision or more recall

ROC allows for same comparison between classifier.

Comparison

Makes it seem that it's better than precesion-recall





ROC

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

F-measure

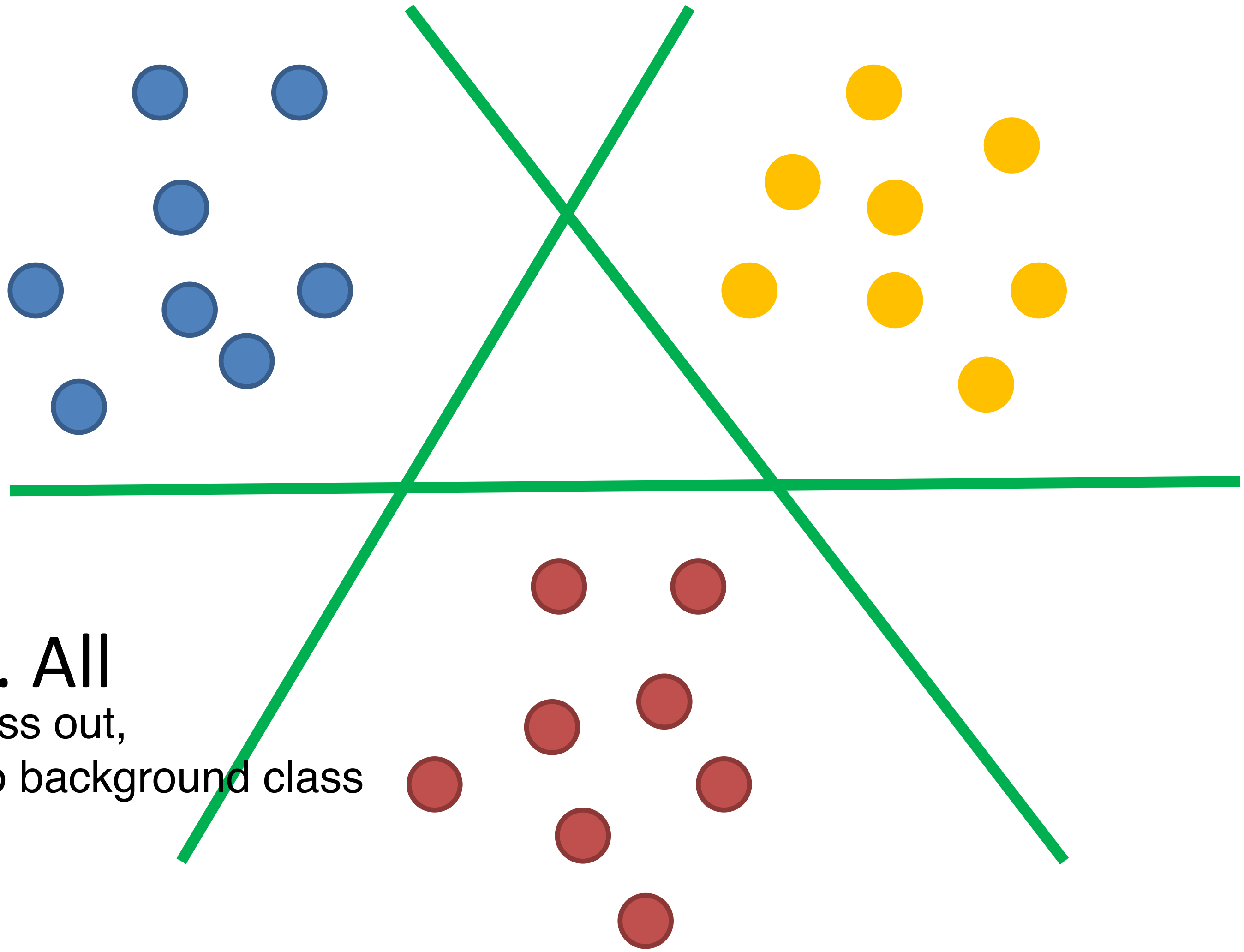
- Weighted average of precision and recall

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

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

Multi Class

Train 3 SVM:
blue vs all
red vs all
yellow vs all.



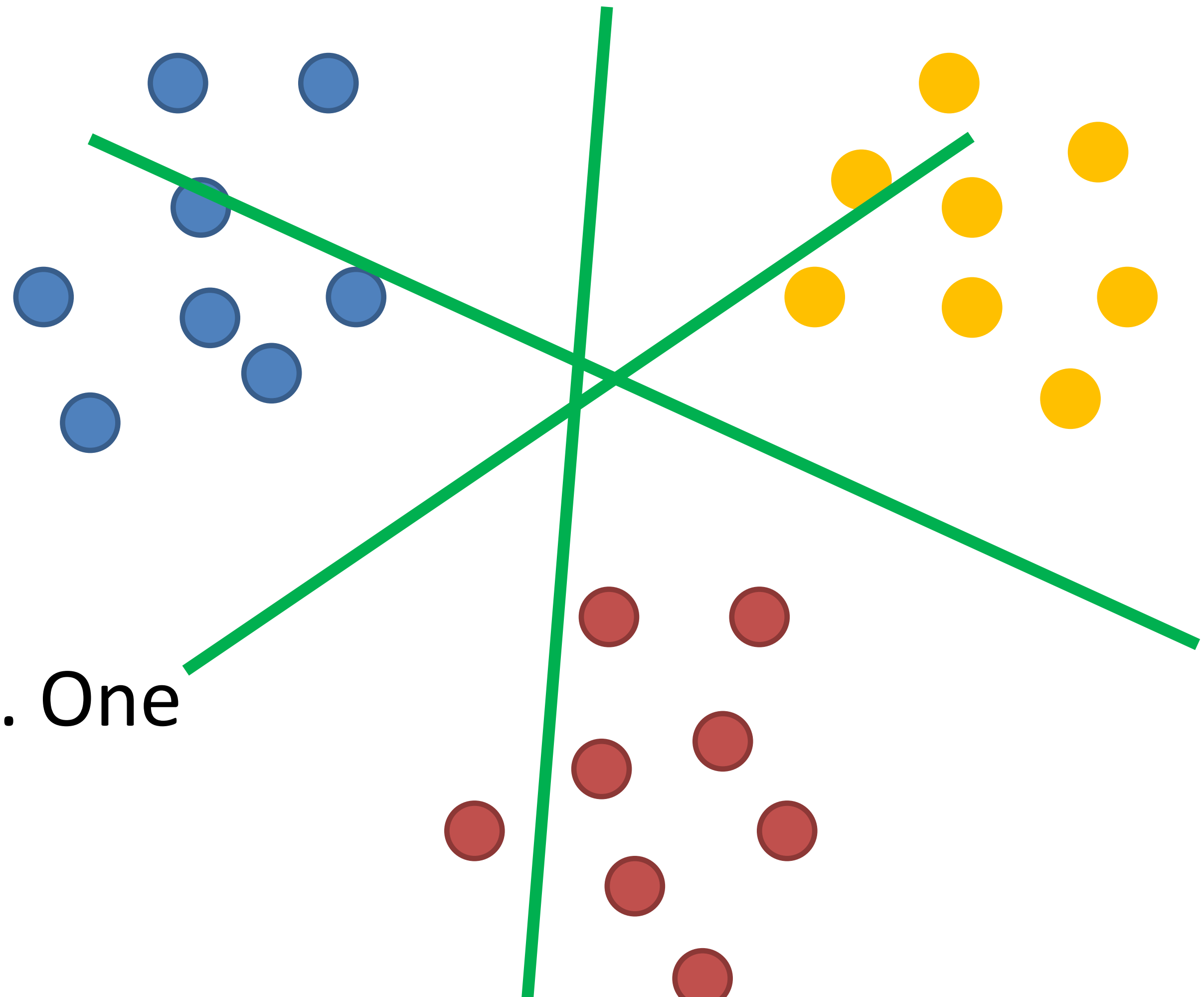
One vs. All

pick one class out,
set others to background class

One vs All

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

Multi Class



One vs. One

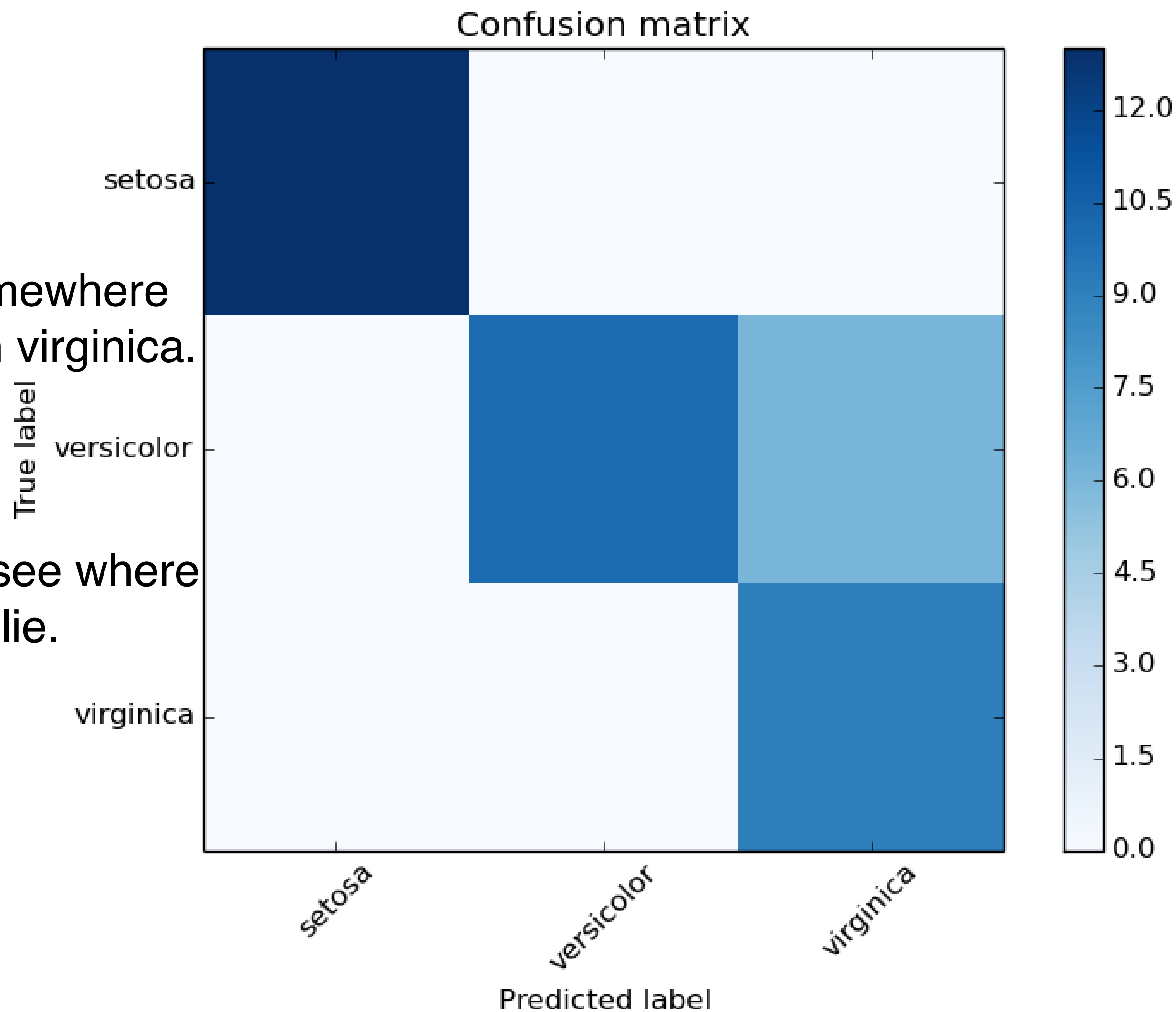
Do pairwise

One vs One

- Train n(n-1)/2 classifiers
- Take majority vote
- Fast training

want diagonal values high (darker color in this graph)

Confusion Matrix



versicolor somewhere confused with virginica.

Can actually see where the problems lie.

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