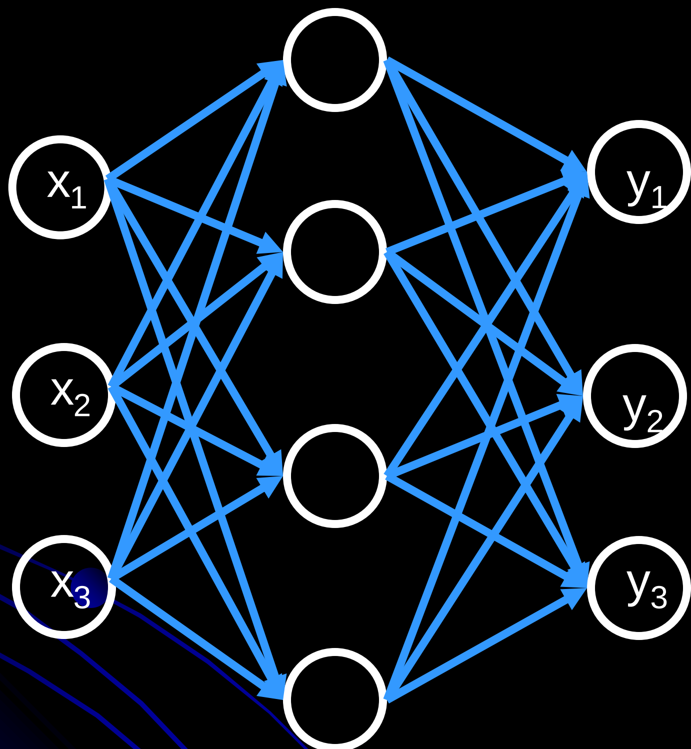
CSSE463: Image Recognition Day 14

- Lab due Weds, 11:59.

- This week:
  - Monday: Neural networks
  - Tuesday: SVM Introduction and derivation
  - Thursday: Project info, SVM demo
  - Friday: SVM lab

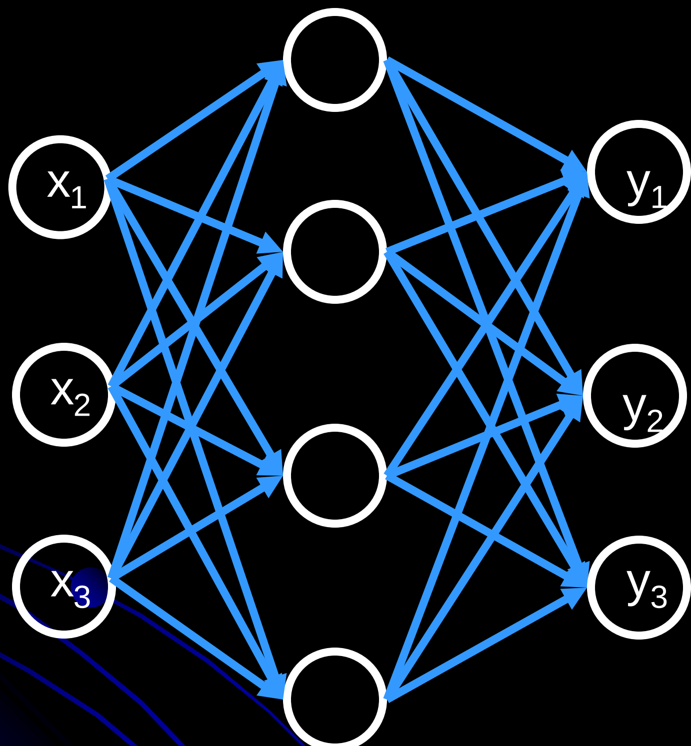
Multilayer feedforward neural nets



- Many perceptrons
- Organized into layers
  - Input (sensory) layer
  - Hidden layer(s): 2 proven sufficient to model any arbitrary function
  - Output (classification) layer
- Powerful!
- Calculates functions of input, maps to output layers
- Example

Sensory (HSV) | Hidden (functions) | Classification (apple/orange/banana)

Backpropagation algorithm



Initialize all weights randomly

- For each labeled example:
  - Calculate output using current network
  - Update weights across network, from output to input, using Hebbian learning
- Iterate until convergence
  - Epsilon decreases at every iteration
- Matlab does this for you. ☺
- neuralNetDemo.m

a. Calculate output (feedforward) →

Repeat

b. Update weights (feedback) ←

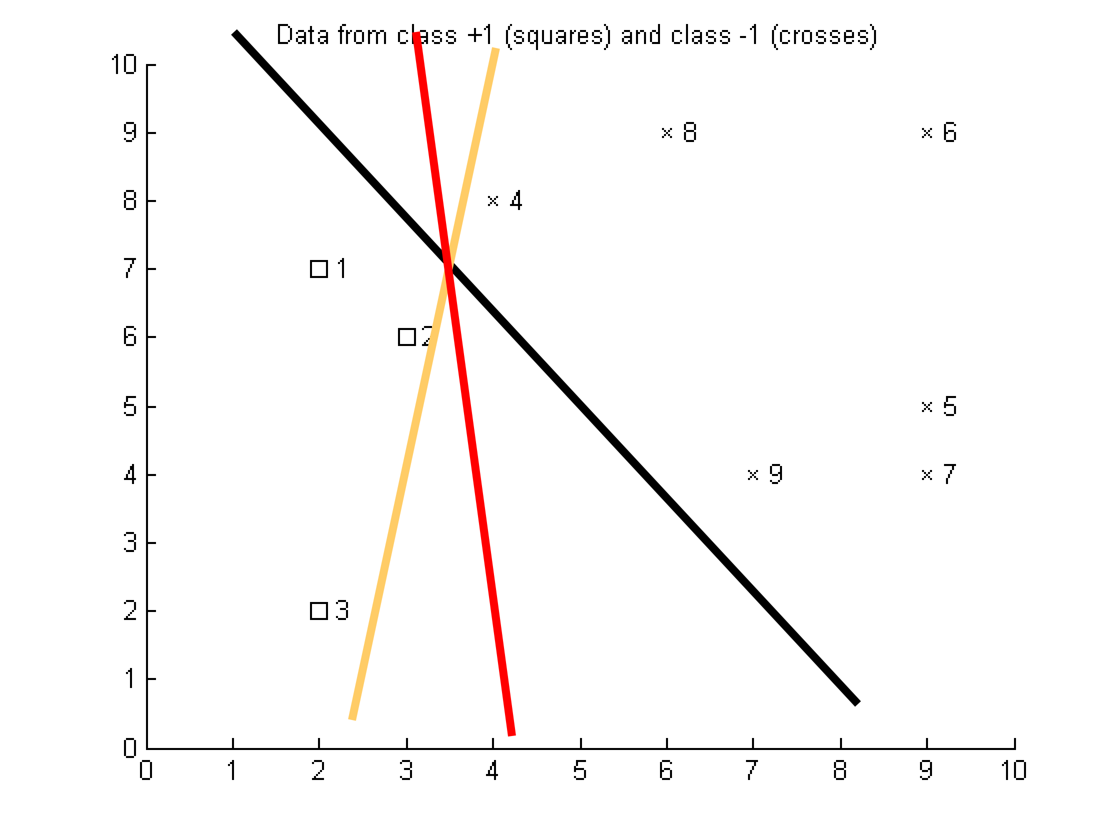
Parameters

- Most networks are reasonably robust with respect to learning rate and how weights are initialized
- However, figuring out how to
  - normalize your input
  - determine the architecture of your net
- is a black art. You might need to experiment. One hint:
  - Re-run network with different initial weights and different architectures, and test performance each time on a validation set. Pick best.

References

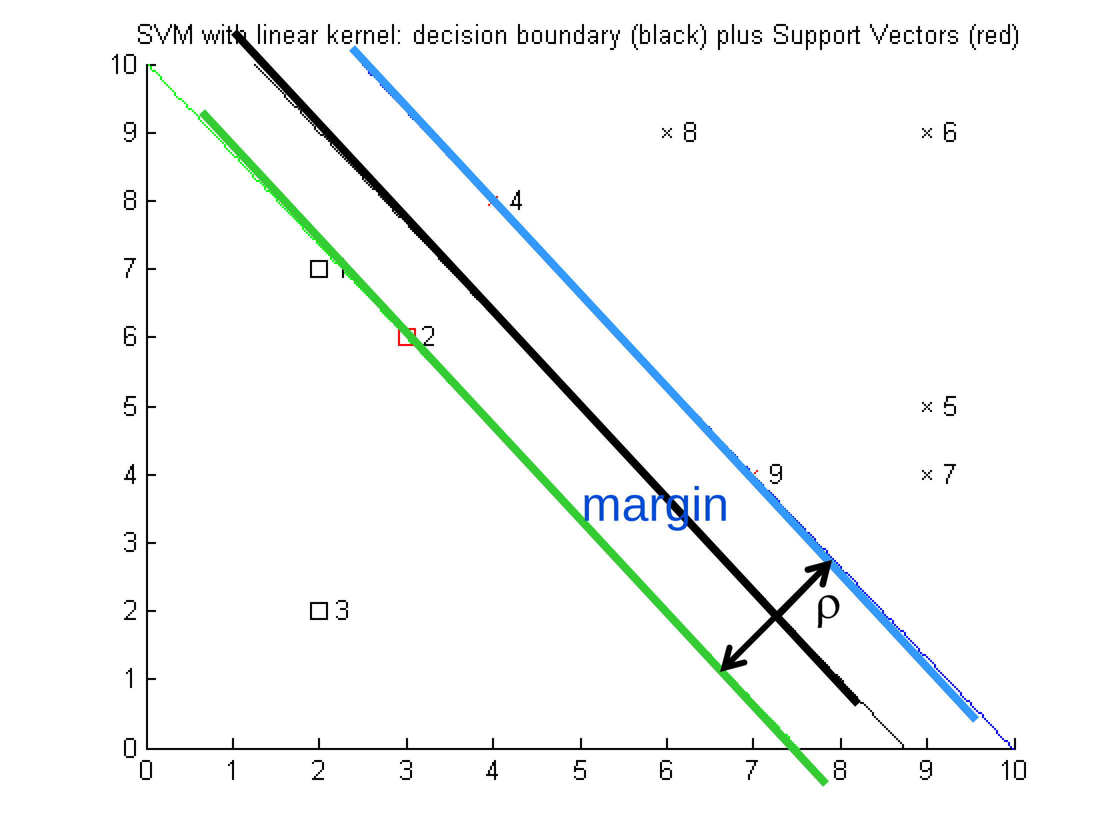
- This is just the tip of the iceberg! See:
  - Sonka, pp. 404-407
  - Laurene Fausett. *Fundamentals of Neural Networks*. Prentice Hall, 1994.
    - Approachable for beginner.
  - C.M. Bishop. *Neural Networks for Pattern Classification*. Oxford University Press, 1995.
    - Technical reference focused on the art of constructing networks (learning rate, # of hidden layers, etc.)
  - Matlab neural net help

SVMs: "Best" decision boundary



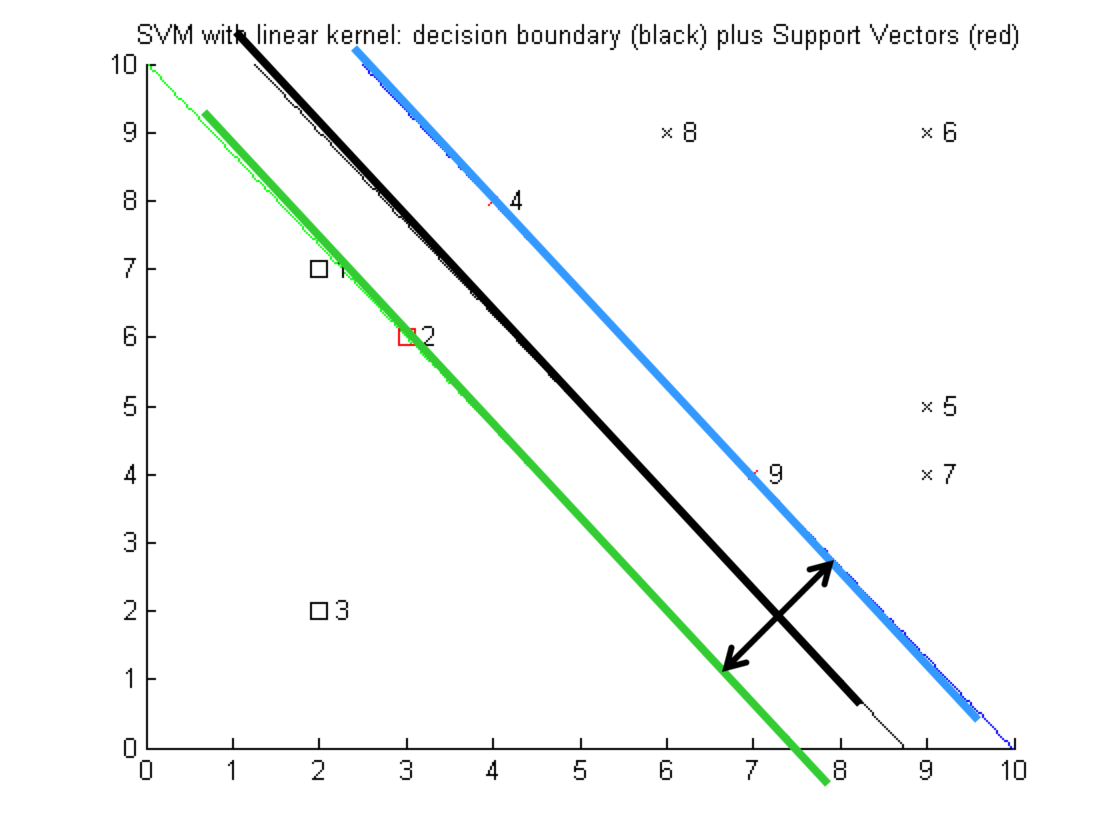
- Consider a 2-class problem
- Start by assuming each class is linearly separable
- There are many separating hyperplanes…
- Which would you choose?

SVMs: "Best" decision boundary



- The "best" hyperplane is the one that *maximizes the margin* between the classes.
- Some training points will always lie on the margin
  - These are called *"support vectors"*
  - #2,4,9 to the left
- Why does this name make sense intuitively?

Support vectors



- The support vectors are the toughest to classify
- What would happen to the decision boundary if we moved one of them, say #4?
- A different margin would have maximal width!

Problem

- Maximize the margin width
- while classifying all the data points correctly…

Mathematical formulation of the hyperplane

- On paper
- Key ideas:
  - Optimum separating hyperplane:
  - Distance to margin:
  - Can show the margin width =
  - Want to maximize margin

$$w_0^T x + b_0$$

$$g(x) = w_0^T x + b_0$$

$$\rho = \frac{2}{\|w_0\|}$$

Finding the optimal hyperplane

- We need to find w and b that satisfy the system of inequalities:

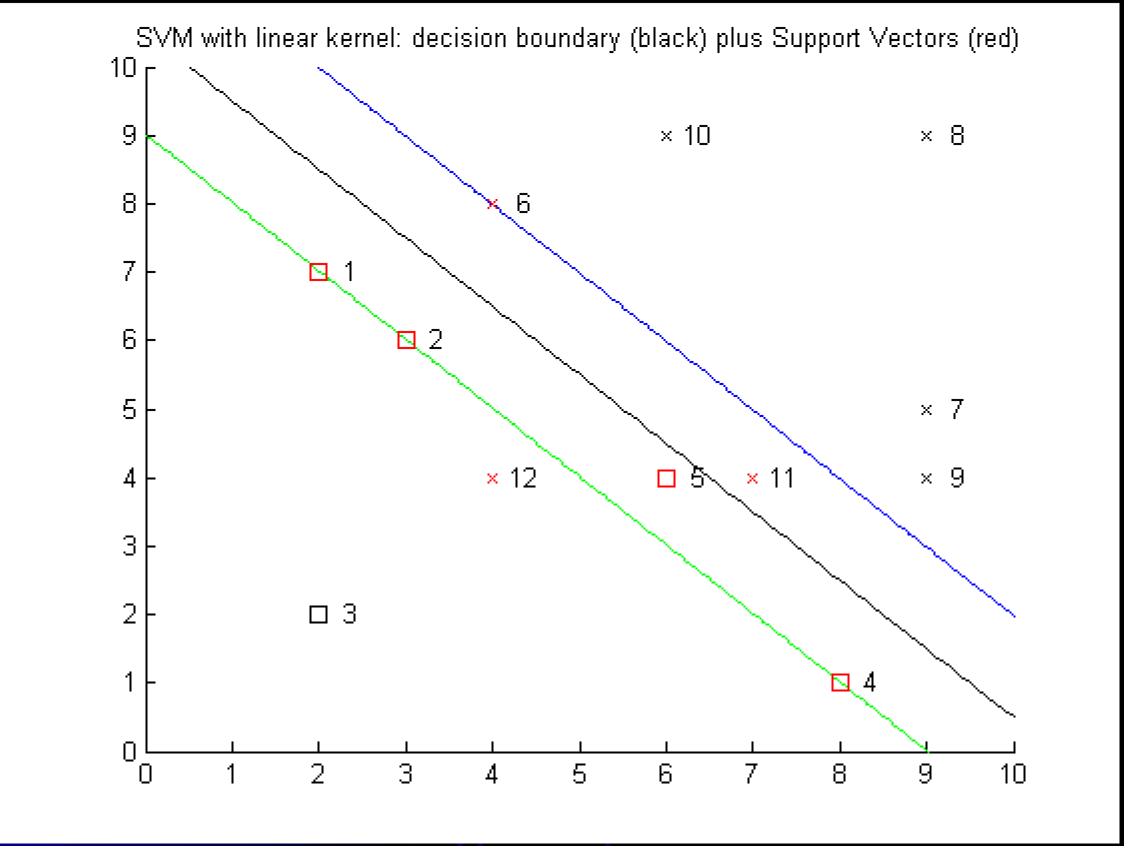
$$d_i(w^T x_i + b) \geq 1 \ for \ i = 1,2,....N$$

- where w minimizes the cost function:

$$\phi(w) = \frac{1}{2} w^T w$$

- (Recall that we want to minimize $||w_0||$, which is equivalent to minimizing $||w_o||^2 = w^T w$)
- Quadratic programming problem
  - Use Lagrange multipliers
  - Switch to the dual of the problem

Non-separable data



- Allow data points to be misclassifed
- But assign a cost to each misclassified point.
- The cost is bounded by the parameter C (which you can set)
- You can set different bounds for each class. Why?
  - Can weigh false positives and false negatives differently

Can we do better?

- Cover's Theorem from information theory says that we can map nonseparable data in the input space to a feature space where the data is separable, with high probability, if:
  - The mapping is nonlinear
  - The feature space has a higher dimension
- The mapping is called a *kernel function*.
- Lots of math would follow here

Most common kernel functions

- Polynomial
- Gaussian Radial-basis function (RBF)
- Two-layer perceptron

$$K(x, x_i) = (x^T x_i + 1)^p$$

$$K(x, x_i) = \exp\left(-\frac{1}{2\sigma^2} \left\| x - x_i \right\|^2 \right)$$

$$K(x, x_i) = \tanh\left(\beta_0 x^T x_i + \beta_1 \right)$$

- You choose p, σ, or β_i
- My experience with real data: **use Gaussian RBF!**

Easy — Difficulty of problem — Hard

p=1, p=2, — higher p — RBF

Q5