# Recognition by finding patterns

- Template matching with correlation (linear filters) is a simple example of recognition by pattern matching
- Some objects behave like quite simple templates
  - Frontal faces

# Recognition by finding patterns

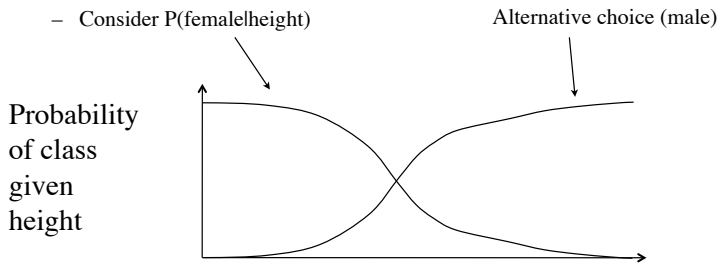
- Example strategy:
  - Find image windows
  - Correct for lighting
  - Pass them to a statistical test (a classifier) that accepts faces and rejects non-faces
  - (Optionally, consider your "reward" function (or "loss" or "risk").

# Basic ideas in classification

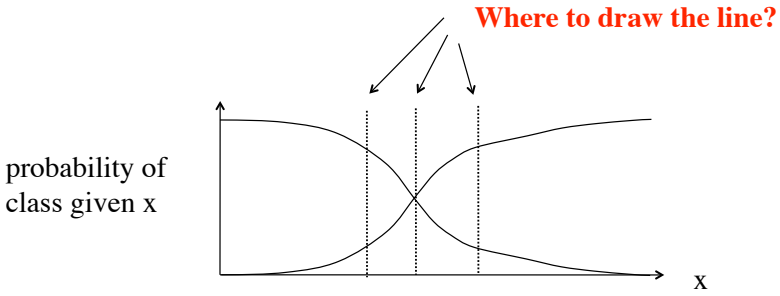
- Concrete example
  - "guess" male / female from height

# Basic ideas in classification

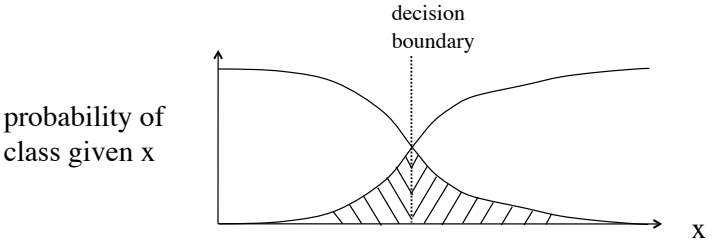
- Concrete example
  - "guess" male / female from height
- Probabilistic approach
  - Consider P(femalelheight)



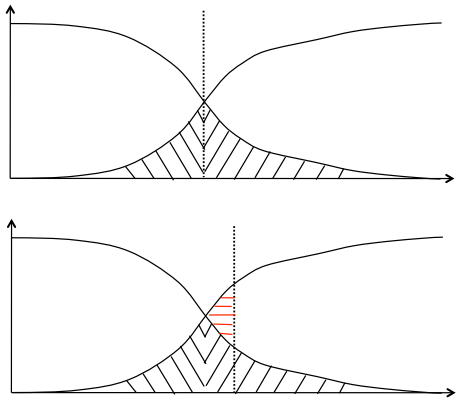
# Basic ideas in classification




# Basic ideas in classification




Area of intersection under curves gives
expected value of making a mistake



Red shows extra
that you get wrong
with different
boundary

# Basic ideas in classification

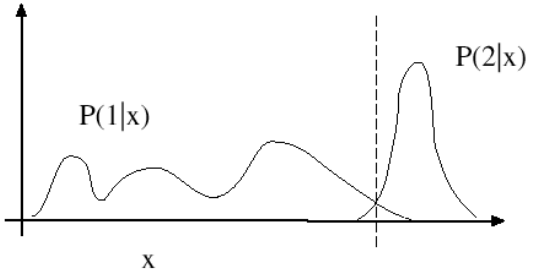
- Concrete example
  - "guess" male / female from height
- Probabilistic approach
  - Consider P(femalelheight)
- Now consider "risk"
  - Suppose you want to give vaccine based on height for a disease that only males get.
  - There is great benefit to males who may be exposed
  - Vaccines have risk as well as benefit
  - Thus there is also some risk to giving females a vaccine they do not need
- This changes the boundary.

Details omitted

# Building classifiers

- Standard scenario
  - Have training data
  - Want to classify new data
- One approach
  - Estimate the probability distributions (we have being thinking about them all along, e.g. P(1|x))
  - Issue: parameter estimates that are "good" may not give optimal classifiers

Finding a decision boundary is not the same as modeling a conditional density.




Important point: P(l|x) can be inaccurate, but the system can work well, as long as the boundary is correct.

Important

# Building classifiers

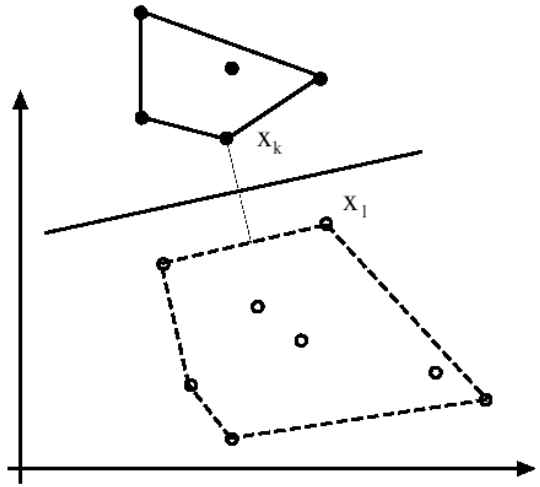
- Standard scenario
  - Have training data
  - Want to classify new data
- One approach
  - Estimate the probability distributions (we have being thinking about them all along, e.g. P(1|x))
  - Issue: parameter estimates that are "good" may not give optimal classifiers
- Another approach
  - Directly go for the boundary

We will consider this one today.

# Support vector machines

- The generic, standard way to do this is with a SVM
- The basic "plug-in classifier" (black box)
- Typically now used for many tasks where before the method of choice was neural networks.
- Very convenient software is now available to do this.
- We will cover the approach briefly

# Support vector machines




# Support vector machines

- If we have a *separating* hyperplane, then if you are on one side $\mathbf{w} \bullet \mathbf{x_i} + b \geq +1$
- If you are on the other side $\mathbf{w} \bullet \mathbf{x_i} + b \leq -1$
- Let $y_i$ be +1 for one class, -1 for the other.

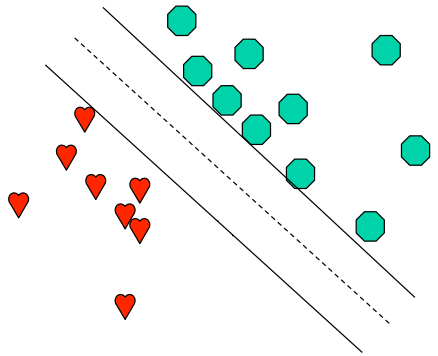
# Support vector machines

- Linearly separable data means that we can chose

$$y_i \left( \boldsymbol{w} \cdot \boldsymbol{x}_i + b \right) \geq 1$$

- Consider the best pair of parallel planes that push against points on the two groups.

# Support vector machines



# Support vector machines

- Consider the best pair of parallel planes that push against points on the two groups.
- The sum of the minimum distances from each group to the other plane can be shown to be:

$$\frac{2}{|\mathbf{w}|}$$

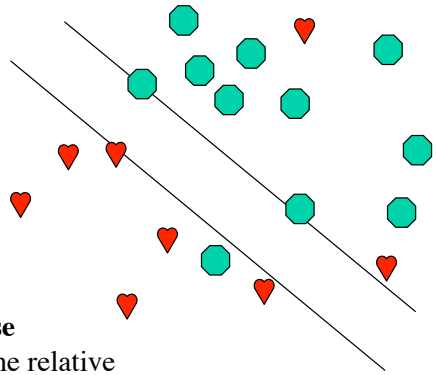
# Support vector machines

- Solved by

$$\begin{aligned} &\text{minimize} && (1/2)\boldsymbol{w}\cdot\boldsymbol{w} \\ &\text{subject to} && y_i\left(\boldsymbol{w}\cdot\boldsymbol{x}_i + b\right) \geq 1 \end{aligned}$$

- (See book, section 22.5 for how to solve it)
- What if the data is not linearly separable
  - Find "best" plane (see book)
  - The boundary is determined by a few points (the support vectors)

# Support vector machines

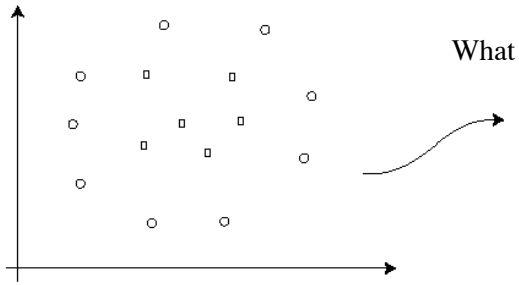


**Non-separable case**
Cost, C, specifies the relative desire to push the planes apart, verses the number of mistakes.

# Support vector machines

- We have found the "best" plane from labeled training data
- How do we classify a new "test" point that has no label
  - Easy---we have a simple formula for determining which side of the plane we are on!
- Pseudo probabilities can be created from the distance to the plane
- This describes a binary classifier. For more than one class, there are two approaches
  - Multiple one against all
  - All against all, and a consensus measure

# Support vector machines (kernel tricks)



What about this case?

# Support vector machines (kernel tricks)

Key observation: The SVM is completely a function of dot products between the vectors.

This means that we can get a non-linear SVM by using a different form of the dot product, K(**x**,**y**).

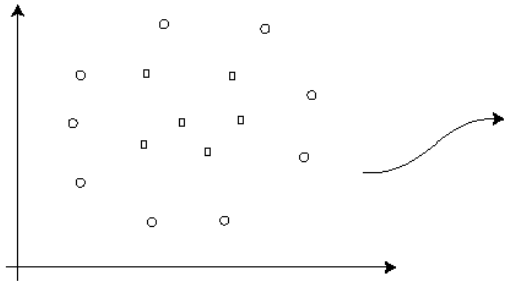
This is equivalent to a linear classification in a much higher dimensional space.

# Support vector machines (kernel tricks)

For example, we can produce a higher dimensional space using polynomials = of the original points, e.g.,

$$(x,y) \rightarrow \left(x^2, xy, y^2, x, y\right) = \left(u_0, u_1, u_2, u_3, u_4\right)$$

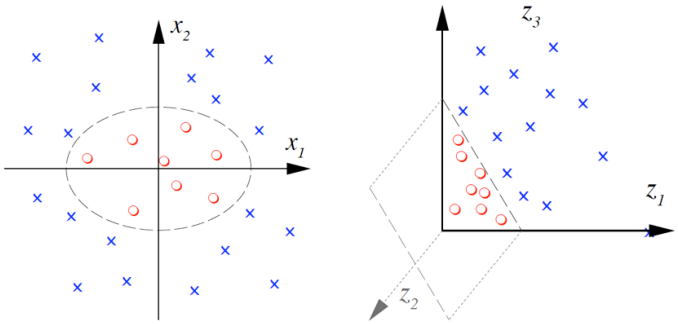
# Support vector machines (kernel tricks)



Can you imagine a 3D space in which the obvious decision boundary is linear?

# A similar example (Originally from Schölkopf and A. J. Smola)

$$\Phi : R^2 \to R^3$$
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{(2)} x_1 x_2, x_2^2)$$




# Testing classifiers

- Standard method is to use Cross-Validation
- Test classification accuracy on data not used in training
- Test generalizability by using data that is progressively different than training data
  - new experiment
  - different camera
  - different researchers

Important

# Classification tutorial summary

- The SVM classifier takes labeled training data and provides a hyper-plane that separates the data into classes
  - The cost parameter controls the trade off between mistakes and separation
- A kernel mapping can enables non-planar boundaries that are hyper-planes in a higher dimension space.
- We classify by testing which side of the hyper-plane we are on.
- We evaluate the robustness of our classifiers using cross validation (testing on held out data).

Important