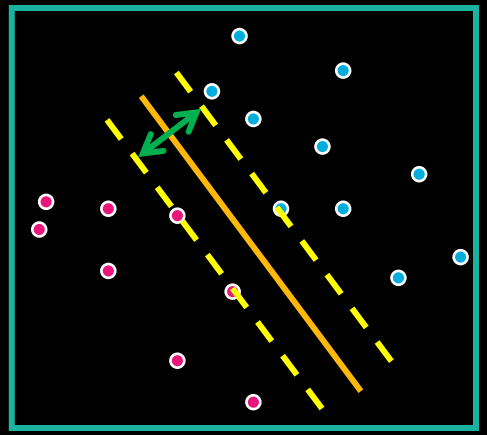
CS4495/6495
Introduction to Computer Vision

8C-L3 *Support Vector Machines*



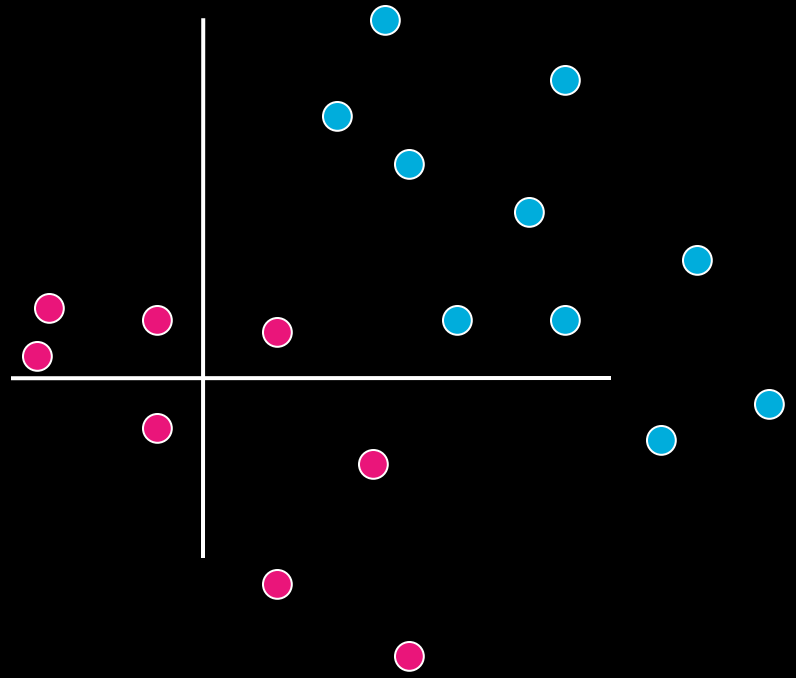
Discriminative classifiers

Discriminative classifiers – find a division (surface) in feature space that separates the classes

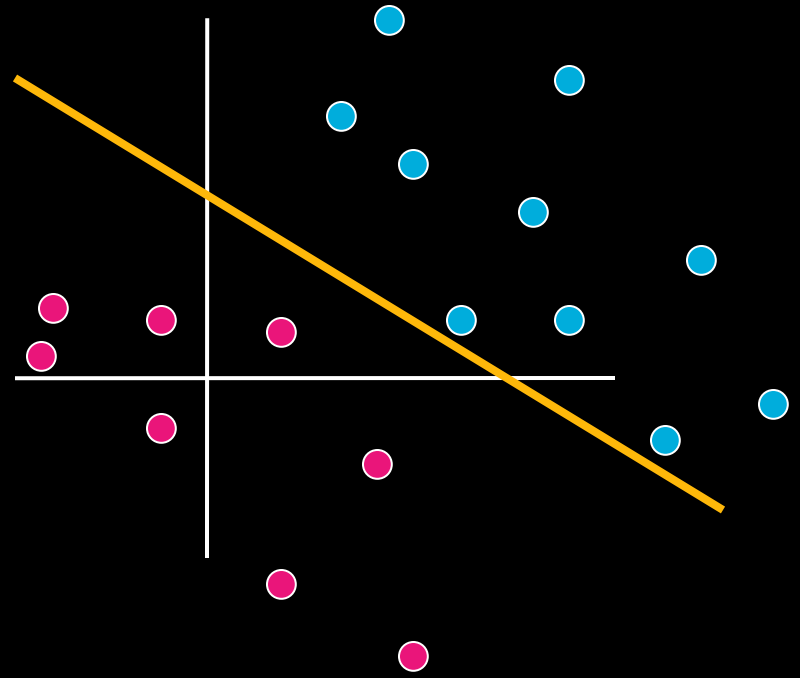
Several methods

- Nearest neighbors
- Boosting
- Support Vector Machines

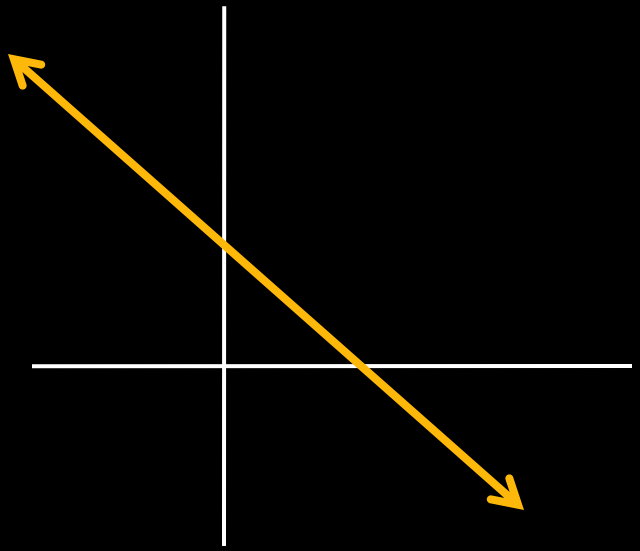
Linear classifiers



Linear classifiers



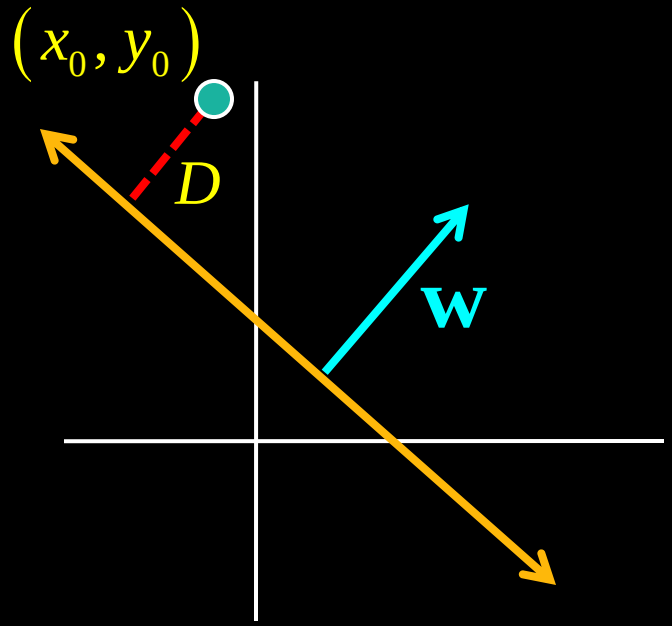
Lines in R^2



Let $\mathbf{w} = \begin{bmatrix} p \\ q \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$px + qy + b = 0$$

Lines in R^2



Let $\mathbf{w} = \begin{bmatrix} p \\ q \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$px + qy + b = 0$$

$$\updownarrow$$

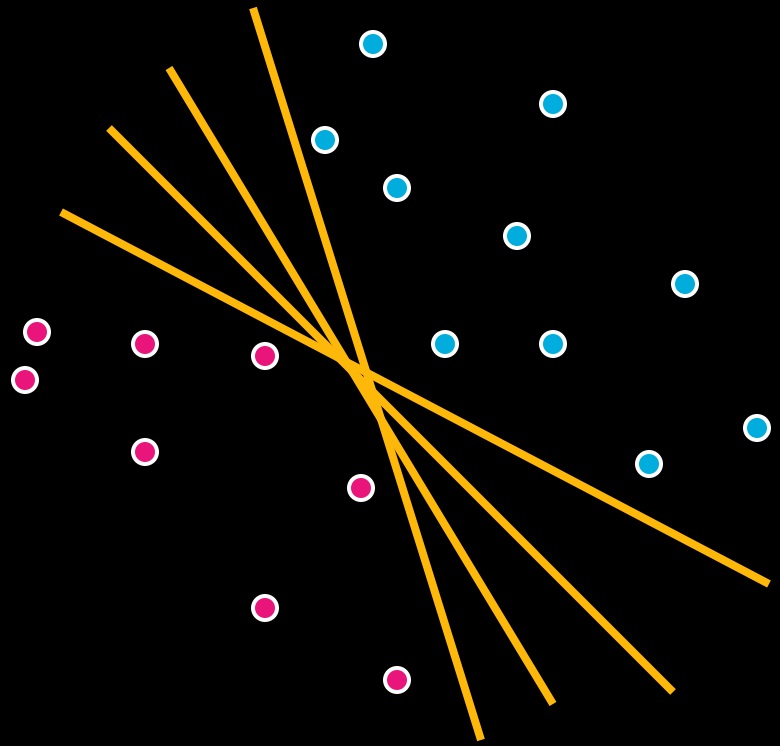
$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

Distance from point to line

$$D = \frac{|px_0 + qy_0 + b|}{\sqrt{p^2 + q^2}} = \frac{\mathbf{w}^{\mathrm{T}}\mathbf{x} + b}{|\mathbf{w}|}$$

Linear classifiers

Find linear function to separate positive and negative examples



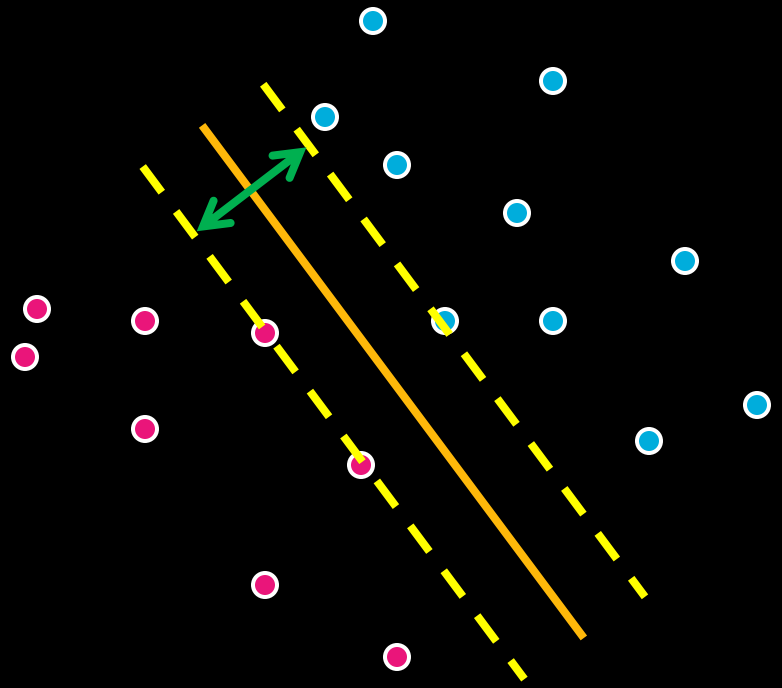
$\mathbf{x}_i$ positive : $\mathbf{x}_i \cdot \mathbf{w} + b \geq 0$

$\mathbf{x}_i$ negative : $\mathbf{x}_i \cdot \mathbf{w} + b < 0$

Which line is best?

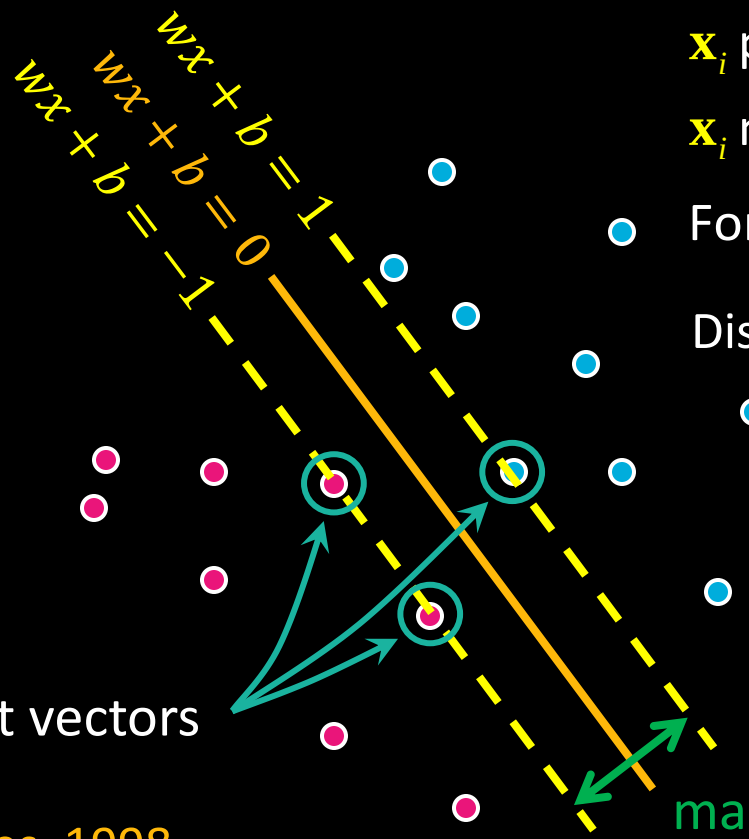
Support Vector Machines (SVMs)

Discriminative classifier based on *optimal separating line* (2D case)



Maximize the *margin* between the positive and negative training examples

How to maximize the *margin*?



$\mathbf{x}_i$ positive $(y_i = 1)$: $\mathbf{x}_i \cdot \mathbf{w} + b \geq 1$

$\mathbf{x}_i$ negative $(y_i = -1)$: $\mathbf{x}_i \cdot \mathbf{w} + b \leq -1$

For support vectors, $\mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$

Dist. between point and line: $\dfrac{|\mathbf{x}_i \cdot \mathbf{w} + b|}{\|\mathbf{w}\|}$

For support vectors: $\dfrac{\mathbf{w}^{\mathrm{T}}\mathbf{x} + b}{\|\mathbf{w}\|} = \dfrac{\pm 1}{\|\mathbf{w}\|}$

$$M = \left| \frac{1}{\|\mathbf{w}\|} - \frac{-1}{\|\mathbf{w}\|} \right| = \boxed{\frac{2}{\|\mathbf{w}\|}}$$

C. Burges, 1998

Finding the maximum margin line

1. Maximize margin $2/\|w\|$
2. Correctly classify all training data points:

$\mathbf{x}_i$ positive $(y_i = 1)$: $\mathbf{x}_i \cdot \mathbf{w} + b \geq 1$

$\mathbf{x}_i$ negative $(y_i = -1)$: $\mathbf{x}_i \cdot \mathbf{w} + b \leq -1$

3. Quadratic optimization problem:

Minimize $\frac{1}{2}\mathbf{w}^T\mathbf{w}$

Subject to $\mathbf{y}_i(\mathbf{x}_i \cdot \mathbf{w} + b) \geq 1$

C. Burges, 1998

Finding the maximum margin line

Solution: $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$

The weights α_i are non-zero only at support vectors

C. Burges, 1998

Finding the maximum margin line

Solution: $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$

$b = y_i - \mathbf{w} \cdot \mathbf{x}_i$ (*for any support vector*)

$$\mathbf{w} \cdot \mathbf{x} + b = \sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b$$

Classification function:

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

if f(x) < 0, classify as negative

$$= \text{sign}\left(\sum_i \alpha_i \mathbf{x}_i \cdot \mathbf{x} + b\right)$$

if f(x) > 0, classify as positive

Dot product only!

C. Burges, 1998

Questions

- What if the features are not 2d?
- What if the data is not linearly separable?
- What if we have more than just two categories?

Questions

- What if the features are not 2d?
- What if the data is not linearly separable?
- What if we have more than just two categories?

Questions

- What if the features are not 2d?
  - Generalizes to d-dimensions
  - Replace line with “hyperplane”

Person detection with HoG's & linear SVM's

- Map each grid cell in the input window to a histogram counting the gradients per orientation
- Train a linear SVM using training set of pedestrian vs. non-pedestrian windows

Dalal and Triggs, CVPR 2005

Code: http://pascal.inrialpes.fr/soft/olt/

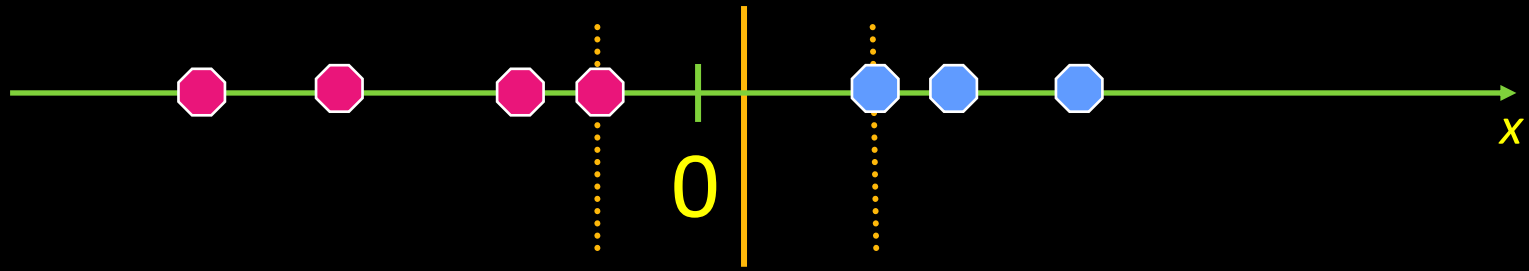

0.62
1.21

Questions

- What if the features are not 2d?
- What if the data is not linearly separable?
- What if we have more than just two categories?

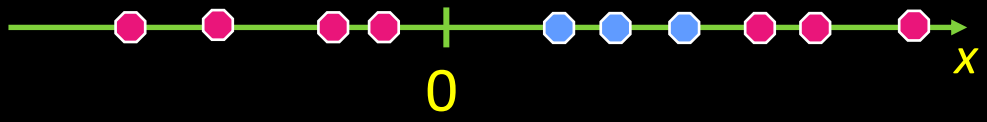
Non-linear SVMs

- Datasets that are linearly separable with some noise work out great:



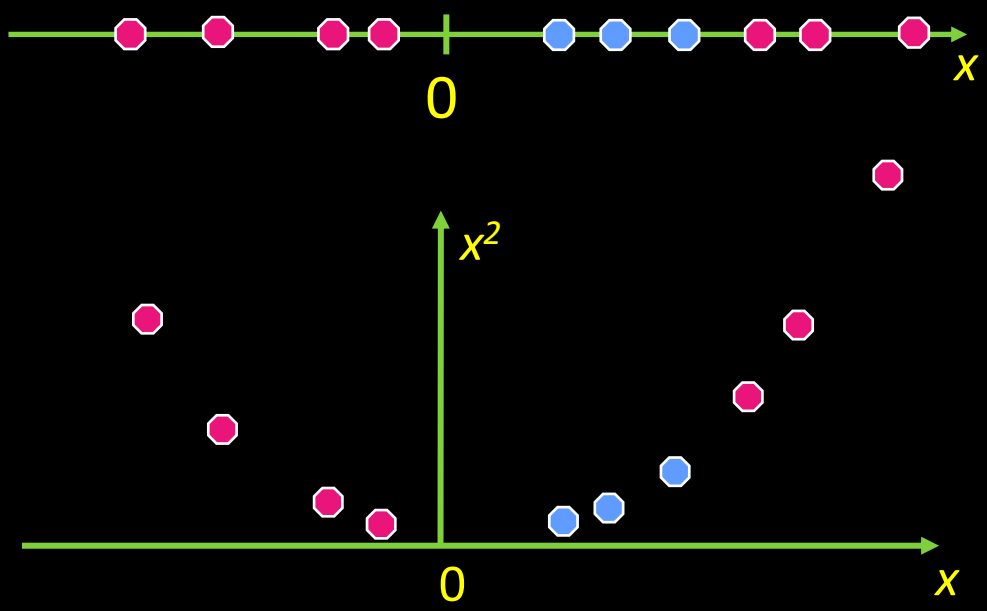
Non-linear SVMs

- But what are we going to do if the dataset is just too hard?



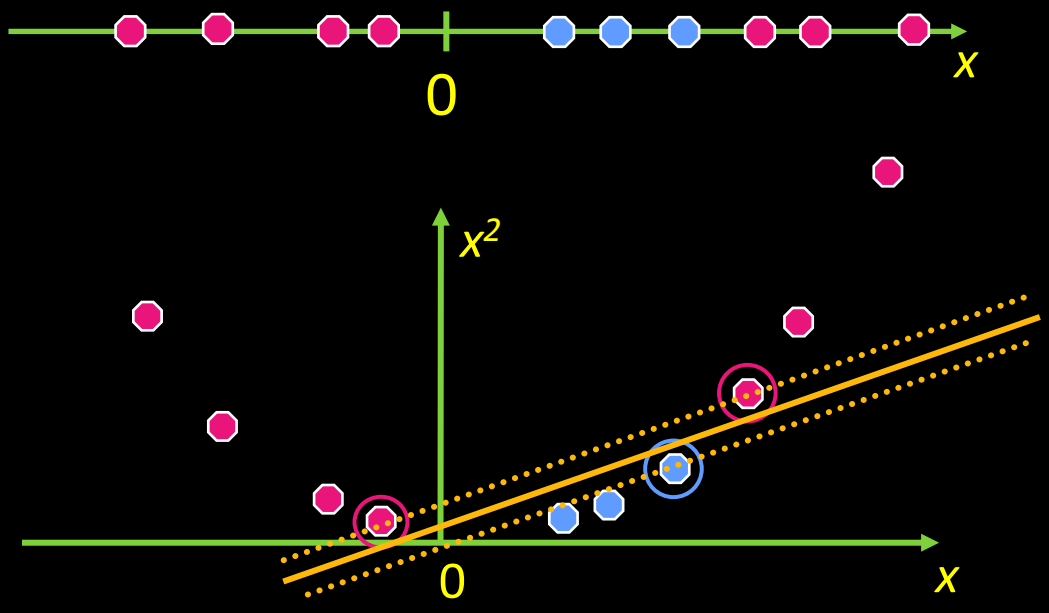
Non-linear SVMs

- How about… mapping data to a higher-dimensional space:



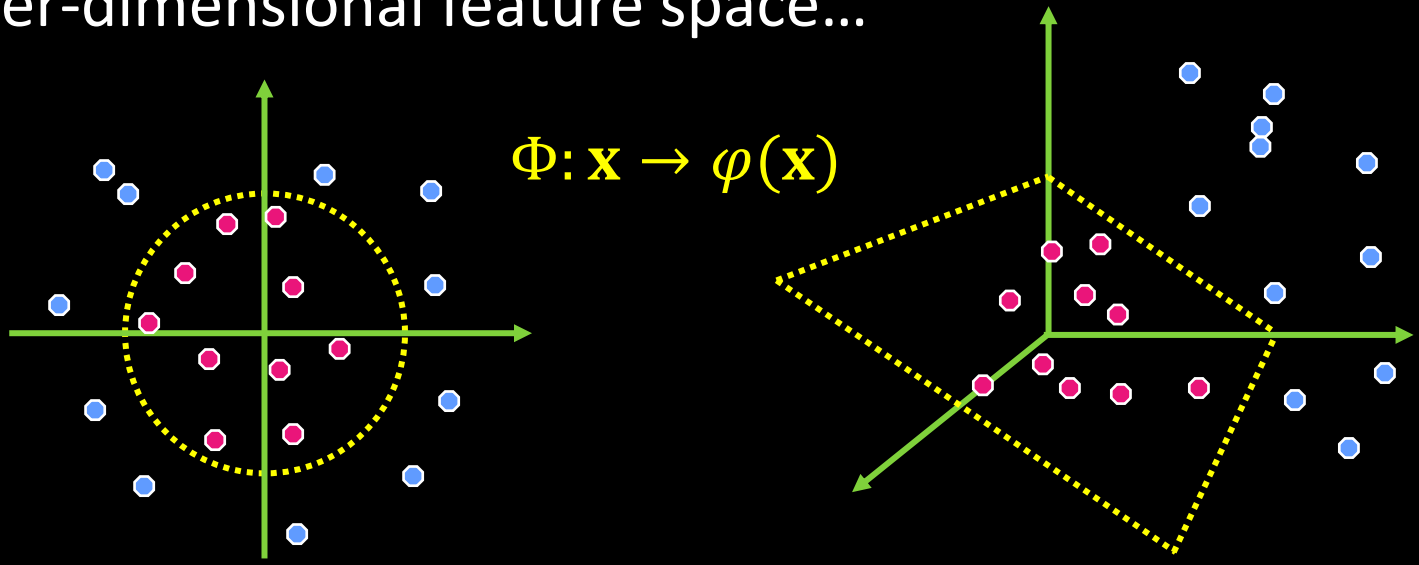
Non-linear SVMs

- How about… mapping data to a higher-dimensional space:



Non-linear SVMs: Feature spaces

General idea: Original input space can be mapped to some higher-dimensional feature space…



…where the training set is easily separable

Andrew Moore

The “kernel” trick

- We saw linear classifier relies on dot product between vectors.
- Define:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j = \mathbf{x}_i^T \mathbf{x}_j$$

The “kernel” trick

If every data point is mapped into high-dimensional space via some transformation

$$\Phi: \mathbf{x} \rightarrow \varphi(\mathbf{x})$$

the dot product becomes:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i)^T \varphi(\mathbf{x}_j)$$

A *kernel function* is a “similarity” function that corresponds to an inner product in some expanded feature space

E.g., 2D vectors $\mathbf{x} = [x_1\ x_2]$ Let $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^2$

Need to show that $K(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i)^T \varphi(\mathbf{x}_j)$:

$$K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^2$$

$$= 1 + x_{i1}^2 x_{j1}^2 + 2x_{i1}x_{j1}x_{i2}x_{j2} + x_{i2}^2 x_{j2}^2 + 2x_{xi1}x_{j1} + 2x_{i2}x_{j2}$$

$$= \begin{bmatrix} 1 & x_{i1}^2 & \sqrt{2}x_{i1}x_{i2} & x_{i2}^2 & \sqrt{2}x_{i1} & \sqrt{2}x_{i2} \end{bmatrix}^T$$

$$\begin{bmatrix} 1 & x_{j1}^2 & \sqrt{2}x_{j1}x_{j2} & x_{j2}^2 & \sqrt{2}x_{j1} & \sqrt{2}x_{j2} \end{bmatrix}$$

$$= \varphi(\mathbf{x}_i)^T \varphi(\mathbf{x}_j)$$

where $\varphi(\mathbf{x}) = \begin{bmatrix} 1 & x_1^2 & \sqrt{2}x_1x_2 & x_2^2 & \sqrt{2}x_1 & \sqrt{2}x_2 \end{bmatrix}$

Andrew Moore

Nonlinear SVMs

The kernel trick: Instead of explicitly computing the lifting transformation $\varphi(\mathbf{x})$, define a kernel function K:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i)^T \varphi(\mathbf{x}_j)$$

This gives a nonlinear decision boundary in the original feature space:

$$\sum_i \alpha_i y_i (\mathbf{x}_i^T \mathbf{x}) + b \rightarrow \sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$$

Examples of kernel functions

- Linear $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$

- Number of dimensions: N (just size of x)

Examples of kernel functions

Gaussian RBF $K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$

Number of dimensions: Infinite

$$\exp\left(-\frac{1}{2}\|\mathbf{x}-\mathbf{x}'\|_2^2\right) = \sum_{j=0}^{\infty}\frac{(\mathbf{x}^\top\mathbf{x}')^j}{j!}\exp\left(-\frac{1}{2}\|\mathbf{x}\|_2^2\right)\exp\left(-\frac{1}{2}\|\mathbf{x}'\|_2^2\right)$$

Examples of kernel functions

Histogram intersection

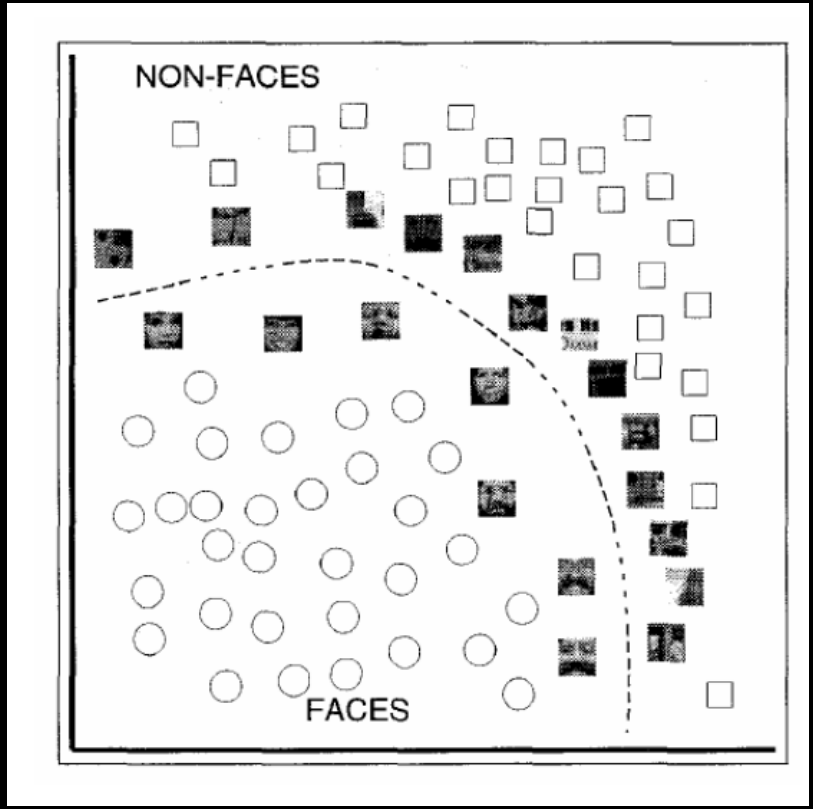
$$K(\mathbf{x}_i, \mathbf{x}_j) = \sum_k \min(\mathbf{x}_i(k), \mathbf{x}_j(k))$$

Number of dimensions: Large. See:

Alexander C. Berg, Jitendra Malik, *"Classification using intersection kernel support vector machines is efficient"*, *CVPR*, 2008

SVMs for recognition: Training

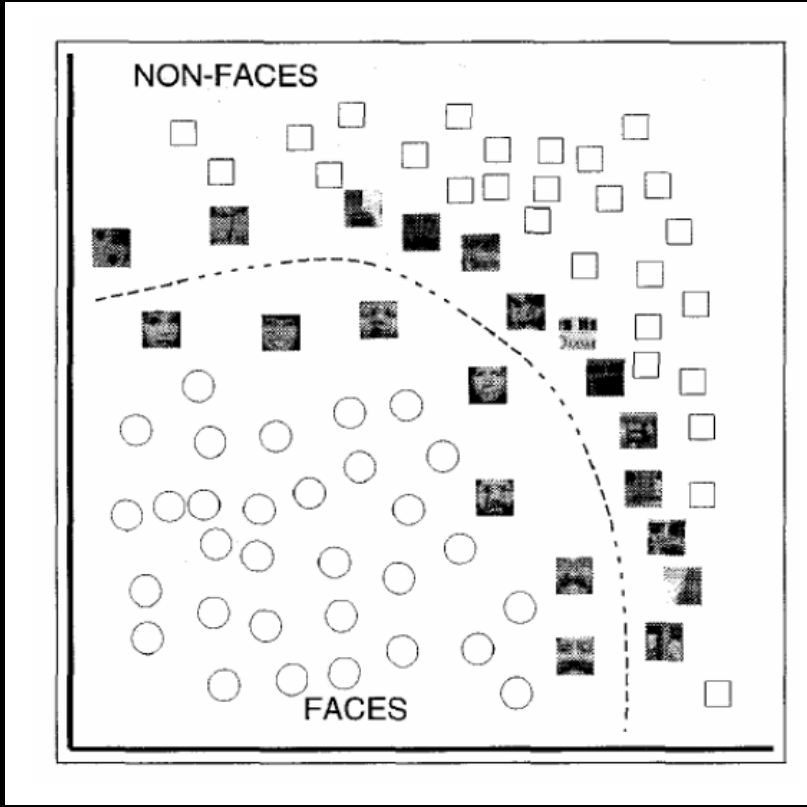
1. Define your representation
2. Select a kernel function
3. Compute pairwise kernel values between labeled examples
4. Use this “kernel matrix” to solve for SVM support vectors & weights



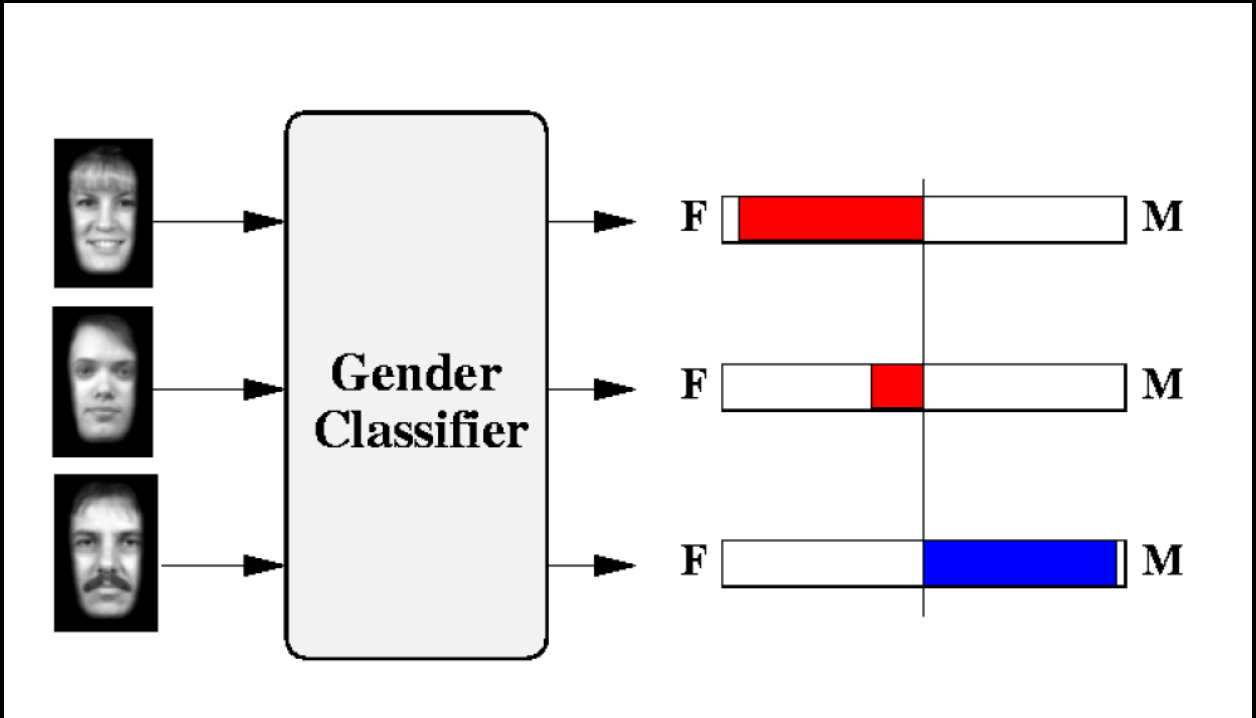
SVMs for recognition: Prediction

To classify a new example:

1. Compute kernel values between new input and support vectors
2. Apply weights
3. Check sign of output

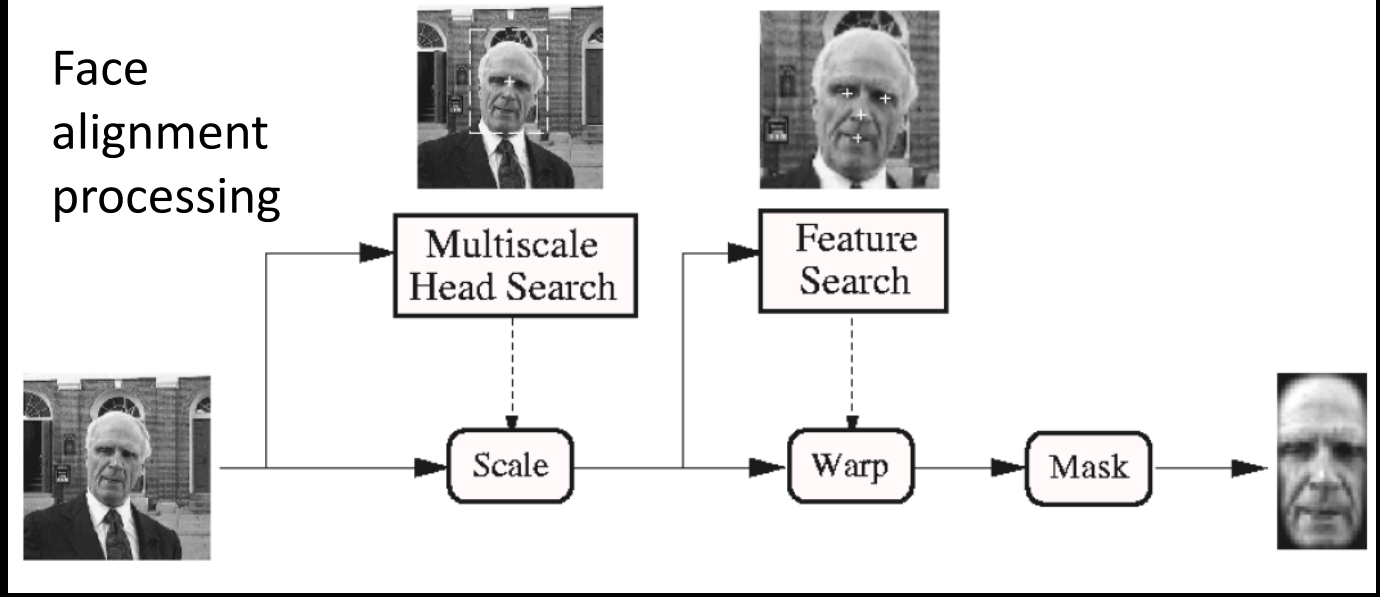


Learning gender with SVMs



Learning Gender with Support Faces [Moghaddam and Yang, TPAMI 2002]

Face alignment processing



Processed faces

Face gender classification by SVM

- Training examples: 1044 males, 713 females
- Images reduced to 21x12 (!!)
- Experiment with various kernels, select Gaussian RBF

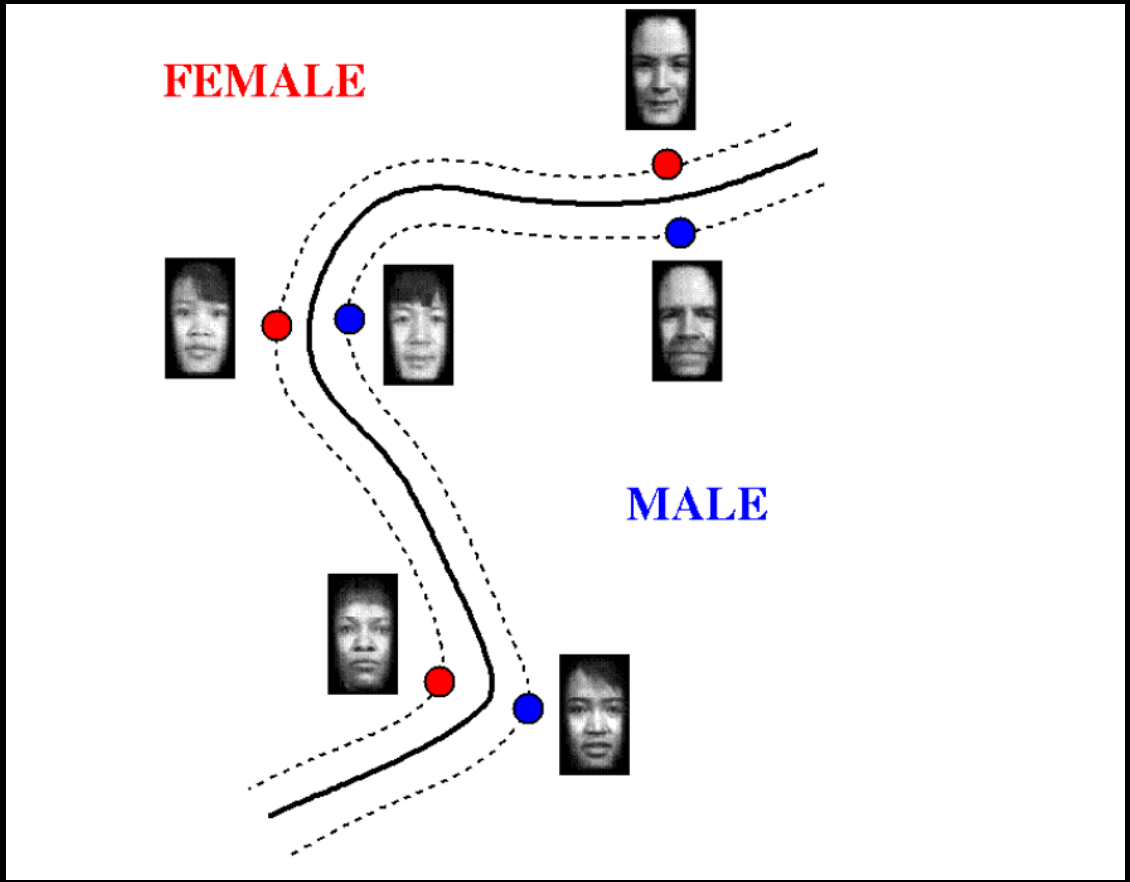
$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\left\| \mathbf{x}_i - \mathbf{x}_j \right\|^2}{2\sigma^2} \right)$$

Face gender classification by SVM

- Training examples: 1044 males, 713 females
- Images reduced to 21x12 (!!)
- Experiment with various kernels, select Gaussian RBF

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

Support Faces



Moghaddam and Yang: Gender Classifier Performance

Classifier	Error Rate		
	Overall	Male	Female
SVM with RBF kernel	**3.38%**	**2.05%**	**4.79%**
SVM with cubic polynomial kernel	**4.88%**	**4.21%**	**5.59%**
Large Ensemble of RBF	5.54%	4.59%	6.55%
Classical RBF	7.79%	6.89%	8.75%
Quadratic classifier	10.63%	9.44%	11.88%
Fisher linear discriminant	13.03%	12.31%	13.78%
Nearest neighbor	27.16%	26.53%	28.04%
Linear classifier	58.95%	58.47%	59.45%

Gender perception experiment: How well can humans do?

Subjects:

- 30 people (22 male, 8 female), ages mid-20's to mid-40's

Test data:

- 254 face images (60% males, 40% females)
- Low res and high res versions

Task:

- Classify as male or female, forced choice
- No time limit

Moghaddam and Yang, Face & Gesture 2000

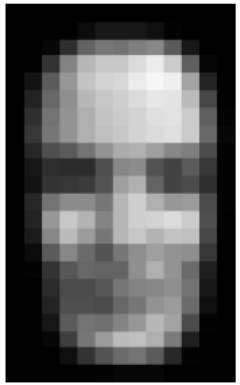
Human Performance

Stimuli →

84 x 48

N = 4032

21 x 12

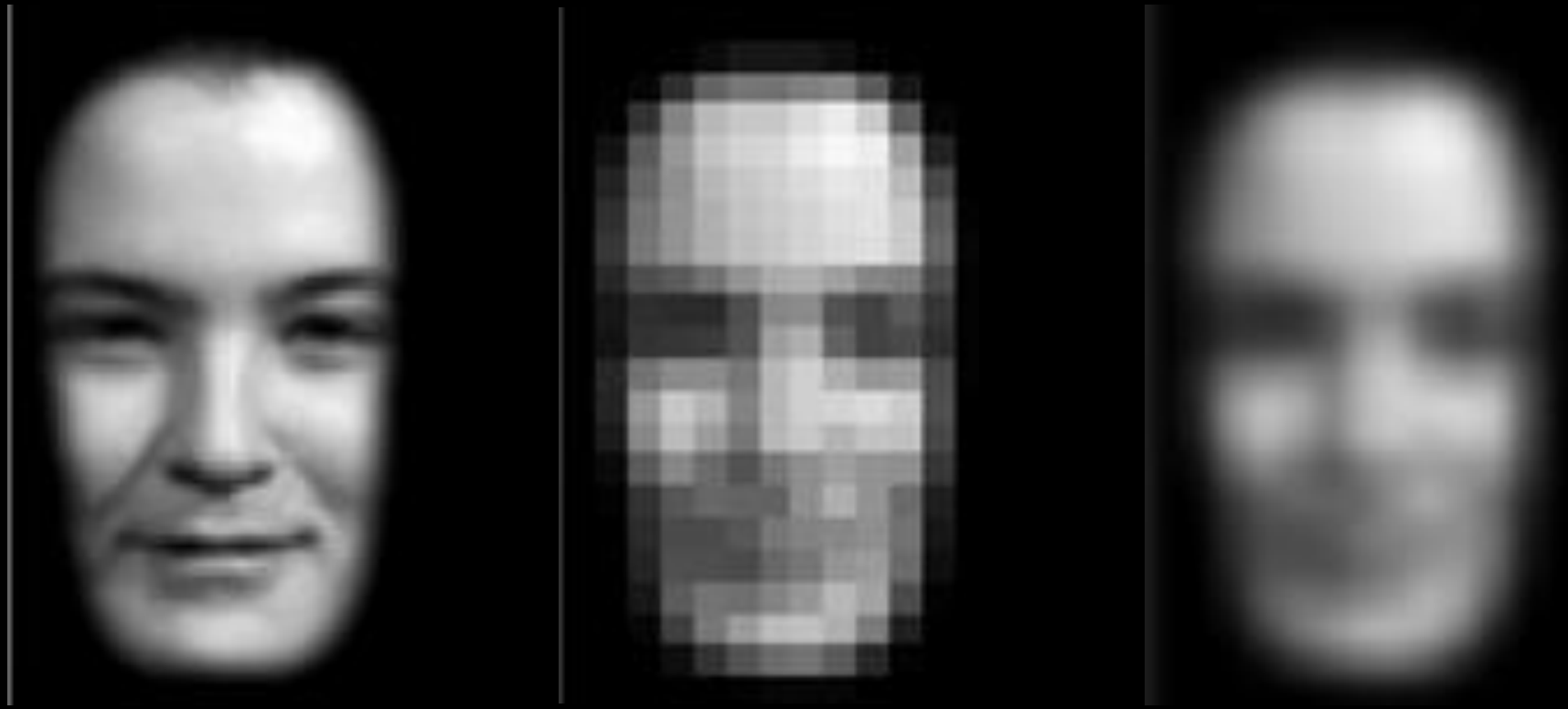


N = 252

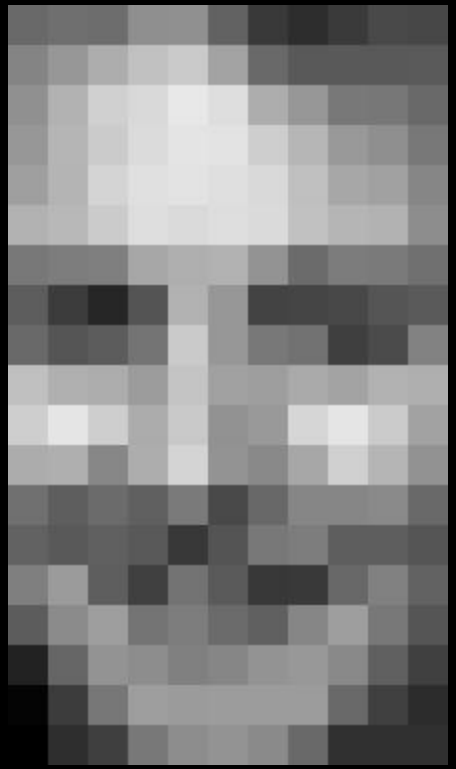
Results →

High-Res	Low-Res
6.54% Error	30.7% Error

Careful how you do things?



Careful how you do things?



Human vs. Machine

Figure 6. SVM vs. Human performance

SVMs performed better than any single human test subject, at either resolution

Hardest examples for humans

Top five human misclassifications

Moghaddam and Yang, Face & Gesture 2000

Questions

- What if the features are not 2d?
- What if the data is not linearly separable?
- What if we have more than just two categories?

Multi-class SVMs

Combine a number of binary classifiers

One vs. all

- Training: Learn an SVM for each class vs. the rest
- Testing: Apply each SVM to test example and assign to it the class of the SVM that returns the highest decision value

One vs. one

- Training: Learn an SVM for each pair of classes
- Testing: Each learned SVM “votes” for a class to be assigned to the test example

SVMs: Pros and cons

Pros

- Many publicly available SVM packages:
  - http://www.kernel-machines.org/software
  - http://www.csie.ntu.edu.tw/~cjlin/libsvm/
- Kernel-based framework is very powerful, flexible
- Often a sparse set of support vectors – compact when testing
- Works very well in practice, even with very small training sample sizes

Adapted from Lana Lazebnik

SVMs: Pros and cons

Cons

- No “direct” multi-class SVM, must combine two-class SVMs
- Can be tricky to select best kernel function for a problem
- Nobody writes their own SVMs
- Computation, memory
  - During training time, must compute matrix of kernel values for every pair of examples
  - Learning can take a very long time for large-scale problems

Adapted from Lana Lazebnik