CSCE 990 Lecture 7: SVMs for Classification*

Stephen D. Scott

February 14, 2006

Introduction

- Finally, we get to put everything together!

- Much of this lecture is material we've covered previously, but now we'll make it specific to SVMs

- We'll also formalize the notion of the margin, introduce soft margin, and argue why we want to minimize $\|\mathbf{w}\|^2$

Outline

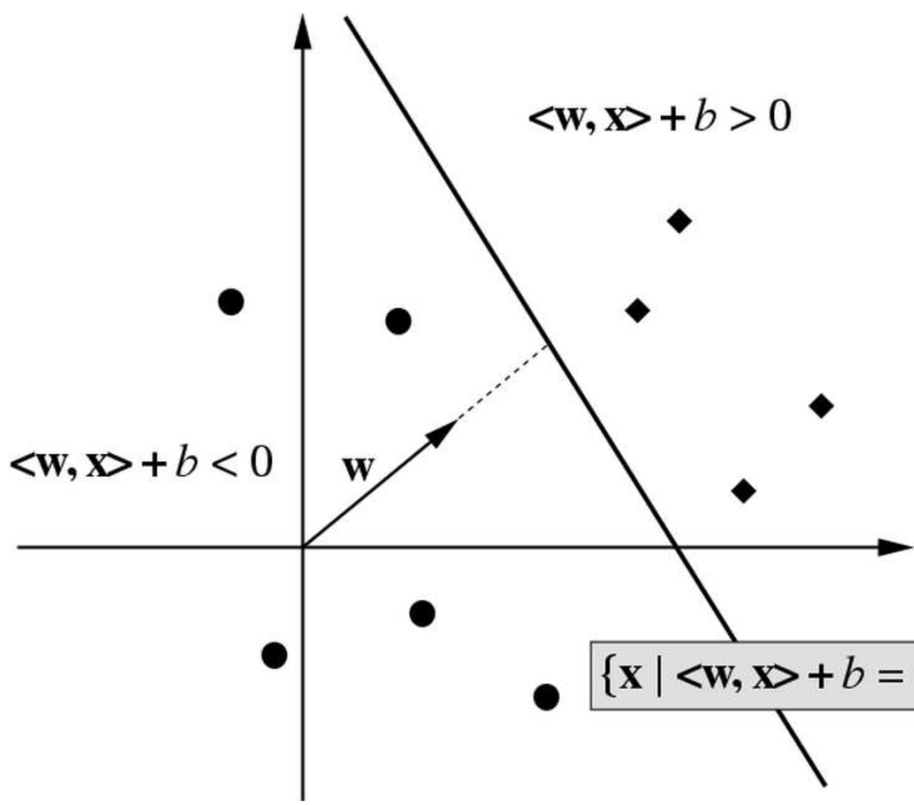
- Canonical hyperplanes
- The (geometrical) margin and the margin error bound
- Optimal margin hyperplanes
- Adding kernels
- Soft margin hyperplanes
- Multi-class classification
- Application: handwritten digit recognition
- Sections 7.1–7.6, 7.8–7.9

Canonical Hyperplanes

- Any hyperplane in a dot product space $\mathcal{H}$ can be written as

$$H = \{\mathbf{x} \in \mathcal{H} \mid \langle \mathbf{w}, \mathbf{x} \rangle + b = 0\}, \mathbf{w} \in \mathcal{H}, b \in \mathbb{R}$$

- $\langle \mathbf{w}, \mathbf{x} \rangle$ is the length of $\mathbf{x}$ in the direction of $\mathbf{w}$, multiplied by $\|\mathbf{w}\|$, i.e. each $\mathbf{x} \in H$ has the same length in the direction of $\mathbf{w}$

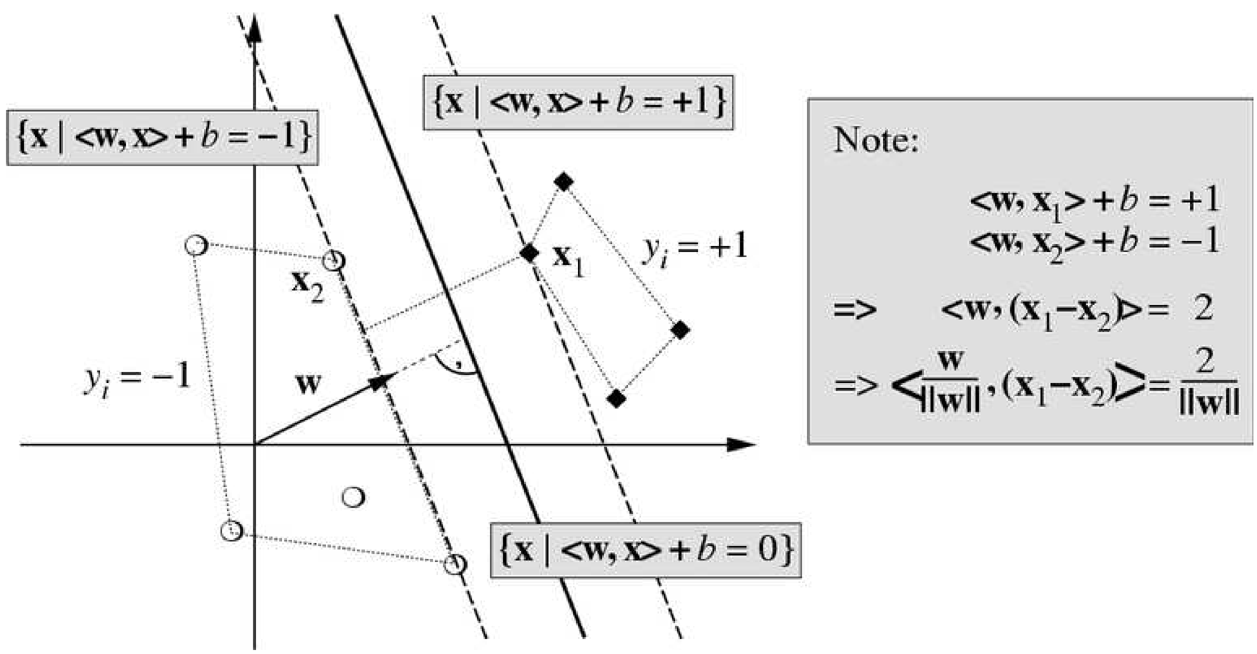


Canonical Hyperplanes
(cont'd)

- Note that if both $\mathbf{w}$ and b are multiplied by the same non-zero constant, H is unchanged

D7.1 The pair $(\mathbf{w}, b) \in \mathcal{H}$ is called a canonical form of the hyperplane H wrt a set of patterns $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathcal{H}$ if it is scaled such that

$$\min_{i=1,\ldots,m} |\langle \mathbf{w}, \mathbf{x}_i \rangle + b| = 1$$



- Given a canonical hyperplane $(\mathbf{w}, b)$, the corresponding decision function is
$f_{\mathbf{w},b}(\mathbf{x}) := \text{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle + b)$

The Margin

D7.2 For a hyperplane $\{\mathbf{x} \in \mathcal{H} \mid \langle \mathbf{w}, \mathbf{x} \rangle + b = 0\}$, define

$$\rho_{\mathbf{w},b}(\mathbf{x}, y) := y(\langle \mathbf{w}, \mathbf{x} \rangle + b)/\|\mathbf{w}\|$$

as the geometrical margin (or simply margin) of the point $(\mathbf{x}, y) \in \mathcal{H} \times \{-1, +1\}$. Further,

$$\rho_{\mathbf{w},b} := \min_{i=1,\ldots,m} \rho_{\mathbf{w},b}(\mathbf{x}_i, y_i)$$

is the (geometrical) margin of $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$ (typically the training set)

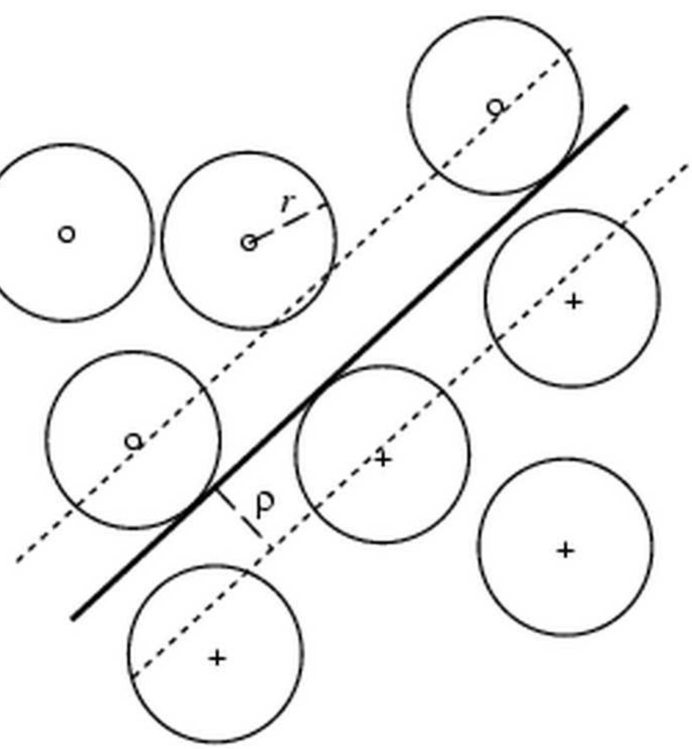
- In D7.2, we are really using the hyperplane $(\hat{\mathbf{w}}, \hat{b}) := (\mathbf{w}/\|\mathbf{w}\|, b/\|\mathbf{w}\|)$, which has unit length
- Further, $\langle \hat{\mathbf{w}}, \mathbf{x} \rangle + \hat{b}$ is $\mathbf{x}$'s distance to this hyperplane, and multiplying by y implies that the margin is positive if $(\mathbf{x}, y)$ is correctly classified
- Since canonical hyperplanes have minimum distance 1 to data points, the margin of a canonical hyperplane is $\rho_{\mathbf{w},b} = 1/\|\mathbf{w}\|$
- I.e. decreasing $\|\mathbf{w}\|$ increases the margin!

Justifications for Large Margin

- Why do we want large margin hyperplanes (that separate the training data)?

- Insensitivity to pattern noise

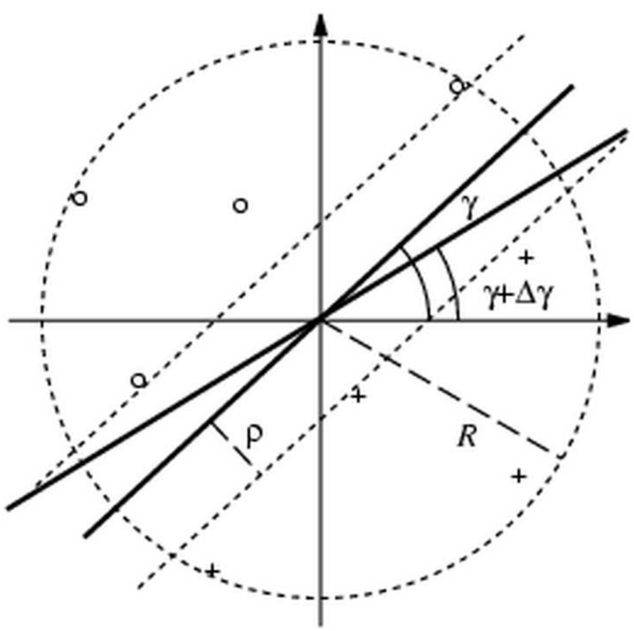
  – E.g. if each (noisy) test point $(\mathbf{x}+\Delta\mathbf{x}, y)$ is near some (noisy) training point $(\mathbf{x}, y)$ with $\|\Delta\mathbf{x}\| < r$, then if $\rho > r$ we correctly classify all test points



Justifications for Large Margin (cont'd)

- Insensitivity to parameter noise
  - If all patterns are at least ρ from $H = (\mathbf{w}, b)$ and all patterns are bounded in length by R, then small changes in the parameters of H will not change classification
  - I.e. can encode H with fewer bits than if we precisely encoded it and still be correct on training set

  $\Rightarrow$ minimum description length/compression of data



Justifications for Large Margin
(cont'd)

T7.3 For decision functions $f(\mathbf{x}) = \text{sgn}\langle \mathbf{w}, \mathbf{x} \rangle$, let $\|\mathbf{w}\| \leq \Lambda$, $\|\mathbf{x}\| \leq R$, $\rho > 0$, and ν be the margin error, i.e. the fraction of training examples with margin $< \rho/\|\mathbf{w}\|$. Then if all training and test patterns are drawn iid, with probability at least $1 - \delta$ the test error is upper bounded by

$$\nu + \sqrt{\frac{c}{m}\left(\frac{R^2\Lambda^2}{\rho^2}\ln^2 m + \ln(1/\delta)\right)}$$

where c is a constant and m is the training set size

- Related to VC dimension of large-margin classifiers, but not exactly what we covered in Chapter 5; e.g. R_{emp}, which was a prediction error rate, is replaced with ν, which is a margin error rate

Justifications for Large Margin

Margin Error Bound

(cont'd)

- Increasing ρ decreases the square root term, but can increase ν
  - Thus we want to maximize ρ while simultaneously minimizing ν
  - Can instead fix $\rho = 1$ (canonical hyperplanes) and minimize $\|\mathbf{w}\|$ while minimizing margin errors
  - In our first quadratic program, we'll set constraints to make $\nu = 0$

Optimal Margin Hyperplanes

- Want hyperplane that correctly classifies all training patters with maximum margin

- When using canonical hyperplanes, implies that we want $y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle + b) \geq 1$ for all $i = 1, \ldots, m$

- We know that we want to minimize the weight vector's length to maximize the margin, so this yields the following constrained quadratic optimization problem:

$$\begin{array}{ll} \underset{\mathbf{w}\in\mathcal{H}, b\in\mathbb{R}}{\text{minimize}} & \tau(\mathbf{w}) = \|\mathbf{w}\|^2/2 \\ \text{s.t.} & y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle + b) \geq 1, i = 1, \ldots, m \end{array} \tag{1}$$

- Another optimization problem. Hey! I have a great idea! Let's derive the dual!

- Langrangian:

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \|\mathbf{w}\|^2/2 - \sum_{i=1}^{m} \alpha_i(y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle + b) - 1)$$

with $\alpha_i \geq 0$

The Dual Optimization Problem (cont'd)

- Recall that at the saddle point, the partial derivatives of L wrt the primal variables must each go to 0:

$$\frac{\partial}{\partial b} L(\mathbf{w}, b, \boldsymbol{\alpha}) = -\sum_{i=1}^{m} \alpha_i y_i = 0$$

$$\frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \boldsymbol{\alpha}) = \mathbf{w} - \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i = 0$$

which imply $\sum_{i=1}^{m} \alpha_i y_i = 0$ and $\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$

- Recall from Chapter 6 that for an optimal feasible solution $\bar{\mathbf{w}}$, $\alpha_i c_i(\bar{\mathbf{w}}, \bar{b}) = 0$ for all constraints c_i, so

$$\alpha_i (y_i (\langle \mathbf{x}_i, \bar{\mathbf{w}} \rangle + \bar{b}) - 1) = 0$$

for all $i = 1, \ldots, m$

The Dual Optimization Problem
(cont'd)

- The $\mathbf{x}_i$ for which $\alpha_i > 0$ are the support vectors, and are the vectors that lie on the margin, i.e. those for which the constraints are tight
  - Other vectors (where $\alpha_i = 0$) are irrelevant to determining the hyperplane $\mathbf{w}$
  - Will be useful later in classification
  - See Prop. 7.8 for relationship between expected number of SVs and test error bound

The Dual Optimization Problem (cont'd)

- Now substitute the saddle point conditions into the Lagrangian

- The kth component of the weight vector is $w_k = \sum_{i=1}^{m} \alpha_i y_i x_{ik}$, so

$$w_k^2 = \left(\sum_{i=1}^{m} \alpha_i y_i x_{ik}\right)\left(\sum_{i=1}^{m} \alpha_i y_i x_{ik}\right)$$

- Thus

$$\begin{aligned}
\|\mathbf{w}\|^2 &= \sum_k \left(\sum_{i=1}^{m} \alpha_i y_i x_{ik}\right)\left(\sum_{i=1}^{m} \alpha_i y_i x_{ik}\right) \\
&= \sum_k \sum_{i,j} \alpha_i \alpha_j y_i y_j x_{ik} x_{jk} \\
&= \sum_{i,j} \alpha_i \alpha_j y_i y_j \sum_k x_{ik} x_{jk} \\
&= \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle
\end{aligned}$$

The Dual Optimization Problem
(cont'd)

- Further,

$$
\begin{aligned}
&\sum_{i=1}^{m} \alpha_i (y_i(\langle \mathbf{x}_i, \mathbf{w} \rangle + b) - 1) \\
&= \sum_{i=1}^{m} \alpha_i y_i \left(\sum_k x_{ik} w_k \right) - \sum_{i=1}^{m} \alpha_i \\
&= \sum_{i=1}^{m} \alpha_i y_i \left(\sum_k x_{ik} \sum_{j=1}^{m} \alpha_j y_j x_{jk} \right) - \sum_{i=1}^{m} \alpha_i \\
&= \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle - \sum_{i=1}^{m} \alpha_i
\end{aligned}
$$

- Combine them:

$$
L(\mathbf{w}, b, \boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle
$$

The Dual Optimization Problem
(cont'd)

- Maximizing the Lagrangian wrt $\boldsymbol{\alpha}$ yields the dual optimization problem:

$$\begin{aligned} \underset{\boldsymbol{\alpha}\in\mathbb{R}^m}{\text{maximize}} \quad & \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \\ \text{s.t.} \quad & \alpha_i \geq 0, i = 1, \ldots, m \\ & \sum_{i=1}^{m} \alpha_i y_i = 0 \end{aligned} \tag{2}$$

- After optimization, we can label new vectors with the decision function:

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{m} \alpha_i y_i \langle \mathbf{x}, \mathbf{x}_i \rangle + b \right)$$

(later we'll discuss finding b)

Adding Kernels

- As discussed before, using kernels is an effective way to introduce nonlinearities to the data
  - Nonlinear remapping might make data (almost) linearly separable in the new space
  - Cover's theorem implies that simply increasing the dimension improves the probability of linear separability
- For given remapping Φ, simply replace $\mathbf{x}$ with $\Phi(x)$
- Thus in dual optimization problem and in decision function, replace $\langle \mathbf{x}, \mathbf{x}_i \rangle$ with $k(x, x_i)$, where k is the PD kernel corresponding to Φ
- If k is PD, then we still have a convex optimization problem
- Once $\boldsymbol{\alpha}$ is found, can e.g. set b to be the average over all $\alpha_j > 0$ of $y_j - \sum_{i=1}^{m} y_i \alpha_i k(x_j, x_i)$ (derived from KKT conditions)

Soft Margin Hyperplanes

- Under a given mapping Φ, the data might not be linearly separable

- There always exists a Φ that will yield separability, but is it a good idea to find one just for the sake of separating?

- If we choose to keep the mapping that corresponds to our favorite kernel, what are our options?

  – Instead of finding a hyperplane that is perfect on the training set, find one that minimizes training errors

    * Computationally intractable to even approximate

  – Instead, we'll soften the margin, allowing for some vectors to get too close to the hyperplane (i.e. margin errors)

Soft Margin Hyperplanes
(cont'd)

- To relax each constraint from (1), add slack variable $\xi_i \geq 0$:

$$y_i(\langle \mathbf{x}_i, \mathbf{w} \rangle + b) \geq 1 - \xi_i, \ i = 1, \ldots, m$$

- Also need to penalize large ξ_i in the objective function to prevent trivial solutions
  - C-SV classifier
  - ν-SV classifier

Soft Margin Hyperplanes

C-SV Classifier

- Weight with $C > 0$ (e.g. $C = 10m$) the importance of minimizing sum of ξ variables:

$$\begin{array}{ll} \underset{\mathbf{w}\in\mathcal{H},b\in\mathbb{R},\boldsymbol{\xi}\in\mathbb{R}^m}{\text{minimize}} & \tau(\mathbf{w},\boldsymbol{\xi}) = \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{m}\sum_{i=1}^{m}\xi_i \\ \text{s.t.} & y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle + b) \geq 1 - \xi_i,\ i = 1,\dots,m \\ & \xi_i \geq 0,\ i = 1,\dots,m \end{array}$$

- First term of τ decreases $\|\mathbf{w}\|$, second term focuses on margin error rate ν, thus together they focus on T7.3

- The dual is similar to that for hard margin:

$$\begin{array}{ll} \underset{\boldsymbol{\alpha}\in\mathbb{R}^m}{\text{maximize}} & W(\boldsymbol{\alpha}) = \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j k(x_i, x_j) \\ \text{s.t.} & 0 \leq \alpha_i \leq C/m,\ i = 1,\dots,m \\ & \sum_{i=1}^{m}\alpha_i y_i = 0 \end{array}$$

- Once $\boldsymbol{\alpha}$ is found, can e.g. set b to be the average over all $\alpha_j \in (0, C)$ of $y_j - \sum_{i=1}^{m} y_i\alpha_i k(x_j, x_i)$

Soft Margin Hyperplanes

ν-SV Classifier

- A more intuitable way to weight the emphasis on reducing margin errors

- Primal:

$$\begin{array}{ll} \underset{\mathbf{w}\in\mathcal{H},\rho,b\in\mathbb{R},\boldsymbol{\xi}\in\mathbb{R}^m}{\text{minimize}} & \tau(\mathbf{w},\boldsymbol{\xi},\rho) = \frac{1}{2}\|\mathbf{w}\|^2 - \nu\rho + \frac{1}{m}\sum_{i=1}^{m}\xi_i \\ \text{s.t.} & y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle + b) \geq \rho - \xi_i, i = 1,\ldots,m \\ & \rho \geq 0, \xi_i \geq 0,\ i = 1,\ldots,m \end{array}$$

- ρ is similar to that in T7.3: for $\boldsymbol{\xi}$ to be 0, all vectors must be at least $\rho/\|\mathbf{w}\|$ from the hyperplane

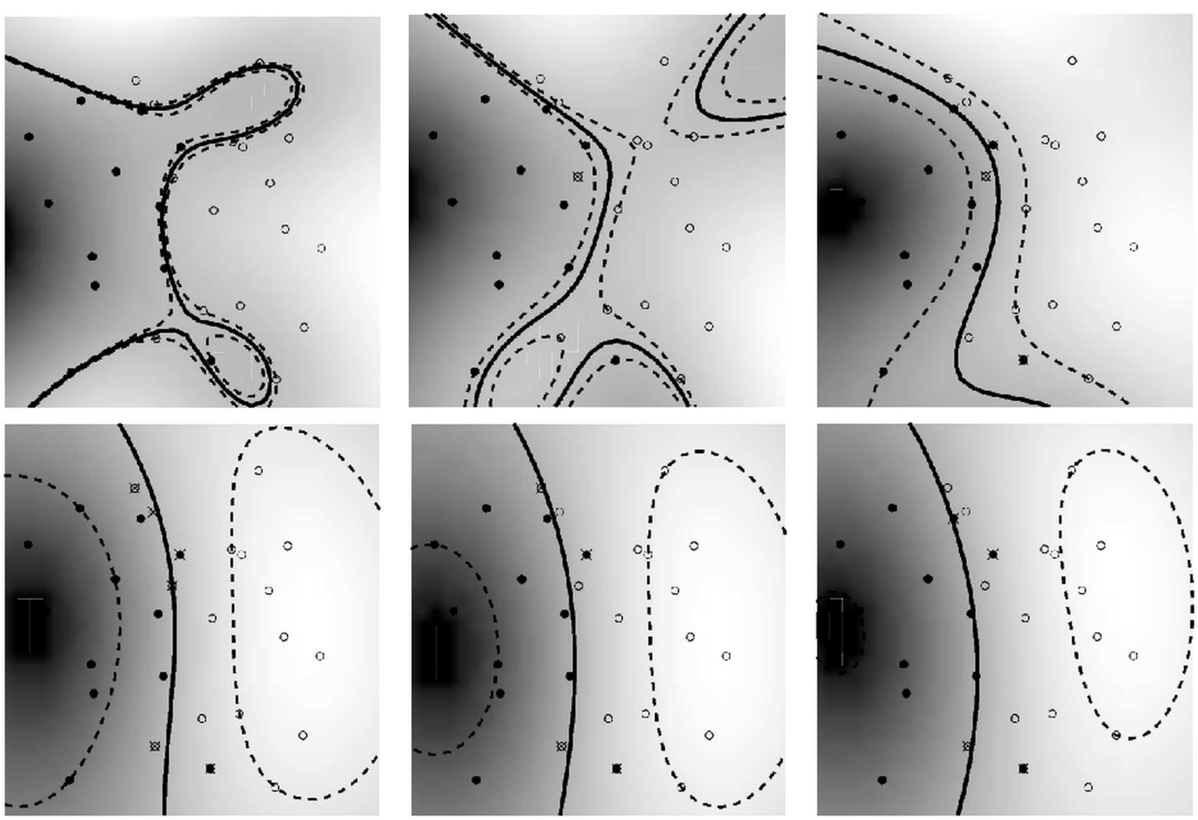
Soft Margin Hyperplanes
ν-SV Classifier
(cont'd)

P7.5 If ν-SVC yields a solution with $\rho > 0$, then

1. ν is an upper bound on the fraction of margin errors

2. ν is a lower bound on the fraction of support vectors

- See Table 7.1, p. 207



Soft Margin Hyperplanes
ν-SV Classifier
(cont'd)

- Derivation of dual form (details omitted) yields:

$$\begin{aligned} \underset{\boldsymbol{\alpha}\in\mathbb{R}^m}{\text{maximize}} \quad & W(\boldsymbol{\alpha}) = -\frac{1}{2}\sum_{i,j=1}^{m}\alpha_i\alpha_j y_i y_j k(x_i, x_j) \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq 1/m, \\ & \sum_{i=1}^{m}\alpha_i y_i = 0 \\ & \sum_{i=1}^{m}\alpha_i \geq \nu \end{aligned}$$

- Let S_+ and S_- be sets of SVs x_i with labels $y_i = +1$ and -1, $0 < \alpha_i < 1$, and $|S_+| = |S_-| = s > 0$, then set

$$b = -\frac{1}{2s}\sum_{x\in S_+\cup S_-}\sum_{i=1}^{m} y_i\alpha_i k(x, x_i)$$

Multi-Class Classification

- What if we want to go beyond binary labels ± 1 to M classes?

- Most methods decompose a multi-class problem into a set of binary ones
  - One vs. rest
  - Error-correcting output codes
  - Pairwise classification
  - Kessler's construction/multi-class objective function (doesn't need to decompose into binary cases)

Multi-Class Classification
One vs. the Rest

- To handle M classes, train a set of M binary classifiers $f^1, \ldots, f^M$, where f^i is trained to distinguish patterns from class i from those not in class i

- If $(\boldsymbol{\alpha}^i, b^i)$ is the classifier learned for class i, then a new pattern x is classified as
$$\underset{j=1,\ldots,M}{\text{argmax}} \left\{ \sum_{i=1}^{m} y_i \alpha_i^j k(x, x_i) + b^j \right\} \quad ,$$
i.e. the class with the most confident prediction among the binary classifiers

- Applicable even if the number of classifiers predicting $+1$ is not exactly 1

- Note that the set of SVs can be different for each class

- Can also let the classifier "punt" if the difference between the top two predictions is small

Multi-Class Classification

Error-Correcting Output Codes (ECOC)

- One vs. rest requires M classifiers to represent M classes

- Is this the minimum amount required?

- E.g. $M = 4$, so use two linear classifiers:

Class	Binary Encoding	
	Classifier 1	Classifier 2
Class 1	-1	-1
Class 2	-1	$+1$
Class 3	$+1$	-1
Class 4	$+1$	$+1$

  and train simultaneously

- Problem: Sensitive to individual classifier errors, so use a set of encodings per class to improve robustness

Multi-Class Classification

Error-Correcting Output Codes (ECOC) (cont'd)

- Similar to principle of error-correcting output codes used in communication networks

  - After all classifiers make their predictions, find the code that is nearest to the bit string returned and use that for the predicted class

- Can provably tolerate some mispredictions by individual classifiers, but doesn't use the margin

Multi-Class Classification
Pairwise Classification

- Instead of training one classifier per class as in one vs. rest, train a classifier for each pair of classes

- Now have $\binom{M}{2}$ classifiers to train rather than $\lceil \log_2 M \rceil$ up to M, but each training set is smaller

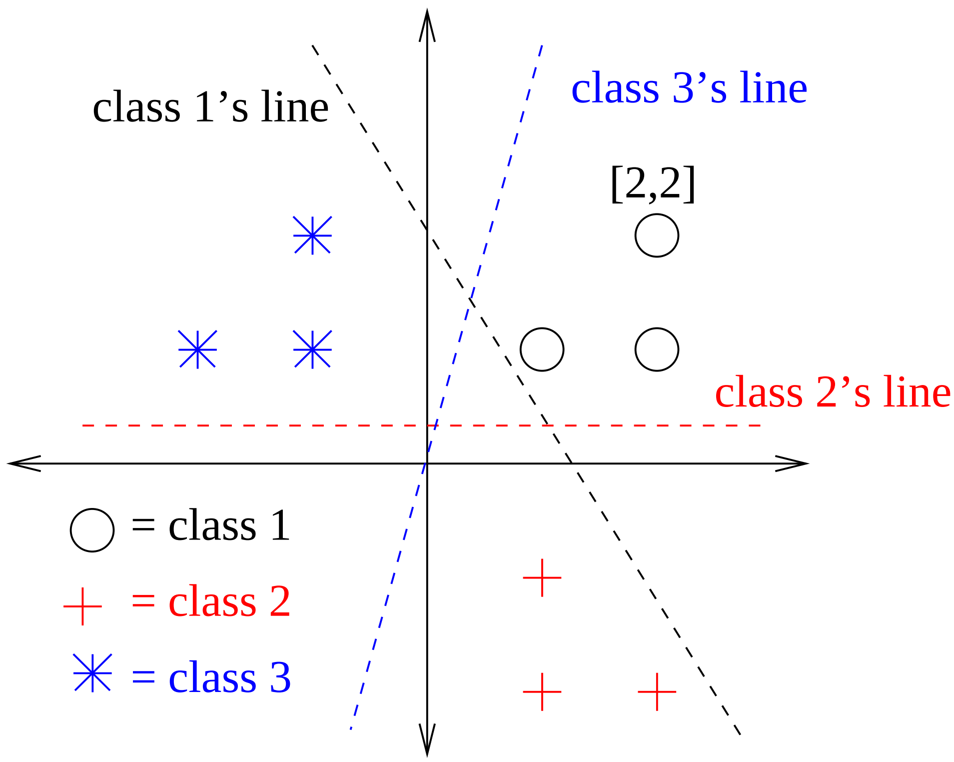
  – Number of SVs smaller for each classifier due to smaller training set and easier learning problem

- To classify new pattern, evaluate it on all classifiers and choose the class that gets the most votes

  – Can avoid running on all classifiers if votes so far imply that some classes are guaranteed to not win

Multiclass learning

Kessler's Construction



- For* $\mathbf{x} = [2, 2, 1]^T$ of class 1, want

$$\sum_{i=1}^{\ell+1} w_{1i}x_i > \sum_{i=1}^{\ell+1} w_{2i}x_i \quad \underline{\text{AND}} \quad \sum_{i=1}^{\ell+1} w_{1i}x_i > \sum_{i=1}^{\ell+1} w_{3i}x_i$$

*The extra 1 is added so threshold can be placed in $\mathbf{w}$.

Multiclass learning
Kessler's Construction (cont'd)

- So map $\mathbf{x}$ to

$$\mathbf{x}_1 = [\overbrace{2, 2, 1,}^{\text{orig.}} \overbrace{-2, -2, -1,}^{\text{neg}} \overbrace{0, 0, 0}^{\text{pad}}]$$
$$\mathbf{x}_2 = [2, 2, 1, 0, 0, 0, -2, -2, -1]$$

(all labels $= +1$) and let

$$\mathbf{w} = [\overbrace{w_{11}, w_{12}, w_{10},}^{\mathbf{w}_1} \overbrace{w_{21}, w_{22}, w_{20},}^{\mathbf{w}_2} \overbrace{w_{31}, w_{32}, w_{30}}^{\mathbf{w}_3}]$$

- Now if $\langle \mathbf{w}^*, \mathbf{x}_1 \rangle > 0$ and $\langle \mathbf{w}^*, \mathbf{x}_2 \rangle > 0$, then

$$\sum_{i=1}^{\ell+1} w_{1i}^* x_i > \sum_{i=1}^{\ell+1} w_{2i}^* x_i \quad \underline{\text{AND}} \quad \sum_{i=1}^{\ell+1} w_{1i}^* x_i > \sum_{i=1}^{\ell+1} w_{3i}^* x_i$$

- In general, map $(\ell + 1) \times 1$ feature vector $\mathbf{x}$ to $\mathbf{x}_1, \ldots \mathbf{x}_{M-1}$, each of size $(\ell + 1)M \times 1$

- $\mathbf{x} \in \omega_i \Rightarrow \mathbf{x}$ in ith block and $-\mathbf{x}$ in jth block, (rest are 0s). Repeat for all $j \neq i$

- Now train to find weights for new vector space via e.g. Perceptron

Multi-Class Classification

Multi-Class Objective Functions

- From the idea of Kessler's construction, can develop a quadratic program for an SVM (C-SV in this case):

$$\begin{aligned}
&\underset{\mathbf{w}_r \in \mathcal{H}, b_r \in \mathbb{R}, \boldsymbol{\xi}^r \in \mathbb{R}^m}{\text{minimize}} && \frac{1}{2}\sum_{r=1}^{M} \|\mathbf{w}_r\|^2 + \frac{C}{m}\sum_{i=1}^{m}\sum_{r \neq y_i} \xi_i^r \\
&\text{s.t.} && \langle \mathbf{x}_i, \mathbf{w}_{y_i} \rangle + b_{y_i} \geq \langle \mathbf{x}_i, \mathbf{w}_r \rangle + b_r + 2 - \xi_i^r, \\
& && \quad r \neq y_i, i = 1, \ldots, m \\
& && \xi_i^r \geq 0, \ \forall i, r
\end{aligned}$$

- Here $y_i \in \{1, \ldots, M\}$ is an integer specifying the class label

Application: Handwritten Digit Recognition

- Experiments using C-SVC on US Postal Service (USPS) database of handwritten digits
  - Human error rate: 2.5%

- Kernels scaled to help avoid over/underflow

poly: $k(x, x') = (\langle x, x' \rangle / 256)^d$

d	1	2	3	4	5	6	7
error (%)	8.9	4.7	4.0	4.2	4.5	4.5	4.7
avg # SVs	282	237	274	321	374	422	491

Gaussian: $k(x, x') = \exp\left(-\|x - x'\|^2 / (256c)\right)$

c	4.0	2.0	1.2	0.8	0.5	0.2	0.1
error (%)	5.3	5.0	4.9	4.3	4.4	4.4	4.5
avg # SVs	266	240	233	235	251	366	722

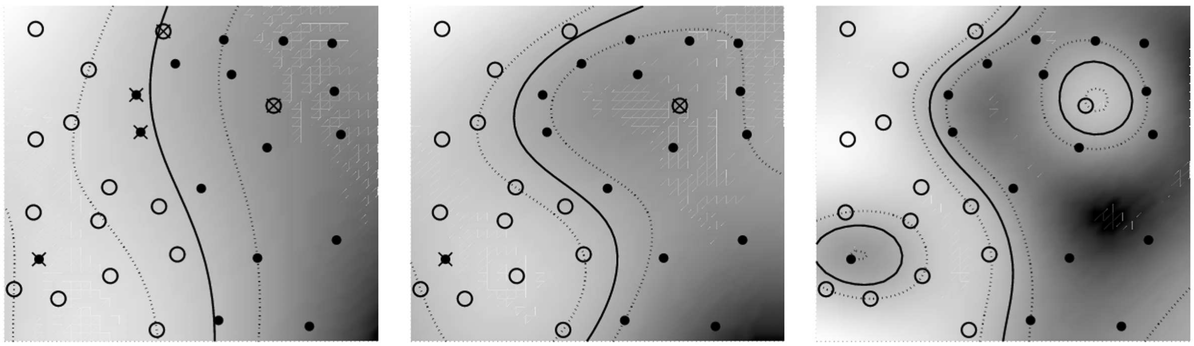
sigmoid: $k(x, x') = \tanh(2 \langle x, x' \rangle / 256 + \Theta)$

$-\Theta$	0.8	0.9	1.0	1.1	1.2	1.3	1.4
error (%)	6.3	4.8	4.1	4.3	4.3	4.4	4.8
avg # SVs	206	242	254	267	278	289	296

- All have comparable min error rates, but sensitive to parameter setting

Parameter Setting

- Gaussian kernel: low, med, high values of c



- How to choose parameter settings?
  - Cross validation
  - Settings that work well for similar problems (rescaled)
  - For ν-SVCs, set ν to e.g. test error from other classifiers
    - $\nu \geq$ margin error $\geq$ train error, which is also $\leq$ test error
  - For C-SVCs, $C \propto 1/R^2$, where R measures range of data in $\mathcal{H}$
    - E.g. $R =$ radius of smallest sphere, max or mean length $k(x_i, x_i)$, or std dev of distance of points to their mean

Overlap of SV Sets

- In the handwritten digit classification experiments, the three kernels typically had 80–93% of their SV sets in common (Table 7.6, p. 220)

- In fact, each kernel got similar error rates when training on SVs of a different kernel rather than the entire training set (Table 7.7)

- Basically, these kernels (dot products) mostly found the same regularities in the data

- Results might vary depending on learning problem/data set

Topic summary due in 1 week!