Machine Learning

CSE 142

Xin (Eric) Wang

Friday, November 5, 2021

T
o
d
a
y

- SVM

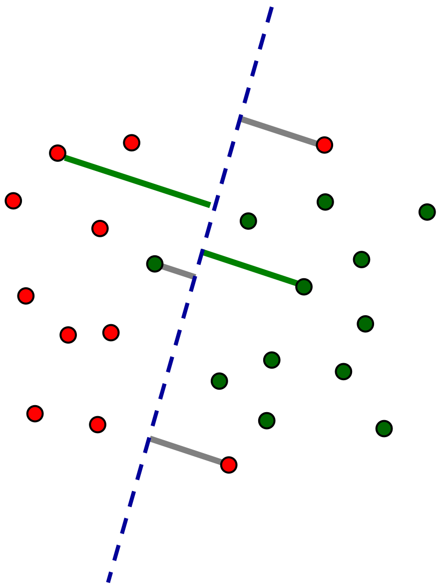
Notes

- HW2 due tonight
  - You should use your ucsc email and ucsc id as the username to register a CodaLab account otherwise we cannot know if you submit the code or not
  - Specify your username in your HW submission to Gradescope
  - You can choose your own teamname to show on the leaderboard for anonymity
- Midterm grades will be out next week

Classifier margin

Review

- The margin (z) of a sample is its distance from the classification boundary
  - Positive if it's correctly classified
  - Negative if it's incorrectly classified



Perceptron margin for point $\boldsymbol{x}$:

$$\boldsymbol{z}(\boldsymbol{x}) = \frac{y(\boldsymbol{w}^T\boldsymbol{x} - t)}{\|\boldsymbol{w}\|} = \frac{m}{\|\boldsymbol{w}\|}$$

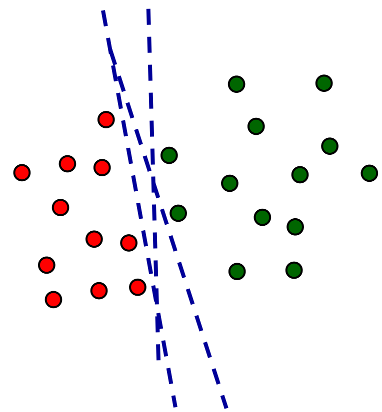
Non-homogeneous representation

Note: m is not the margin; it's the result of plugging $\boldsymbol{x}_i$ into $y(\boldsymbol{w}^T\boldsymbol{x} - t)$

Classifiers and margins

Review

- The class margin (on the training set) is the minimum margin of the data points for that class
- The classifier margin is the sum of the class margins
- There are an infinite number of linear classifiers that can perfectly separate linearly separable data



Infinitely many lines can separate the two classes

- But which (of all these) is the best linear classifier?
  - Perhaps the one that maximizes the classifier margin

Support vector machine (SVM)

- A support vector machine (SVM) is a linear classifier whose decision boundary is a linear combination of the support vectors (training samples at the margins)
- In an SVM, we find classifier parameters $(\boldsymbol{w}, t)$ that maximize the classifier margin
- Since $m = y(\boldsymbol{w}^T\boldsymbol{x} - t)$ and we wish to maximize the margin $\frac{m}{||\boldsymbol{w}||}$, we can instead fix $m = 1$ and minimize $||w||$
  - Provided that none of the training points fall inside the margin
- This leads to a constrained optimization problem:

$$\mathbf{w}^*, t^* = \underset{\mathbf{w}, t}{\arg\min} \frac{1}{2}||\mathbf{w}||^2 \qquad \text{subject to } y_i(\mathbf{w} \cdot \mathbf{x}_i - t) \geq 1, 1 \leq i \leq n$$

  - Then, after some quadratic optimization based on Lagrange multiplier (Page 212-214)….

Support vector machine (SVM)

…we get the following result:

$$\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i \qquad \text{where } \alpha_i \text{ are non-negative reals s.t. } \sum_{i=1}^{n} \alpha_i y_i = 0$$

$\alpha_i > 0$ only for the support vectors!

Other data $\boldsymbol{x}_i$ for which $\alpha_i = 0$ can be removed from the training set without affecting the learned decision boundary

I.e., the decision boundary is defined only by the (typically few) support vectors from the training set – those that are nearest to the decision boundary (at the margin)

And thus the weight vector $\boldsymbol{w}$ is merely a linear combination of the (typically few) support vectors

The threshold t can be found by solving $m = 1 = \boldsymbol{w}^T\boldsymbol{x} - t$ for any support vector $\boldsymbol{x}$

Support vector machine (SVM) in dual form

- How do we find the α_i values?
  - Dual form of the optimization—a function of Lagrange multipliers only
  - Via a quadratic optimization solver!
  - In some simple problems we can do them by hand

$$\alpha_1^*, \ldots, \alpha_n^* = \underset{\alpha_1,\ldots,\alpha_n}{\operatorname{argmax}} \left[-\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \boxed{\mathbf{x}_i \cdot \mathbf{x}_j} + \sum_{i=1}^{n} \alpha_i \right]$$

Note the pairwise dot products between training instances—the entries of the Gram matrix

$$\text{subject to } \alpha_i \geq 0, 1 \leq i \leq n \text{ and } \sum_{i=1}^{n} \alpha_i y_i = 0$$

1. Quadratic optimization to solve for $\alpha_1, \ldots, \alpha_n$
   - Non-zero α_i corresponds to support vector $\boldsymbol{x}_i$
2. Create $\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i$
3. Solve for t by plugging in for any support vector $\boldsymbol{x}_i$

$$m = 1 = \boldsymbol{w}^T \boldsymbol{x}_i - t$$

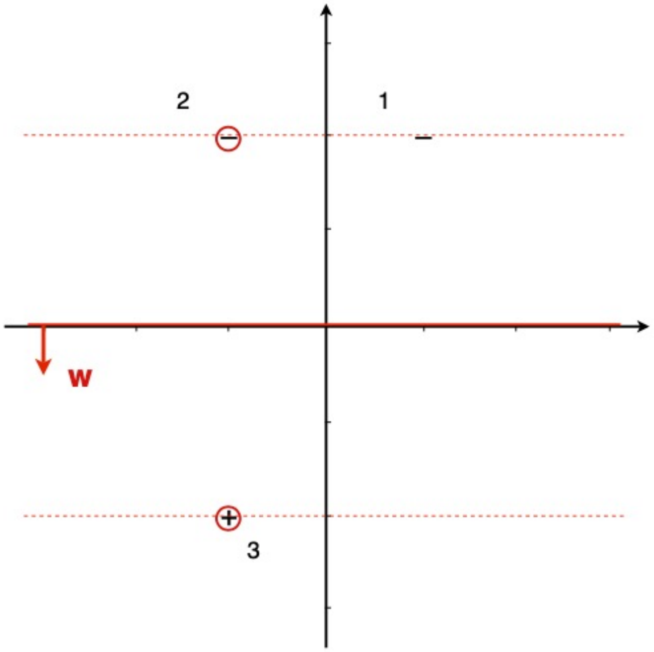
The support vectors $\boldsymbol{x}_i$ fully determine the decision boundary!

SVM classifier example (Fig. 7.8)

$$\mathbf{X} = \begin{pmatrix} 1 & 2 \\ -1 & 2 \\ -1 & -2 \end{pmatrix} \quad \mathbf{y} = \begin{pmatrix} -1 \\ -1 \\ +1 \end{pmatrix} \quad \mathbf{X}' = \begin{pmatrix} -1 & -2 \\ 1 & -2 \\ -1 & -2 \end{pmatrix}$$

- The matrix $\mathbf{X}'$ incorporates the class labels: the rows are $y_i\mathbf{X_i}$
- The Gram matrix is:

$$\mathbf{XX}^{\mathrm{T}} = \begin{pmatrix} 5 & 3 & -5 \\ 3 & 5 & -3 \\ -5 & -3 & 5 \end{pmatrix} \quad \mathbf{X}'\mathbf{X}'^{\mathrm{T}} = \begin{pmatrix} 5 & 3 & 5 \\ 3 & 5 & 3 \\ 5 & 3 & 5 \end{pmatrix}$$



- The dual optimization problem is thus

$$\underset{\alpha_1,\alpha_2,\alpha_3}{\arg\max} -\frac{1}{2}\left(5\alpha_1^2+3\alpha_1\alpha_2+5\alpha_1\alpha_3+3\alpha_2\alpha_1+5\alpha_2^2+3\alpha_2\alpha_3+5\alpha_3\alpha_1+3\alpha_3\alpha_2+5\alpha_3^2\right)+\alpha_1+\alpha_2+\alpha_3$$

$$= \underset{\alpha_1,\alpha_2,\alpha_3}{\arg\max} -\frac{1}{2}\left(5\alpha_1^2+6\alpha_1\alpha_2+10\alpha_1\alpha_3+5\alpha_2^2+6\alpha_2\alpha_3+5\alpha_3^2\right)+\alpha_1+\alpha_2+\alpha_3$$

subject to $\alpha_1 \geq 0, \alpha_2 \geq 0, \alpha_3 \geq 0$ and $-\alpha_1-\alpha_2+\alpha_3=0$.

SVM classifier example (Fig. 7.8)

- Using the equality constraint we can eliminate one of the variables, say α_3, and simplify the objective function to

$$\operatorname*{arg\,max}_{\alpha_1,\alpha_2,\alpha_3} -\frac{1}{2}(20\alpha_1^2 + 32\alpha_1\alpha_2 + 16\alpha_2^2) + 2\alpha_1 + 2\alpha_2$$

- Setting partial derivatives to 0 we obtain $-20\alpha_1 - 16\alpha_2 + 2 = 0$ and $-16\alpha_1 - 16\alpha_2 + 2 = 0$ (notice that, because the objective function is quadratic, these equations are guaranteed to be linear).

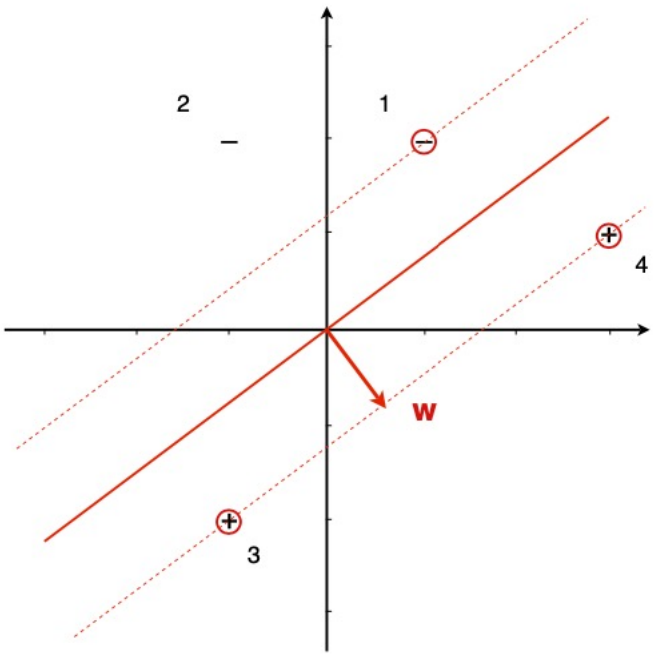
- We therefore obtain the solution $\alpha_1 = 0$ and $\alpha_2 = \alpha_3 = 1/8$. We then have $\mathbf{w} = 1/8(\mathbf{x}_3 - \mathbf{x}_2) = \begin{pmatrix} 0 \\ -1/2 \end{pmatrix}$, resulting in a margin of $1/||\mathbf{w}|| = 2$.

- Finally, t can be obtained from any support vector, say $\mathbf{x}_2$, since $y_2(\mathbf{w} \cdot \mathbf{x}_2 - t) = 1$; this gives $-1 \cdot (-1 - t) = 1$, hence $t = 0$.

SVM classifier example (Fig. 7.8)

We now add an additional positive at (3, 1):

$$\mathbf{X}' = \begin{pmatrix} -1 & -2 \\ 1 & -2 \\ -1 & -2 \\ 3 & 1 \end{pmatrix} \qquad \mathbf{X}'\mathbf{X}'^{\mathrm{T}} = \begin{pmatrix} 5 & 3 & 5 & -5 \\ 3 & 5 & 3 & 1 \\ 5 & 3 & 5 & -5 \\ -5 & 1 & -5 & 10 \end{pmatrix}$$

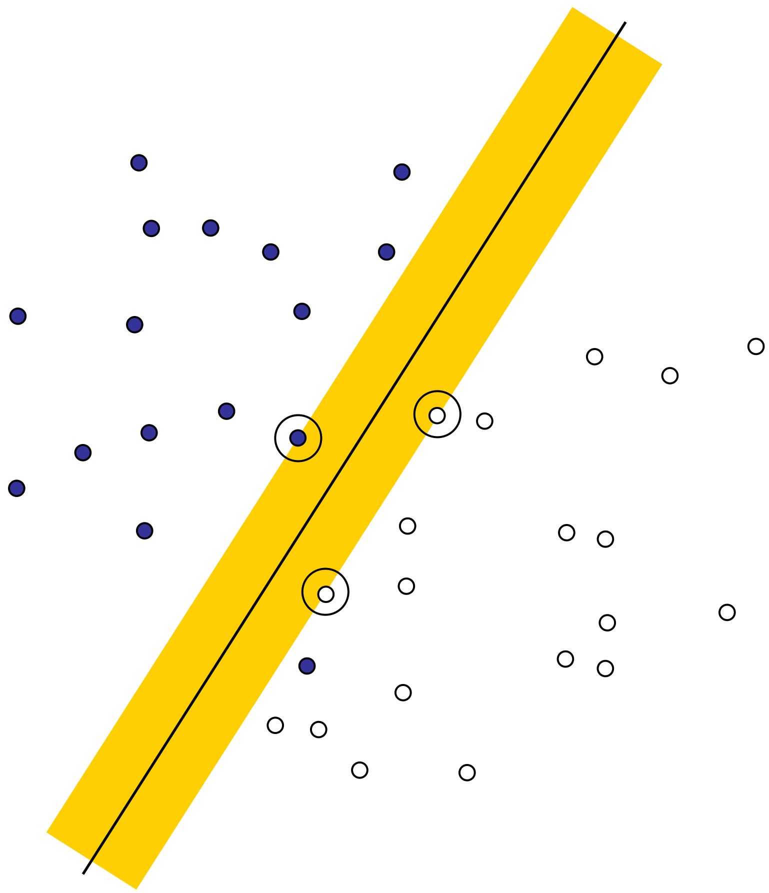


- It can be verified by similar calculations to those above that the margin decreases to 1 and the decision boundary rotates to $\mathbf{w} = \begin{pmatrix} 3/5 \\ -4/5 \end{pmatrix}$.
- The Lagrange multipliers now are $\alpha_1 = 1/2$, $\alpha_2 = 0$, $\alpha_3 = 1/10$ and $\alpha_4 = 2/5$. Thus, only $\mathbf{x}_3$ is a support vector in both the original and the extended data set.

Support vector machine (SVM)

What if the data is not linearly separable?

Or a data point "strays" into an otherwise nice margin?



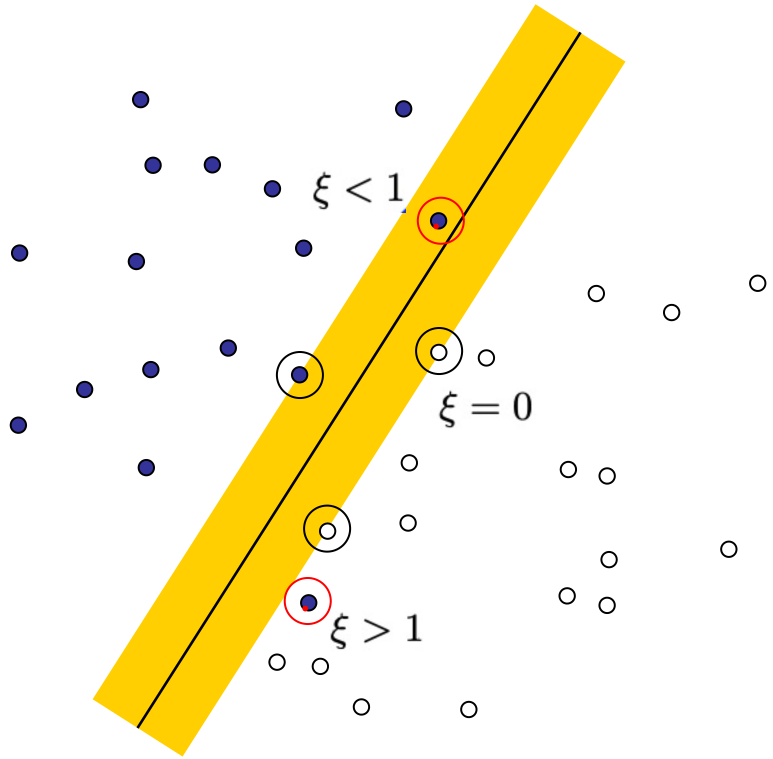
This can be solved with a Soft Margin SVM

Soft margin SVM

ξ_i = "xi"

- We introduce a slack variable ξ_i for each training example to account for margin errors
  - Points that are inside the margin
  - Points that are on the wrong side of the decision boundary

$$\boldsymbol{w}^T \boldsymbol{x}_i - t \geq 1 - \xi_i$$



Soft margin SVM

ξ_i = "xi"

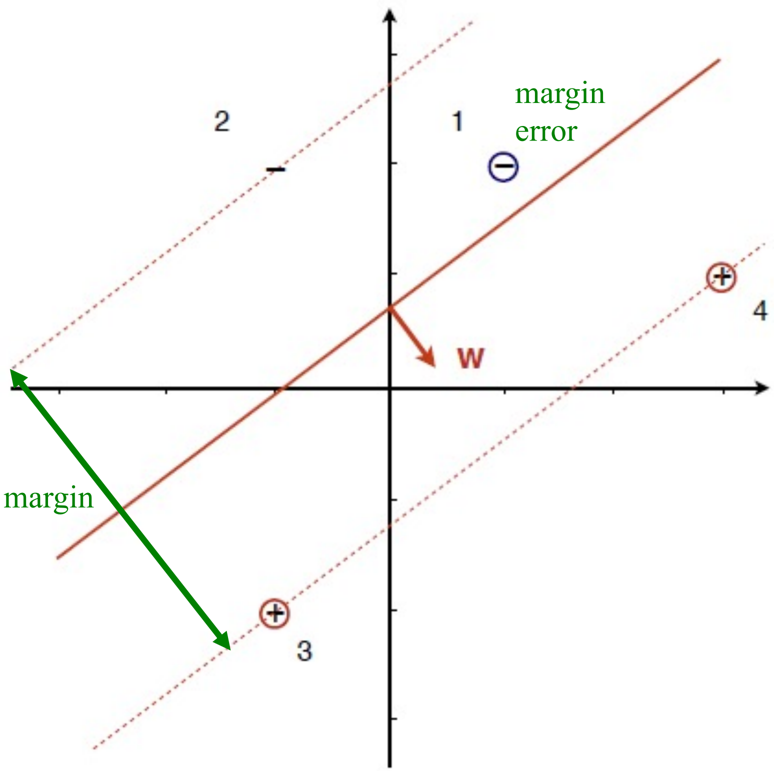
- Results in the soft margin optimization problem:

$$\mathbf{w}^*, t^*, \xi_i^* = \underset{\mathbf{w}, t, \xi_i}{\arg\min} \left[\frac{1}{2} ||\mathbf{w}||^2 + C \sum_{i=1}^{n} \xi_i \right]$$

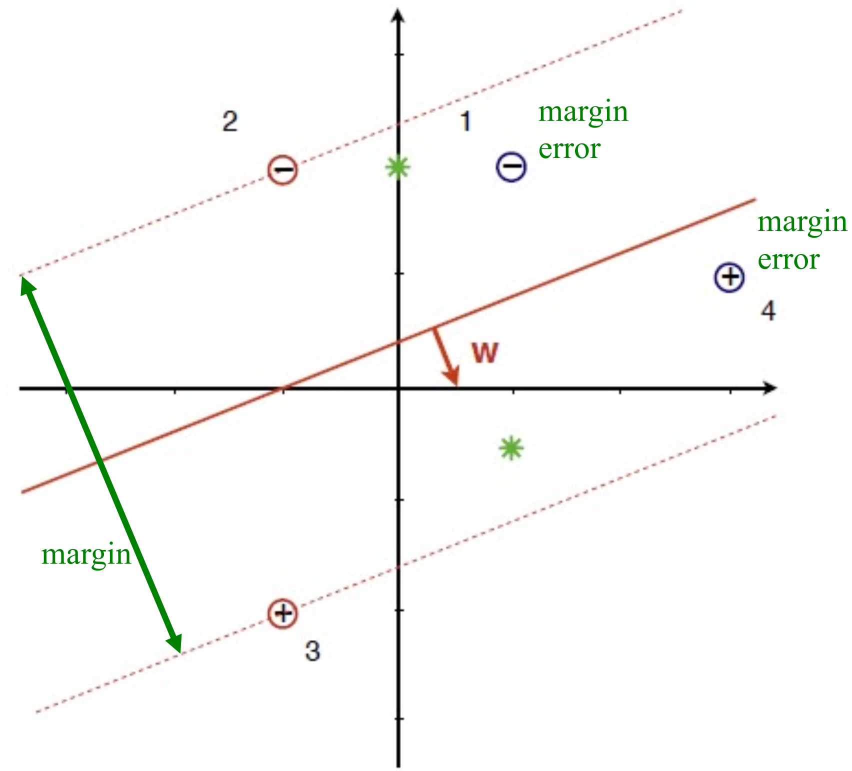
$$\text{subject to } y_i(\mathbf{w} \cdot \mathbf{x}_i - t) \geq 1 - \xi_i \text{ and } \xi_i \geq 0, 1 \leq i \leq n$$

- The complexity parameter C is a user-defined parameter that allows for a tradeoff between maximizing the margin (lower C) and minimizing the margin errors (higher C)
  - A high value of C means that margin errors incur a high penalty
  - A low value permits more margin errors (possibly including misclassifications) in order to achieve a large margin
  - *Note that when $C = 0$, this gives no penalty to outliers – which makes it equivalent to our basic linear classifier!*

Soft margin SVM



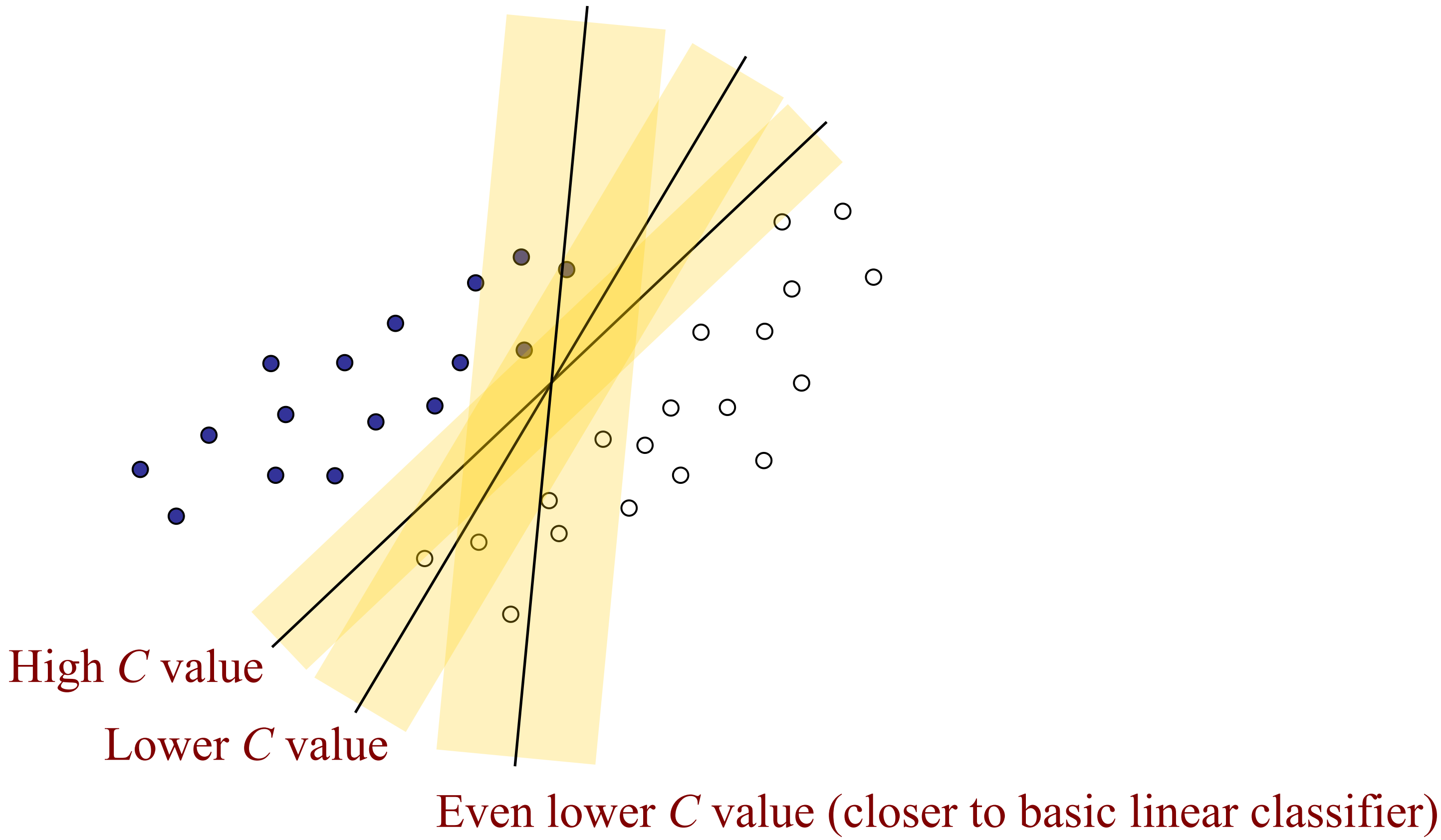
$C = \frac{5}{16}$ Smaller margin
Fewer margin errors



$C = \frac{1}{10}$ Larger margin
More margin errors

Soft margin SVM

A minimal-complexity (low *C*) soft margin classifier summarizes the classes by their class means in a way very similar to the basic linear classifier



Quiz: Perceptron and SVM

- For both perceptron and SVM classifiers, $\boldsymbol{w} = \sum_{\boldsymbol{i}} \alpha_i y_i \boldsymbol{x}_i$, how are α_i and $\boldsymbol{x}_i$ different in perceptron and SVM?

Perceptron and SVM binary classifiers – summary

- In the perceptron model, we iteratively learn the linear discriminant $\boldsymbol{w}$, which is a linear combination of the misclassified input vectors $\boldsymbol{x}_i$

$$\boldsymbol{w} = \sum_{\boldsymbol{i}} \alpha_i y_i \boldsymbol{x}_i$$

α_i – # of times $\boldsymbol{x}_i$ was misclassified
y_i – class label of $\boldsymbol{x}_i$ $\{+1, -1\}$

  – After training, a new input is classified as a member of the positive class if $\boldsymbol{w}^T\boldsymbol{x} > 0$ (using homogeneous representation)

- In SVM learning, we solve a constrained optimization problem:

$$\alpha_1^*, \ldots, \alpha_n^* = \underset{\alpha_1, \ldots, \alpha_n}{\operatorname{argmax}} \left[-\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j + \sum_{i=1}^{n} \alpha_i \right]$$

subject to $\alpha_i \geq 0, 1 \leq i \leq n$ and $\sum_{i=1}^{n} \alpha_i y_i = 0$

which leads us to $\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i$ where $\alpha_i = 0$ except for the support vectors

Non-homogeneous

Perceptron and SVM binary classifiers – summary

- In both perceptron and SVM learning, the linear decision boundary is a linear combination of the training data points
  - In the perceptron, just the ones that get misclassified in the iterative training
  - In the SVM, just the (few) support vectors

- Both learning methods have a dual form in which the dot product of training data points $\boldsymbol{x}_i^T \boldsymbol{x}_j$ is part of the main computation
  - All values of $\boldsymbol{x}_i^T \boldsymbol{x}_j$ are contained in the Gram matrix

$$\boldsymbol{G} = \boldsymbol{X}^T \boldsymbol{X} = [\boldsymbol{x}_1 \quad \boldsymbol{x}_2 \quad \dots \quad \boldsymbol{x}_k]^T [\boldsymbol{x}_1 \quad \boldsymbol{x}_2 \quad \dots \quad \boldsymbol{x}_k]$$

  so it's often efficient to compute the Gram matrix in advance and index into it, rather than computing the dot products over and over again

Perceptron and SVM binary classifiers – summary

- Perceptron and (basic) SVM learning only converge to a solution if the training data is linearly separable
- If the data is not linearly separable, we can employ a soft margin SVM, where we introduce a *slack variable* ξ_i for each training data point, allowing for margin errors:

$$\boldsymbol{w}^T \boldsymbol{x}_i - t \geq 1 - \xi_i \qquad \xi_i > 0 \;\rightarrow\; \boldsymbol{x}_i \text{ is not a support vector}$$

and leading to this optimization problem:

$$\mathbf{w}^*, t^*, \xi_i^* = \underset{\mathbf{w}, t, \xi_i}{\arg\min} \left[\frac{1}{2} ||\mathbf{w}||^2 + C \sum_{i=1}^{n} \xi_i \right]$$

$$\text{subject to } y_i(\mathbf{w} \cdot \mathbf{x}_i - t) \geq 1 - \xi_i \text{ and } \xi_i \geq 0, 1 \leq i \leq n$$

where the complexity parameter C is a user-defined parameter that allows for a tradeoff between maximizing the margin (lower C) and minimizing the margin errors (higher C)