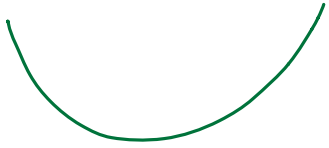


Linear classification

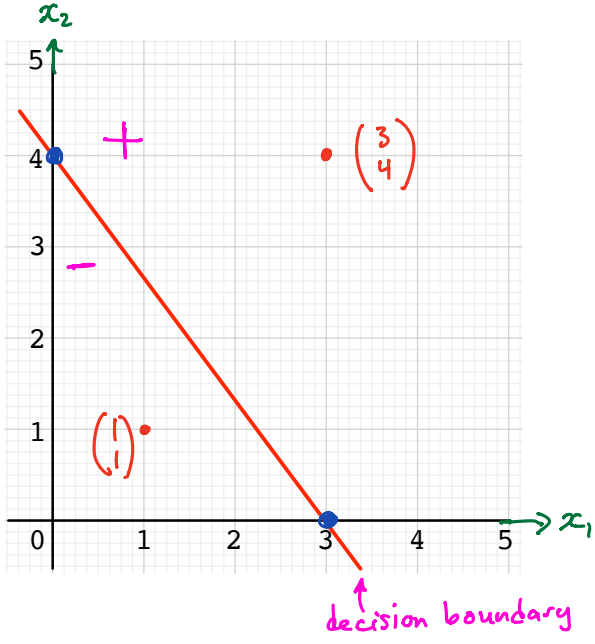
DSE 220

Outline

1. Linear decision boundary for binary classification
2. The Perceptron algorithm
3. Maximizing the margin
4. The soft-margin SVM

Linear decision boundary for classification: example

$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$



<linear function> = 0

Slope = $\frac{-4}{3}$

$x_2 = -\frac{4}{3} x_1 + 4$

$\Leftrightarrow \quad 4x_1 + 3x_2 - 12 = 0$

Classify $x = \begin{pmatrix} 3 \\ 4 \end{pmatrix}$

Plug into $4x_1 + 3x_2 - 12$:

$4 \cdot 3 + 3 \cdot 4 - 12 = 12 > 0$ } predict +1

For $x = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$:

$4 \cdot 1 + 3 \cdot 1 - 12 = -5 < 0$ } -1

- What is the formula for this boundary?
- What label would we predict for a new point x?

Prediction:

SIGN $(4x_1 + 3x_2 - 12)$

Linear classifiers

Binary classification problem: data $x \in \mathbb{R}^d$ and labels $y \in \{-1, +1\}$

- Linear classifier:
  - Parameters: $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$
  - Decision boundary $w \cdot x + b = 0$
  - On point x, predict label $\text{sign}(w \cdot x + b)$
- If the true label on point x is y:
  - Classifier correct if $y(w \cdot x + b) > 0$

Say true label for x is y.

Our prediction is $\text{sign}(w \cdot x + b)$

Correct $\iff$ $y = \text{sign}(w \cdot x + b)$

$\iff$ EITHER $y = +1$ and $w \cdot x + b > 0$

OR $y = -1$ and $w \cdot x + b < 0$

$\iff$ $y(w \cdot x + b) > 0$

Outline

1. Linear decision boundary for binary classification
2. The Perceptron algorithm
   - A loss function for classification
   - A stochastic gradient descent approach
   - The Perceptron
   - Convergence
3. Maximizing the margin
4. The soft-margin SVM

A loss function for classification

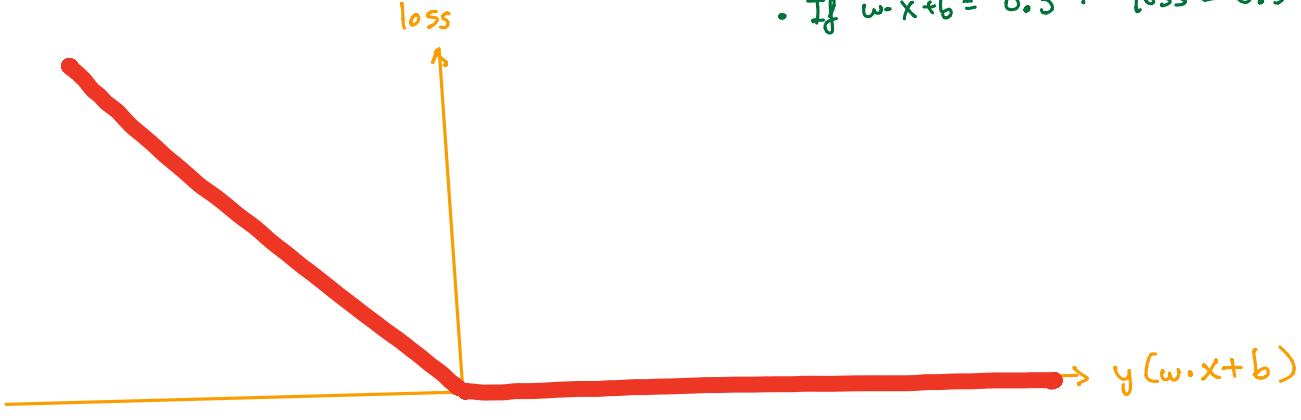
What is the **loss** of our linear classifier (given by w, b) on a point (x, y)?

One idea for a loss function:

- If $y(w \cdot x + b) > 0$: correct, no loss
- If $y(w \cdot x + b) < 0$: loss $= -y(w \cdot x + b)$

E.g. $y = -1$

- If $w \cdot x + b = -3$: correct prediction, no loss
- If $w \cdot x + b = 3$: loss = 3
- If $w \cdot x + b = 0.3$: loss = 0.3



A simple learning algorithm

Fit a linear classifier w, b to the training set using **stochastic gradient descent**.

- Update w, b based on just one data point (x, y) at a time
- If $y(w \cdot x + b) > 0$: zero loss, no update (correct on (x,y))
- If $y(w \cdot x + b) \leq 0$: loss is $-y(w \cdot x + b)$ (wrong on (x,y))

Wrong on (x,y):

Loss $\ell = -y(w \cdot x + b)$

$\frac{d\ell}{dw} = -yx$

$\frac{d\ell}{db} = -y$

learning rate

SGD update

$w \leftarrow w + \eta\, yx$

$b \leftarrow b + \eta\, y$

just use $\eta = 1$

A simple learning algorithm

Fit a linear classifier w, b to the training set using **stochastic gradient descent**.

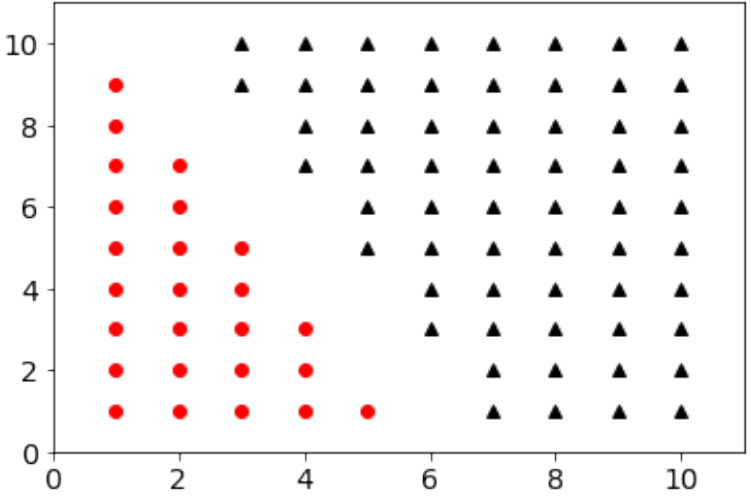
- Update w, b based on just one data point (x, y) at a time
- If $y(w \cdot x + b) > 0$: zero loss, no update
- If $y(w \cdot x + b) \leq 0$: loss is $-y(w \cdot x + b)$

The Perceptron algorithm

- Initialize $w = 0$ and $b = 0$
- Keep cycling through the training data (x, y):
  - If $y(w \cdot x + b) \leq 0$ (i.e. point misclassified):
    - $w = w + yx$
    - $b = b + y$

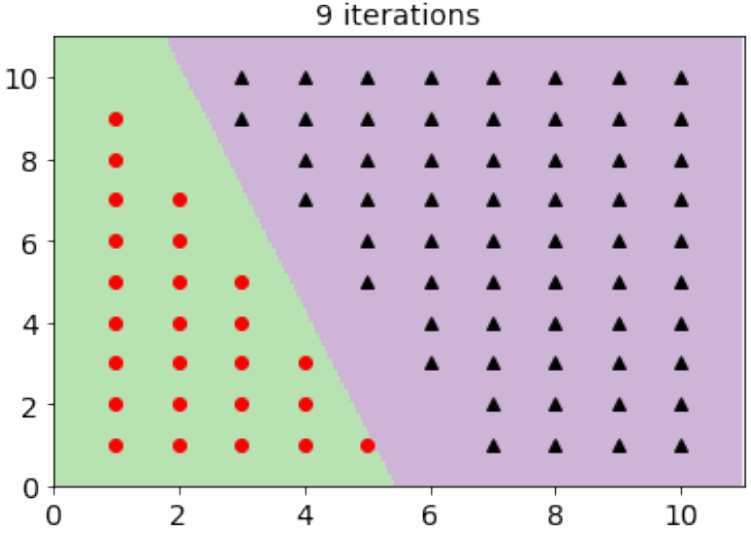
The Perceptron in action

85 data points, linearly separable.



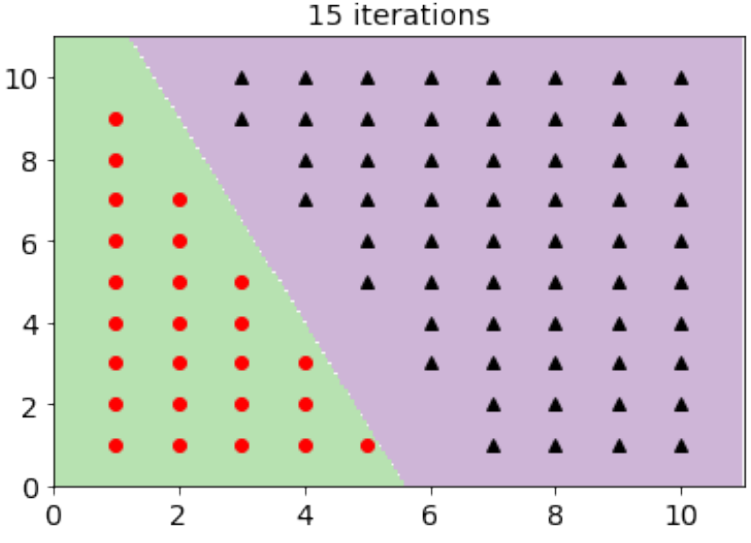
The Perceptron in action

85 data points, linearly separable.



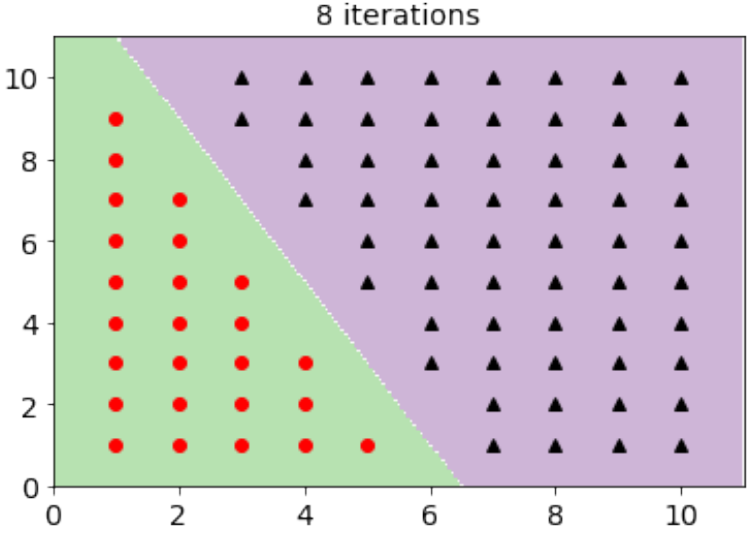
The Perceptron in action

85 data points, linearly separable.



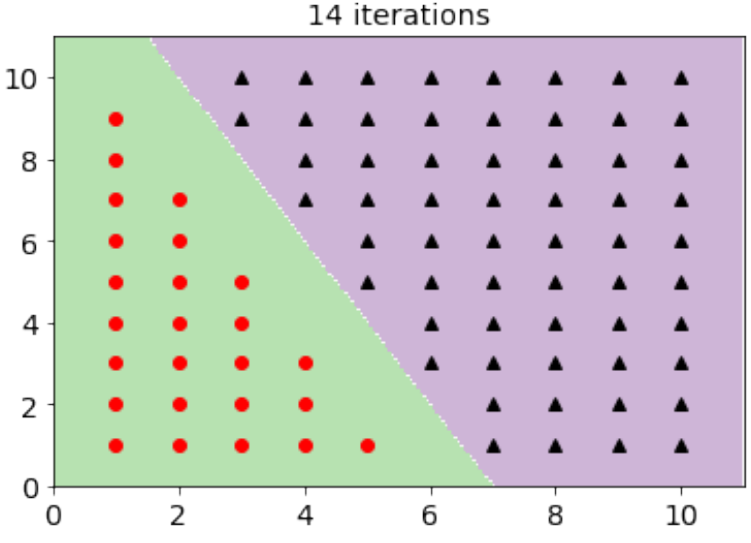
The Perceptron in action

85 data points, linearly separable.



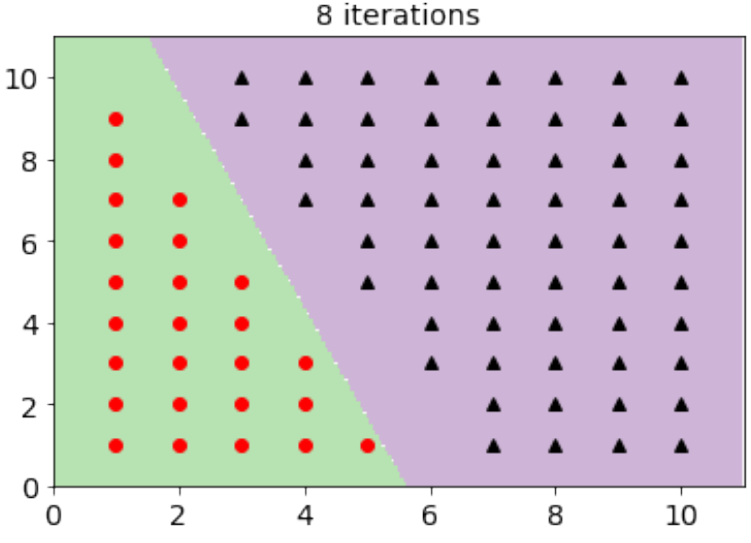
The Perceptron in action

85 data points, linearly separable.



The Perceptron in action

85 data points, linearly separable.



Perceptron: convergence

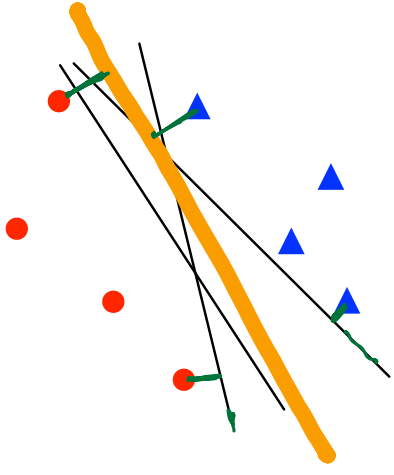
If the training data is linearly separable:

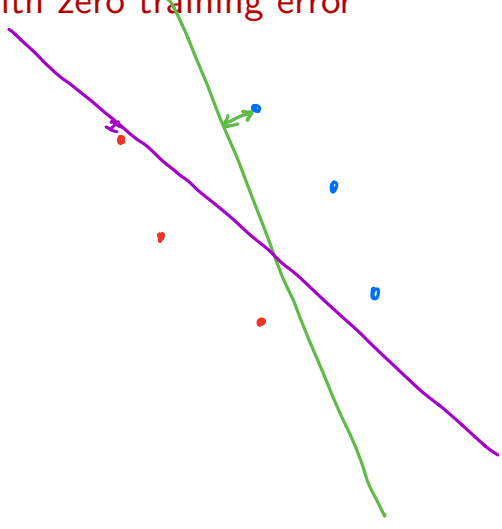
- The Perceptron algorithm will find a linear classifier with zero training error
- It will converge within a finite number of steps.

Perceptron: convergence

If the training data is linearly separable:

- The Perceptron algorithm will find a linear classifier with zero training error
- It will converge within a finite number of steps.





But is there a better, more systematic choice of separator?

Outline

Worksheet 9 # 1, 2, 3

Lab 2 # 1

1. Linear decision boundary for binary classification
2. The Perceptron algorithm
3. Maximizing the margin
   - The margin of a linear classifier
   - Maximizing the margin
   - A convex optimization problem
   - Support vectors
4. The soft-margin SVM

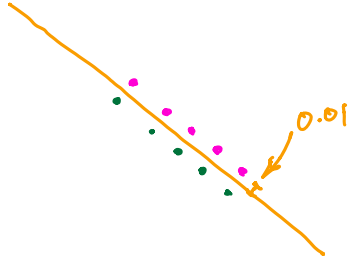
Support vector machines

The learning problem

Given: training data $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)}) \in \mathbb{R}^d \times \{-1, +1\}$.

Find: $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$ such that $y^{(i)}(w \cdot x^{(i)} + b) > 0$ for all i.

The learning problem



Given: training data $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)}) \in \mathbb{R}^d \times \{-1, +1\}$.

Find: $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$ such that $y^{(i)}(w \cdot x^{(i)} + b) > 0$ for all i.

By scaling w, b, can equivalently ask for

multiply w, b by a suitable factor

$$y^{(i)}(w \cdot x^{(i)} + b) \geq 1 \quad \text{for all } i$$

If $y^{(i)}(w \cdot x^{(i)} + b) > 0$ for all data points i,
then multiplying w, b by a scaling factor will give
$y^{(i)}(w \cdot x^{(i)} + b) \geq 1$ for all i.

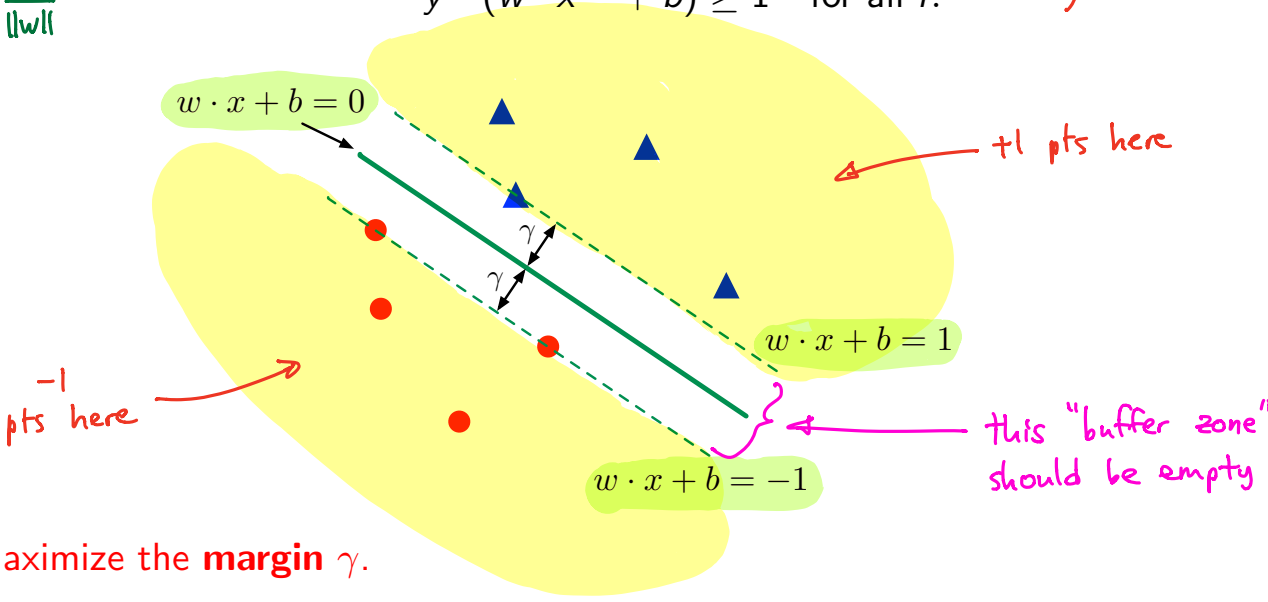
Maximizing the margin

Given: training data $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)}) \in \mathbb{R}^d \times \{-1, +1\}$.
Find: $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$ such that

$$y^{(i)}(w \cdot x^{(i)} + b) \geq 1 \quad \text{for all } i.$$

} data is correctly classified

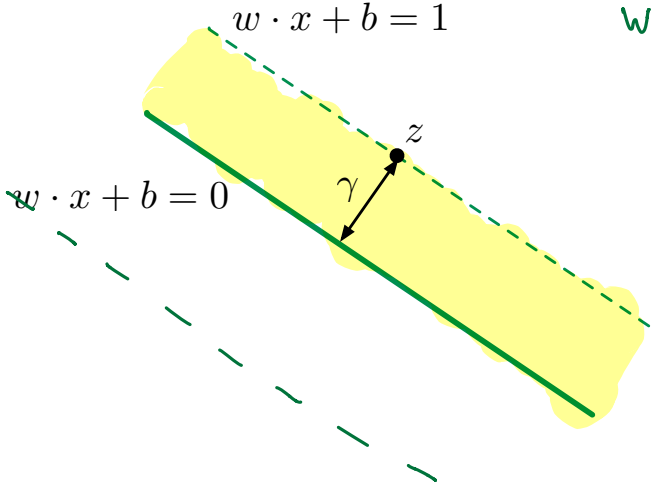
$\gamma = \frac{1}{\|w\|}$



Maximize the **margin** γ.

A formula for the margin

Close-up of a point z on the positive boundary.



A quick calculation shows that $\gamma = 1/\|w\|$.

In short: to maximize the margin, minimize $\|w\|$.

$$\|w\| = \sqrt{w_1^2 + w_2^2 + \cdots + w_d^2}$$

$$w = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{pmatrix}$$

Goal: maximize margin γ

$\equiv$ minimize $\|w\|$

$\equiv$ minimize $\|w\|^2$

Maximum-margin linear classifier

- Given $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)}) \in \mathbb{R}^d \times \{-1, +1\}$

$$\begin{aligned} &\min_{w \in \mathbb{R}^d, b \in \mathbb{R}} \quad \|w\|^2 \\ &\text{s.t.:} \quad y^{(i)}(w \cdot x^{(i)} + b) \geq 1 \quad \text{for all } i = 1, 2, \ldots, n \end{aligned}$$

Maximum-margin linear classifier

convex function $w_1^2 + w_2^2 + \ldots + w_d^2$

- Given $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)}) \in \mathbb{R}^d \times \{-1, +1\}$

this is the optimization problem we want to solve

$$\min_{w \in \mathbb{R}^d, b \in \mathbb{R}} \|w\|^2$$

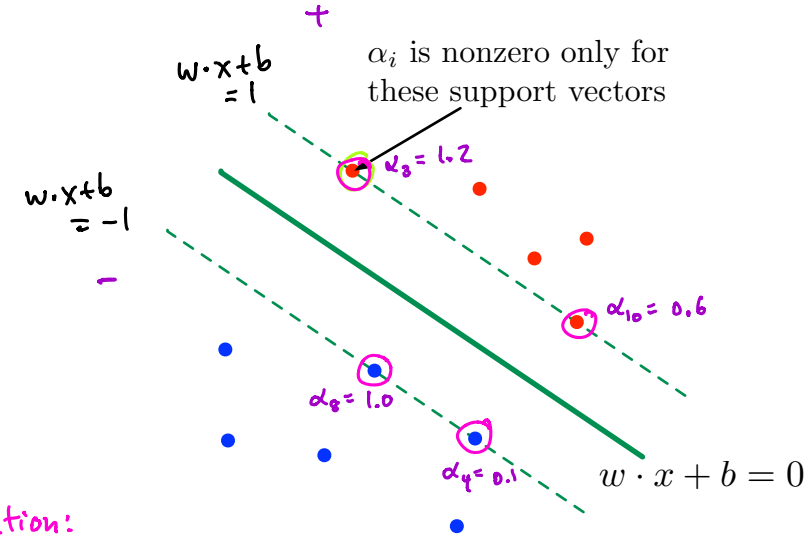
$$\text{s.t.: } y^{(i)}(w \cdot x^{(i)} + b) \geq 1 \quad \text{for all } i = 1, 2, \ldots, n$$

- This is a **convex optimization problem**:
  - Convex objective function
  - Linear constraints

- This means that:
  - the optimal solution can be found efficiently
  - **duality** gives us information about the solution

Support vectors

Support vectors: training points that lie exactly on the margin, i.e. $y^{(i)}(w \cdot x^{(i)} + b) = 1$.



$W = 1.2\, x^{(3)} + 0.6\, x^{(10)} - 1.0\, x^{(8)} - 0.1\, x^{(4)}$

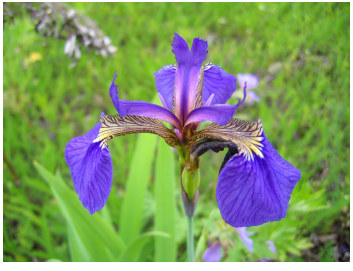
Form of the solution:

$w = \sum_{i=1}^{n} \alpha_i y^{(i)} x^{(i)}$ is a function of just the support vectors.

← these coefficients are all ≥ 0 and they > 0 only on support vectors.

Small example: Iris data set

Fisher's **iris** data



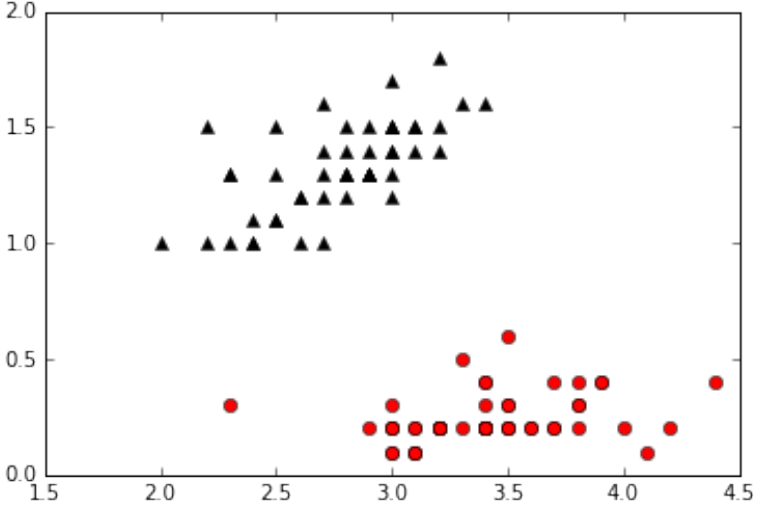
150 data points from three classes:

- iris setosa
- iris versicolor
- iris virginica

Four measurements: petal width/length, sepal width/length

Small example: Iris data set

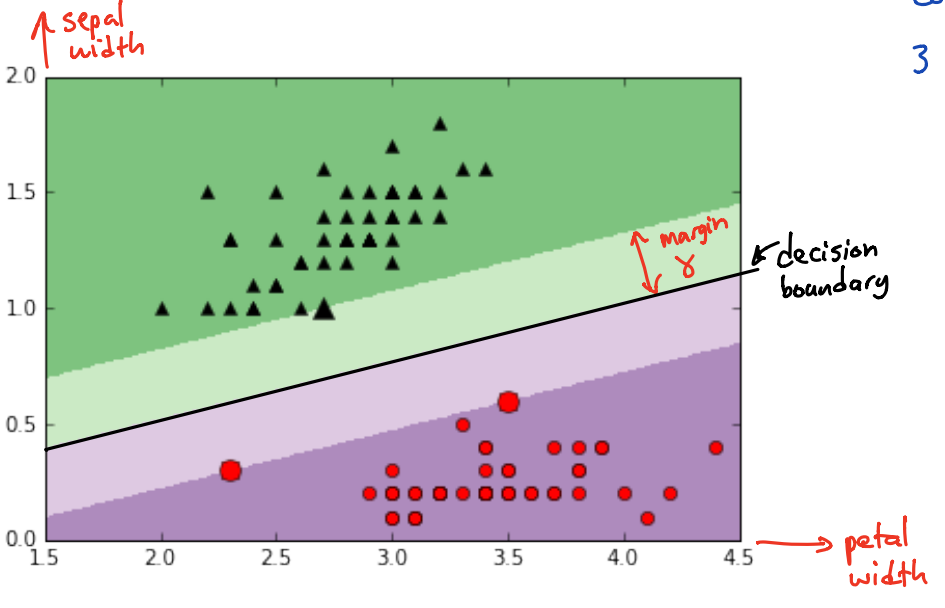
Two features: sepal width, petal width.
Two classes: setosa (red circles), versicolor (black triangles)



Small example: Iris data set

Two features: sepal width, petal width.

Two classes: setosa (red circles), versicolor (black triangles)



3 support vectors

⇒ w is a linear combination of those 3 data points

Outline

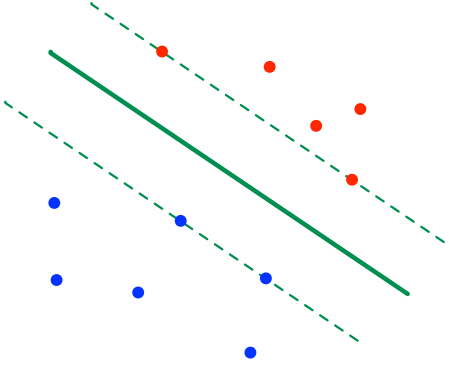
1. Linear decision boundary for binary classification
2. The Perceptron algorithm
3. Maximizing the margin
4. The soft-margin SVM ← the version of the SVM that is used in practice
   - Data that isn't linearly separable
   - Adding slack variables for each point
   - Revised convex optimization problem
   - Setting the slack parameter

Recall: maximum-margin linear classifier

Given: $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)}) \in \mathbb{R}^d \times \{-1, +1\}$.
Find: the linear separator w that perfectly classifies the data and has maximum margin.

$$\min_{w \in \mathbb{R}^d, b \in \mathbb{R}} \|w\|^2$$

s.t.: $y^{(i)}(w \cdot x^{(i)} + b) \geq 1 \quad \text{for all } i = 1, 2, \ldots, n$



Solution $w = \sum_{i=1}^{n} \alpha_i y^{(i)} x^{(i)}$ is a function of just the support vectors.

Recall: maximum-margin linear classifier

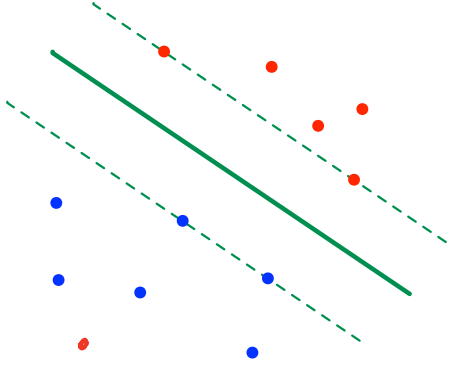
Given: $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)}) \in \mathbb{R}^d \times \{-1, +1\}$.

Find: the linear separator w that perfectly classifies the data and has maximum margin.

$$\min_{w \in \mathbb{R}^d, b \in \mathbb{R}} \|w\|^2$$

$$\text{s.t.: } y^{(i)}(w \cdot x^{(i)} + b) \geq 1 \quad \text{for all } i = 1, 2, \ldots, n$$

no longer possible to satisfy all these constraints; we must give up on some of them...



Solution $w = \sum_{i=1}^{n} \alpha_i y^{(i)} x^{(i)}$ is a function of just the support vectors.

What if data is not separable?

The non-separable case

maximize margin as before

minimize total amount of slack used

$$\min_{w \in \mathbb{R}^d, b \in \mathbb{R}, \xi \in \mathbb{R}^n} \|w\|^2 + C \sum_{i=1}^{n} \xi_i$$

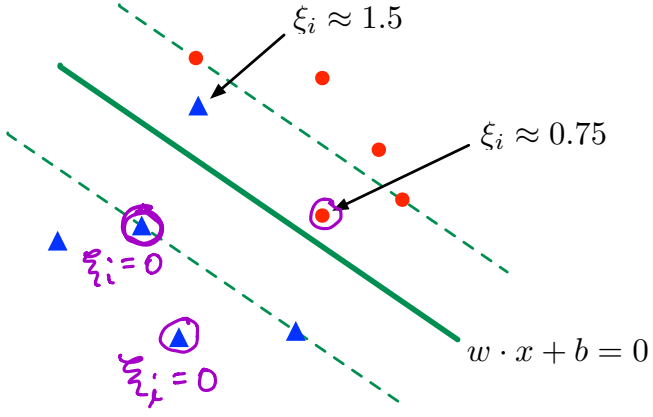
s.t.: $y^{(i)}(w \cdot x^{(i)} + b) \geq 1 - \xi_i$ for all $i = 1, 2, \ldots, n$

$\xi \geq 0$

Allow each data point $x^{(i)}$ some **slack** ξ_i.

the extent to which the constraint on $(x^{(i)}, y^{(i)})$ is violated

$\xi \approx 3$

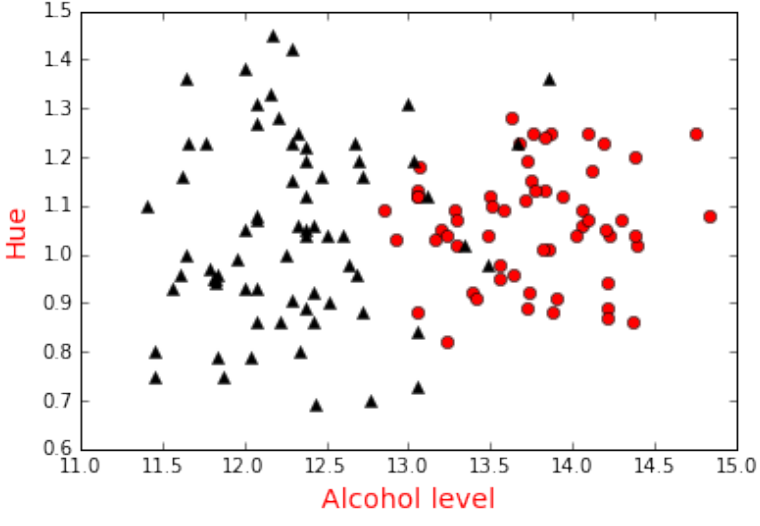


$x_i : \xi$

total slack used is $\sum_{i=1}^{n} \xi_i \approx 2.25$

Wine data set

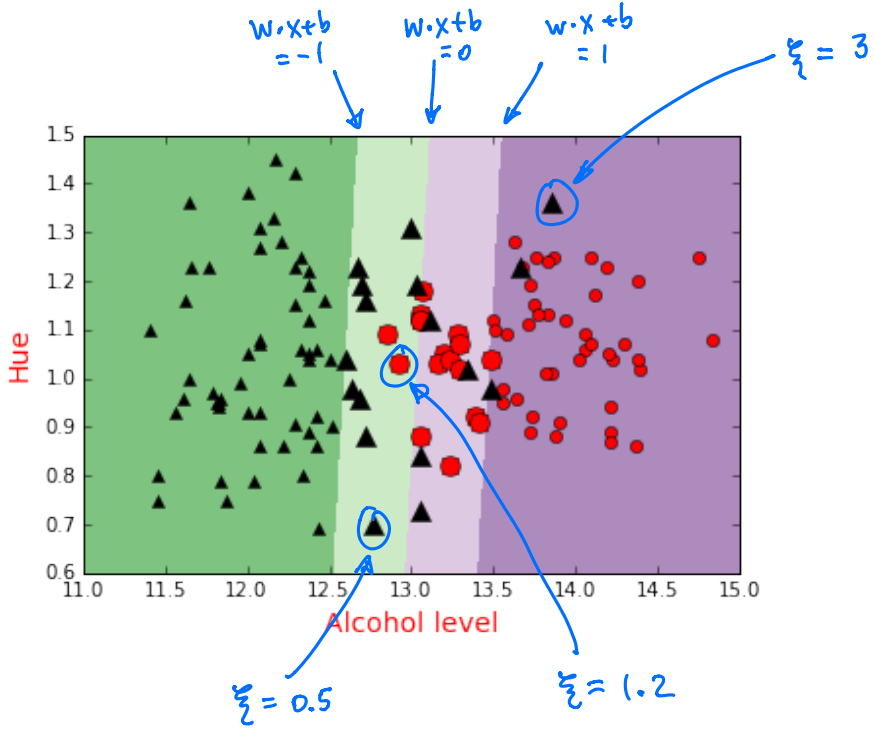
Here $C = 1.0$



Wine data set

Here $C = 1.0$

Support vectors = all points right on the margin AND all points on which we have used slack



The tradeoff between margin and slack

What effect does this have?

$$\min_{w \in \mathbb{R}^d, b \in \mathbb{R}, \xi \in \mathbb{R}^n} \quad \|w\|^2 + C \sum_{i=1}^{n} \xi_i$$

$$\text{s.t.:} \quad y^{(i)}(w \cdot x^{(i)} + b) \geq 1 - \xi_i \quad \text{for all } i = 1, 2, \ldots, n$$

$$\xi \geq 0$$

$C = 0$

Slack is FREE

Misclassify as many pts as we like

We'll end up with $w = 0$ (infinite margin)

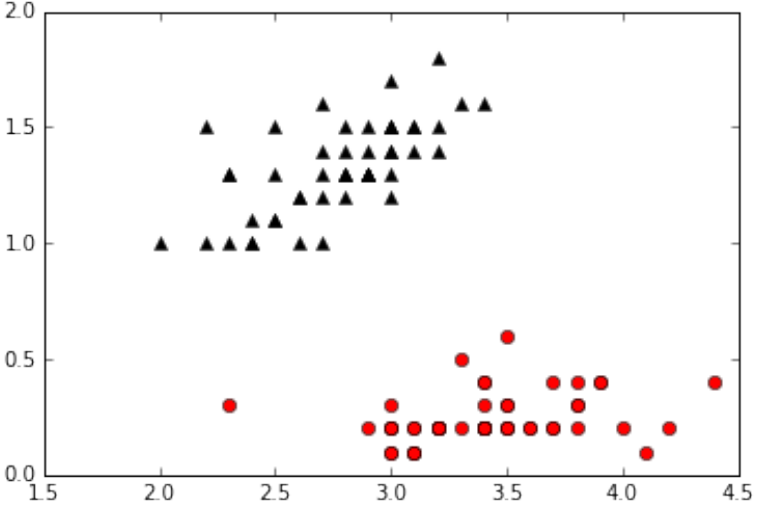
$C \to \infty$

Slack is EXPENSIVE

Like the hard-margin SVM that assumed separability

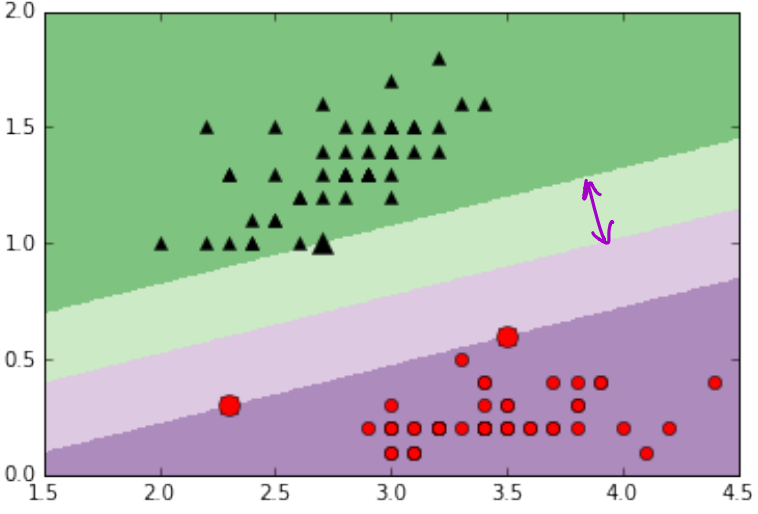
Back to Iris

$C = 10$



Back to Iris

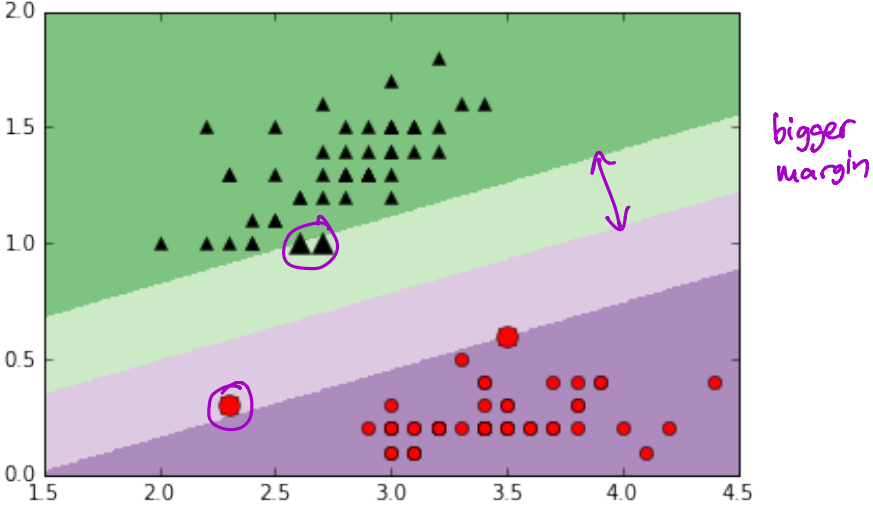
$C = 10$



no slack used

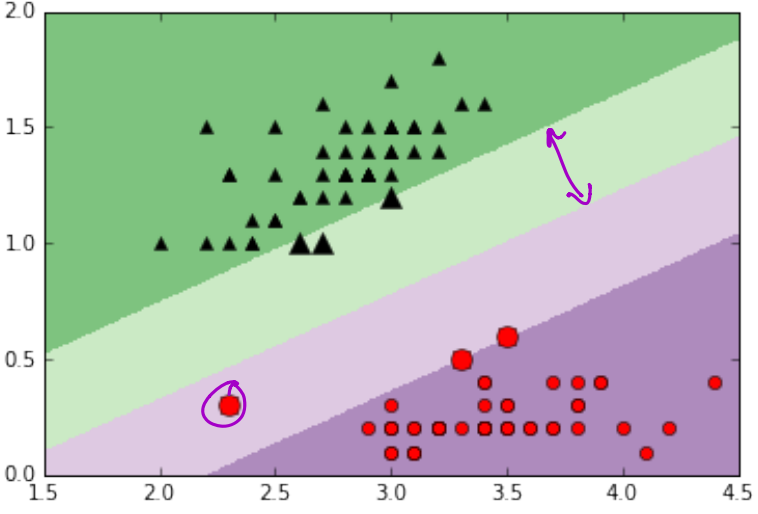
Back to Iris

$C = 3$



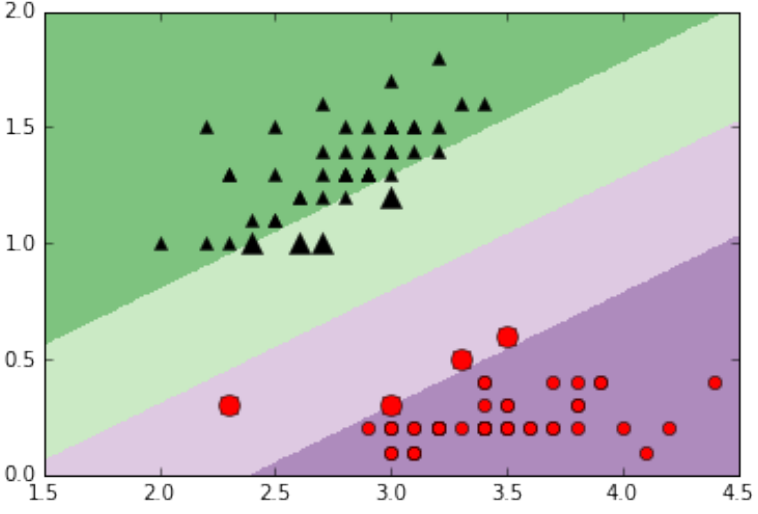
Back to Iris

$C = 2$



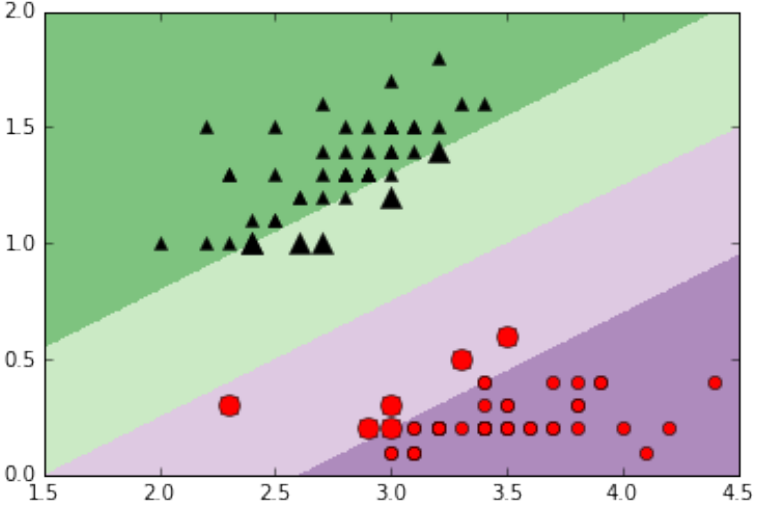
Back to Iris

$C = 1$



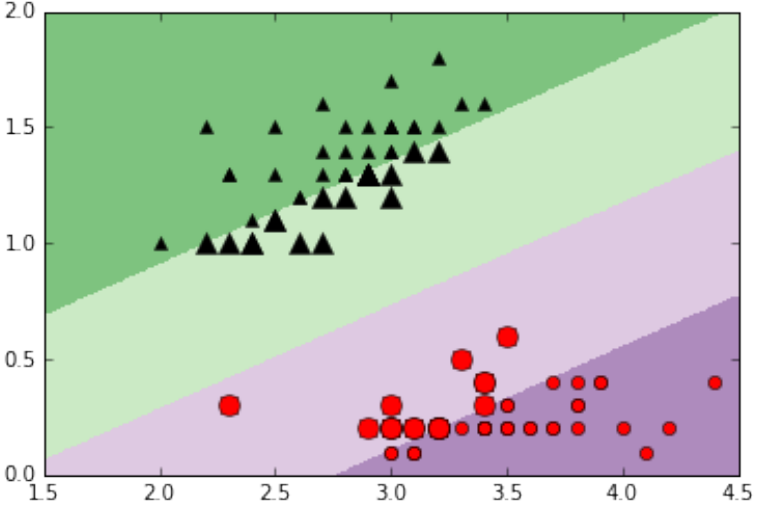
Back to Iris

$C = 0.5$



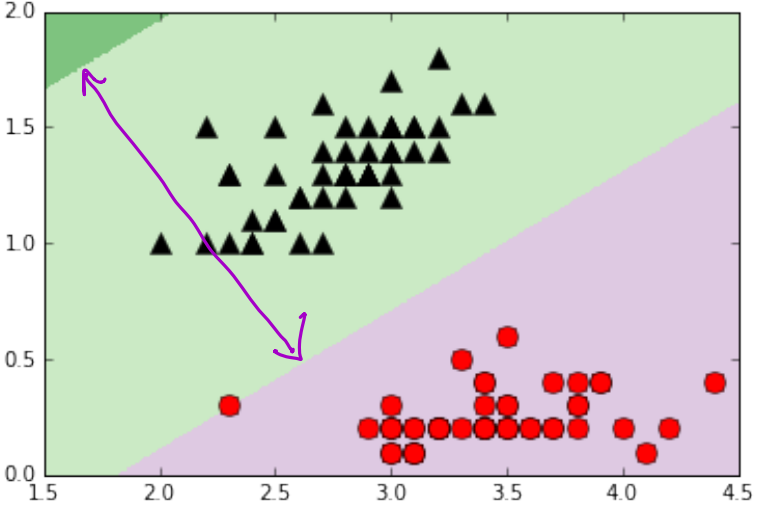
Back to Iris

$C = 0.1$



Back to Iris

$C = 0.01$



Sentiment data

Sentences from reviews on Amazon, Yelp, IMDB, each labeled as positive or negative.

- `Needless to say, I wasted my money.`
- `He was very impressed when going from the original battery to the extended battery.`
- `I have to jiggle the plug to get it to line up right to get decent volume.`
- `Will order from them again!` +1

Dimension $d = 4500$

Num pts $n = 2500$

Data details:

- Bag-of-words representation using a vocabulary of size 4500
- 2500 training sentences, 500 test sentences

What C to use?

As C grows, the charge for violations grows
$\Rightarrow$ fewer violations in training set
$\Rightarrow$ smaller training error

C	training error (%)	test error (%)	# support vectors
0.01	23.72	28.4	2294
0.1	7.88	18.4	1766
1	1.12	16.8	1306
10	0.16	19.4	1105
100	0.08	19.4	1035
1000	0.08	19.4	950

How to choose C?

Cross-validation.

Why is this decreasing as C grows?

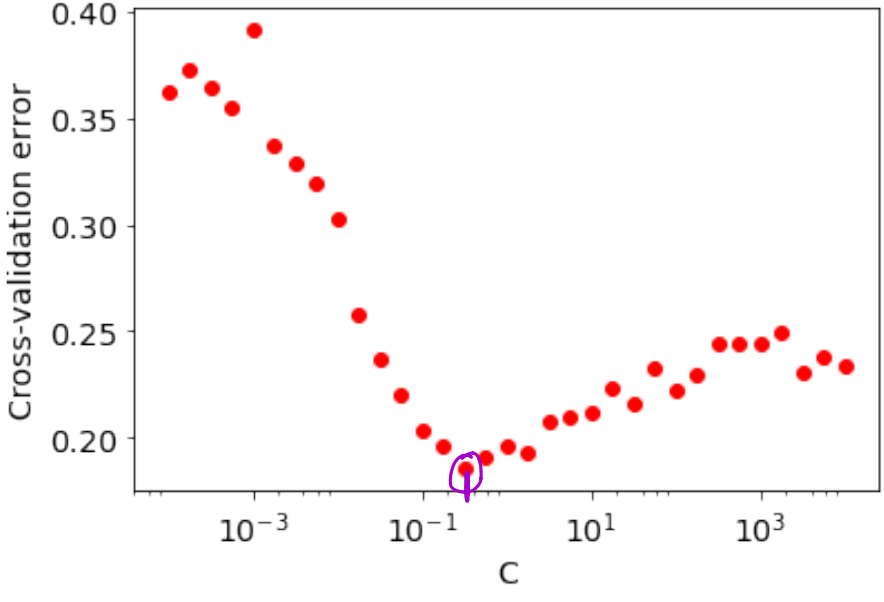
Fewer violations (every violation is a support vector)

Cross-validation

Training set size = 2500

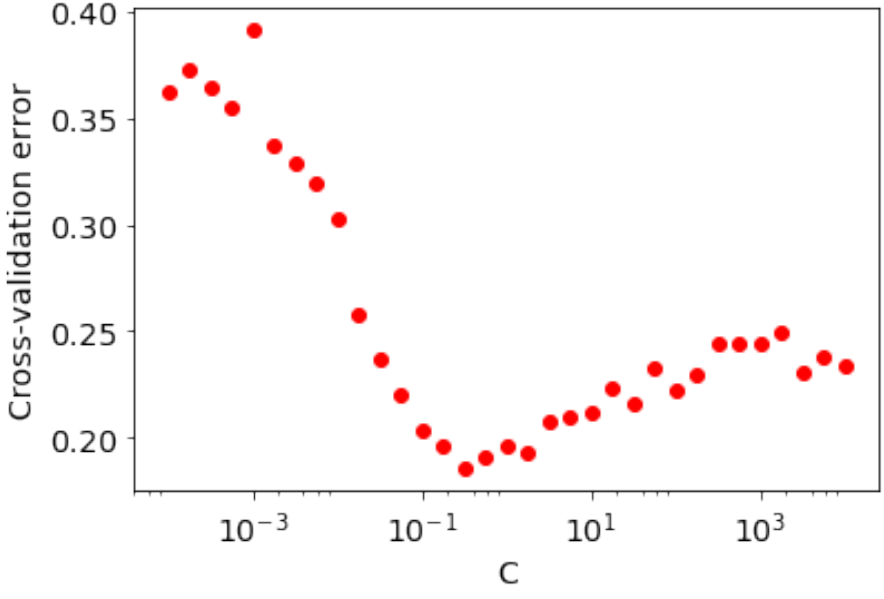
Results of 5-fold cross-validation:

5 trials;
each validation set has size 500



Cross-validation

Results of 5-fold cross-validation:



Chose $C = 0.32$. Test error: 15.6%

(recall LR error was > 21%)

Worksheet 9: all

Lab 2: #1-4

Worksheet 9 probs