Perceptron

Research Skills: Machine Learning

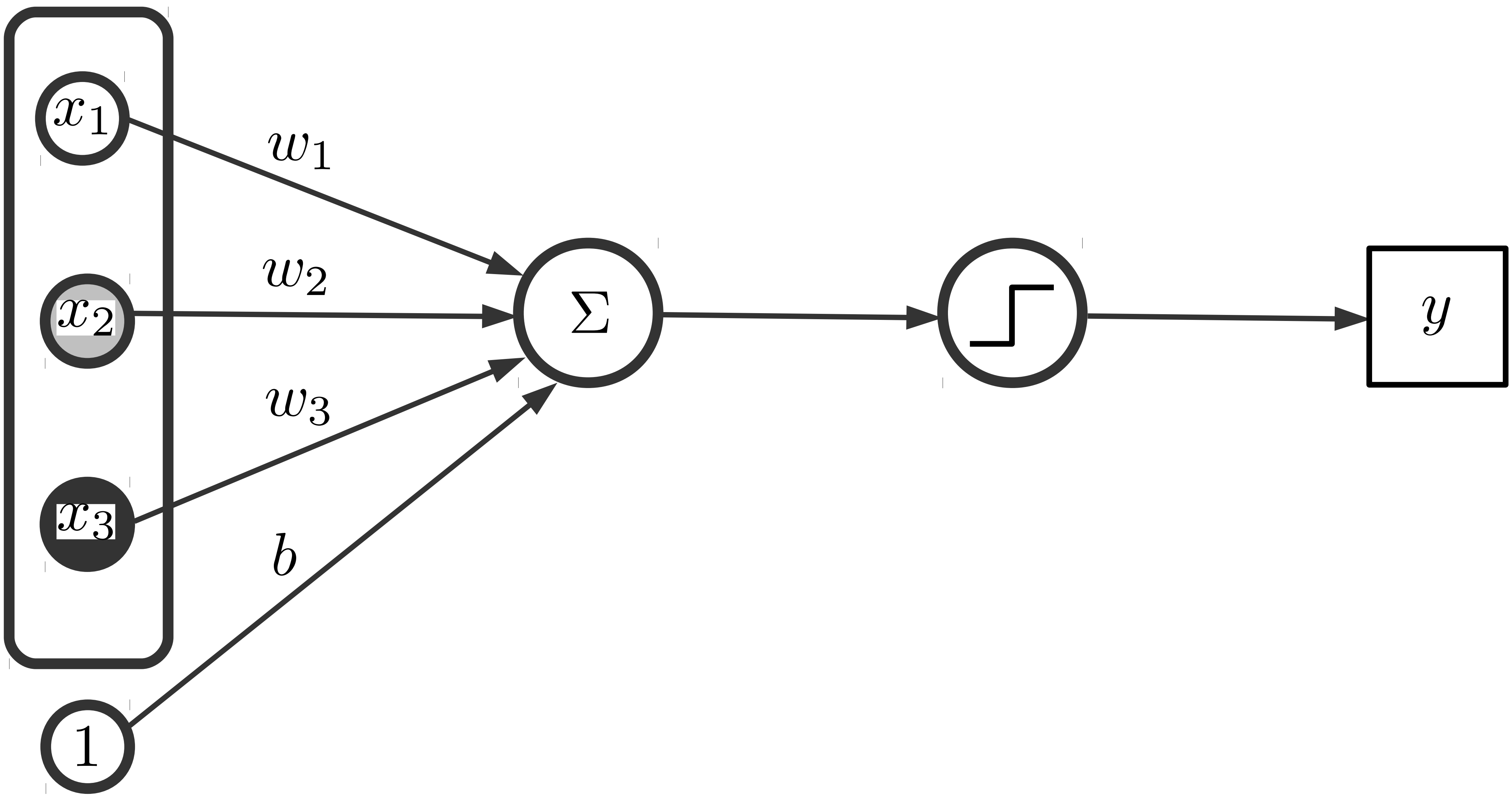
Grzegorz Chrupała
g.chrupala @ uvt.nl

Learning from examples

- kNN
  - memorize examples
- Decision Trees
  - learn nested **if-then-else** rules
- Linear classifiers
  - find simple boundaries in space

Frank Rosenblatt 1928 – 1971. Psychologist, inventor of the perceptron algorithm.

Perceptron



Perceptron classification rule

- Perceptron uses a simple rule to classify objects
  - It computes the weighted sum of the input features (plus **bias**)
  - If this sum is greater than or equal to **0**, it outputs positive class **+1**
  - Otherwise it outputs negative class **-1**

Example: movie reviews

	#good	#dark	#mediocre	#the	
▪ $\mathbf{x}^1$ = (	2,	0,	0,	5	)
▪ $\mathbf{x}^2$ = (	0,	1,	2,	7	)
▪ $\mathbf{w}$ = (	2.5,	0.5,	-4.0,	0.0	)

- b = 0.5
- score $f(\mathbf{x}^1) = +5.5$, $y^1 = +1$
- score $f(\mathbf{x}^2) = -7.0$, $y^2 = -1$

Discriminant function

$$f(\mathbf{x}) = \left(\sum_{i=1}^{N} w_i x_i \right) + b$$

$$y = \begin{cases} +1 & \text{if } f(\mathbf{x}) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

Dot product notation

$$\mathbf{w} \cdot \mathbf{x} = \sum_{i=1}^{N} w_i x_i$$

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

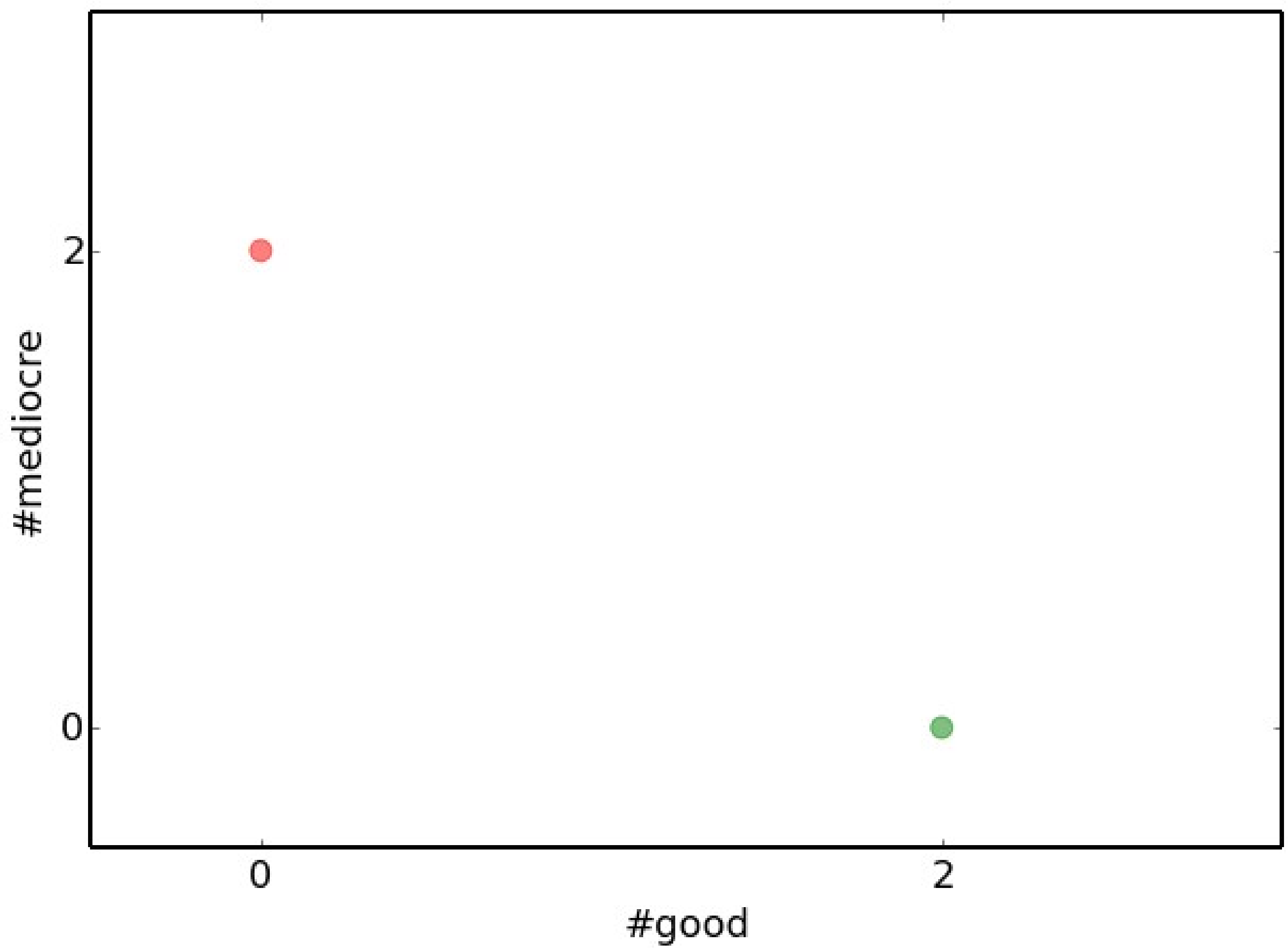
Role of bias

- When $\mathbf{w} \cdot \mathbf{x} \approx 0,$
  bias decides which class to predict
- Makes the default decision
- Biases the classifier towards positive or negative class

Geometric interpretation in 2D

	#good	#mediocre
▪ $\mathbf{x}^1$ = (	2,	0)
▪ $\mathbf{x}^2$ = (	0,	2)
▪ $\mathbf{w}$ = (	2.5,	-4.0)

- b = 0.5



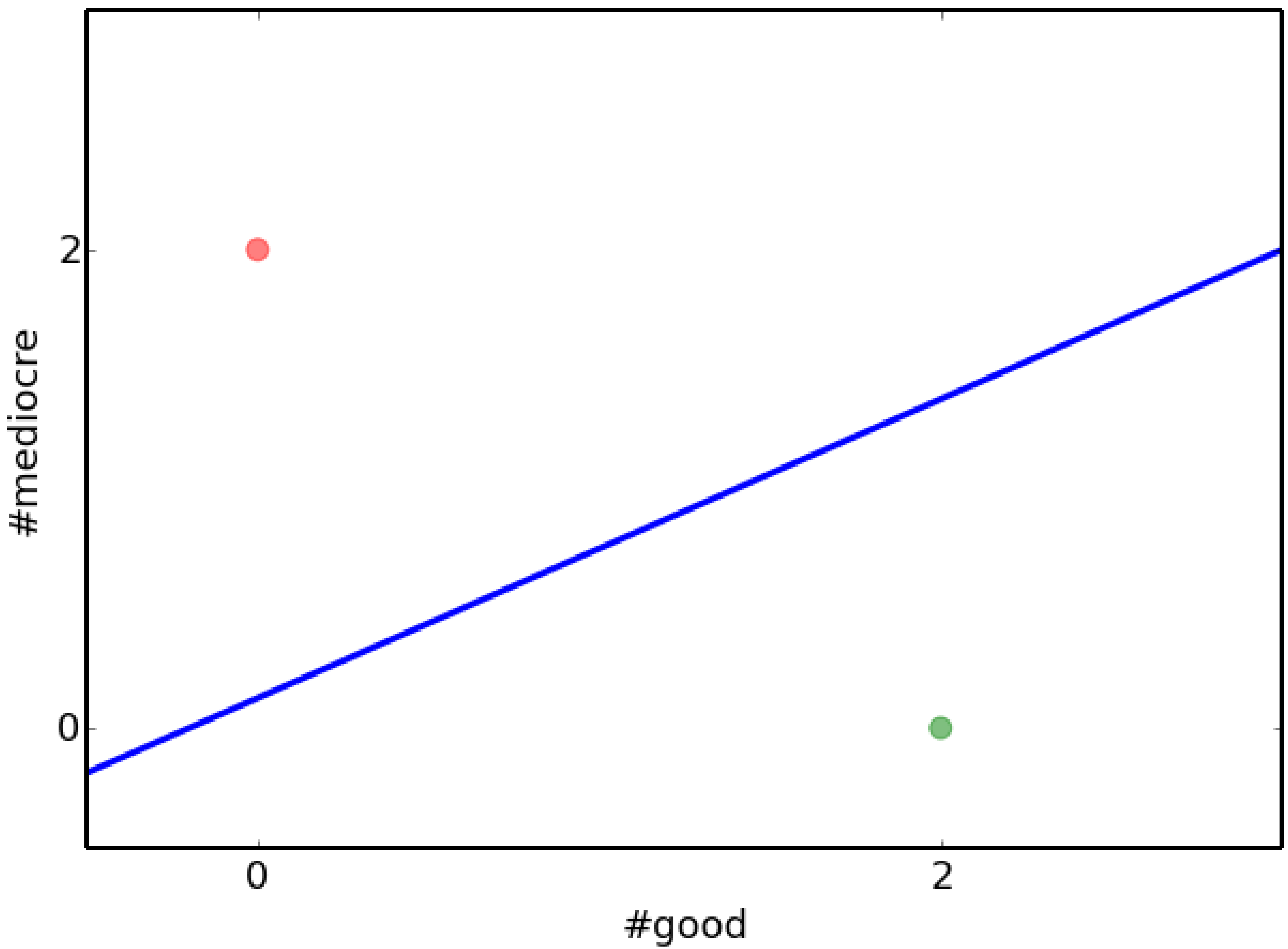
2
0
0
2
#mediocre
#good

Decision boundary

$$w_1x_1 + w_2x_2 + b = 0$$

Solve for x_2

$$x_2 = \underbrace{-\frac{w_1}{w_2}}_{\text{slope}}x_1 + \underbrace{\frac{b}{w_2}}_{\text{intercept}}$$



2
0
#mediocre
0
2
#good

How can we find good (w,b) ?

- Go through examples one by one
- Try classifying current example with current (**w**,b)
- If correct, keep going
- If not correct, adjust (**w**,b)

How to adjust (w,b)?

- Example ($\mathbf{x}$, +1)
- With current ($\mathbf{w}$, b), the score $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$ is less than 0
- How do we change b to make it higher?
- How do we change $\mathbf{w}$ to make it higher?

Example

- $\mathbf{x}^1$ = (2, 0, 0, 5)
- $\mathbf{w}$ = (-0.5, 1.0, -2.0, 0.0)
- **b** = 0
- $f(\mathbf{x}^1) = \mathbf{w} \cdot \mathbf{x}^1 + b = -1.0$
- Change b to increase $f(\mathbf{x}^1)$
- Change $\mathbf{w}$ to increase $f(\mathbf{x}^1)$

Update rule: example (x,y), model (w,b)

$y_{\text{pred}} = \text{predict}((\mathbf{w}, b), \mathbf{x})$
if $y = +1$ and $y_{\text{pred}} = -1$ **then**
    $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}$
    $b \leftarrow b + 1$
else if $y = -1$ and $y_{\text{pred}} = +1$ **then**
    $\mathbf{w} \leftarrow \mathbf{w} - \mathbf{x}$
    $b \leftarrow b - 1$

Vector addition and subtraction

$$\mathbf{c} = \mathbf{a} + \mathbf{b}$$

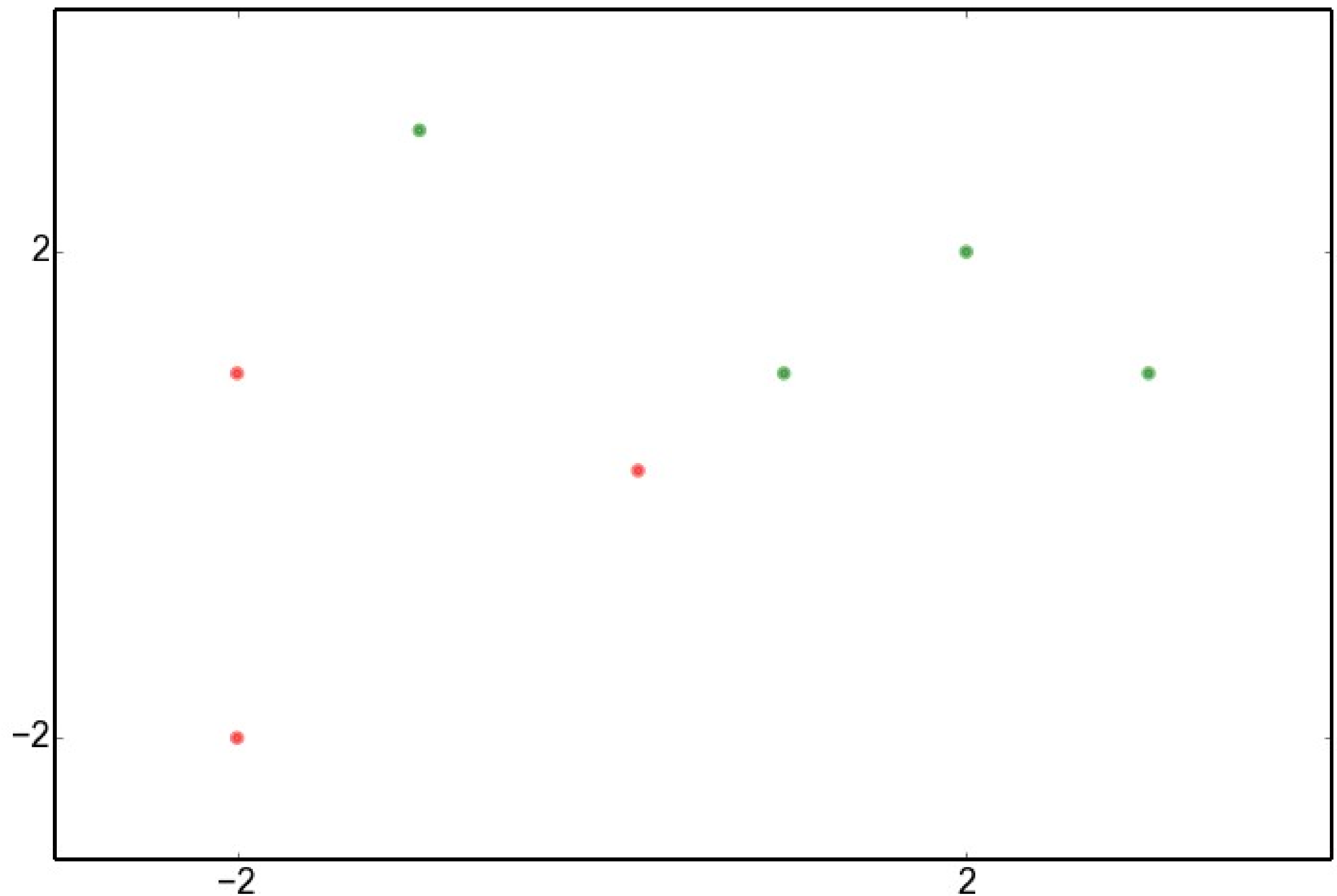
$$\text{for all } i,\ c_i = a_i + b_i$$

```
x¹    = (    2,     0,     0,      5 )
w     = ( -0.5,   1.0,  -2.0,    0.0 )
w+x¹  = (  1.5,   1.0,  -2.0,    5.0 )
```

One iteration over *N* examples

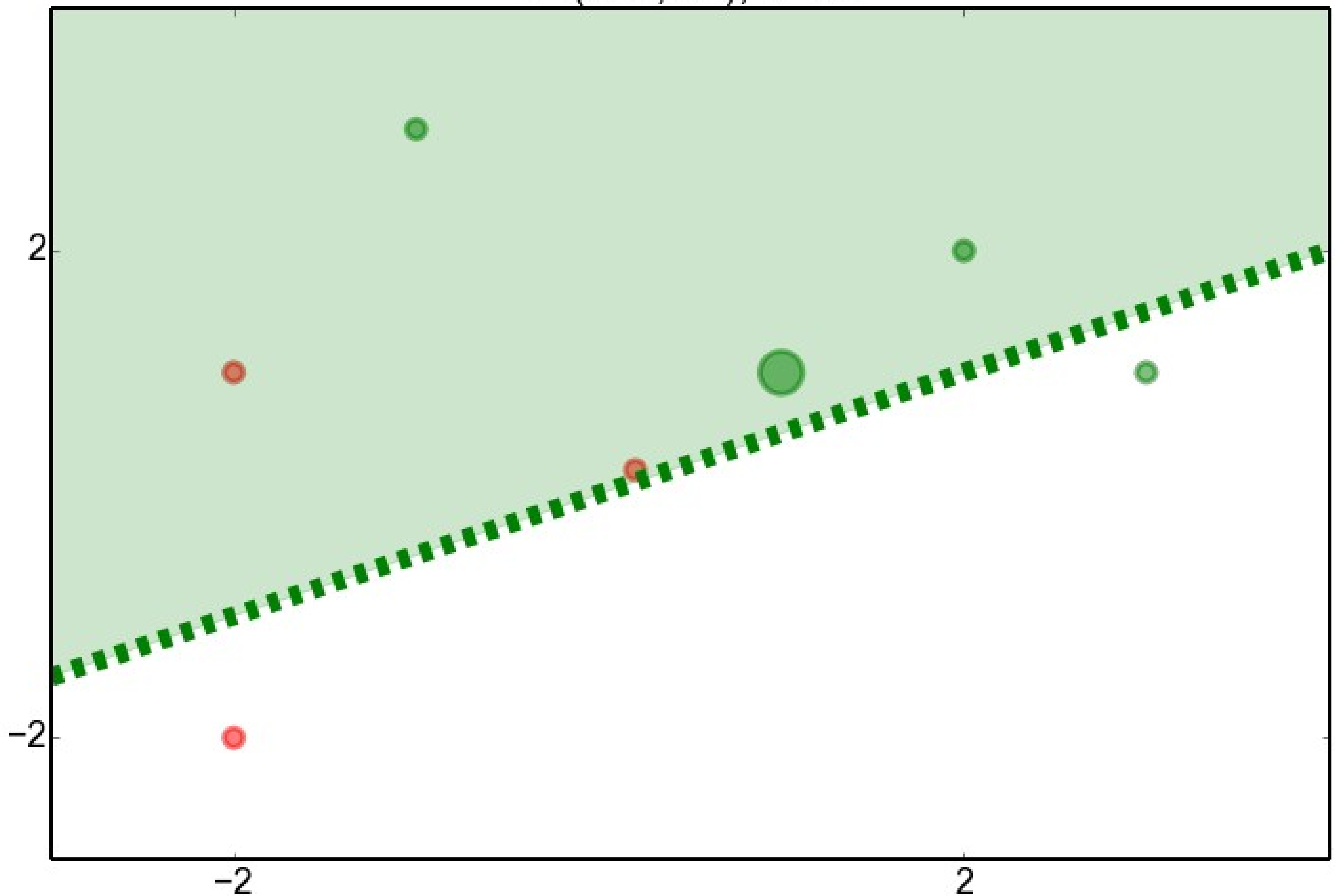
$\mathbf{w} \leftarrow \mathbf{0}$
$b \leftarrow 0$
for $n = 1..N$ **do**
  $y^n_{\text{pred}} = \text{predict}((\mathbf{w}, b), \mathbf{x}^n)$
  if $y^n = +1$ and $y^n_{\text{pred}} = -1$ **then**
    $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}^n$
    $b \leftarrow b + 1$
  else if $y^n = -1$ and $y^n_{\text{pred}} = +1$ **then**
    $\mathbf{w} \leftarrow \mathbf{w} - \mathbf{x}^n$
    $b \leftarrow b - 1$

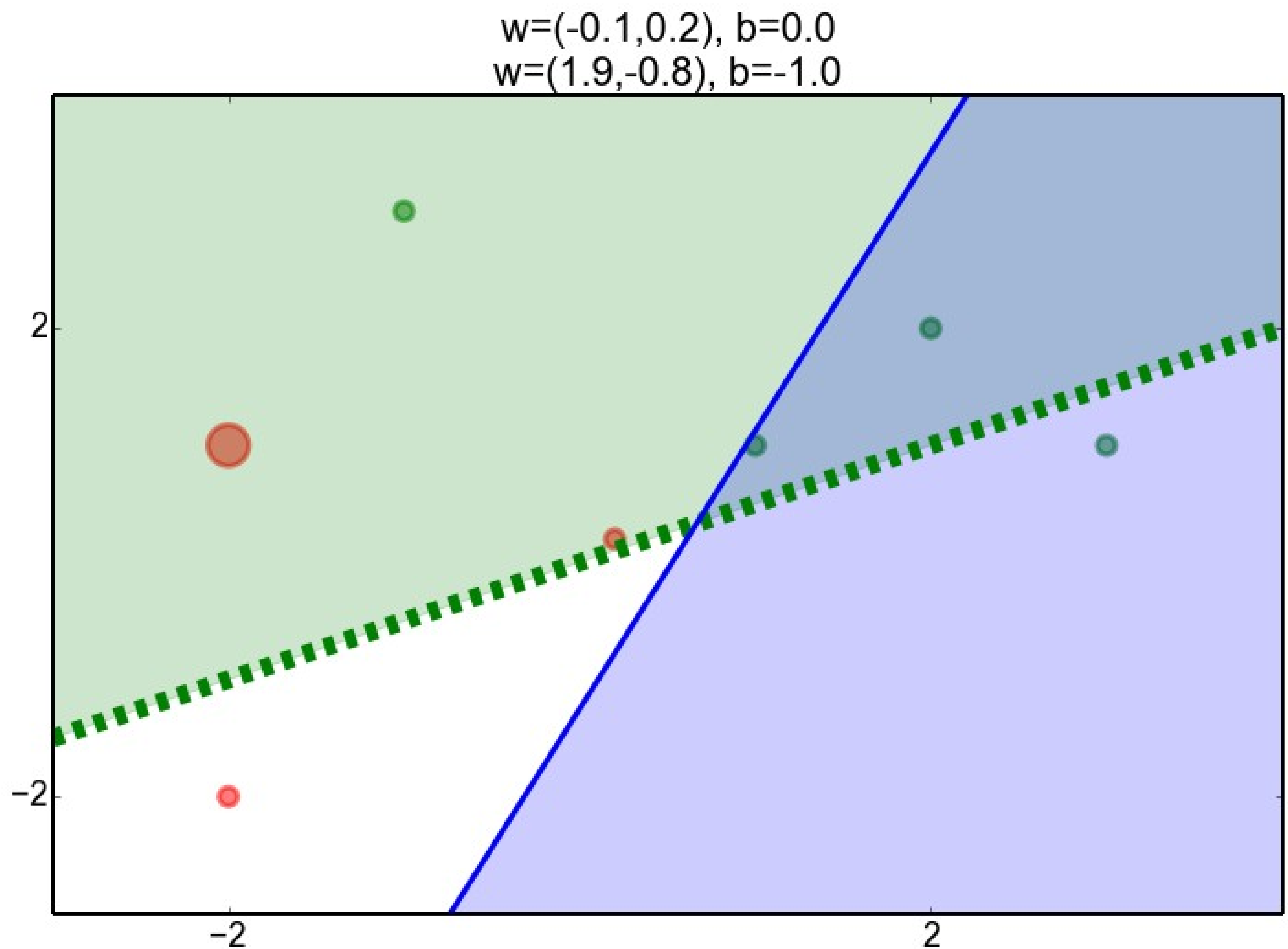
Example

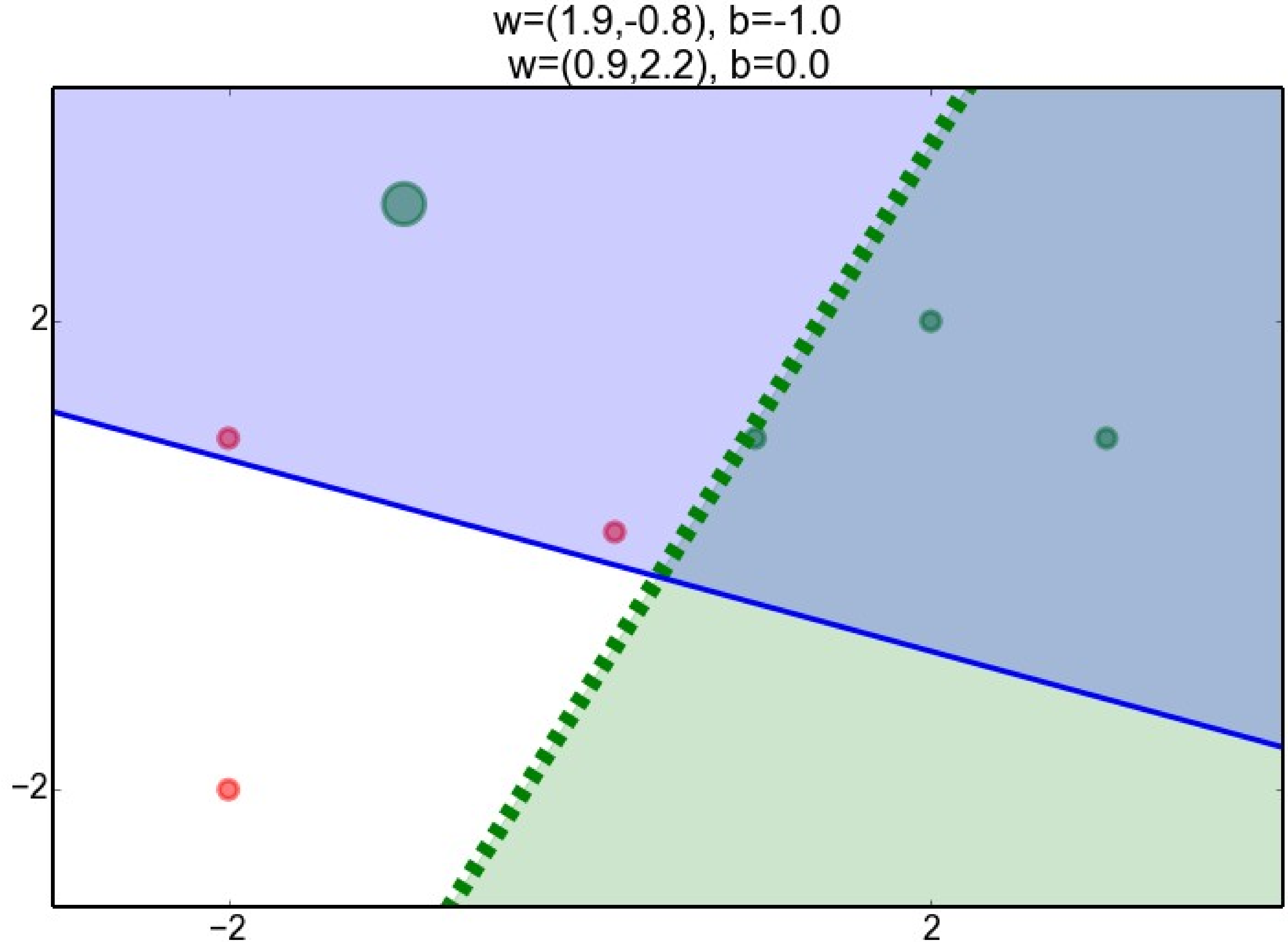


2
−2
−2
2

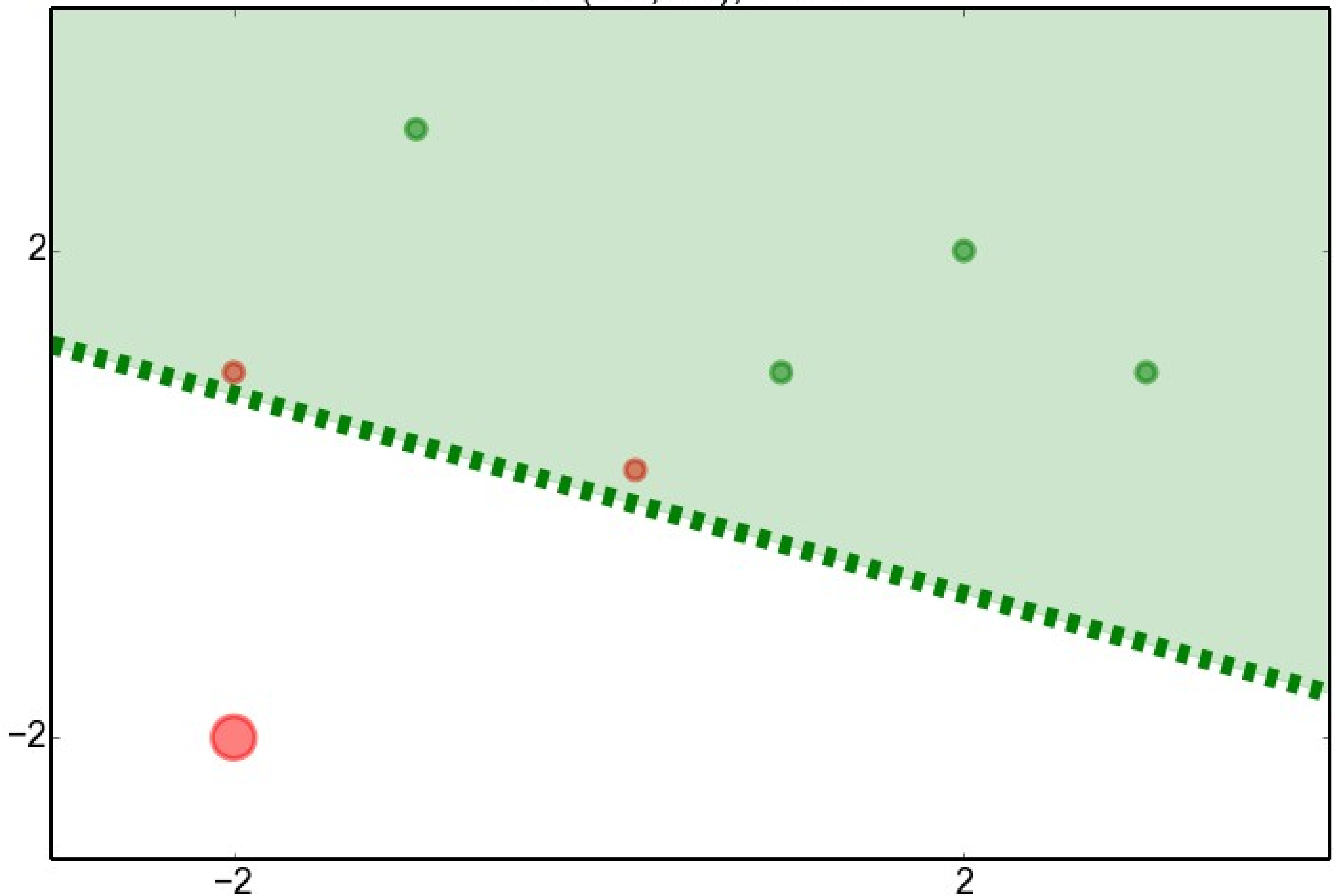
w=(-0.1,0.2), b=0.0
w=(-0.1,0.2), b=0.0



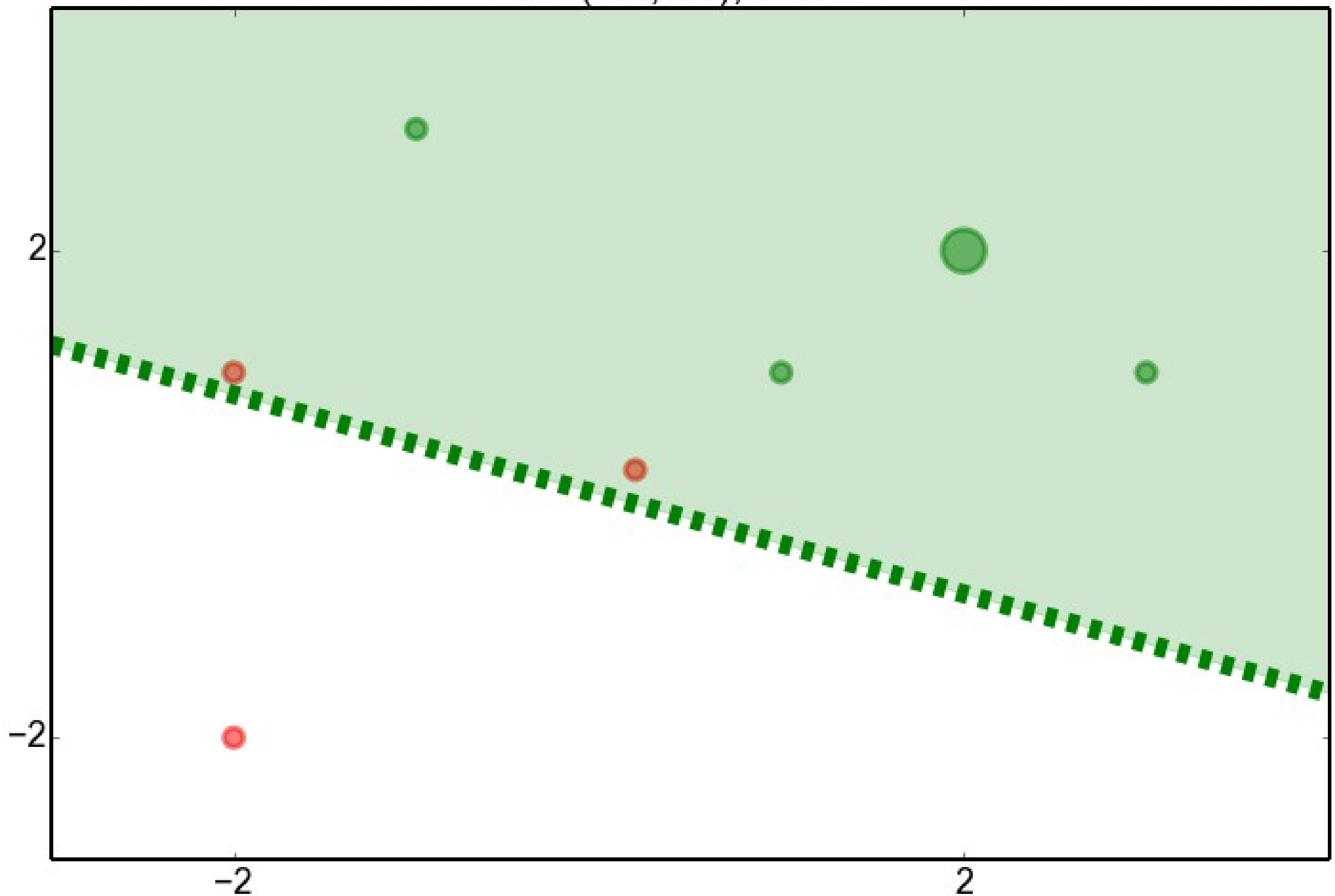

w=(-0.1,0.2), b=0.0
w=(1.9,-0.8), b=-1.0
2
−2
−2
2

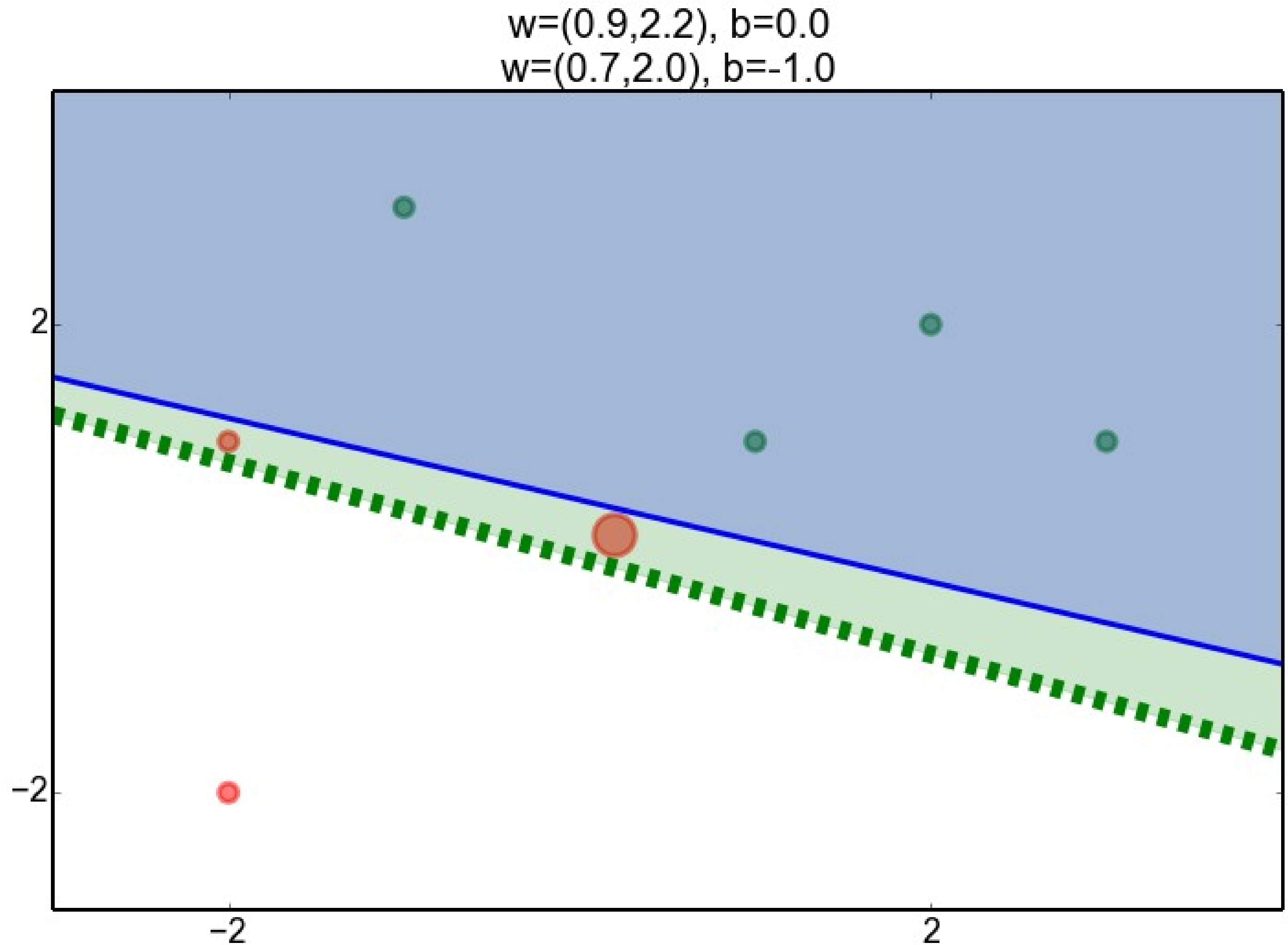

w=(1.9,-0.8), b=-1.0
w=(0.9,2.2), b=0.0
2
−2
−2
2

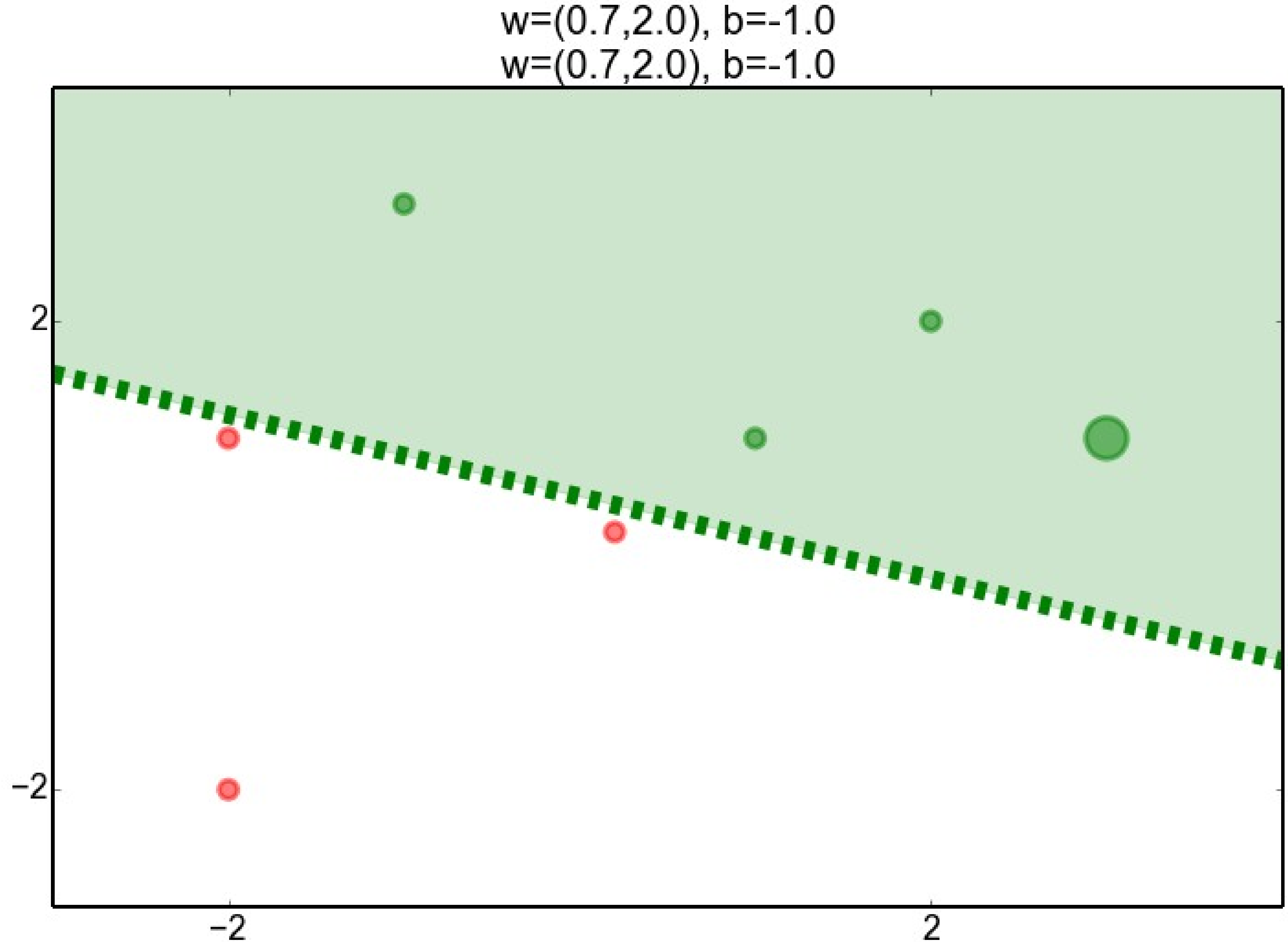
w=(0.9,2.2), b=0.0
w=(0.9,2.2), b=0.0



w=(0.9,2.2), b=0.0
w=(0.9,2.2), b=0.0




w=(0.9,2.2), b=0.0
w=(0.7,2.0), b=-1.0
2
−2
−2
2


w=(0.7,2.0), b=-1.0
w=(0.7,2.0), b=-1.0
2
−2
−2
2

Termination

It can be proven that if there is a linear boundary separating **+1** from **-1**, the perceptron algorithm will find it.

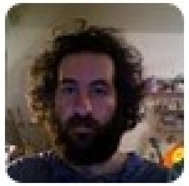
Online learning

- Perceptron looks at one example at a time
- Online learners are good for streams of data
  - Social media posts
  - Photo uploads
  - User queries on a search engine

Jacob Eisenstein @jacobeisenstein · 2h

Dot-producting a dense parameter matrix by a sparse feature vector is maybe the most basic #scipy operation you'd want to do in NLP (3/2)

1



yoav goldberg @yoavgo · 39m

@jacobeisenstein I find numpy's sparse stuff to be quite cumbersome to work with, and suggest rolling your own sparse dot products in cython

1

Razib Khan @razibkhan · 43m

Anger of Suspect in Danish Killings Is Seen as Only Loosely Tied to Islam nyti.ms/17gDRF9 lots of violent radicals aren't most pious

1

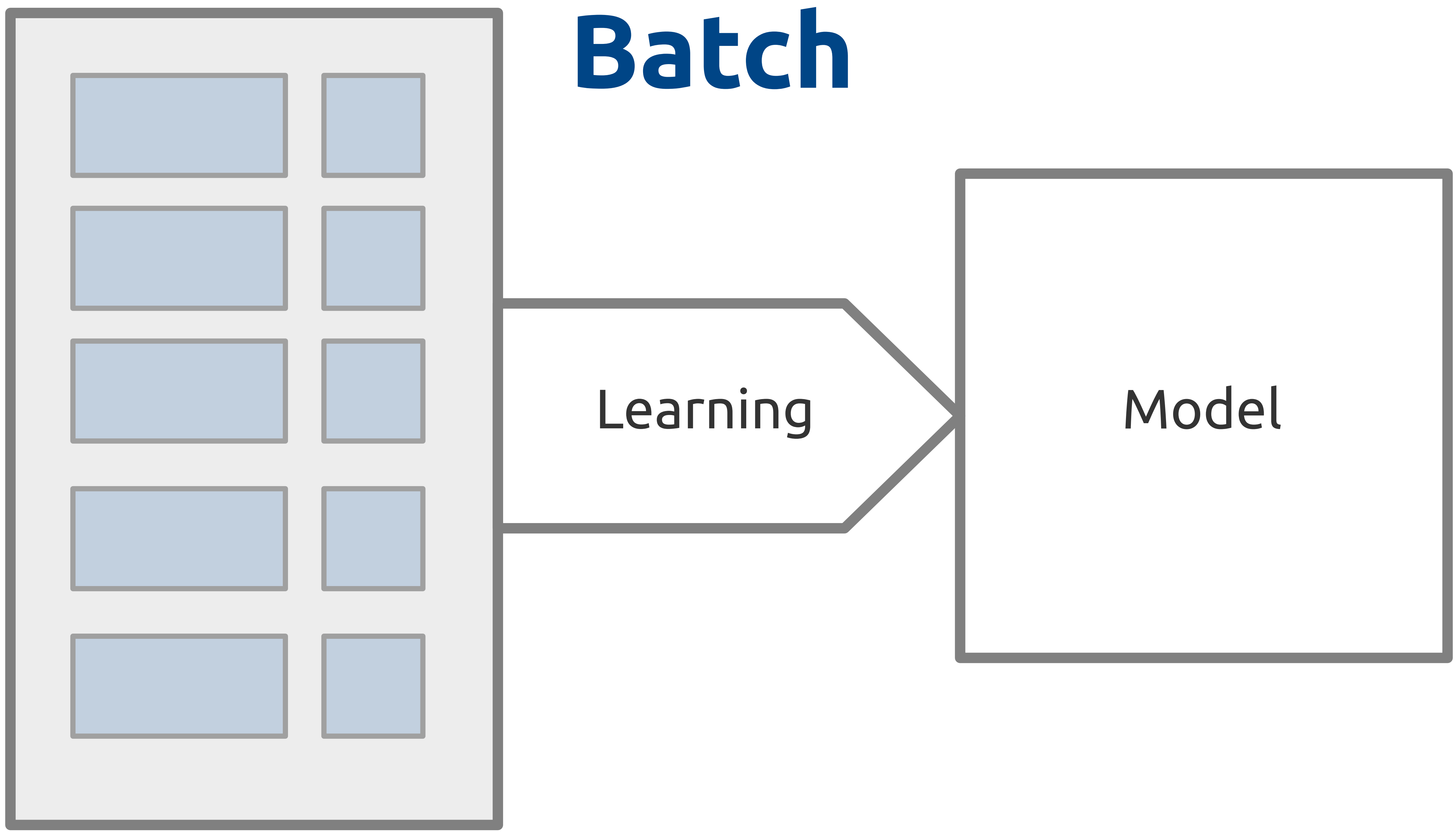
View summary

Brian Switek @Laelaps · 50m

Airports should have special "Out of my way, slowpokes!" lanes for people with less than 40 minutes to connect to their next flight.

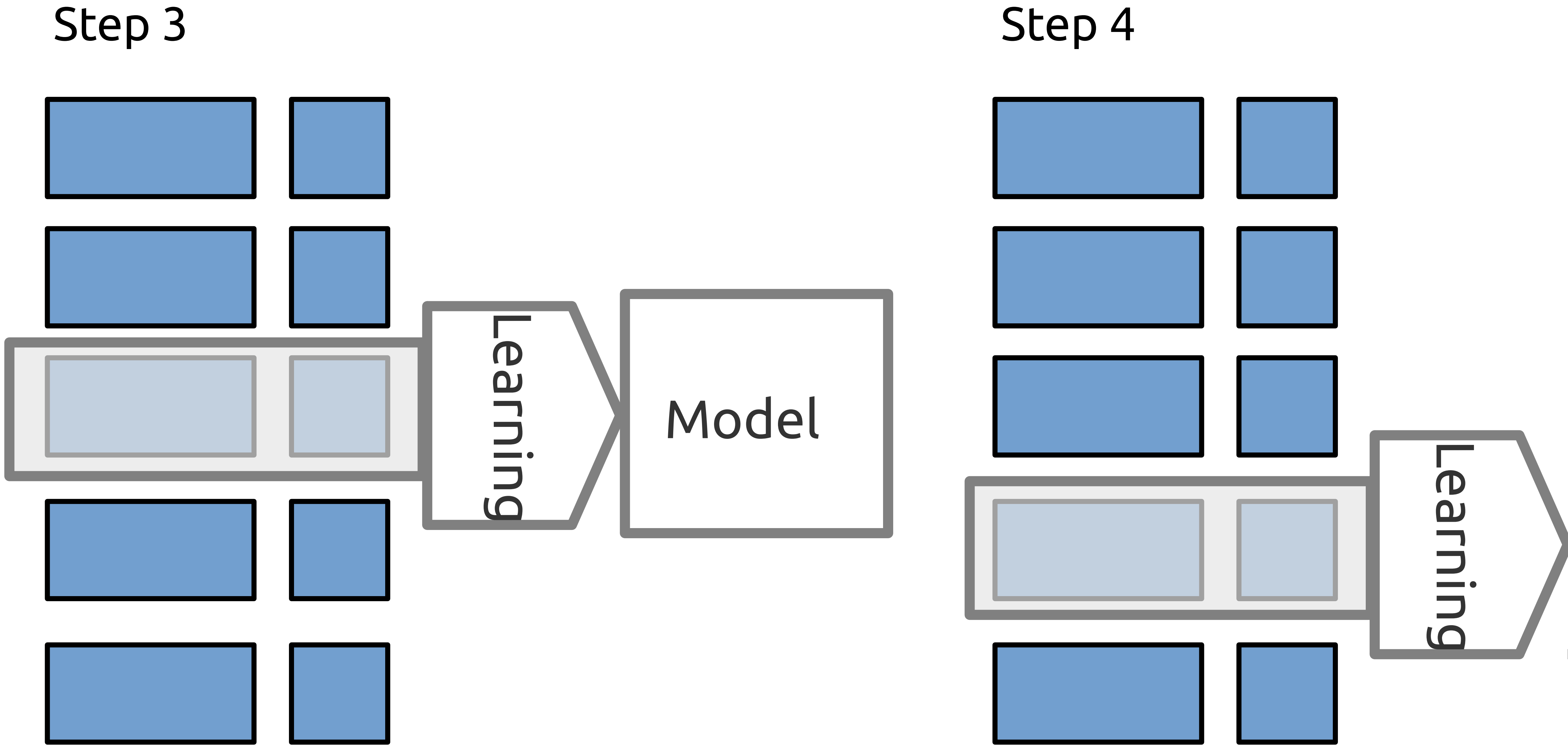
2 9

Brian Switek @Laelaps · 51m



Batch
Learning
Model

Online



Online vs Batch

- **Batch** algorithm has to remember whole dataset
- **Online** algorithm only remembers current example
- Perceptron can imitate batch learning by iterating over data several times

Evaluation in pure online learning

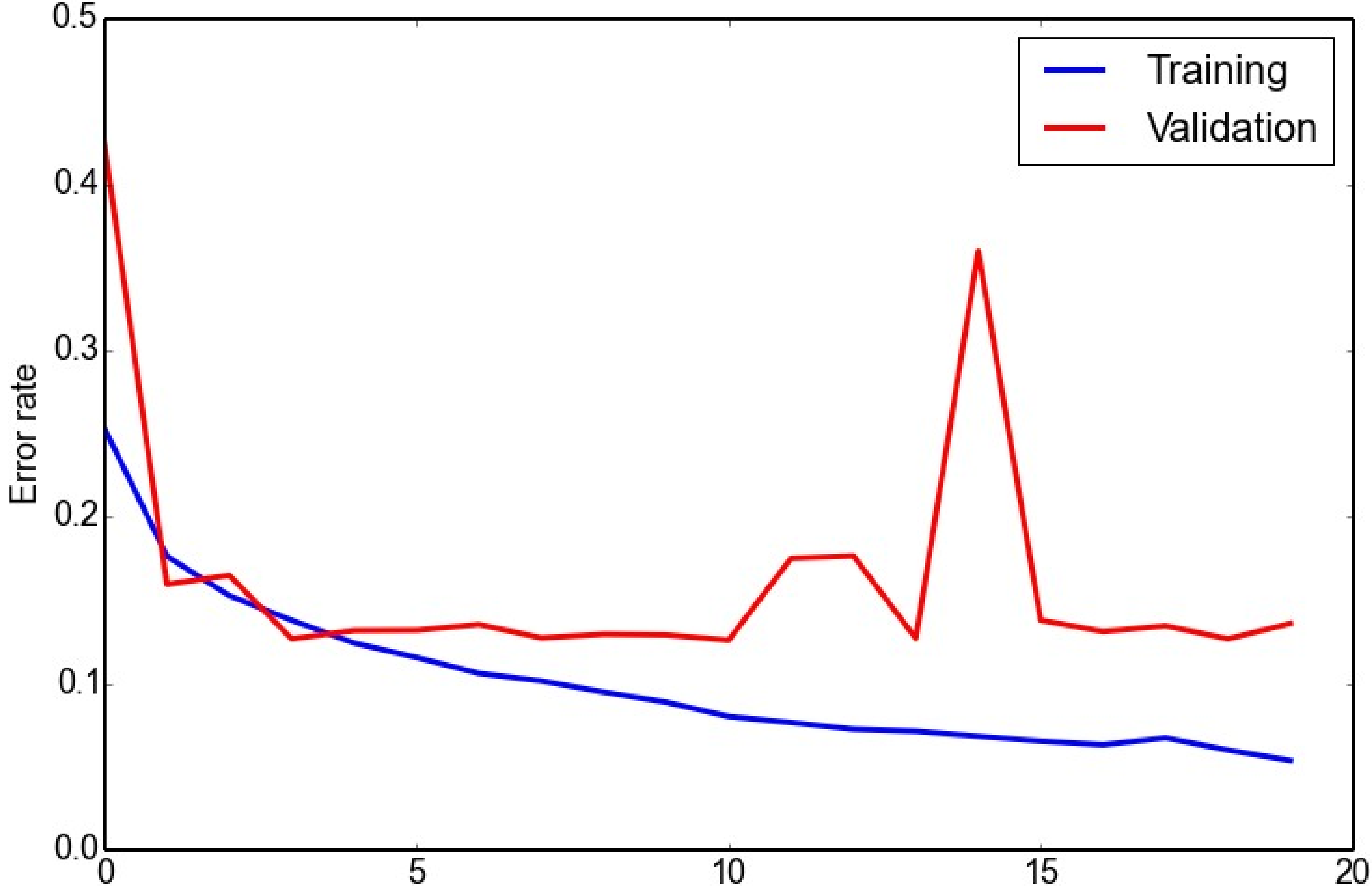
- Make prediction for current example
- Record if correct or not
- (Update model), go to next example
- At each point in time:
  - error rate = proportion of mistakes made so far, to total examples seen so far

Evaluation with multiple iterations

- When using **multiple iterations**, we would be evaluating on previously seen examples
- Use separate **development** set!

Learning ratings of movies in the sentiment dataset

- 25,000 movie reviews, positive and negative
- Use 5,000 for validation, 20,000 for training
- 20 iterations


Training
Validation
Error rate
0.0
0.1
0.2
0.3
0.4
0.5
0
5
10
15
20

Early stopping

- Number of iterations is a kind of hyper-parameter of the “batch” Perceptron
- Stop training when error on validation data stops dropping
- When training error goes down, but validation goes up, we're **overfitting**

Which are the most important features?

- Bottom 10
  - waste worst poorly mess awful disappointment fails lacks annoying worse
- Top 10
  - subtitles captures enjoyable subtle noir surprisingly today excellent wonderfully perfect
- Around 0:
  - very character since during you're second stories particularly yourself hit

Sparseness

$$\begin{pmatrix} 0 & 3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 7 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 7 \\ 0 & 0 & 0 & 5 & 0 & 0 & 0 & 8 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 9 & 0 & 0 & 0 & 4 & 0 \\ 0 & 1 & 0 & 7 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 7 & 0 & 0 & 0 & 0 & 5 & 0 & 0 \end{pmatrix}$$

Representing text

- Text document often represented with word counts
- How many elements in the vector?
  - Many tens or hundreds of thousands
- Use a sparse representation which omits zero values

Dense

	#good	#dark	#mediocre	#the
▪ $\mathbf{x}^1 = ($	2,	0,	0,	5 $)$
▪ $\mathbf{X}^2 = ($	0,	1,	2,	7 $)$

Sparse

V = (#good #dark #mediocre #the)

- $\mathbf{x}^1$ = { 1:2, 4:5 }
- $\mathbf{x}^2$ = { 2:1, 3:2, 4:7 }

All absent values are implicitly zero.

Sparse vectors in Python

- Python dictionaries
- Sparse matrices in **scipy**
- Assignment 2
  - Implement perceptron algorithm
  - work with dictionaries as sparse vectors

Exercises
Cosine similarity

- The cosine of the angle between two vectors **u** and **v** is:

$$\text{cosine}(\mathbf{u}, \mathbf{v}) = \frac{\sum_{i=1}^{V} u_i v_i}{||\mathbf{u}|| \times ||\mathbf{v}||}, \text{ where } ||\mathbf{u}|| = \sqrt{\sum_{i=1}^{V} u_i^2}$$

Exercises

(1) Define the norm $||\mathbf{u}||$ of vector $\mathbf{u}$ represented as a Python dictionary

(2) Define the cosine distance between two vectors represented as Python dictionaries

Examples

```
>>> u = {1:1,3:-1}
>>> v = {1:1,2:-1}
>>> print norm(u)
1.4142135623730951
>>> print cosine(u, u)
1.0
>>> print cosine(u, v)
0.5
```

Problem:
Most positive example

- Given a dataset of movie reviews and a model trained on it, how can we find the most positive review (according to the model?)

(Advanced) Problem: Multiclass classification

- The perceptron algorithm as presented in the lecture works for binary classification. How could it be adapted to learn classification with more than two classes?
- Discuss the solutions on course forum.
- (There are at least two common approaches.)

Image credits

- Frank Rosenblatt http://www.rutherfordjournal.org/images/TAHC_rosenblatt-sepia.jpg