Gradient Descent

Machine Learning

Agenda

- Introduction
- Linear models
- Gradient Descent for quadratic function
- Stochastic Gradient Descent

Two aspects of a learner

- **How it uses inputs to predict outputs**
  - DT – traverse tree asking questions until arrive at leaf with **target**
  - Perceptron – predict **+1** if $\mathbf{w} \cdot \mathbf{x} + b >= 0$
- **How it learns**
  - DT – split data using best question and recurse on splits
  - Perceptron – update **w** and b if made a mistake

Two aspects of a learner

- How it uses inputs to predict outputs
  - **Model**
- How it learns
  - **Optimization algorithm**

Modularity

- Often **parameters** (e.g. weights) of the same model can be found in many different ways
- Standard **optimization** algorithms for many types of problems
    - Often can be treated as a **black box**

How to participate?

1 Go to wooclap.com

2 Enter the event code in the top banner

Event code

ALFXDE

Enable answers by SMS

Copy participation link

wooclap

100 %

55

You cannot vote anymore

Which of the following ML methods is(are) a non-linear classifier?

1	Decision tree	48%	15 ✓
2	Perceptron	13%	4
3	Both of them	23%	7
4	Neither of them	16%	5

wooclap

100 %

48% correct

31 / 55

Linear models

- Linear models are based on a weighted sum of features
- (Multiclass) Classification
- (Multivariate) Regression

Go to **wooclap.com** and use the code **ALFXDE**

What is the most common model for regression?

Linear regresseion

Simple linear regression

linear

Linear regression

Lineair Regression

ols

Logistic regression

Regression

Ordinary least squares

wooclap

100 %

25

For example linear regression

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

Go to **wooclap.com** and use the code **ALFXDE**

How can we find the best w?

Correct for the error
actual
classified
difference
Minimize the errors
prediction
value
loss
tuning
MSE
find
predicted
Ols
minimizing
minimum
correctly
error
objects
w
derivaat
get
adjust dependent variables
Derive it
reducing error
make the least mse

wooclap

100 %

17

How can we find best w?

- Use specialized formula for linear regression

OR

- Convert into problem of finding minimum of function
- Standard solvers
  - Newton's method
  - (Stochastic) gradient descent
- **This approach works for many types of models**

Go to **wooclap.com** and use the code **ALFXDE**

What is the error metric for measuring how well our regression line fits the data points?

MSE

RMSE

R2 or mse

R2

R squared

MAE

SSE

MSE and RMSE

MAD

R square

R2 mse

Median absolute error

wooclap

100 %

27

Sum of squared errors

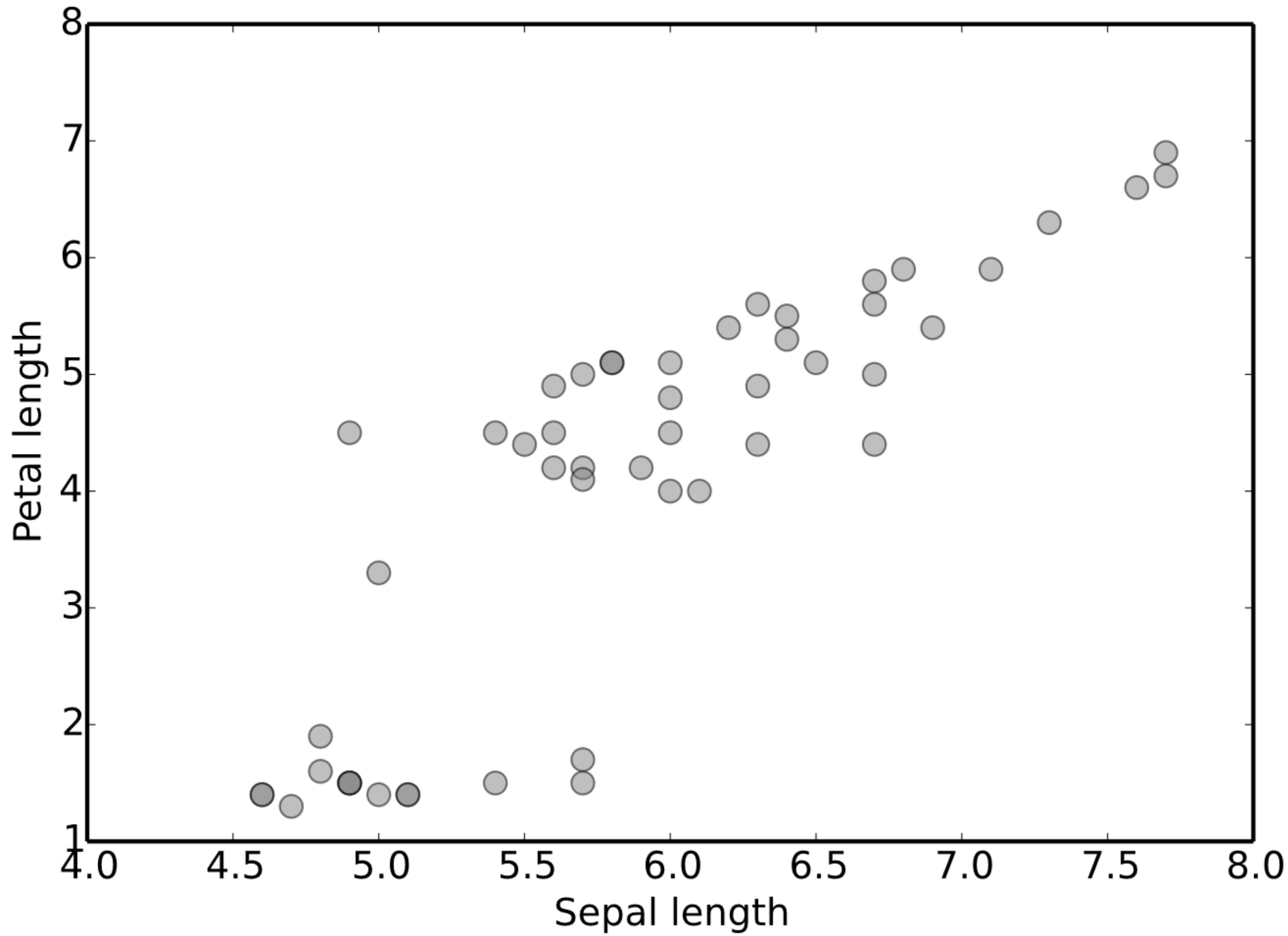
- Want to find **w**,b for which error on training data is smallest
- We can use SSE as a measure of error

$$\mathrm{SSE} = \sum_{i=1}^{N} (y^i_{\mathrm{pred}} - y^i)^2$$

Error as a function of w,b

$$\text{Error}(\mathbf{w}, b) = \sum_{i=1}^{N} (\mathbf{w} \cdot \mathbf{x}^i + b - y^i)^2$$

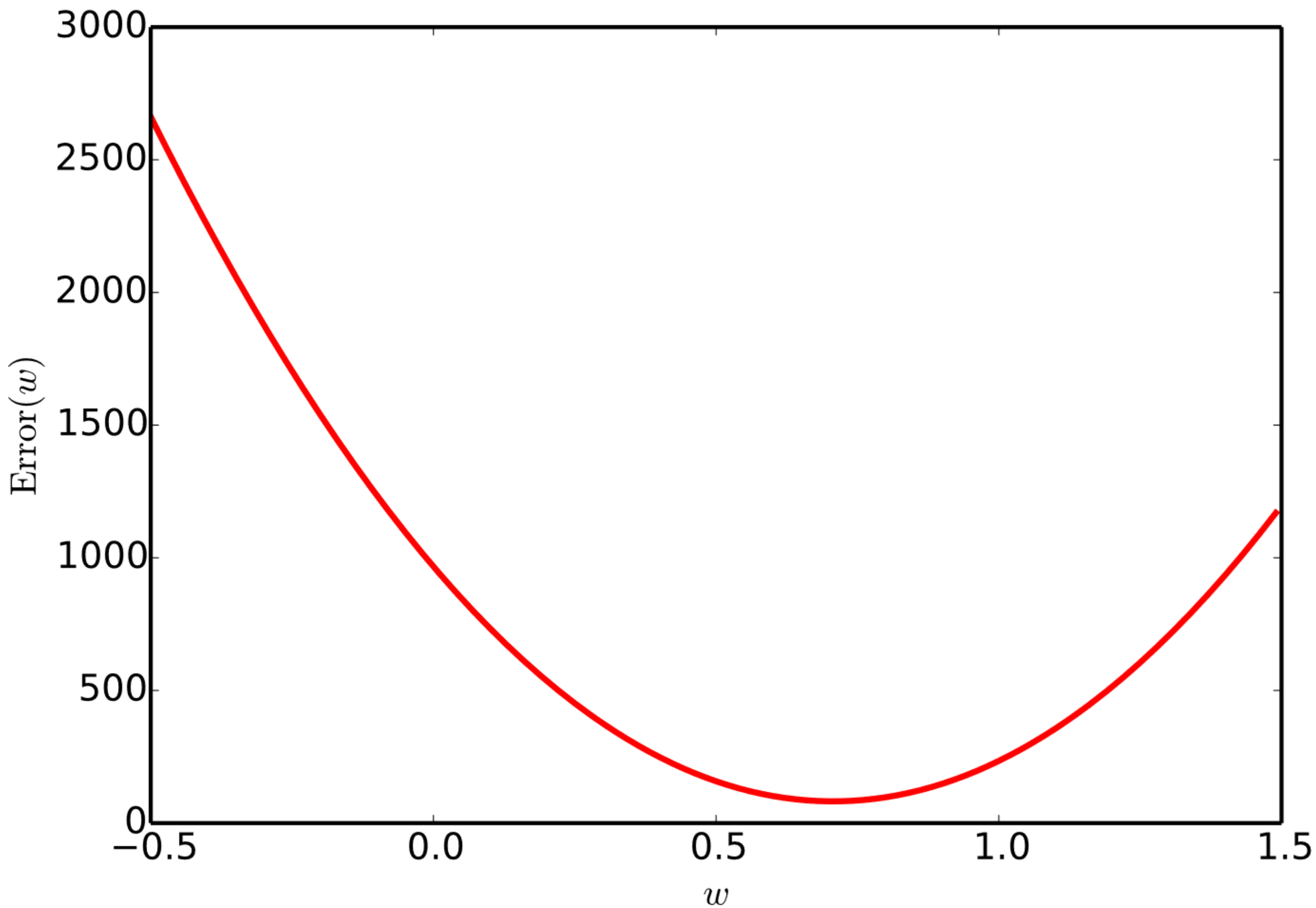
Regression example – Iris



Iris

- Find regression line which predicts **petal length** from **sepal length**
- For simplicity, fix $b = 0$
- How does **Error(w)** change as we vary **w**?

Find w for which Error(w) is lowest

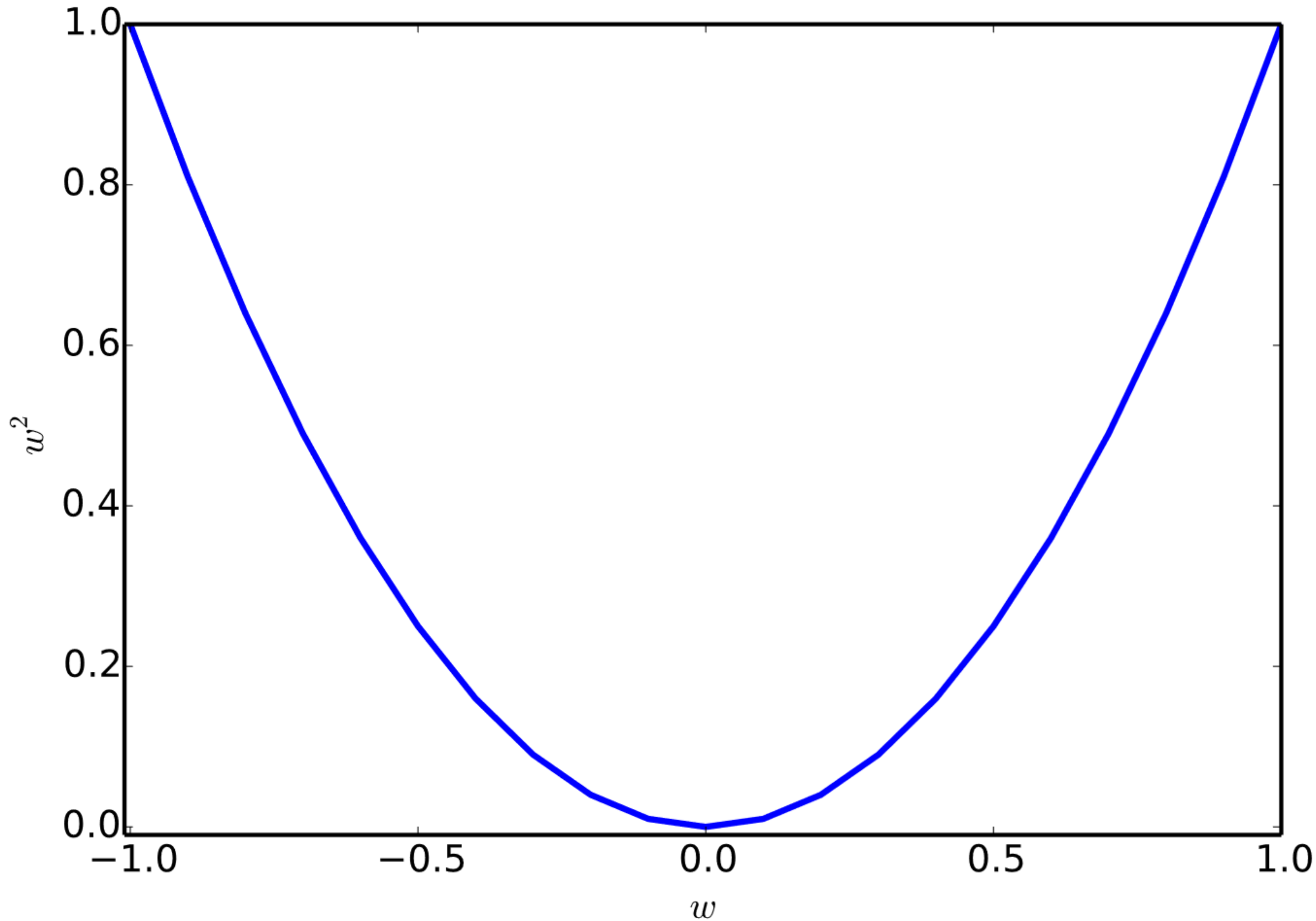


Start with something even simpler

$$\mathrm{Error}(\mathbf{w}, b) = \sum_{i=1}^{N} (\mathbf{w} \cdot \mathbf{x}^i + b - y^i)^2$$

- Work through example of a simpler function:

$$f(w) = w^2$$



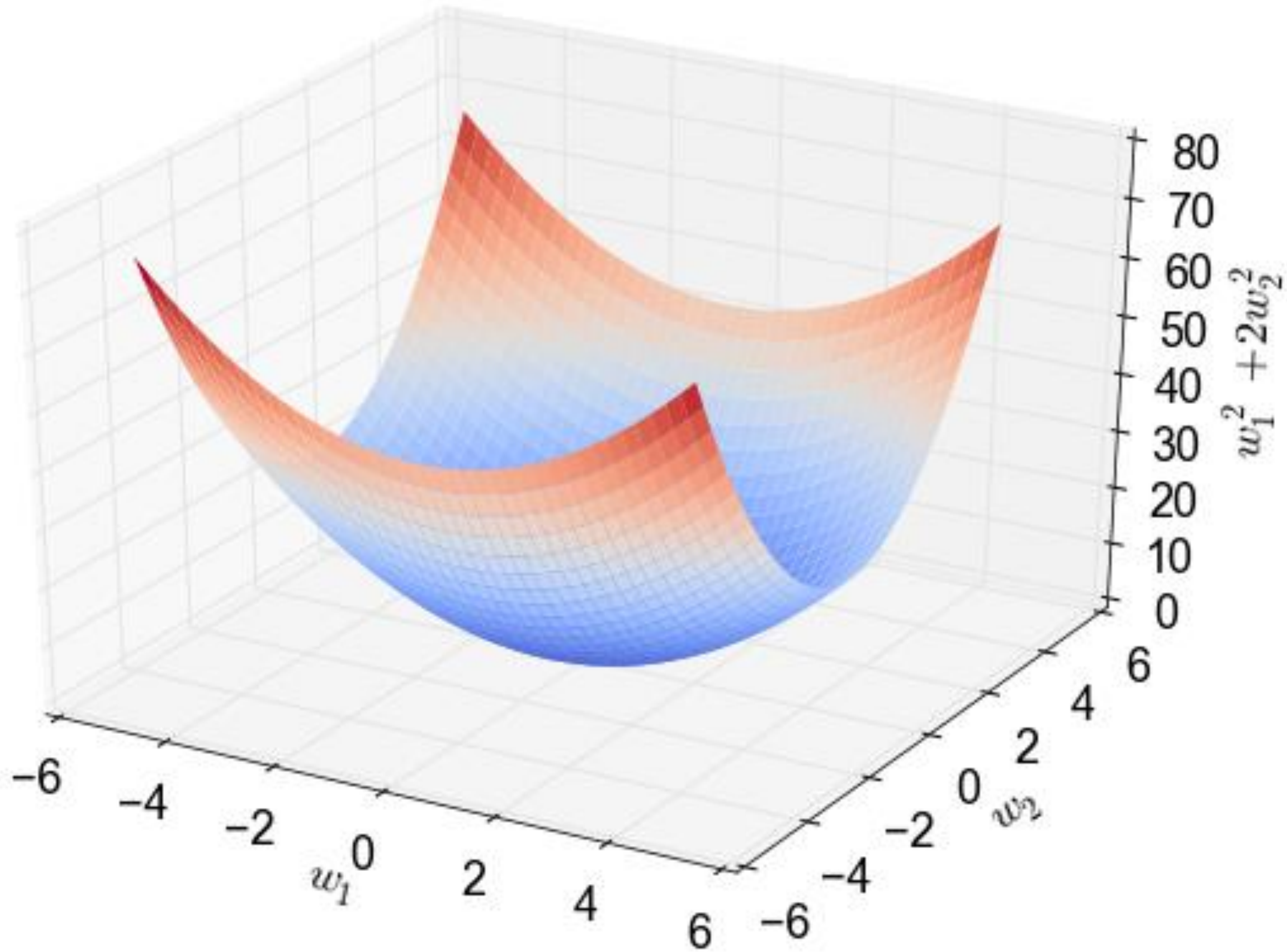
1.0
0.8
0.6
0.4
0.2
0.0
w^2
−1.0
−0.5
0.0
0.5
1.0
w

How can we find the value of *w* which minimizes *f(w)*?

- Start at a random value of *w*
- Check slope of function at this point
- Descend the slope: adjust *w* to decrease *f(w)*

Gradient vs slope

- **Slope** describes steepness of a single dimension
- We usually work with functions with vectors as arguments , e.g. Error(**w**)
- **Gradient** is the collection of slopes, one for each dimension


$w_1^2 + 2w_2^2$
80
70
60
50
40
30
20
10
0
w_1
−6
−4
−2
0
2
4
6
w_2
−6
−4
−2
0
2
4
6

Agenda

- Introduction
- Linear models
- Gradient Descent for quadratic function
- Stochastic Gradient Descent

Gradient descent for $f(w) = w^2$

Go to **wooclap.com** and use the code **ALFXDE**

How do we compute the slope of a function?

first derivative
calculate
points
slope
Pythagora theorem
1st derivative
y/x
another
Calculus
Break
increase
point
Line tangent
x
Needs break
unit
2
derivative
right
Find
next
Change in y per 1 unit of x
Delta y/ Delta x
delta y / delta x

wooclap

100 %

20

How do we compute the slope of a function?

- First derivative
- For function *f*, first derivative can be written *f'*
- Then $f'(a)$ is the slope of function *f* at point *a*

First derivative

- If we define

$$f(w) = w^2$$

- The first derivative is

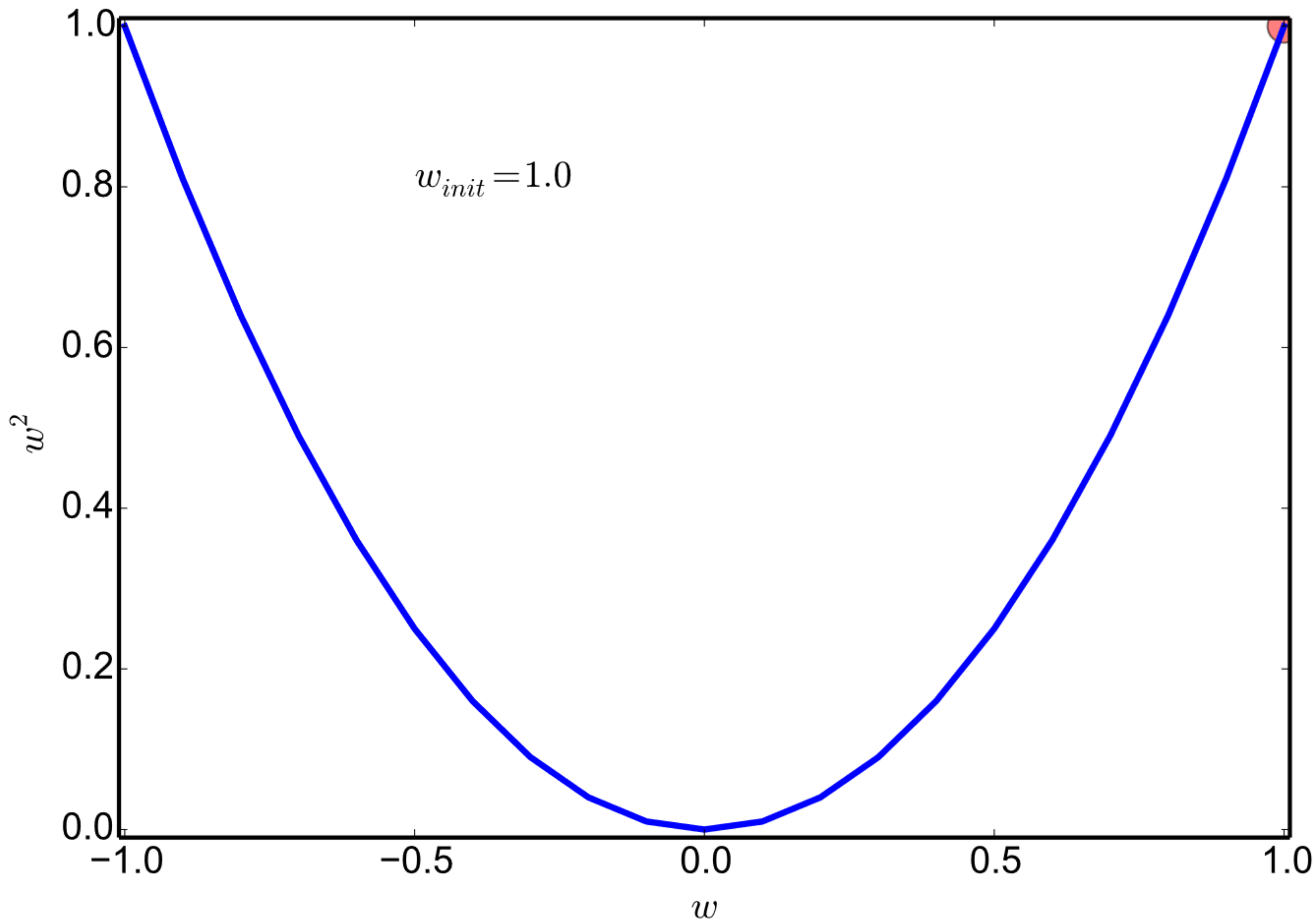
$$f'(w) = 2w$$

Ready to descend

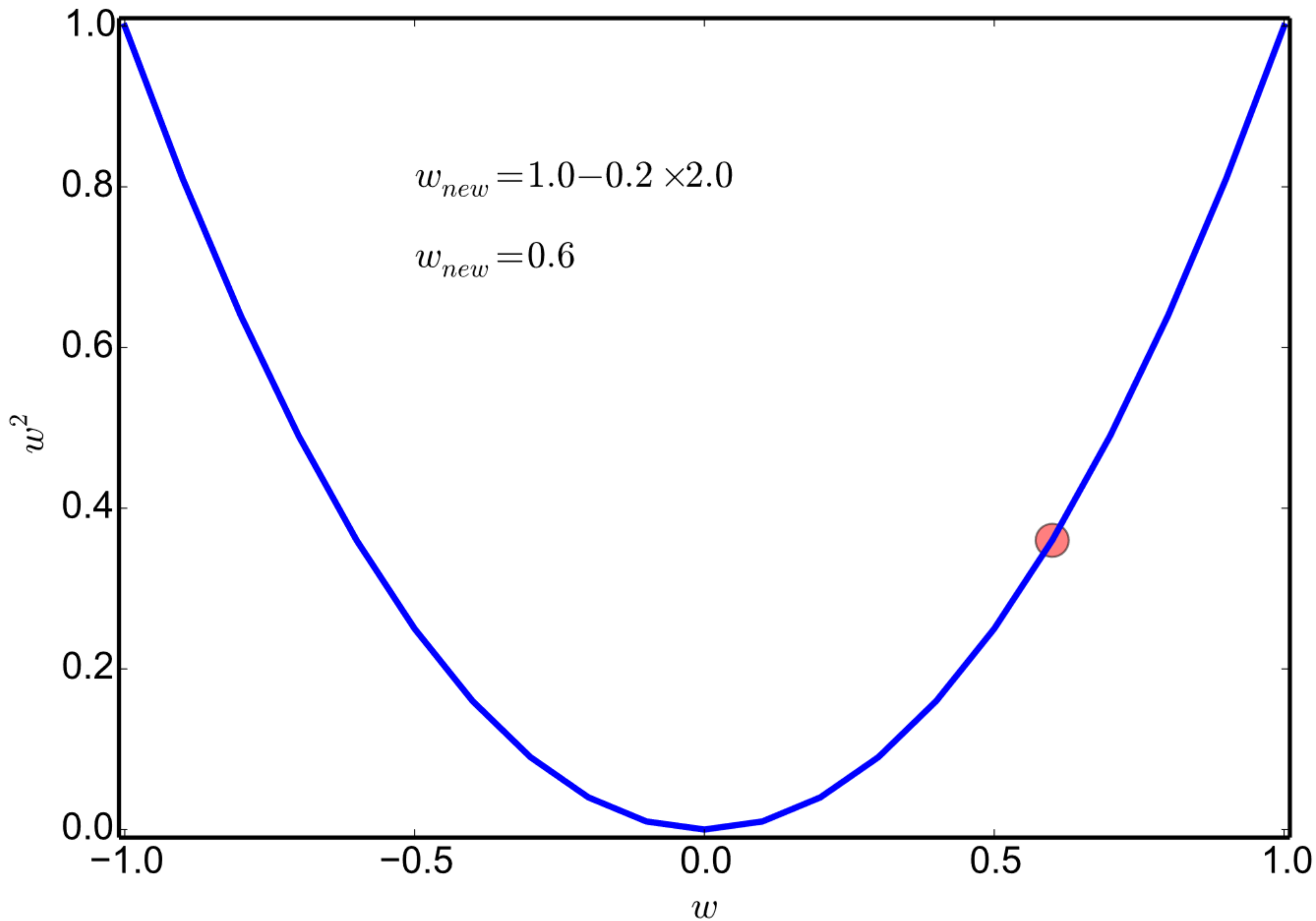
- Initialize *w* to some value (e.g. 1.0)
- Update:

$$w_{\text{new}} = w_{\text{old}} - \eta \times f'(w_{\text{old}})$$

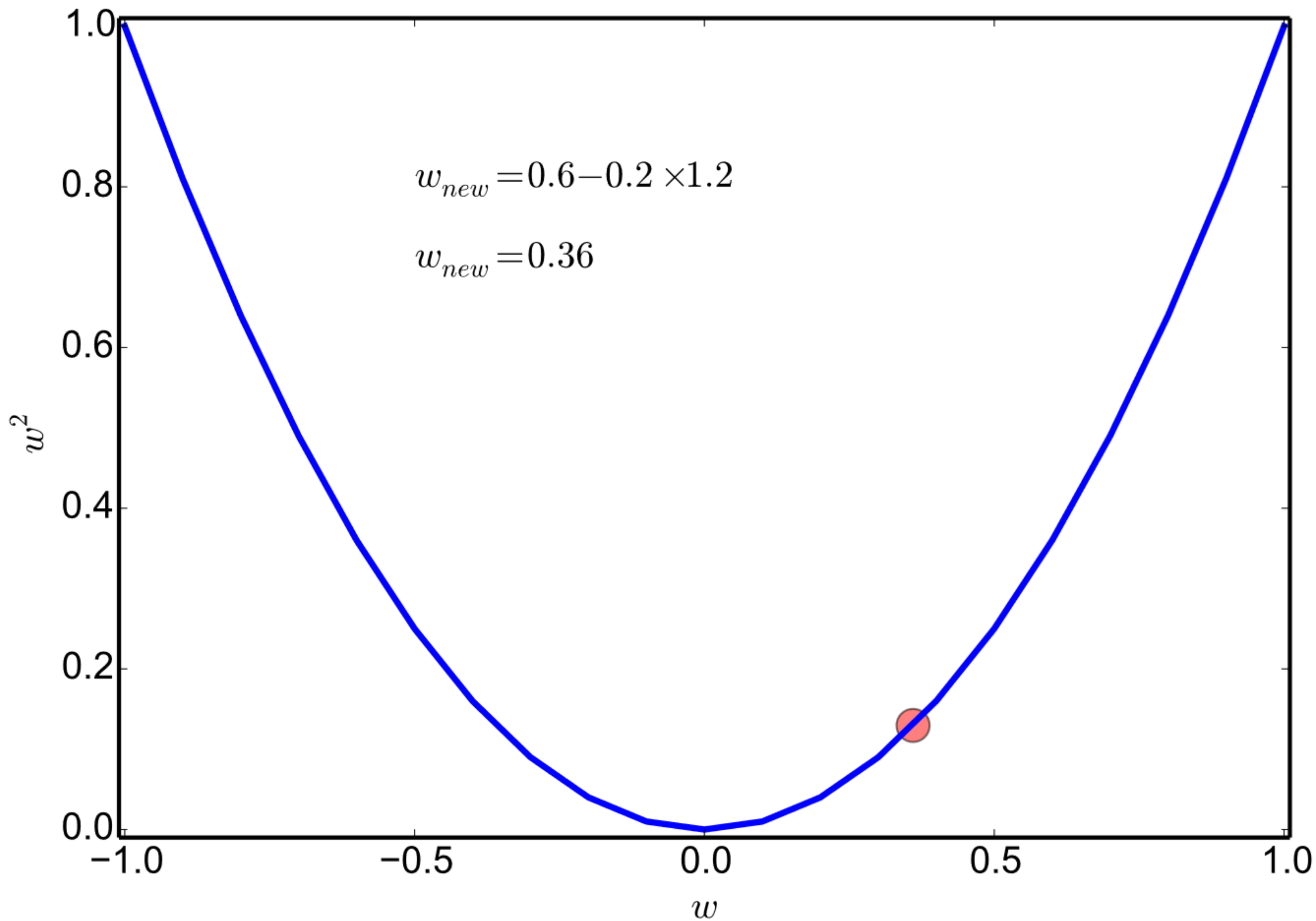
- η is the **learning rate**, controling speed of descent (e.g. 0.01 or 0.2)
- Stop when ***w*** doesn't change much any more

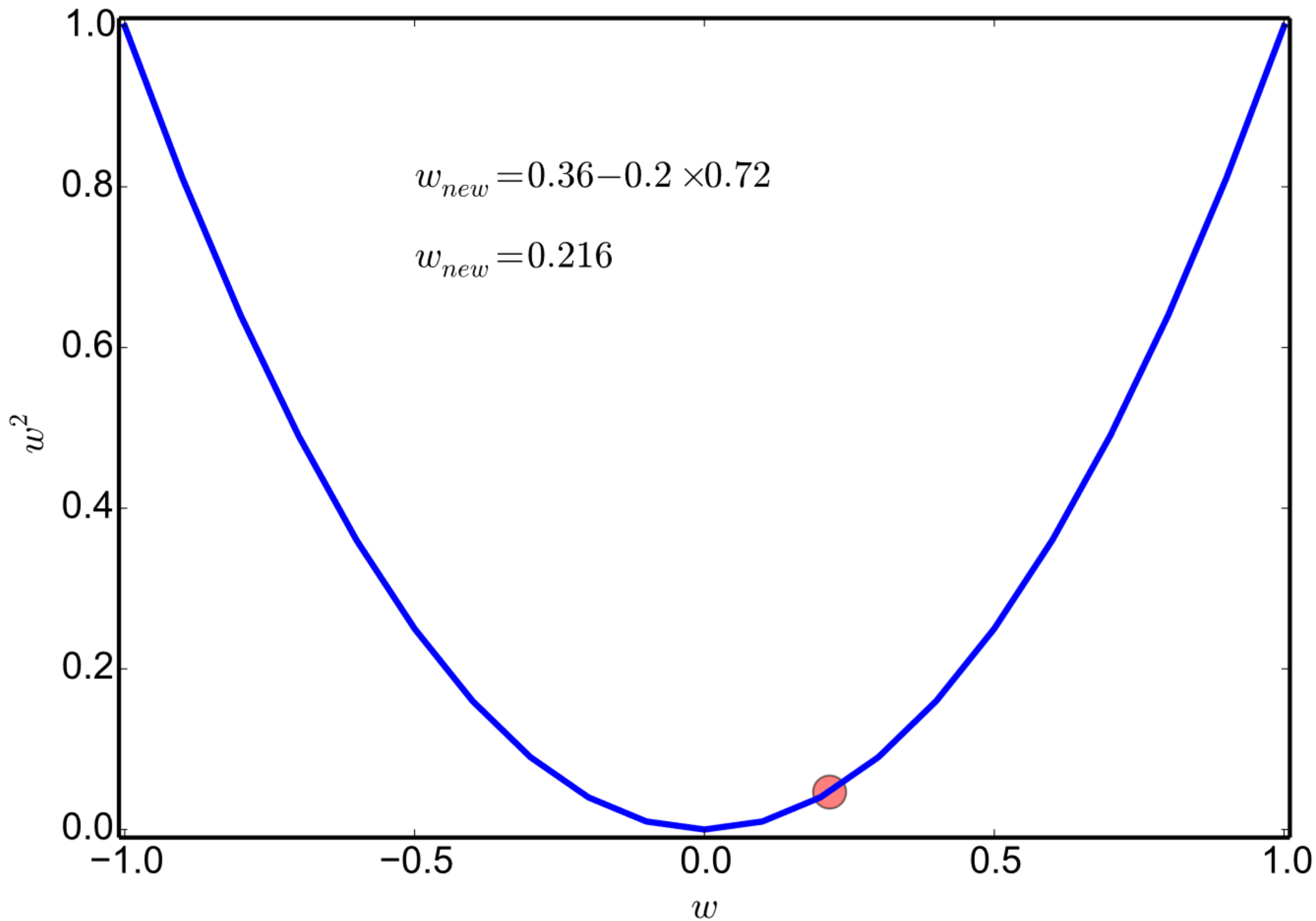


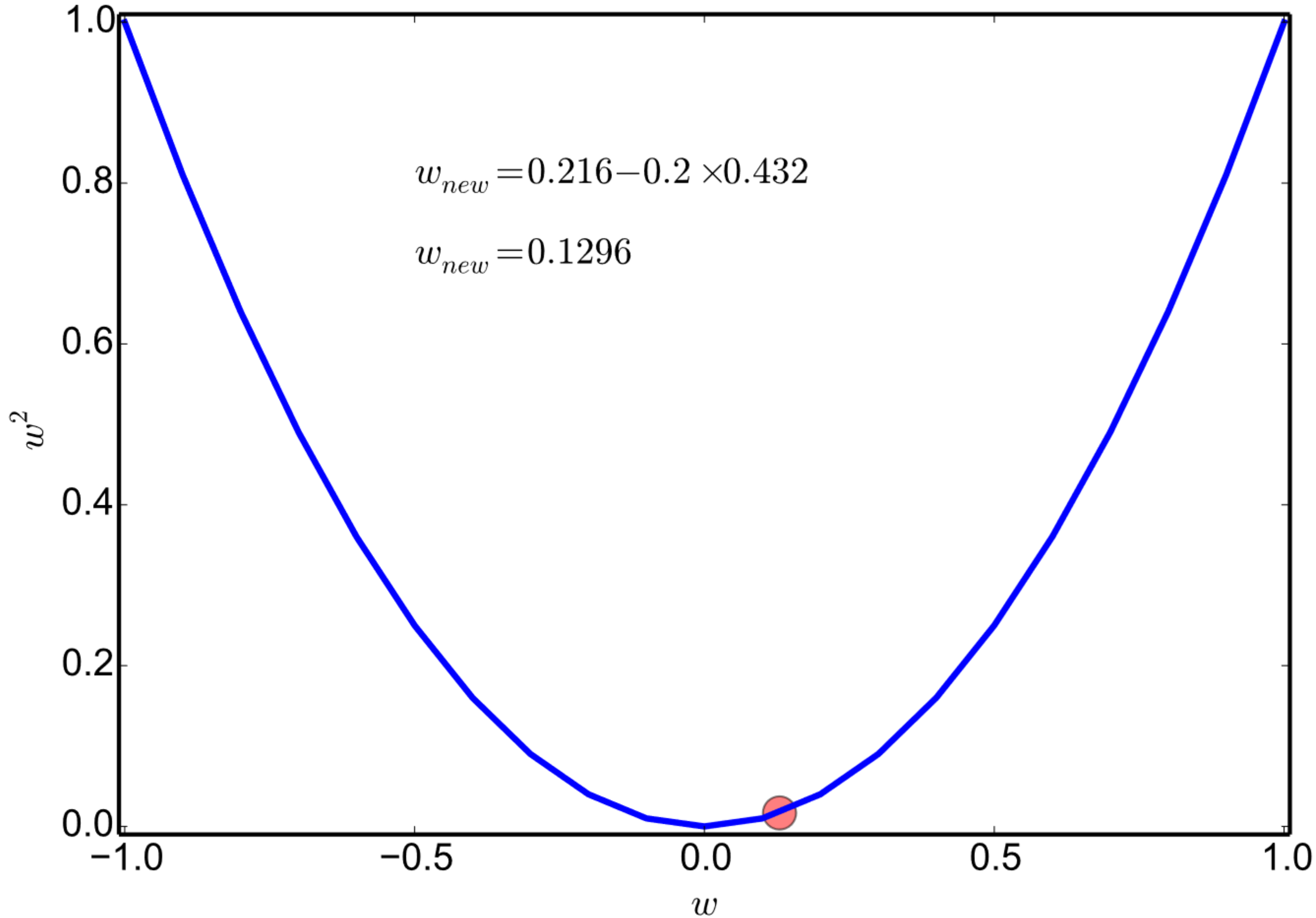
1.0
0.8
0.6
0.4
0.2
0.0
w^2
$w_{init} = 1.0$
−1.0
−0.5
0.0
0.5
1.0
w

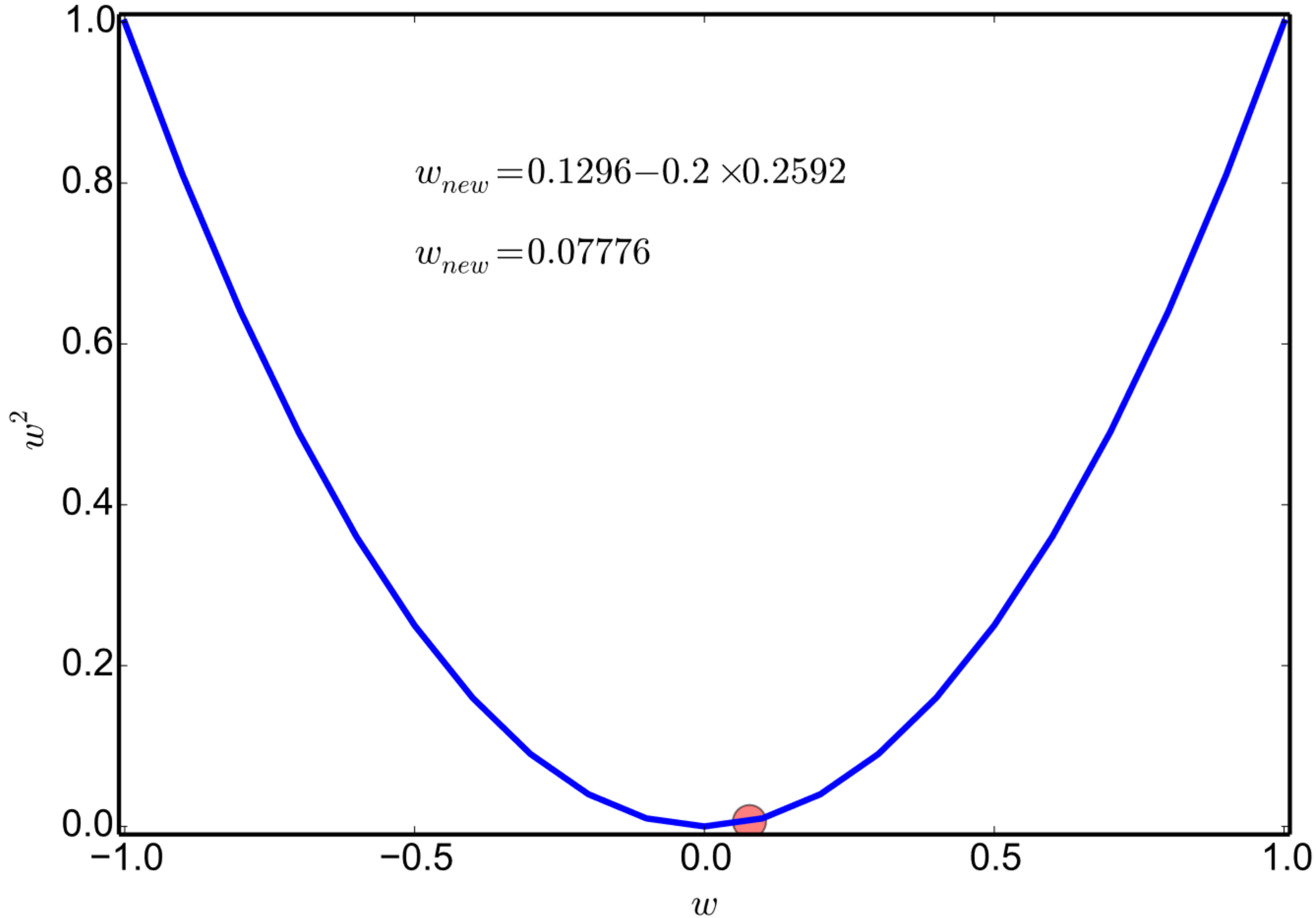


$w_{new} = 1.0 - 0.2 \times 2.0$
$w_{new} = 0.6$
1.0
0.8
0.6
0.4
0.2
0.0
w^2
−1.0
−0.5
0.0
0.5
1.0
w


$w_{new} = 0.6 - 0.2 \times 1.2$
$w_{new} = 0.36$
1.0
0.8
0.6
0.4
0.2
0.0
w^2
−1.0
−0.5
0.0
0.5
1.0
w


$w_{new} = 0.36 - 0.2 \times 0.72$
$w_{new} = 0.216$
1.0
0.8
0.6
0.4
0.2
0.0
w^2
−1.0
−0.5
0.0
0.5
1.0
w


$w_{new} = 0.216 - 0.2 \times 0.432$
$w_{new} = 0.1296$
1.0
0.8
0.6
0.4
0.2
0.0
w^2
−1.0
−0.5
0.0
0.5
1.0
w


$w_{new} = 0.1296 - 0.2 \times 0.2592$
$w_{new} = 0.07776$
1.0
0.8
0.6
0.4
0.2
0.0
w^2
−1.0
−0.5
0.0
0.5
1.0
w

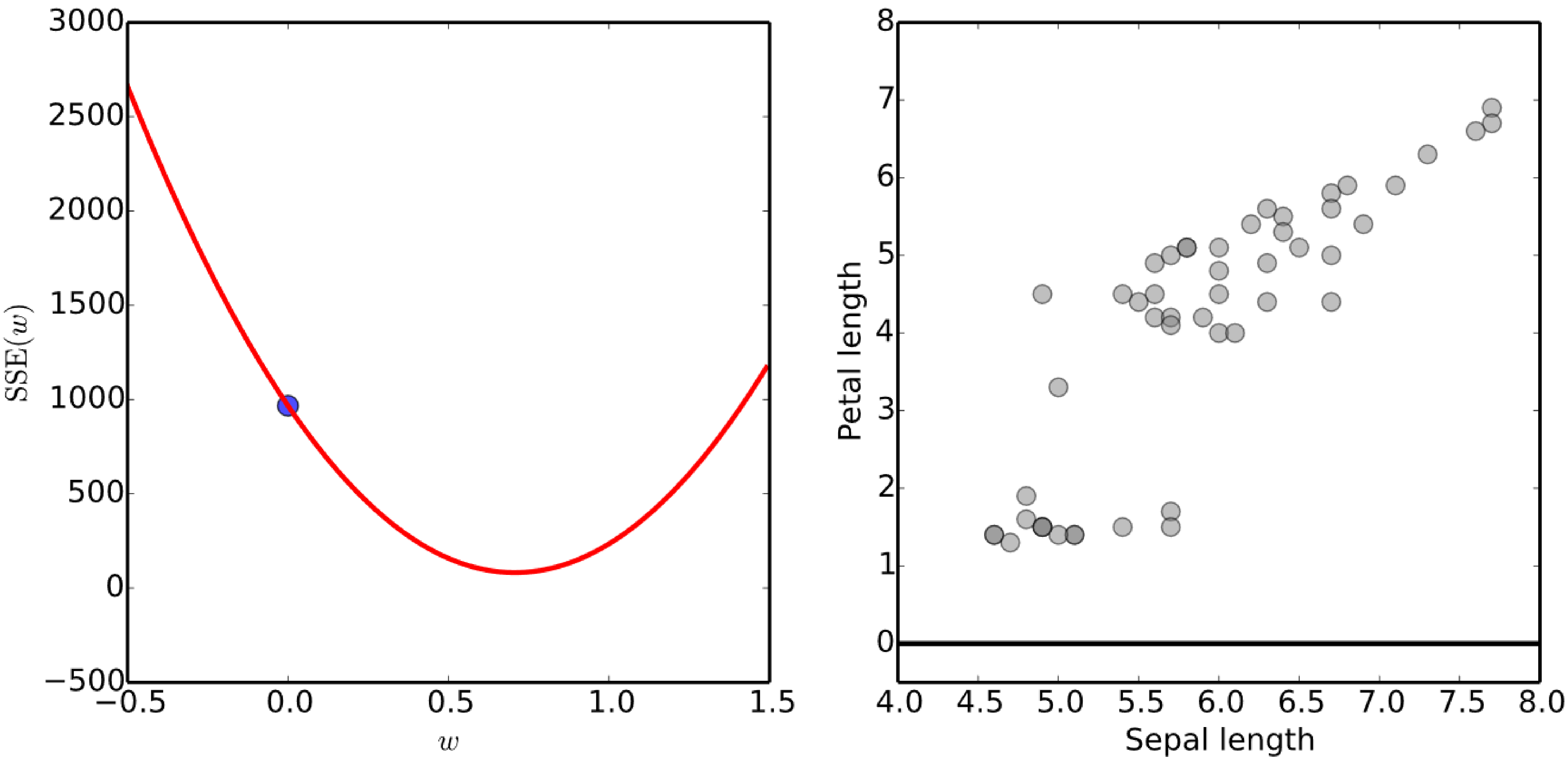
Back to function Error(**w**,b)

$$b_{\text{new}} = b_{\text{old}} - \eta \times 2\sum_{i=1}^{N}(y^i_{\text{pred}} - y^i)$$

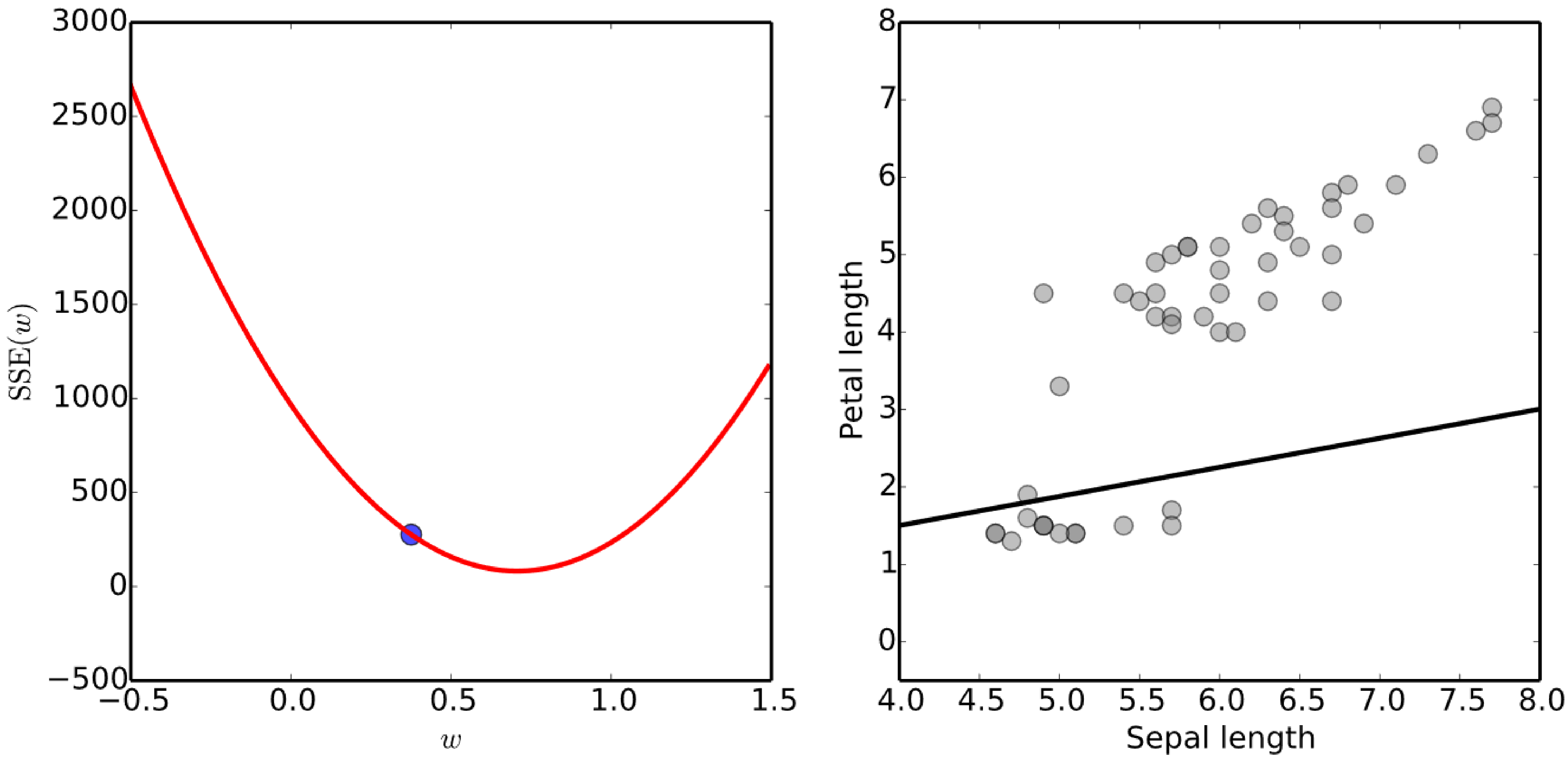
$$\mathbf{w}_{\text{new}} = \mathbf{w}_{\text{old}} - \eta \times 2\sum_{i=1}^{N}(y^i_{\text{pred}} - y^i)\mathbf{x}^i$$

Visualization

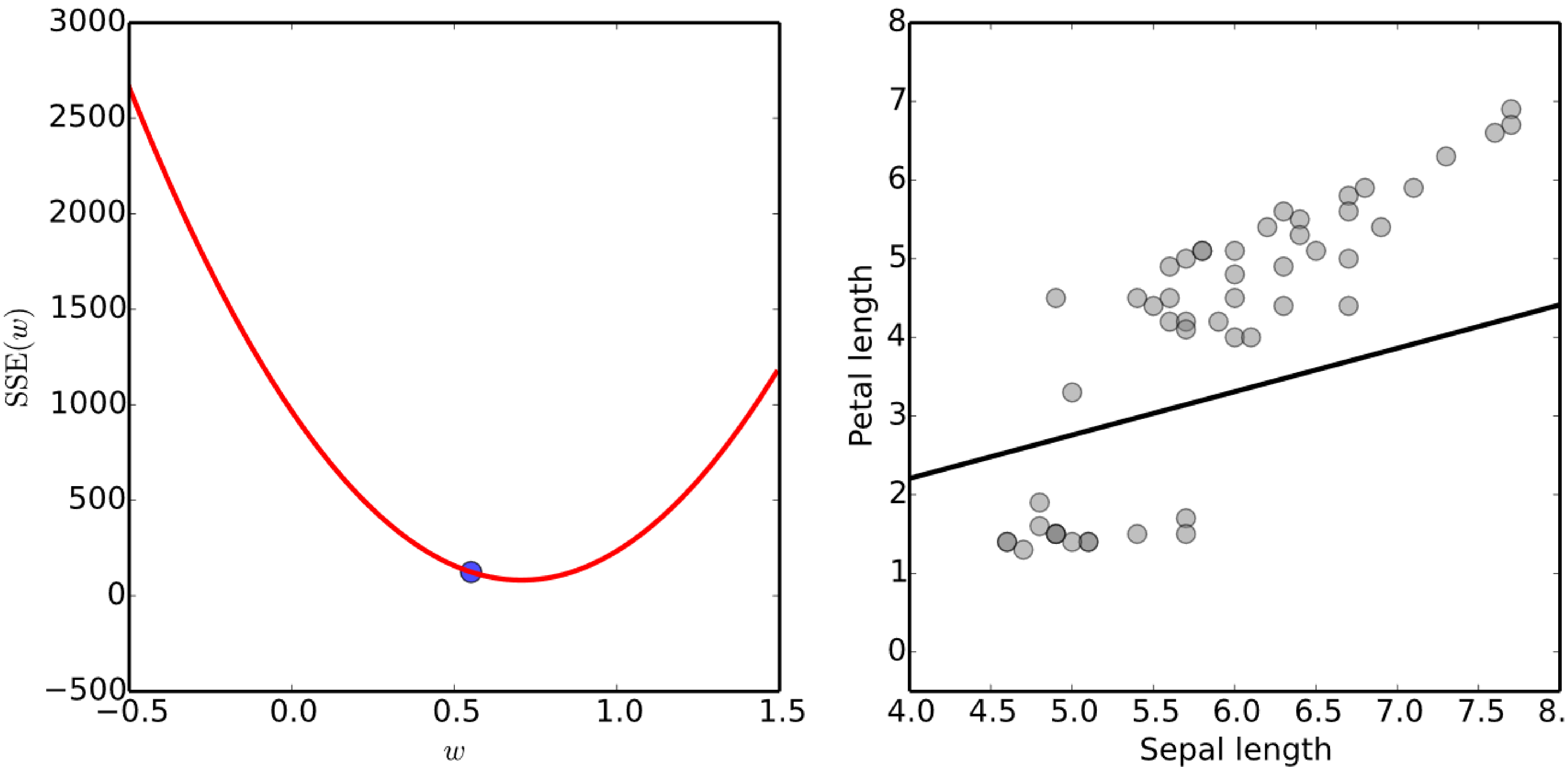
(Keep fixed b=0 to make it easier to plot)

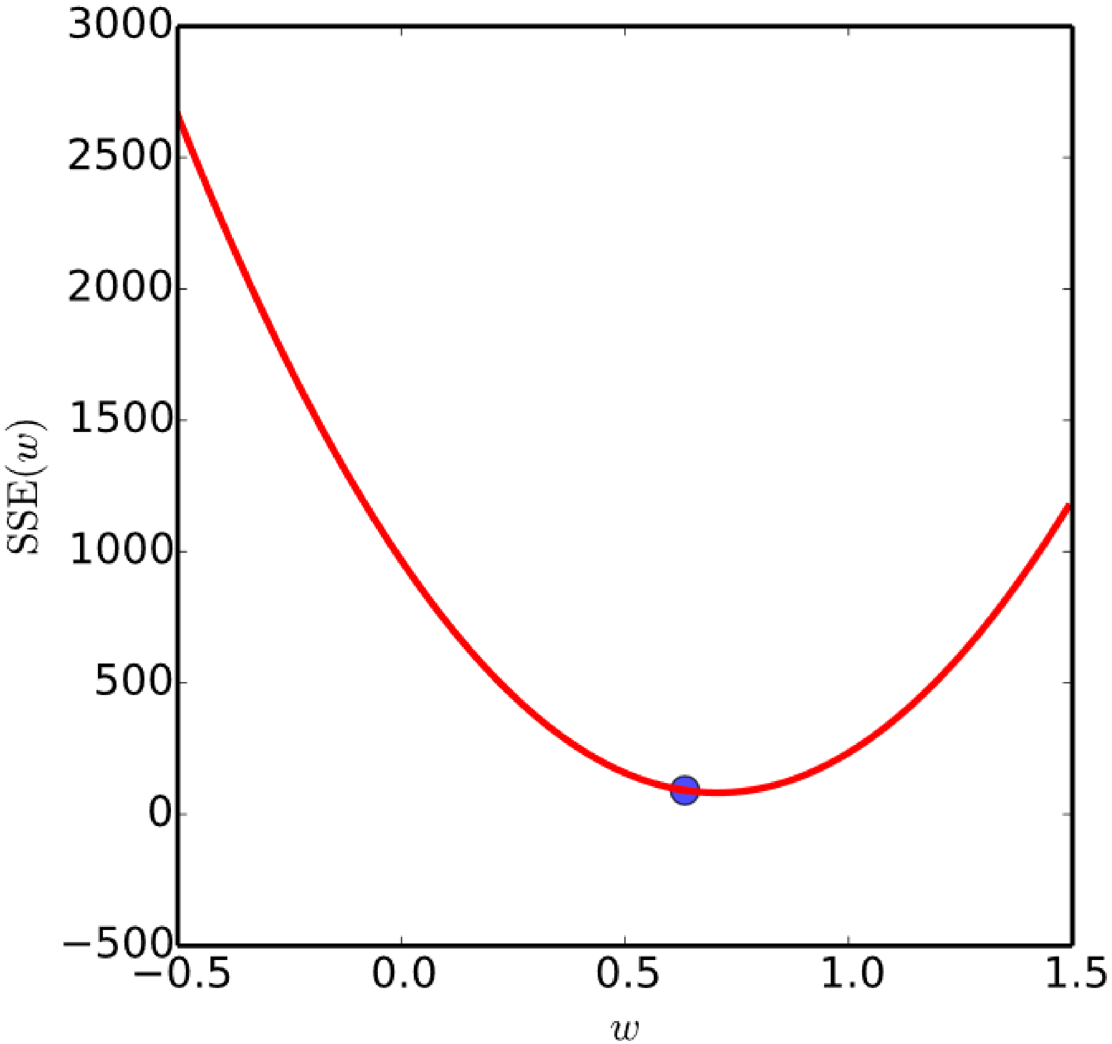


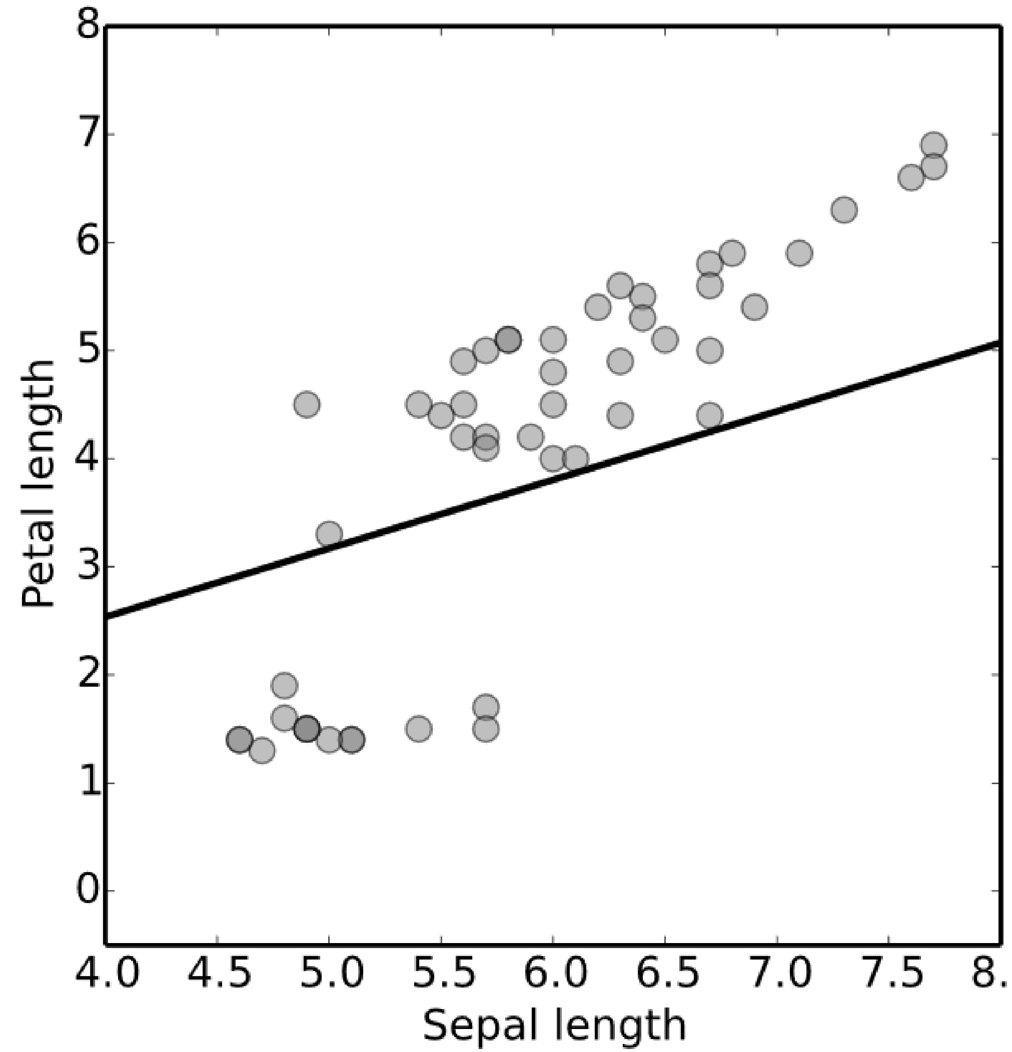
3000
2500
2000
1500
1000
500
0
−500
−0.5
0.0
0.5
1.0
1.5
w
$\mathrm{SSE}(w)$
8
7
6
5
4
3
2
1
0
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
Sepal length
Petal length

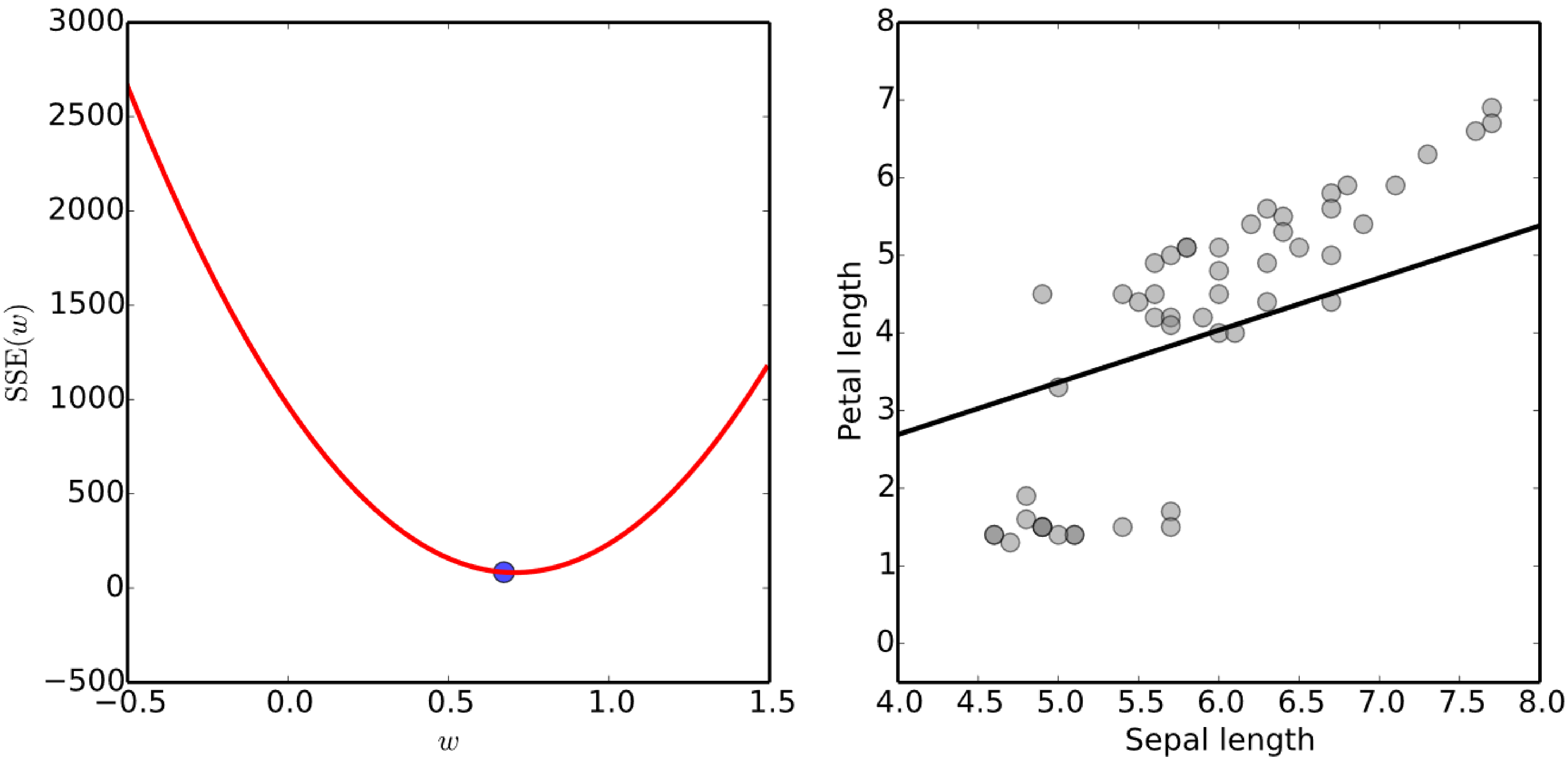


3000
2500
2000
1500
1000
500
0
−500
−0.5
0.0
0.5
1.0
1.5
w
$\mathrm{SSE}(w)$
8
7
6
5
4
3
2
1
0
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
Sepal length
Petal length

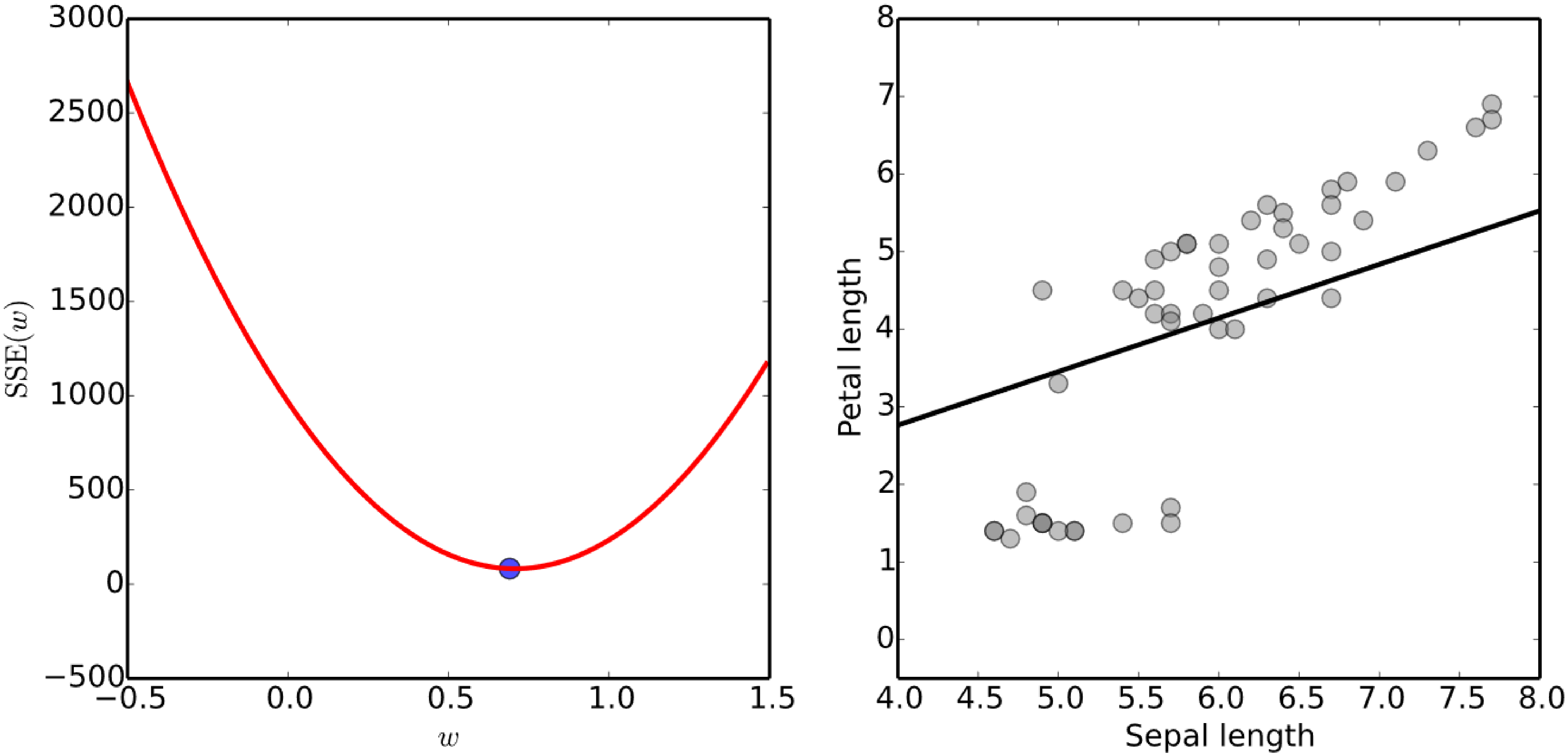

3000
2500
2000
1500
1000
500
0
−500
SSE(w)
−0.5
0.0
0.5
1.0
1.5
w
8
7
6
5
4
3
2
1
0
Petal length
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
Sepal length


3000
2500
2000
1500
1000
500
0
−500
−0.5
0.0
0.5
1.0
1.5
SSE(w)
w

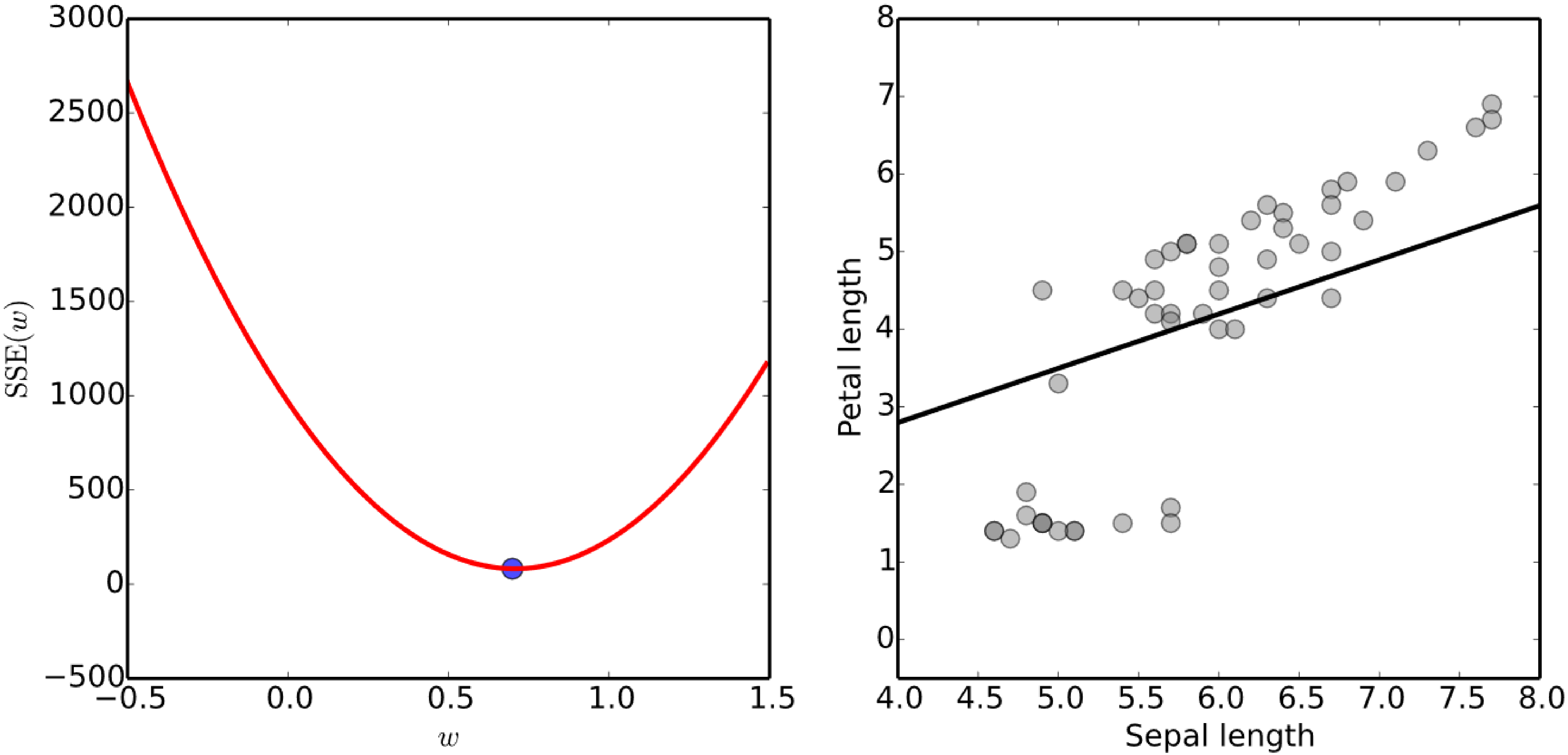

8
7
6
5
4
3
2
1
0
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
Petal length
Sepal length

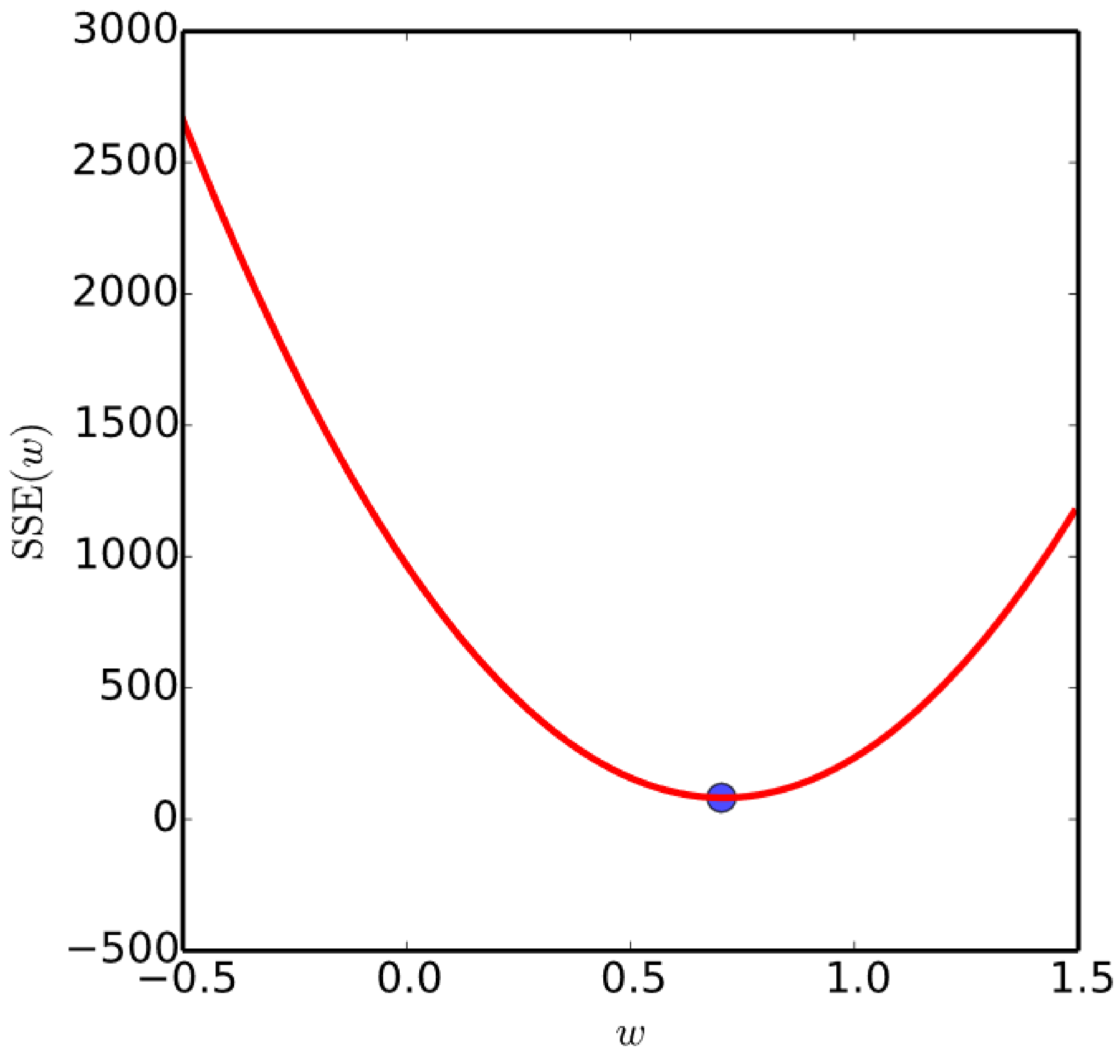


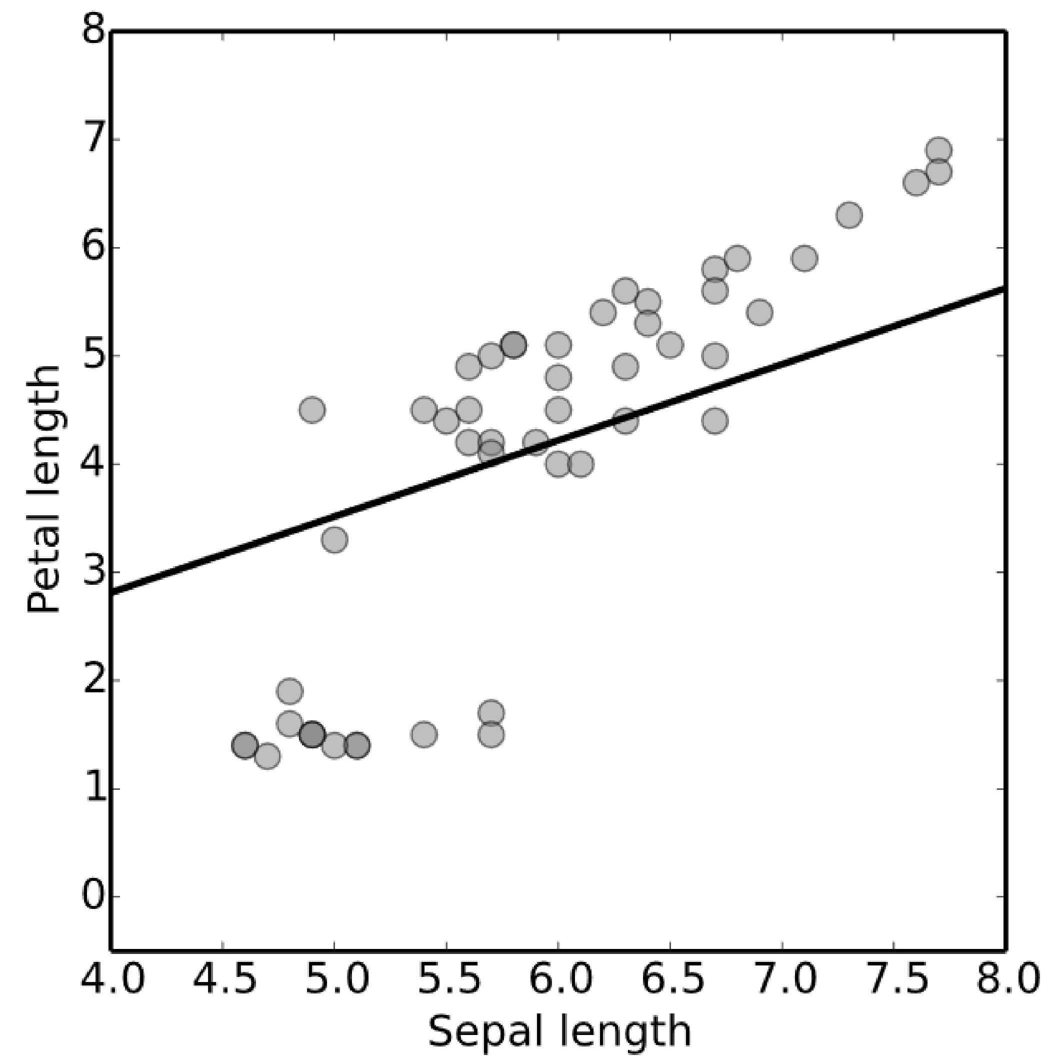
3000
2500
2000
1500
1000
500
0
−500
−0.5
0.0
0.5
1.0
1.5
w
$\mathrm{SSE}(w)$
8
7
6
5
4
3
2
1
0
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
Sepal length
Petal length



3000
2500
2000
1500
1000
500
0
−500
−0.5
0.0
0.5
1.0
1.5
w
SSE(w)
8
7
6
5
4
3
2
1
0
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
Sepal length
Petal length


3000
2500
2000
1500
1000
500
0
−500
−0.5
0.0
0.5
1.0
1.5
SSE(w)
w
8
7
6
5
4
3
2
1
0
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
Petal length
Sepal length


3000
2500
2000
1500
1000
500
0
−500
−0.5
0.0
0.5
1.0
1.5
w
$\mathrm{SSE}(w)$


8
7
6
5
4
3
2
1
0
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
Sepal length
Petal length

Gradient descent

- For many types of models, we can find good model parameters with gradient descent
- Important to use appropriate learning rate!

Go to **wooclap.com** and use the code **ALFXDE**

What do you think will happen if the learning rate is too big (small)?

Skip over minimal value
datasets
smalls already It's
Miss small points
Overfitting
Overshooting overshoot time
not effective matter
complexity skip learning minimum failure / first big
variability Search
consuming
value process might miss leading
iterate increases
miss the minimum
Time consuming
You might miss the minimum
skip the minimum

wooclap

100 %

18

Problem: learning rate

- What will happen if the learning rate is too small?
- And if it's too big?

Gradient descent with big datasets

$$\mathbf{w}_{\text{new}} = \mathbf{w}_{\text{old}} - \eta \times 2 \sum_{i=1}^{N} (y^i_{\text{pred}} - y^i)\mathbf{x}^i$$

We have to compute predictions and calculate residuals for all the examples before making an update. Can we do something faster?

Idea: update more often

- Instead, we could update after every example (or after every 100):

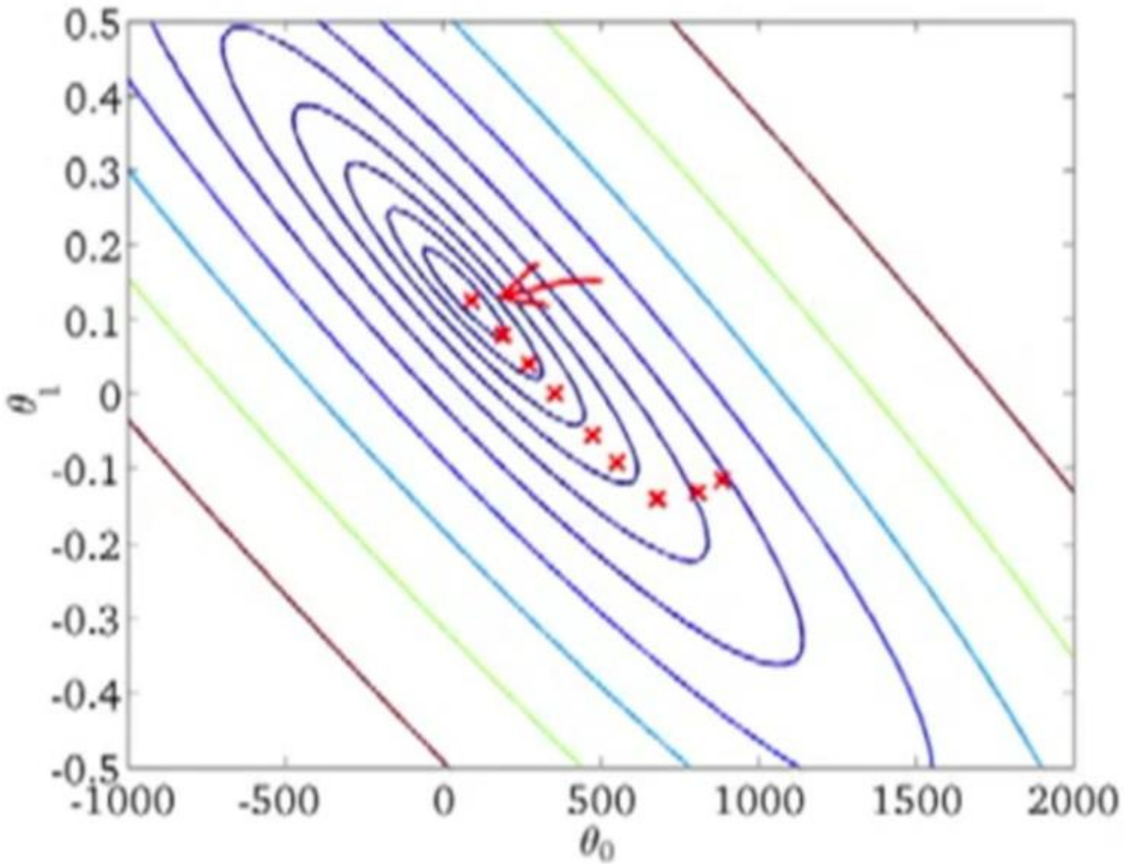
1: **for** $i = 1$ **to** N **do**

2: $\quad \mathbf{w}_{\text{new}} = \mathbf{w}_{\text{old}} - \eta \times 2(y^i_{\text{pred}} - y^i)\mathbf{x}^i$

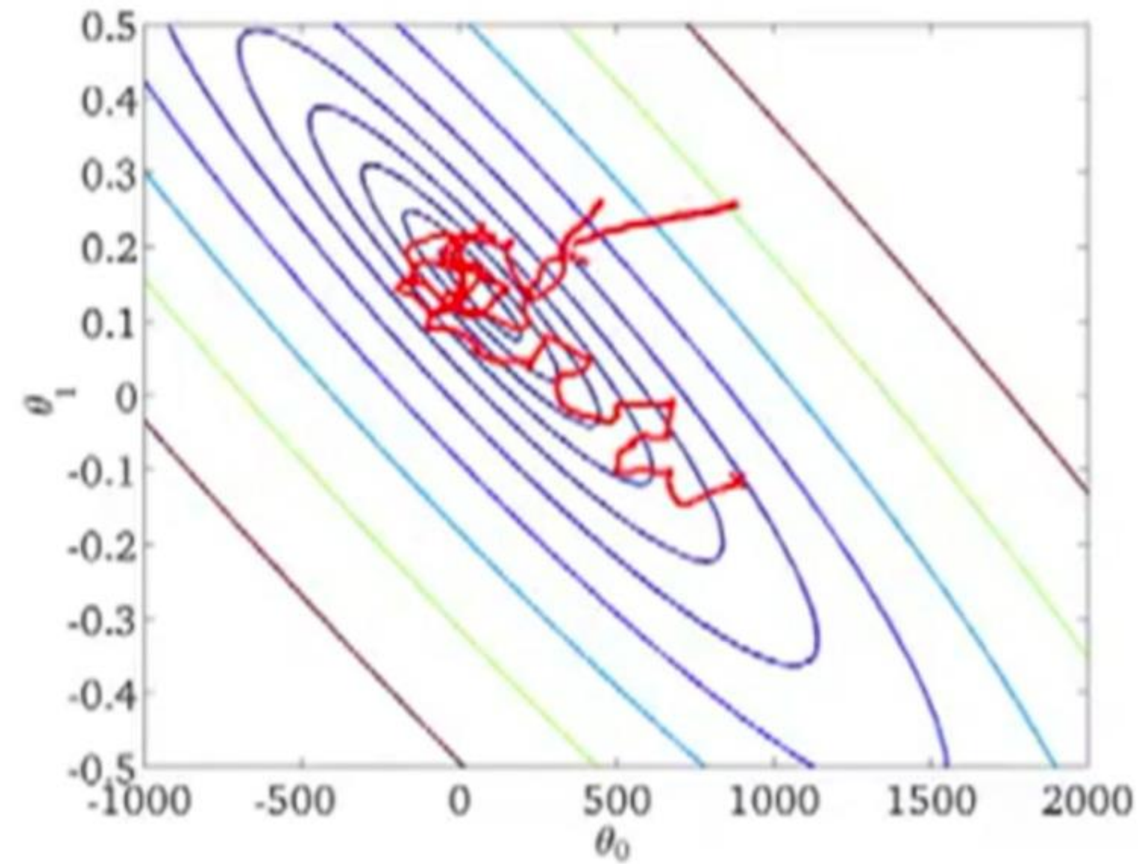
Stochastic Gradient Descent

- **Stochastic** gradient descent (aka **SGD**)

GD vs SGD



Batch Gradient Descent



Stochastic Gradient Descent

https://www.quora.com/Is-stochastic-gradient-descent-online-How-do-I-intuitively-prove-that-it-converges

Stochastic Gradient Descent

- **Stochastic** gradient descent (aka **SGD**)
- Suitable for online learning scenarios
- Compare with the Perceptron update rule
- Workhorse of modern Machine Learning
- Large, deep neural networks

Go to **wooclap.com** and use the code **ALFXDE**

Can we change the learning rate between the iterations?

1	Yes	65%	17
2	No	19%	5
3	I'm not sure!	15%	4

wooclap

100 %

26 / 55

Can we use a variable learning rate?

- How would we vary it?

Momentum

- A modification to SGD which smooths gradient estimates with memory
- No modification to learning rate

$$\mathbf{u}_t = \beta \mathbf{u}_{t-1} + (1 - \beta) f'(\mathbf{w}_{\text{old}})$$

$$\mathbf{w}_{\text{new}} = \mathbf{w}_{\text{old}} - \eta \mathbf{u}_t$$

How do we find the derivatives?

- Symbolic or automatic differentiation
- We can get gradients for complicated functions composed of differentiable operations
- Automatic application of chain rule
- Tensorflow, PyTorch

What about local minima?

- Potential problem for non-linear models
- e.g. Neural Networks
- In practice, not necessarily a big problem in high-dimensional data

Summary

- Modular learning:
  Model + Optimization
- Gradient descent to find model parameters with lowest error
- Stochastic version widely used.