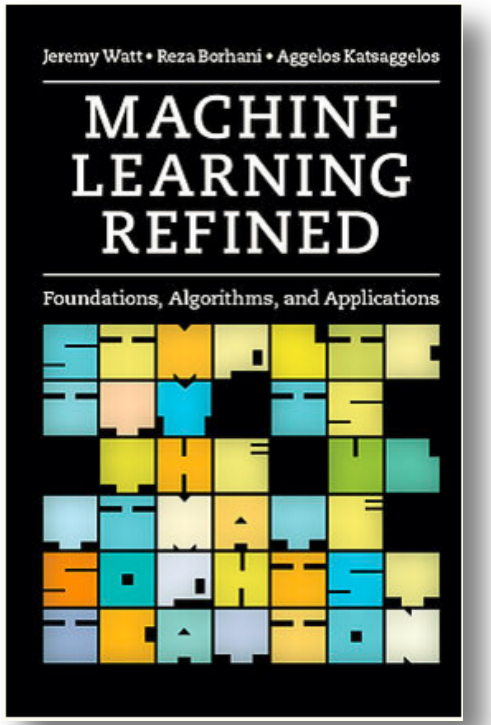
# Mathematical Optimization for Machine Learning

# What is this talk based on?

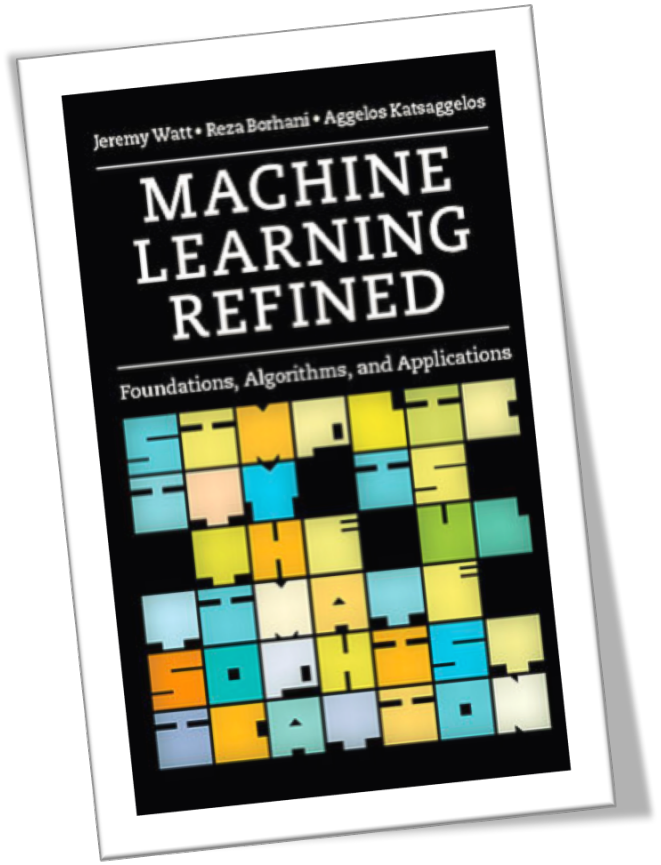



A new ML textbook!
Cambridge University Press
*July 2016*

Several large ML courses taught at
Northwestern University
*Winter 2014, Winter 2015, Fall 2015*

# What makes our book unique?

- A presentation built on lucid geometric intuition
  (w/ 200+ color illustrations)
- A learning experience where students learn by doing
  (w/ 50+ in depth coding exercises)
- Inclusion of real application to motivate concepts
  (computer vision, natural language processing, economics, neuroscience, recommender systems, physics, biology, etc.)

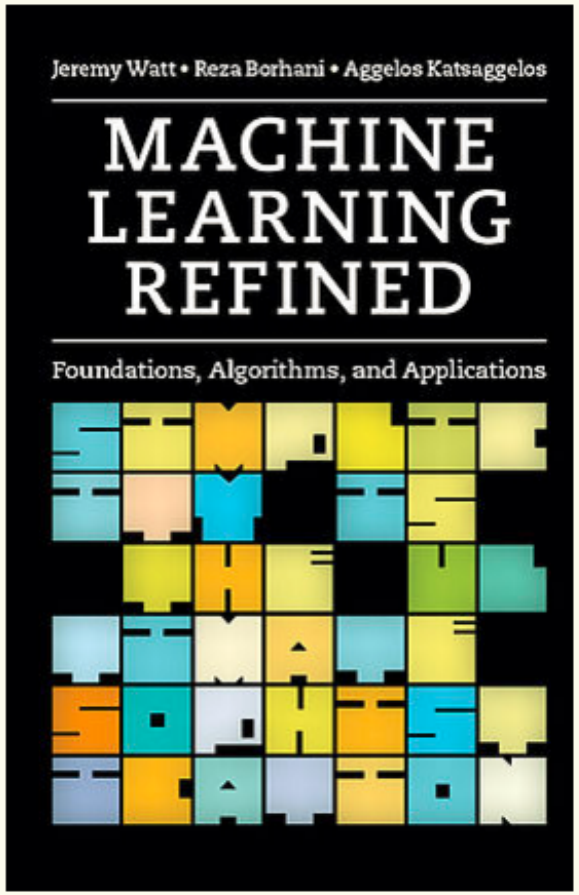



- A rigorous yet user friendly presentation of state-of-the-art numerical techniques
  (gradient descent, Newton's method, stochastic gradient descent, accelerated gradient descent, Lipschitz step-length rule, adaptive step-length selection, and more)

# Where to download the slides?

mlrefined.com

**www.MLrefined.com**




Pre-order on Amazon

Machine Learning Refined (to appear early 2016 with Cambridge University Press) is a new machine learning textbook containing fresh and intuitive yet rigorous descriptions of the most fundamental concepts necessary to conduct research, build products, tinker, and play. The text includes a plethora of practical examples and exercises, written in both Python and Matlab/Octave programming languages, to help readers gain complete mastery of the subject.

To learn more about what makes our textbook so great click here, and then see for yourself by downloading free sample chapters via the link below!


**click here**

# Intended audience

To get the most from this lesson you should be generally familiar with

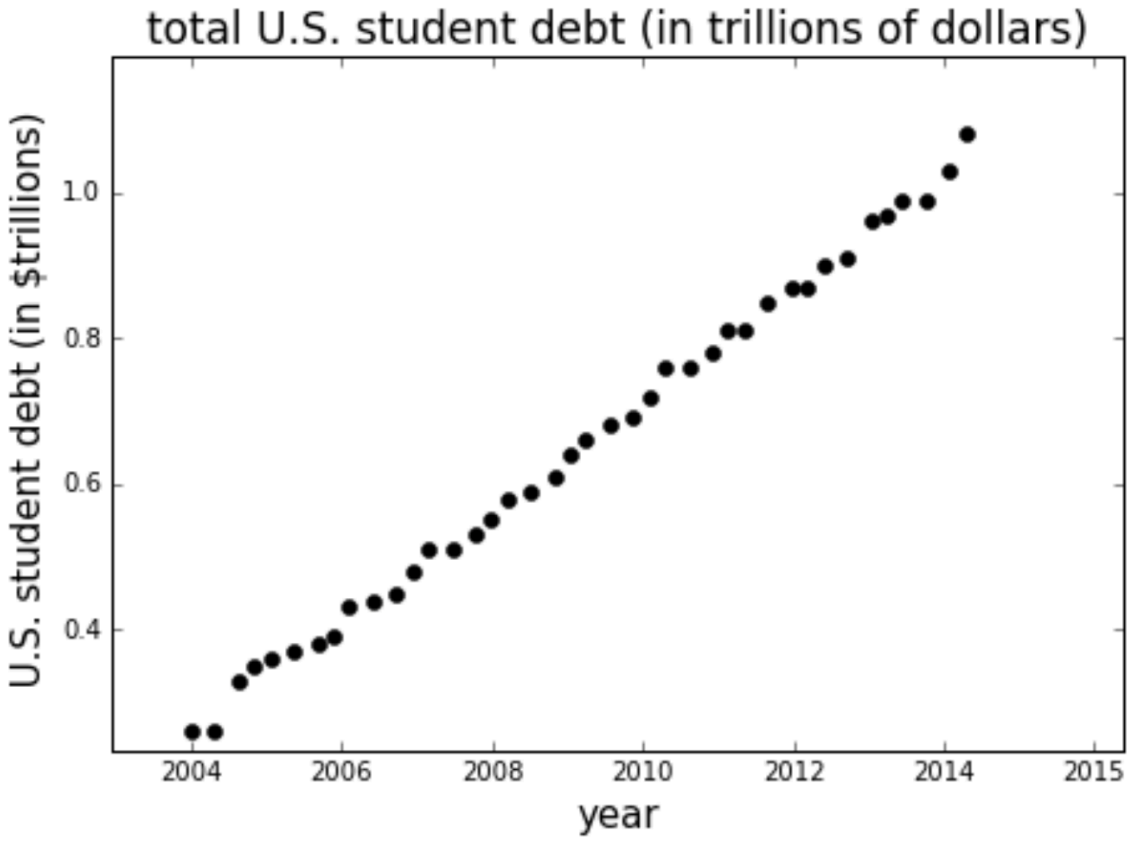
- function notation and the notion of a derivative (from calculus)
- python programming and the IPython interactive notebook

# Outline

1. Motivation: linear regression
2. The search for the best parameters
3. The big picture on mathematical optimization
4. The gradient descent algorithm: how does it generally work?

# Linear regression

**Regression** ⇒ summarize a trend or predict future ourput values




data shown here taken from [2]

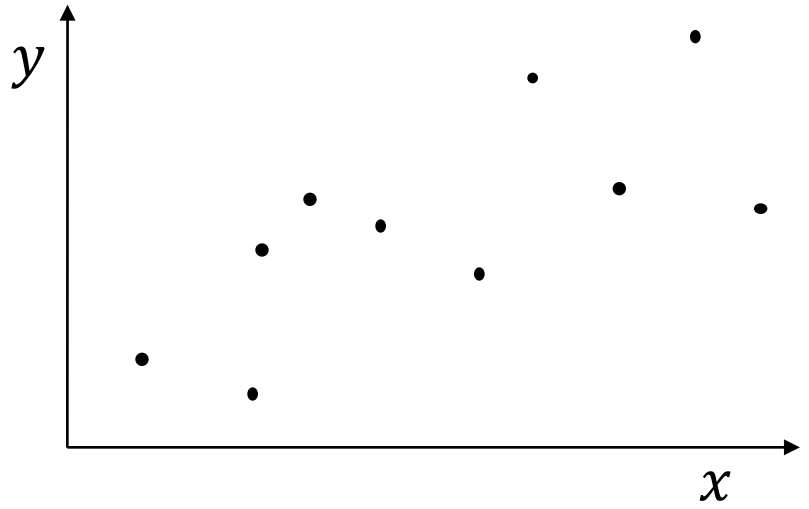
# Linear regression

**Data:** set of $P$ input/output pairs

$$(x_1, y_1), (x_2, y_2), \cdots, (x_P, y_P)$$

**Output**

**Input**



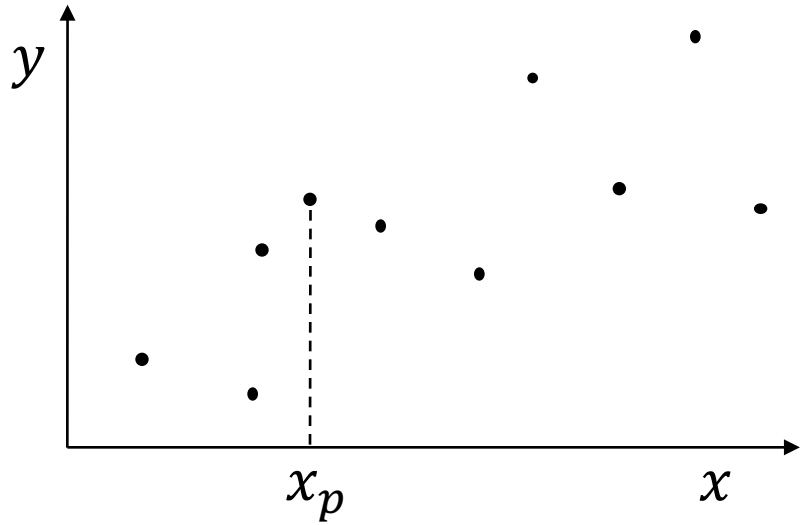
# Linear regression

**Data:** set of $P$ input/output pairs

$$(\boldsymbol{x_1}, \boldsymbol{y_1}), (\boldsymbol{x_2}, \boldsymbol{y_2}), \cdots, (\boldsymbol{x_P}, \boldsymbol{y_P})$$

**Output**

**Input**



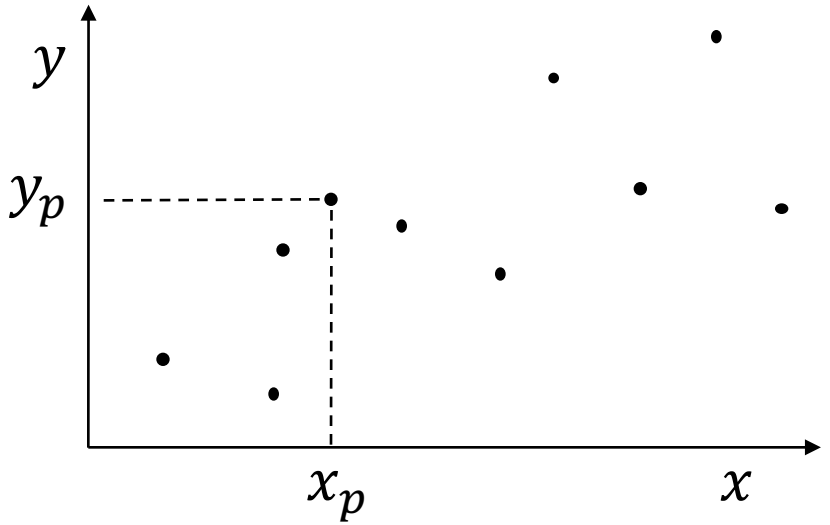
# Linear regression

**Data:** set of $P$ input/output pairs

$$(\boldsymbol{x_1}, \boldsymbol{y_1}), (\boldsymbol{x_2}, \boldsymbol{y_2}), \cdots, (\boldsymbol{x_P}, \boldsymbol{y_P})$$

**Output**

**Input**



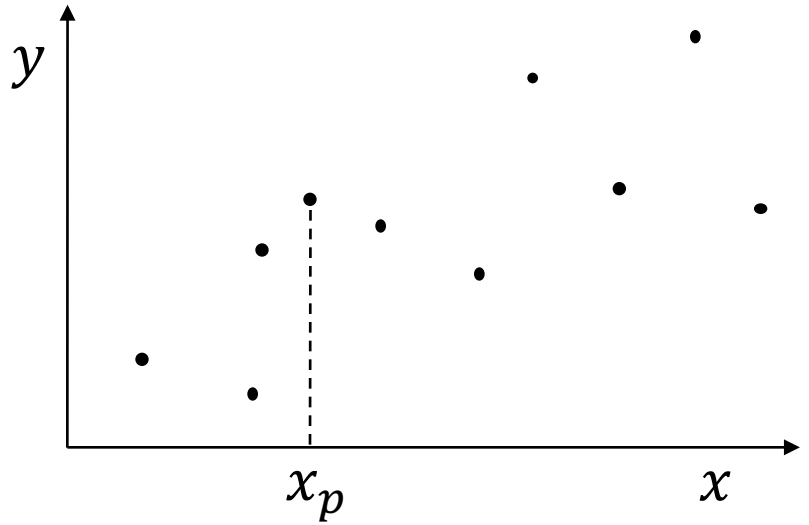
# Linear regression

**Data:** set of $P$ input/output pairs

$$(\boldsymbol{x_1}, \boldsymbol{y_1}), (\boldsymbol{x_2}, \boldsymbol{y_2}), \cdots, (\boldsymbol{x_P}, \boldsymbol{y_P})$$

**Output**

**Input**



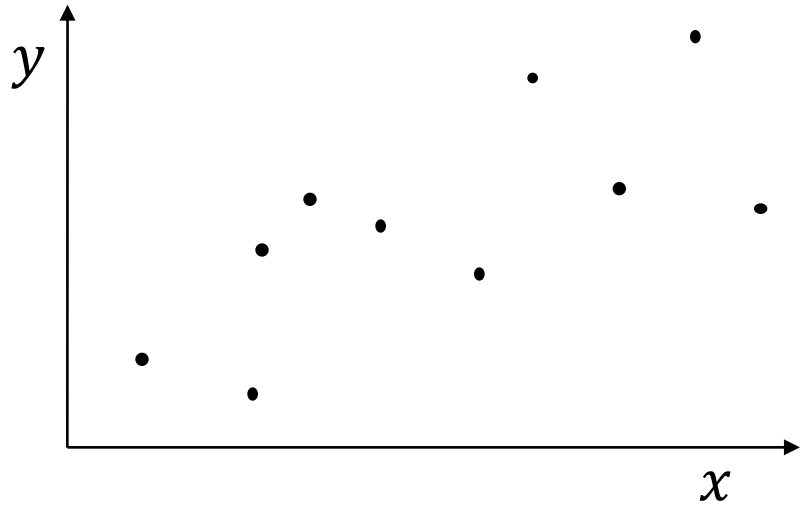
# Linear regression

**Data:** set of $P$ input/output pairs

$$(x_1, y_1), (x_2, y_2), \cdots, (x_P, y_P)$$

**Output**

**Input**



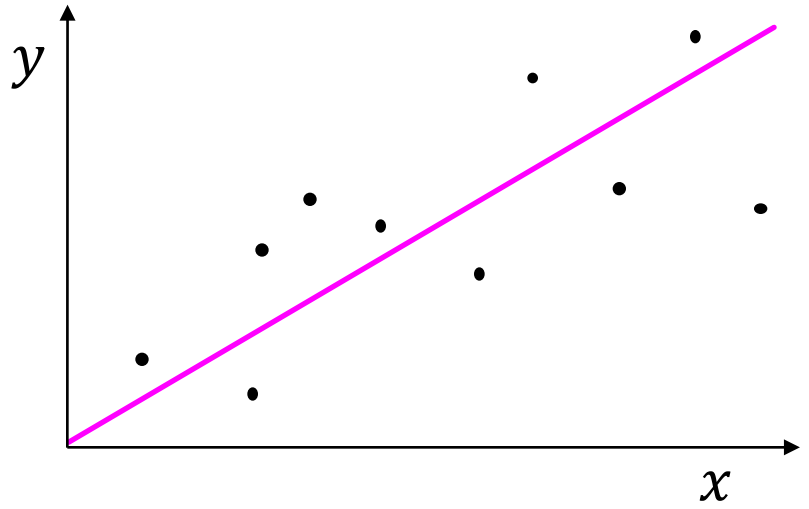
# Linear regression

**Data:** set of $P$ input/output pairs

$$(x_1, y_1), (x_2, y_2), \cdots, (x_P, y_P)$$

**Output**

**Input**



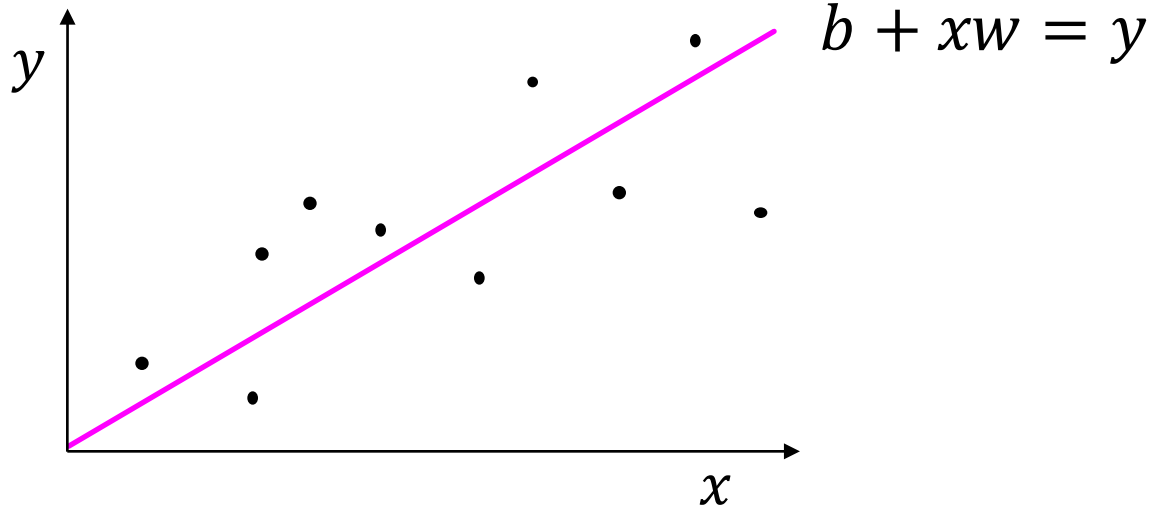
# Linear regression

**Data:** set of $P$ input/output pairs

$$(\boldsymbol{x_1}, \boldsymbol{y_1}), (\boldsymbol{x_2}, \boldsymbol{y_2}), \cdots, (\boldsymbol{x_P}, \boldsymbol{y_P})$$

**Output**

**Input**



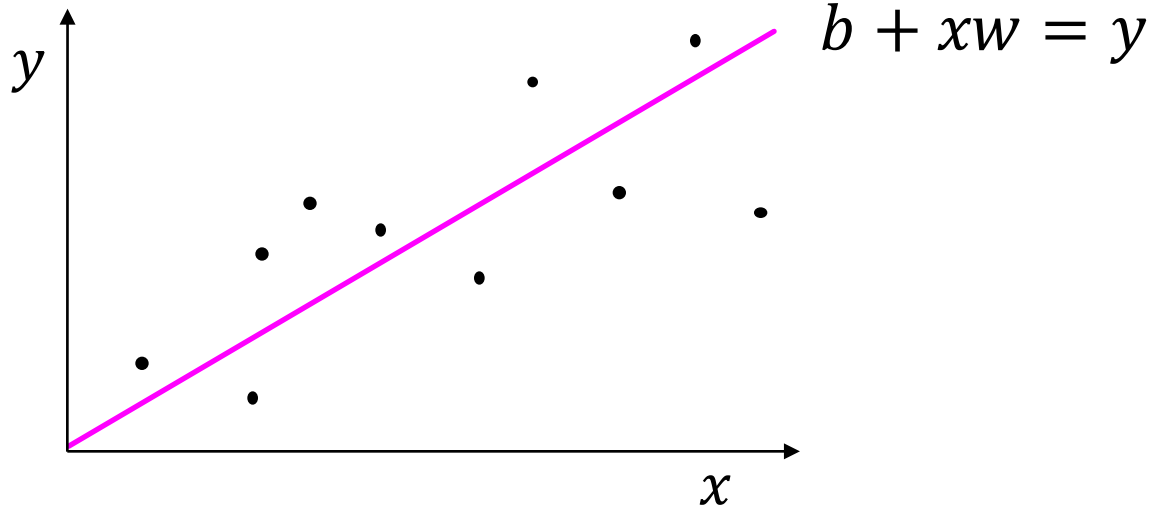
# Linear regression

**Data:** set of $P$ input/output pairs

$$(\boldsymbol{x_1}, \boldsymbol{y_1}), (\boldsymbol{x_2}, \boldsymbol{y_2}), \cdots, (\boldsymbol{x_P}, \boldsymbol{y_P})$$

**Output**

**Input**




Find intercept $b$ and slope $w$ so line passes through all points as best as possible

$$b + x_p w \approx y_p$$

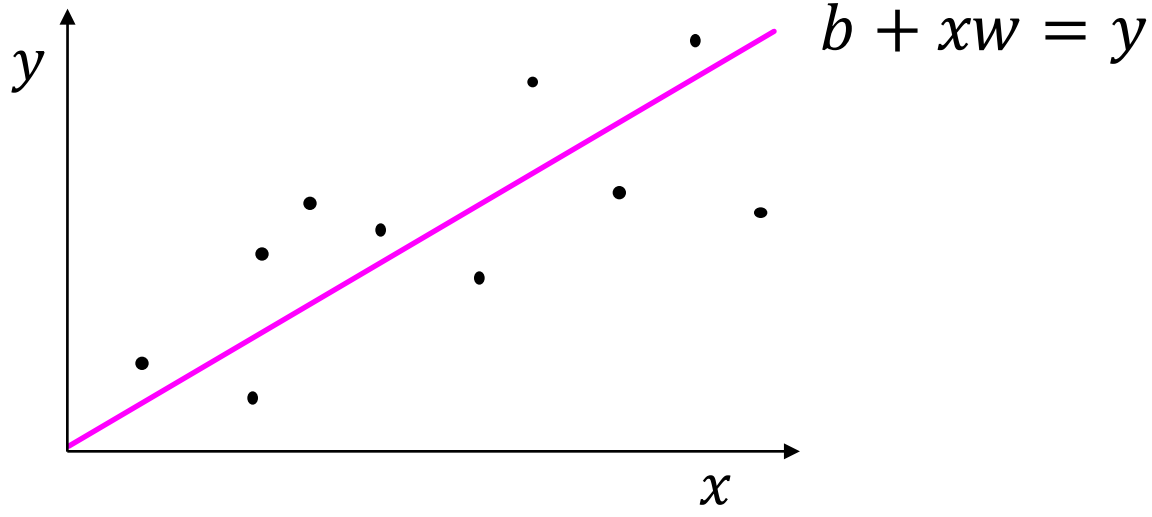
# Linear regression

**Data:** set of $P$ input/output pairs

$$(\boldsymbol{x_1}, \boldsymbol{y_1}), (\boldsymbol{x_2}, \boldsymbol{y_2}), \cdots, (\boldsymbol{x_P}, \boldsymbol{y_P})$$

**Output**

**Input**

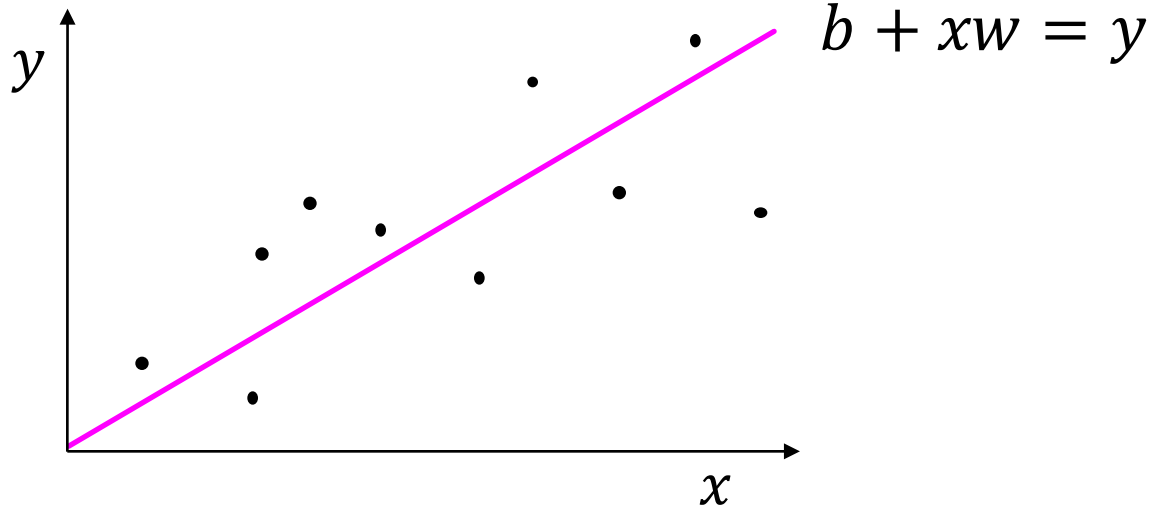



Find intercept $b$ and slope $w$ so line passes through all points as best as possible

$$b + x_p w \approx y_p$$

Want this for all $p = 1 \ldots P$

# Linear regression

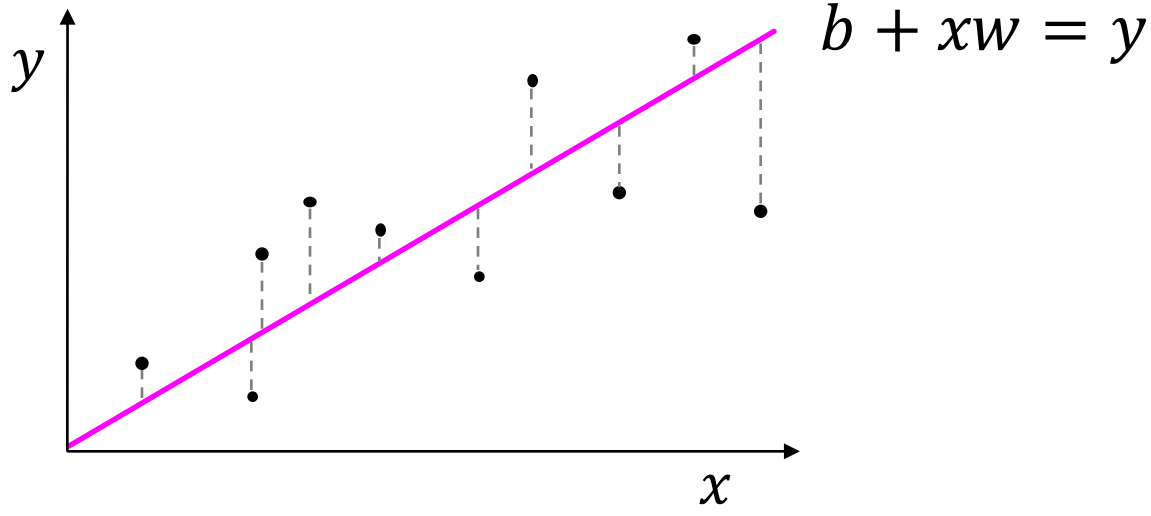



Find intercept $b$ and slope $w$ so line passes through all points as best as possible

$$b + x_p w \approx y_p$$

Want this for all $p = 1 \ldots P$

# Linear regression

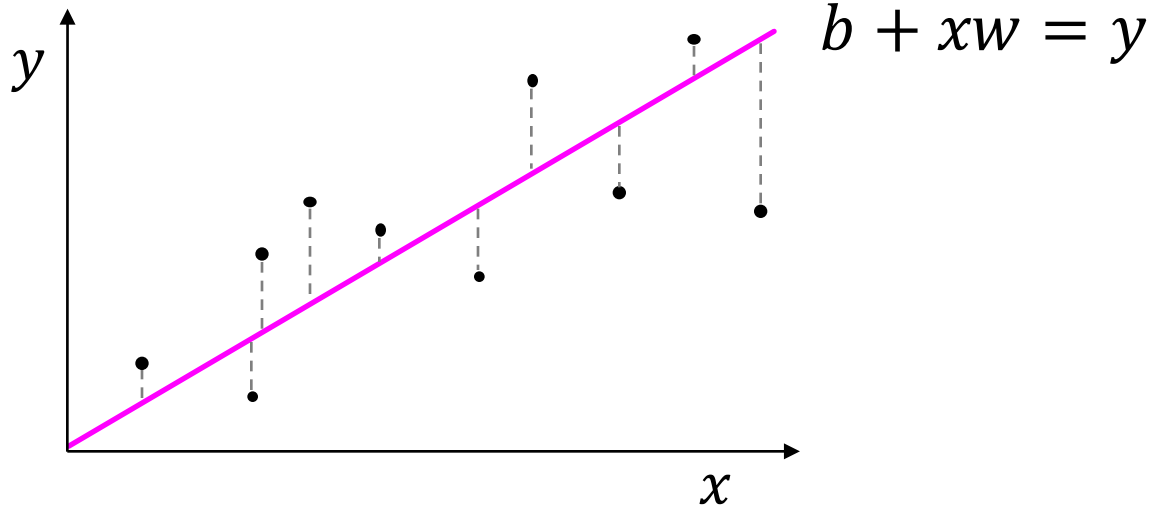



Find intercept $b$ and slope $w$ so line passes through all points as best as possible

$$b + x_p w \approx y_p$$

Want this for all $p = 1 \ldots P$

# Linear regression

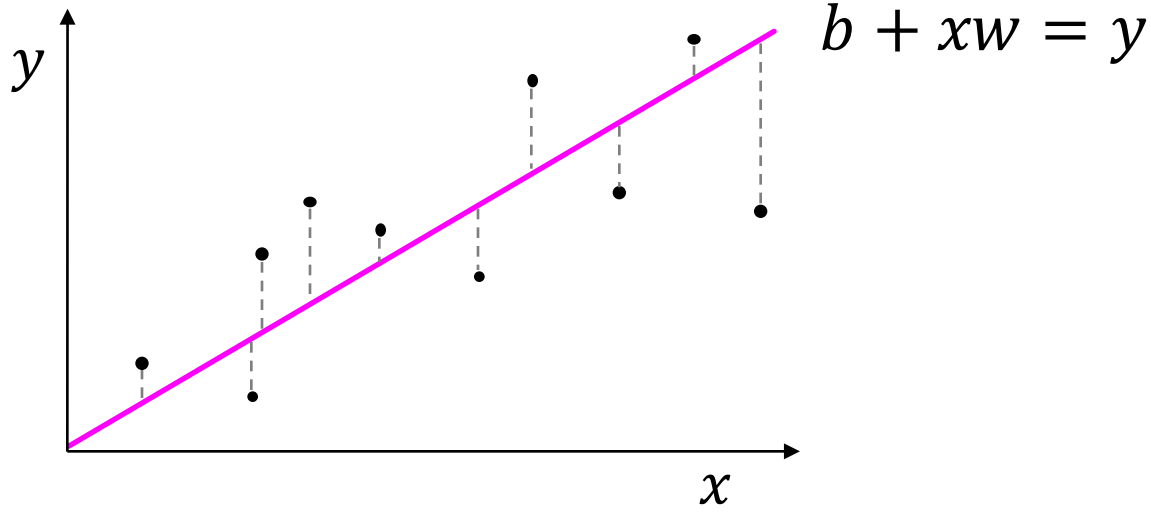



Find intercept $b$ and slope $w$ so line has ***smallest*** squared error with data points

$$b + x_p w \approx y_p$$

Want this for all $p = 1 \ldots P$

# Linear regression

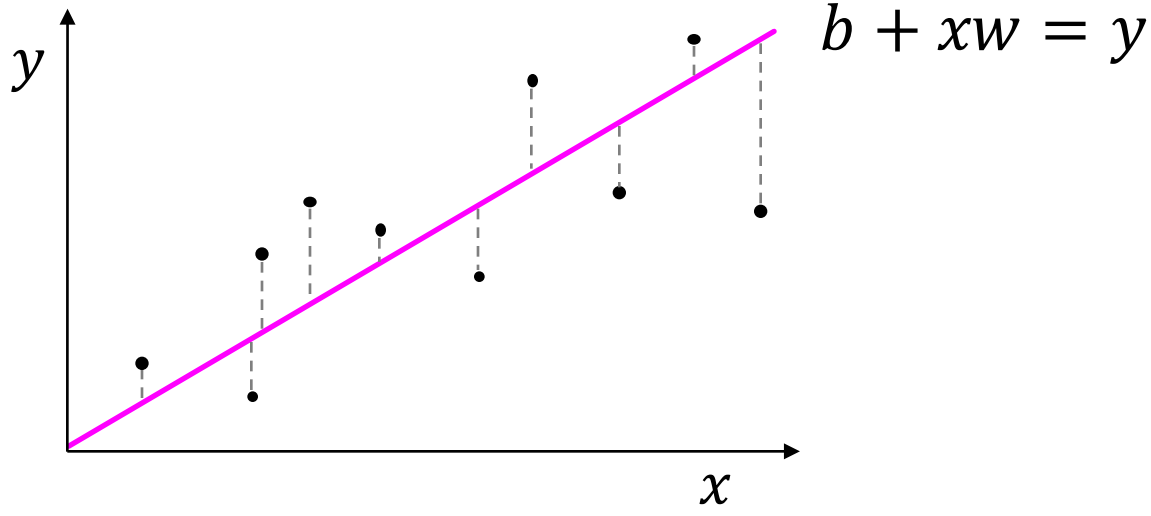



Find intercept $b$ and slope $w$ so line has ***smallest*** squared error with data points

$$\left(b + x_p w - y_p\right)^2$$

Want this for all $p = 1 \ldots P$

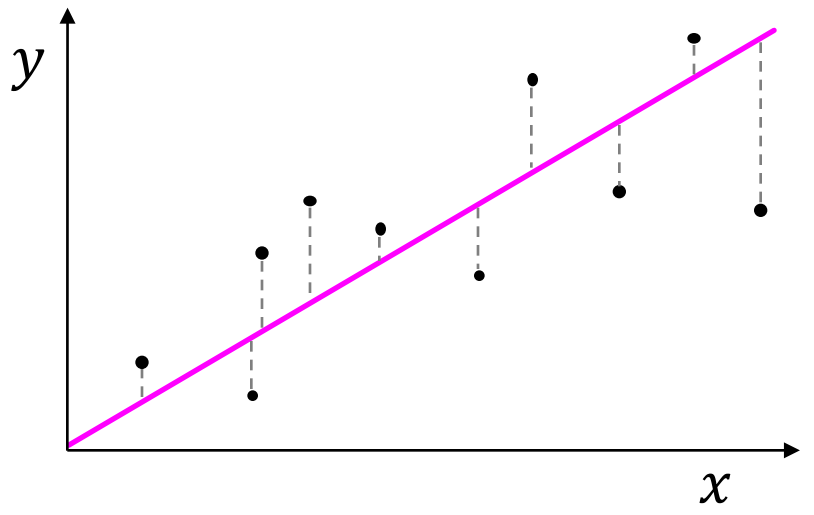
# Linear regression




Find intercept $b$ and slope $w$ so line has ***smallest*** squared error with data points

$$\sum_{p=1}^{P} \left(b + x_p w - y_p\right)^2$$

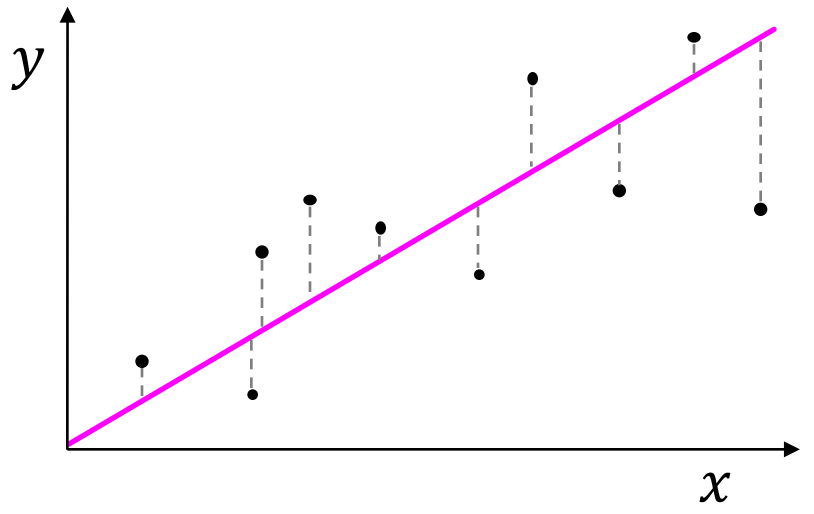
# Linear regression




Find intercept $b$ and slope $w$ so line has ***smallest*** squared error with data points

$$g(b, w) = \sum_{p=1}^{P} \left(b + x_p w - y_p\right)^2$$

# Linear regression




Find intercept $b$ and slope $w$ so line has ***smallest*** squared error with data points

$$g(b,w) = \sum_{p=1}^{P} \left(b + x_p w - y_p\right)^2$$

In other words–want to find $b$ and slope $w$ so *cost function* $g(b,w)$ is minimized

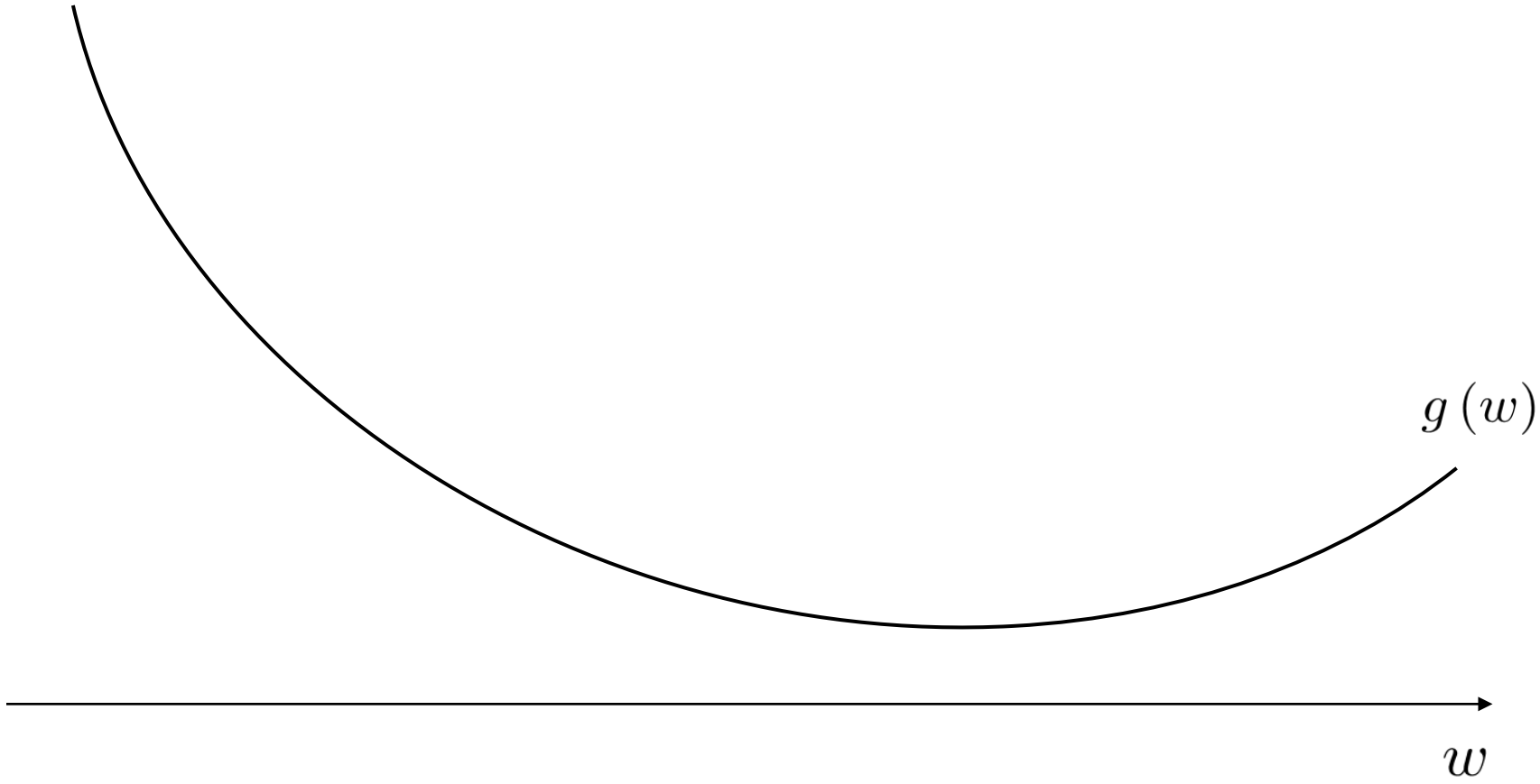
# Demo 1: linear regression cost function

- to the notebook!

# the search for ideal parameters

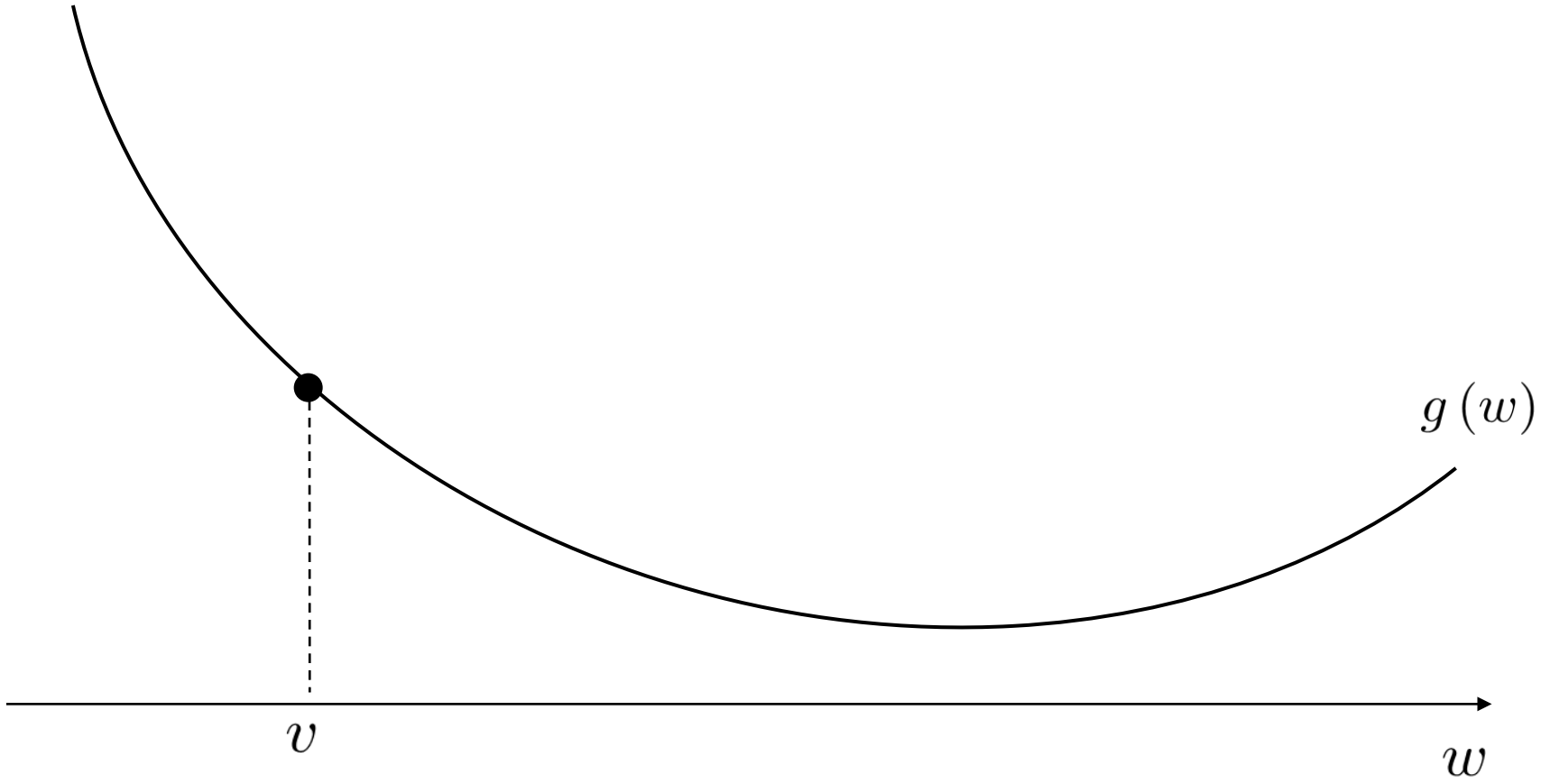
# the derivative

- differentiable function $g\,(w)$ with single input *w*
- derivative $g'\,(w)$ gives the slope of tangent line at *w*



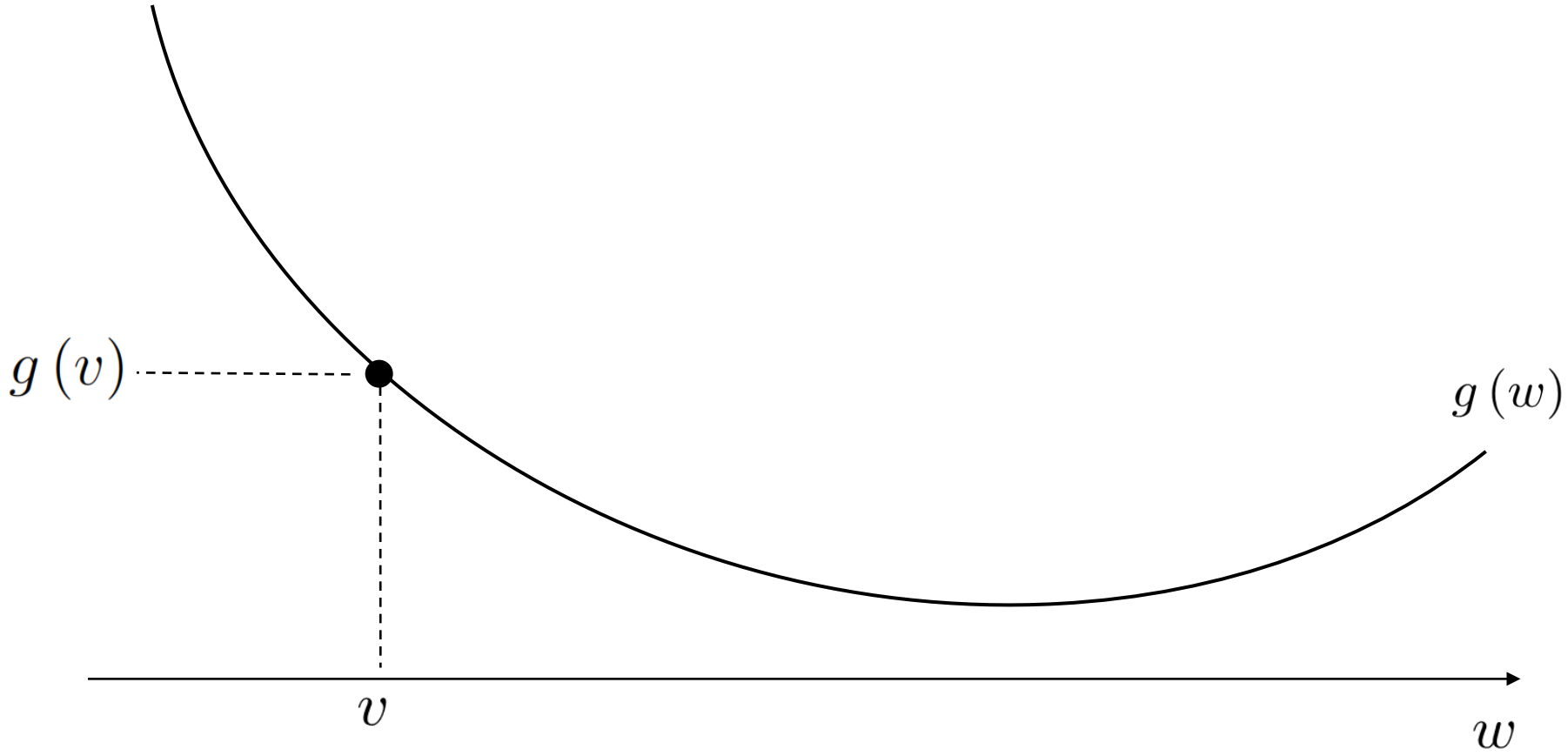
# the derivative

- differentiable function $g\,(w)$ with single input *w*
- derivative $g'\,(w)$ gives the slope of tangent line at *w*



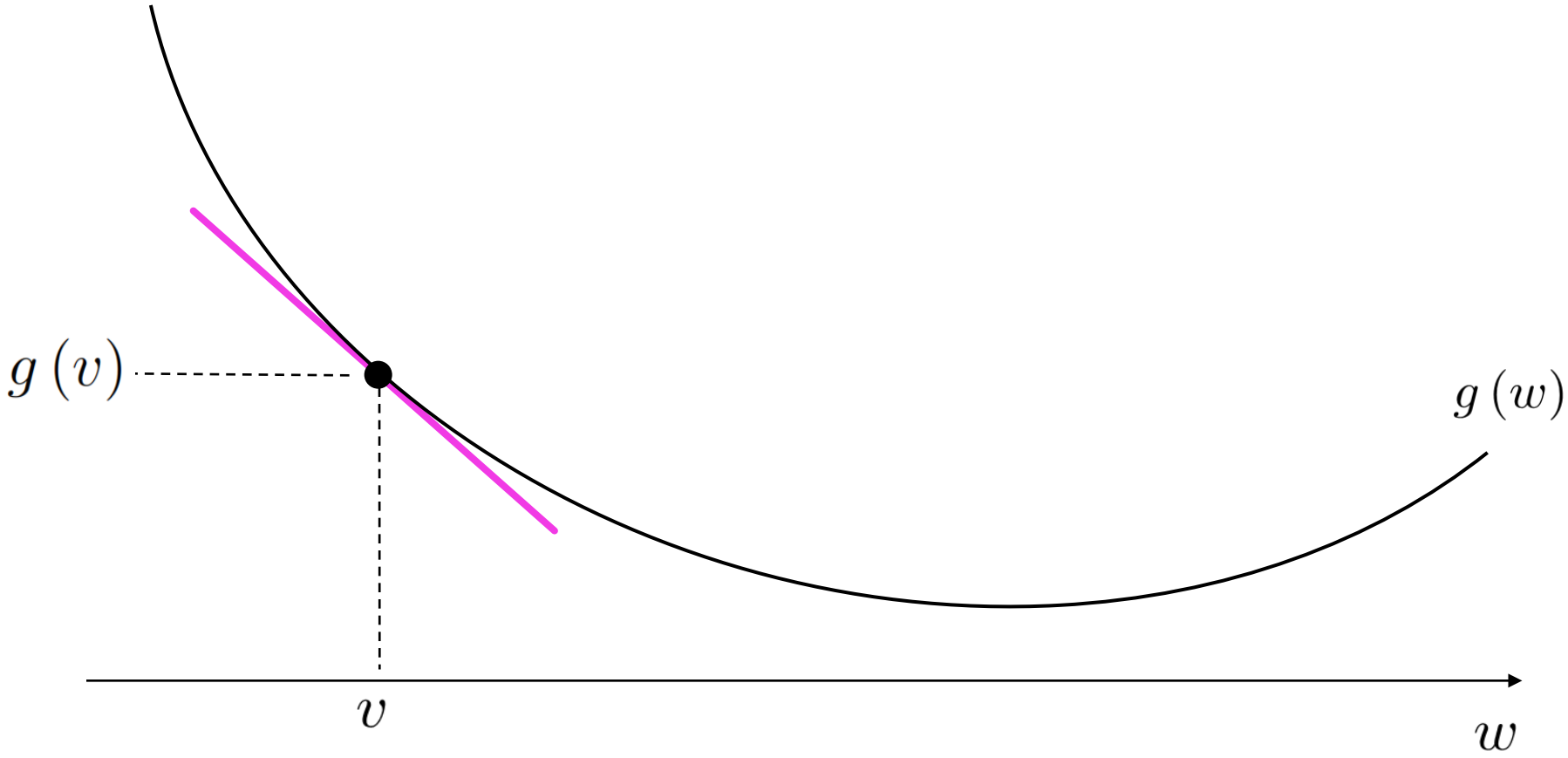
# the derivative

- differentiable function $g(w)$ with single input $w$
- derivative $g'(w)$ gives the slope of tangent line at $w$



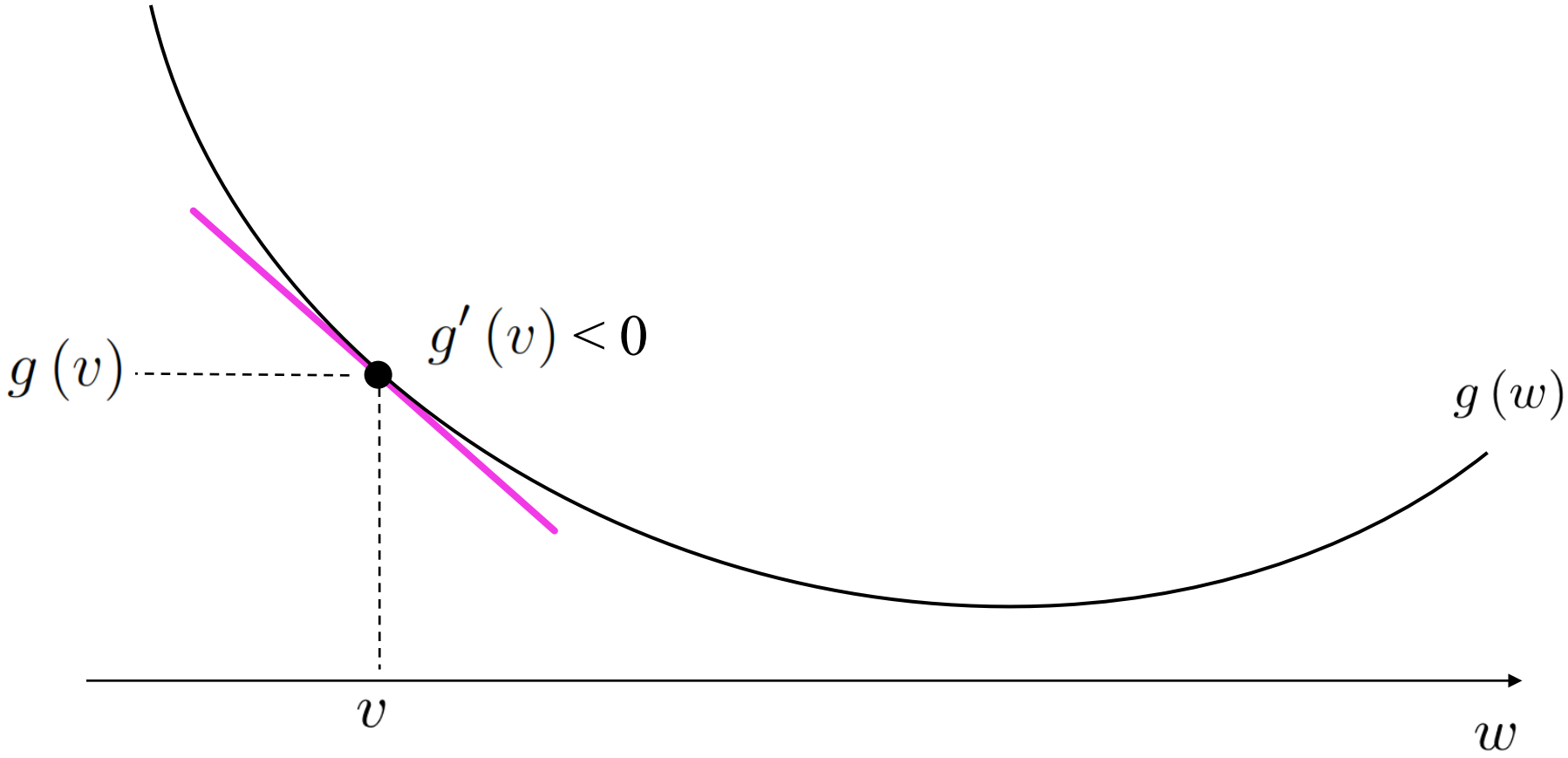
# the derivative

- differentiable function $g(w)$ with single input $w$
- derivative $g'(w)$ gives the slope of tangent line at $w$



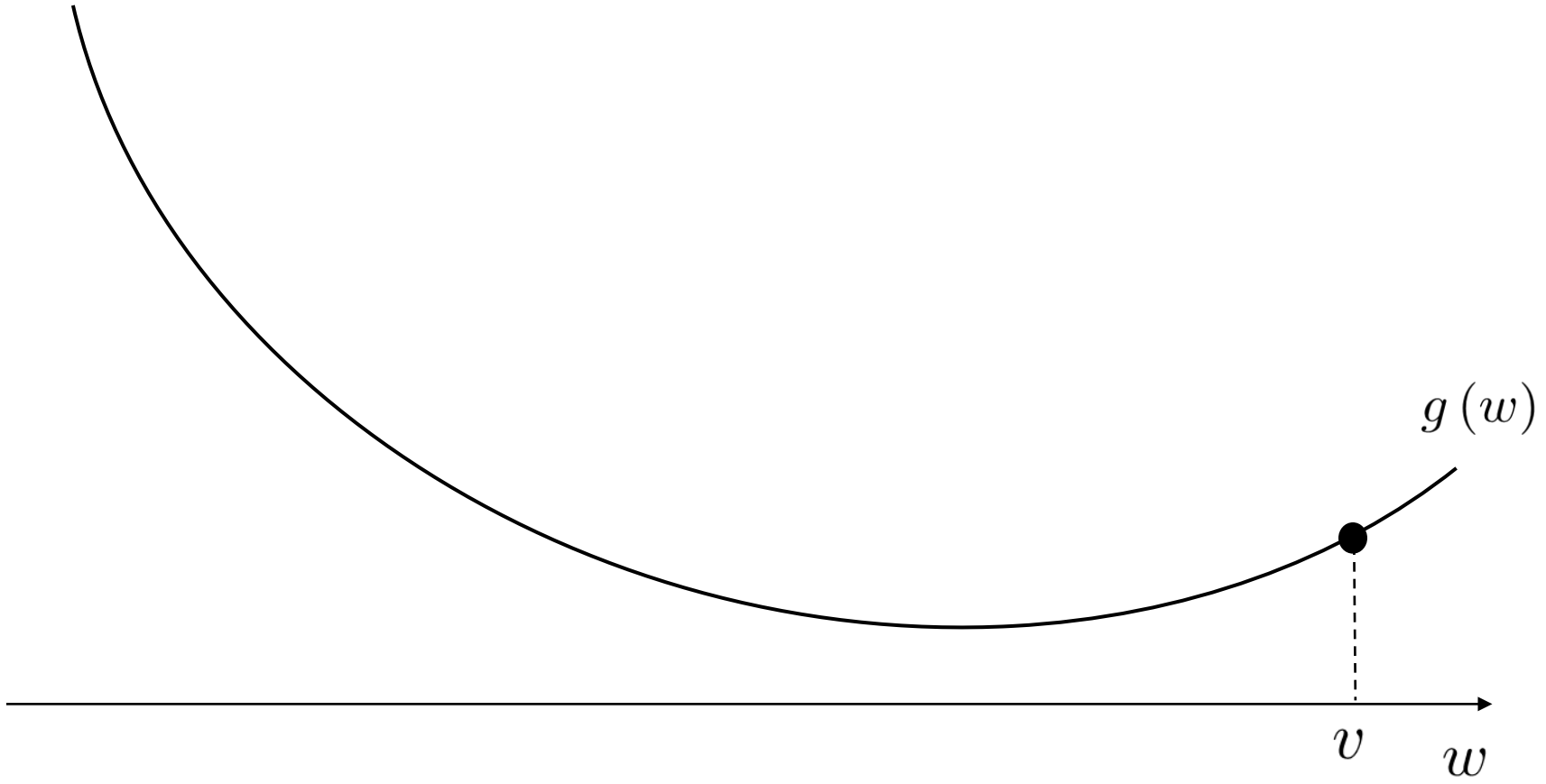
# the derivative

- differentiable function $g(w)$ with single input $w$
- derivative $g'(w)$ gives the slope of tangent line at $w$



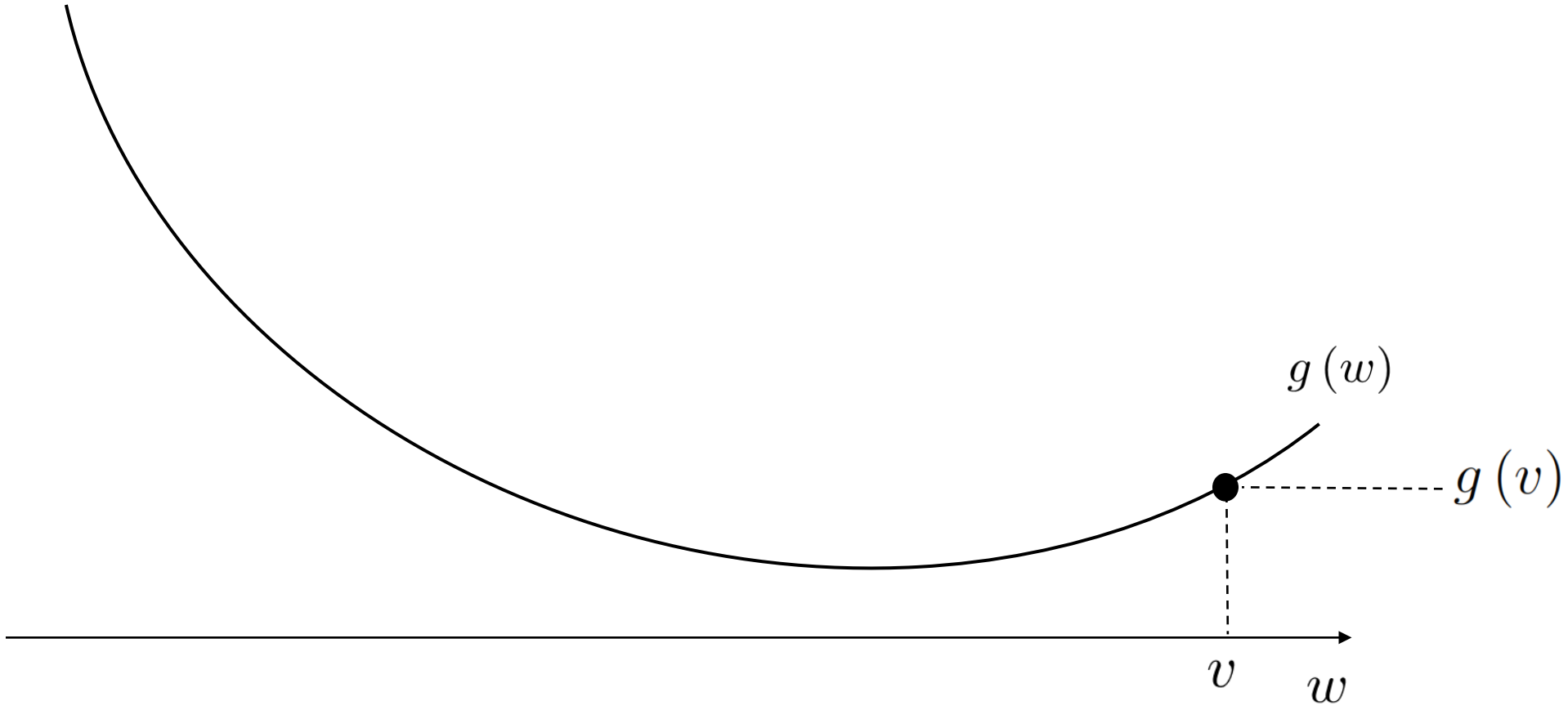
# the derivative

- differentiable function $g\,(w)$ with single input *w*
- derivative $g'\,(w)$ gives the slope of tangent line at *w*



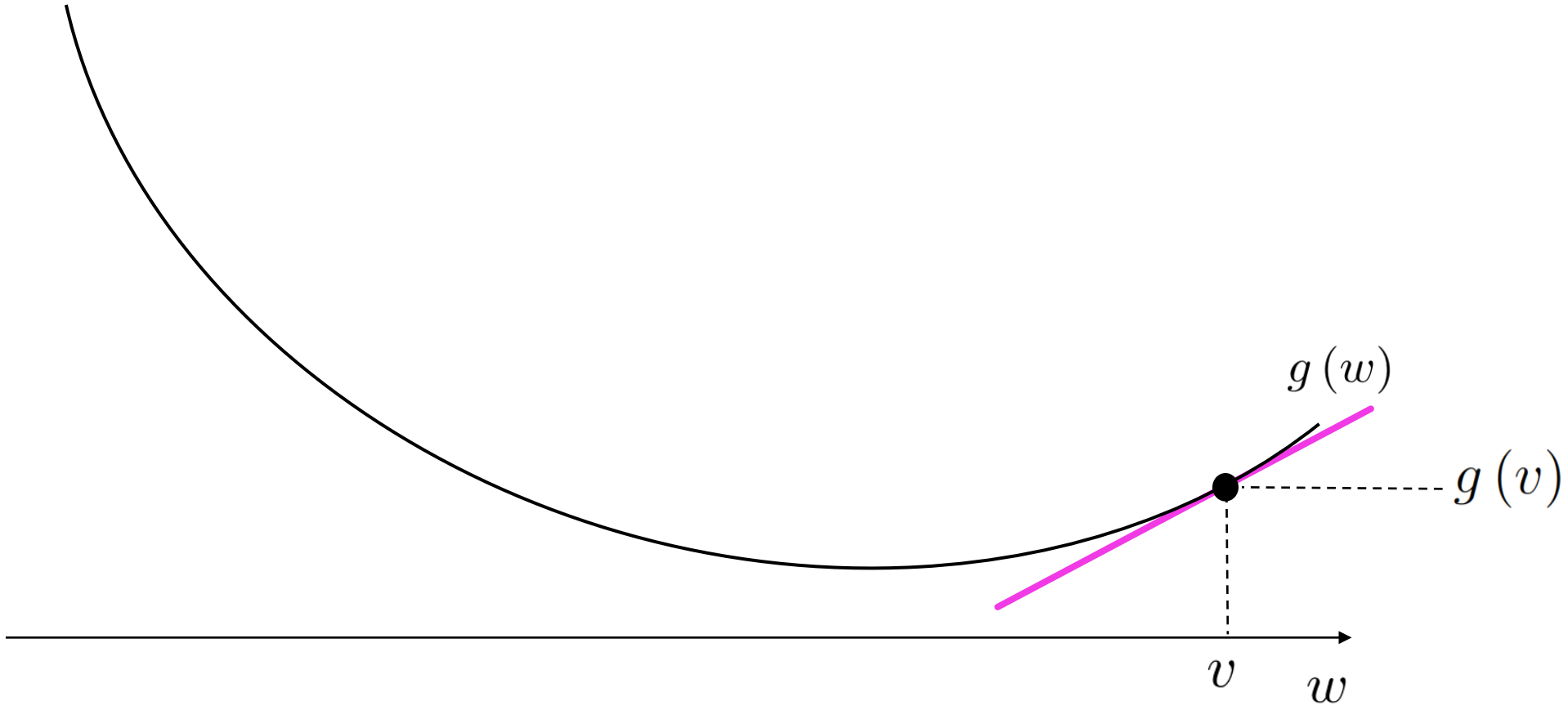
# the derivative

- differentiable function $g\,(w)$ with single input $w$
- derivative $g'\,(w)$ gives the slope of tangent line at $w$



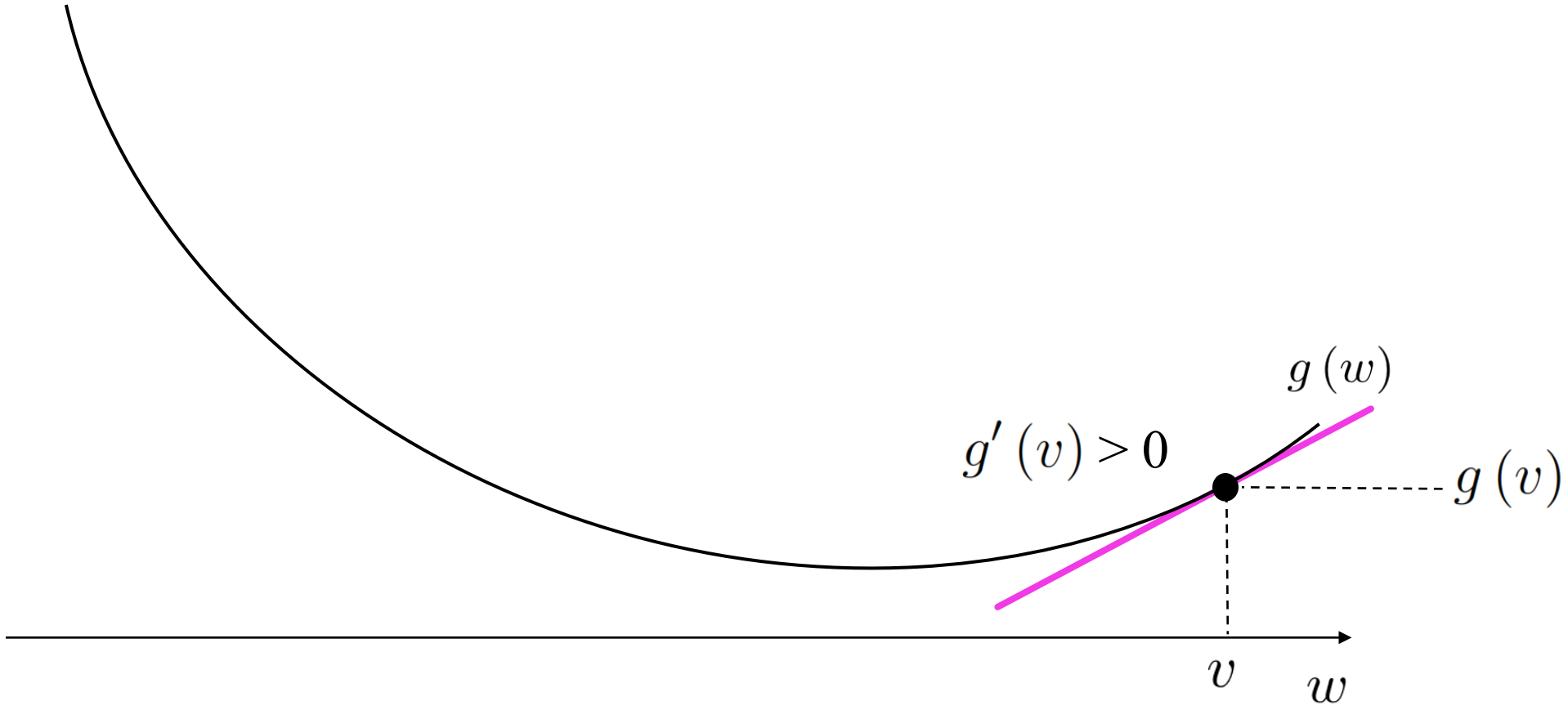
# the derivative

- differentiable function $g\,(w)$ with single input $w$
- derivative $g'\,(w)$ gives the slope of tangent line at $w$



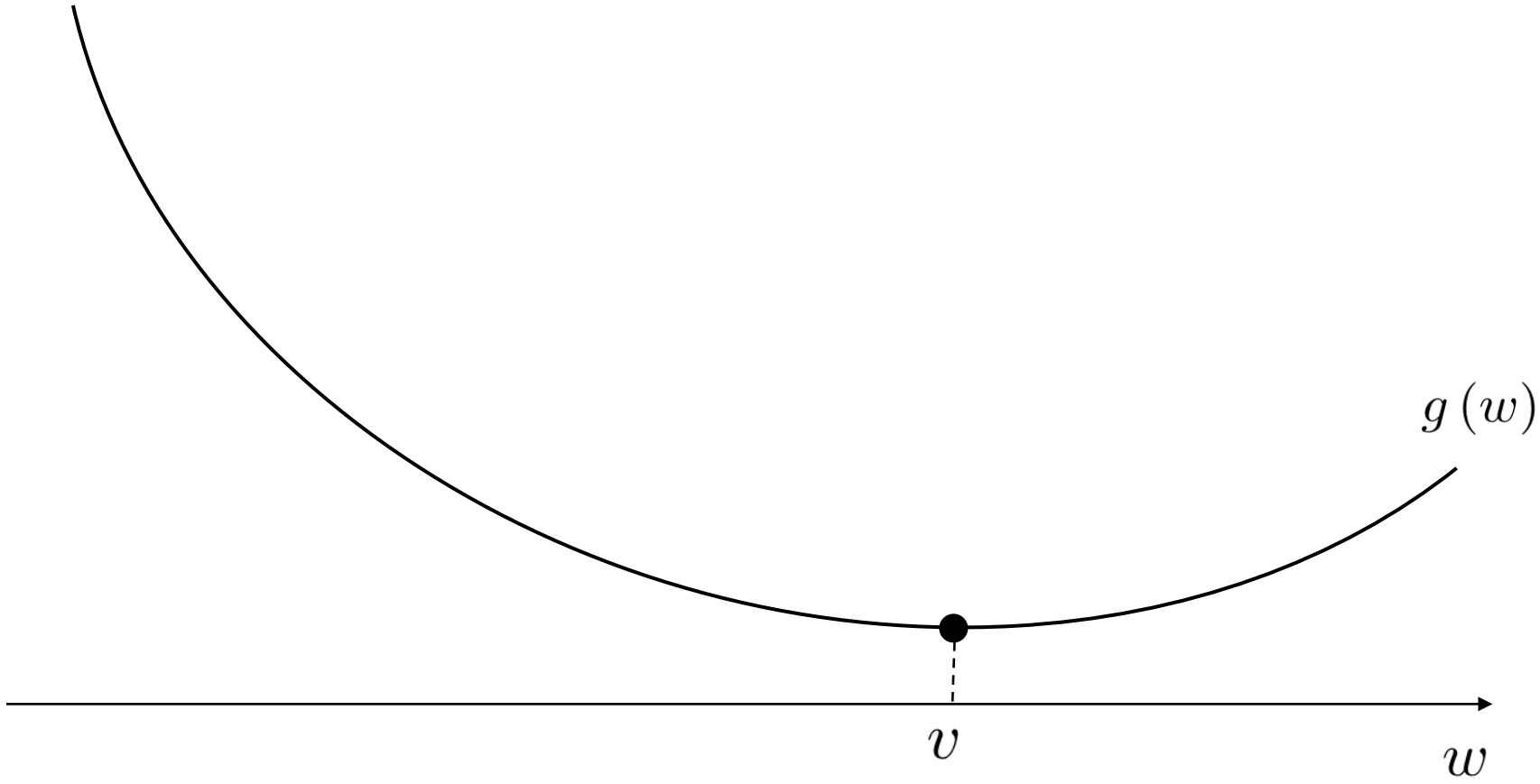
# the derivative

- differentiable function $g\,(w)$ with single input $w$
- derivative $g'\,(w)$ gives the slope of tangent line at $w$



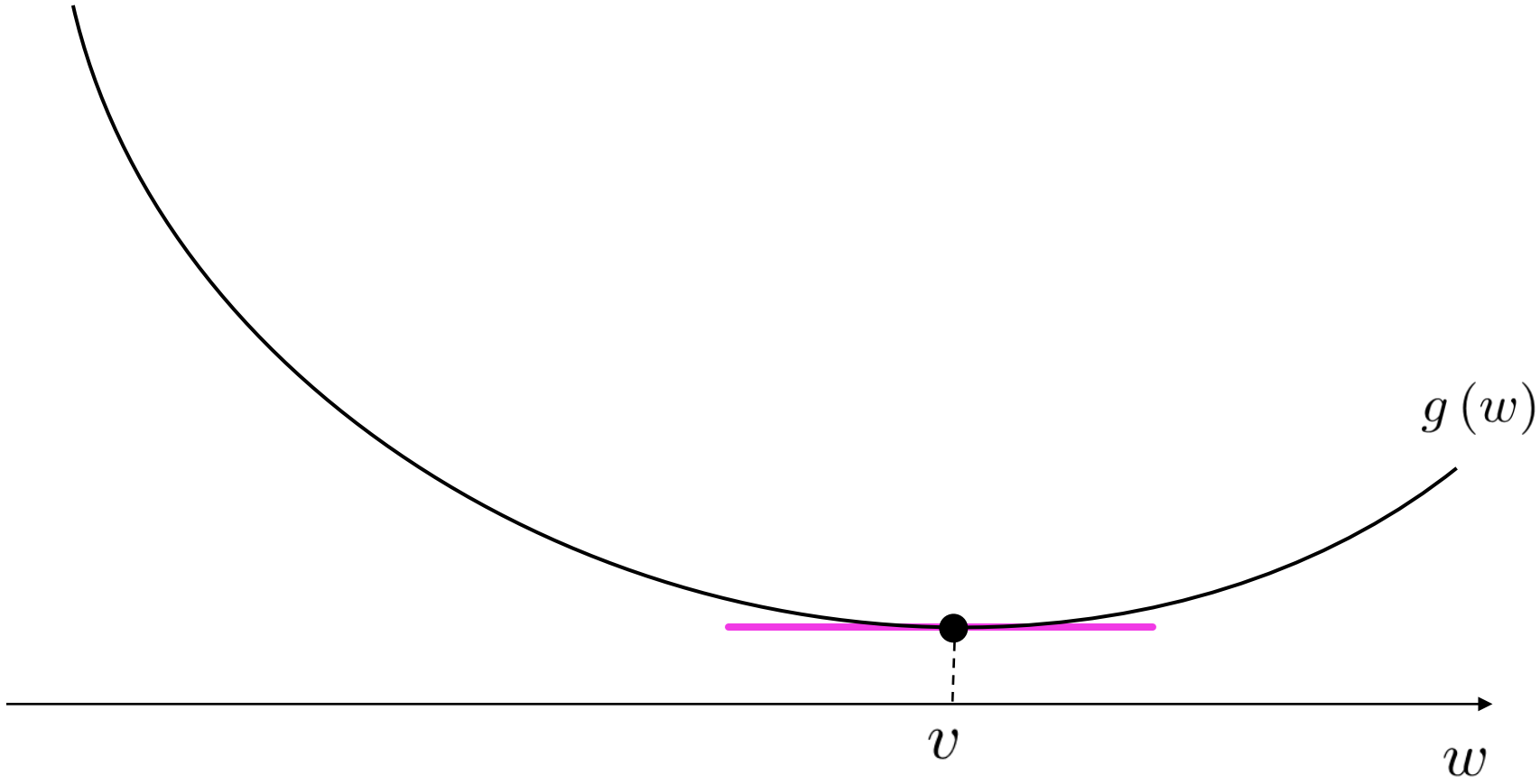
# the derivative

- differentiable function $g\left(w\right)$ with single input *w*
- derivative $g'\left(w\right)$ gives the slope of tangent line at *w*



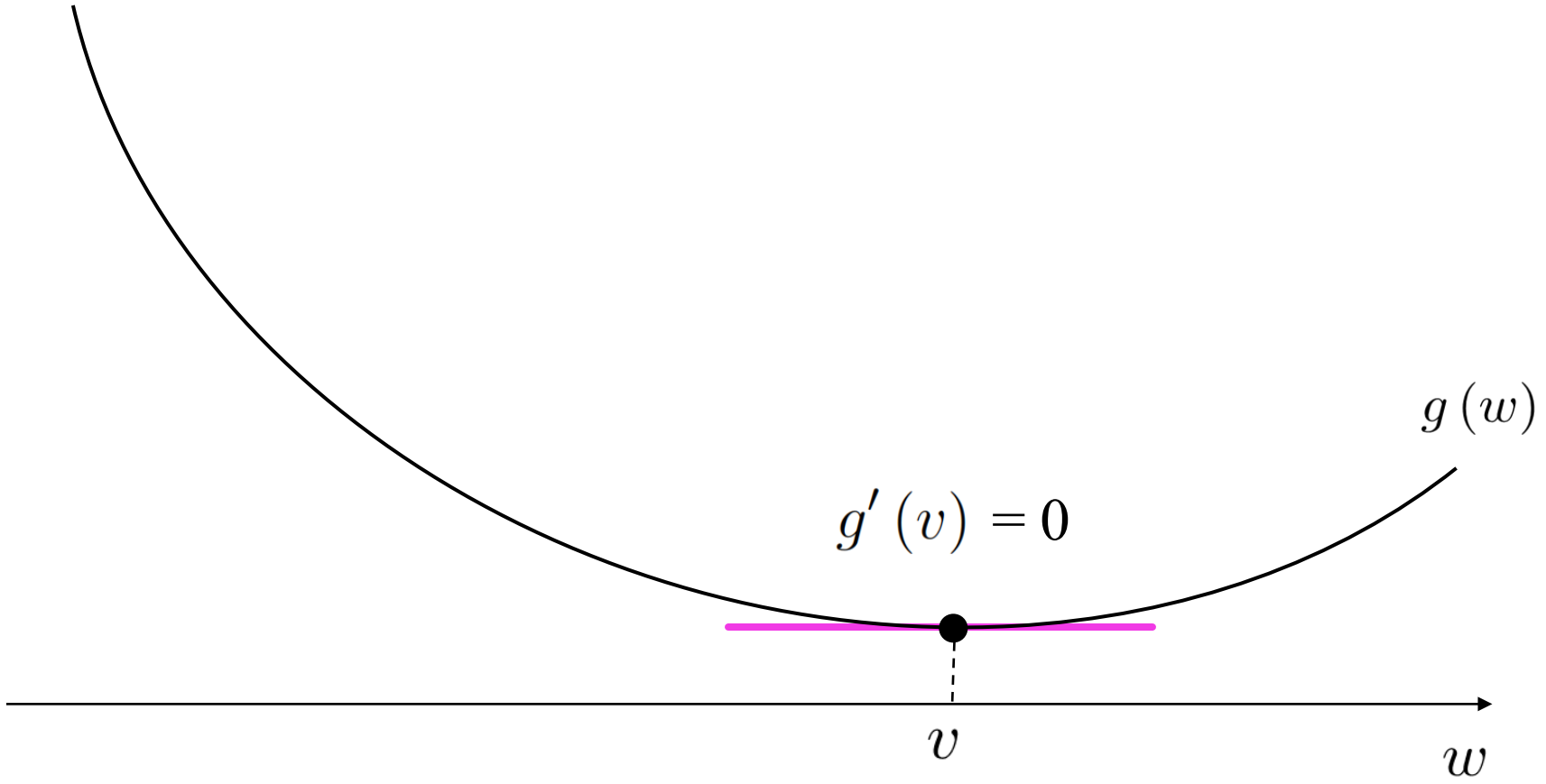
# the derivative

- differentiable function $g\left(w\right)$ with single input $w$
- derivative $g'\left(w\right)$ gives the slope of tangent line at $w$



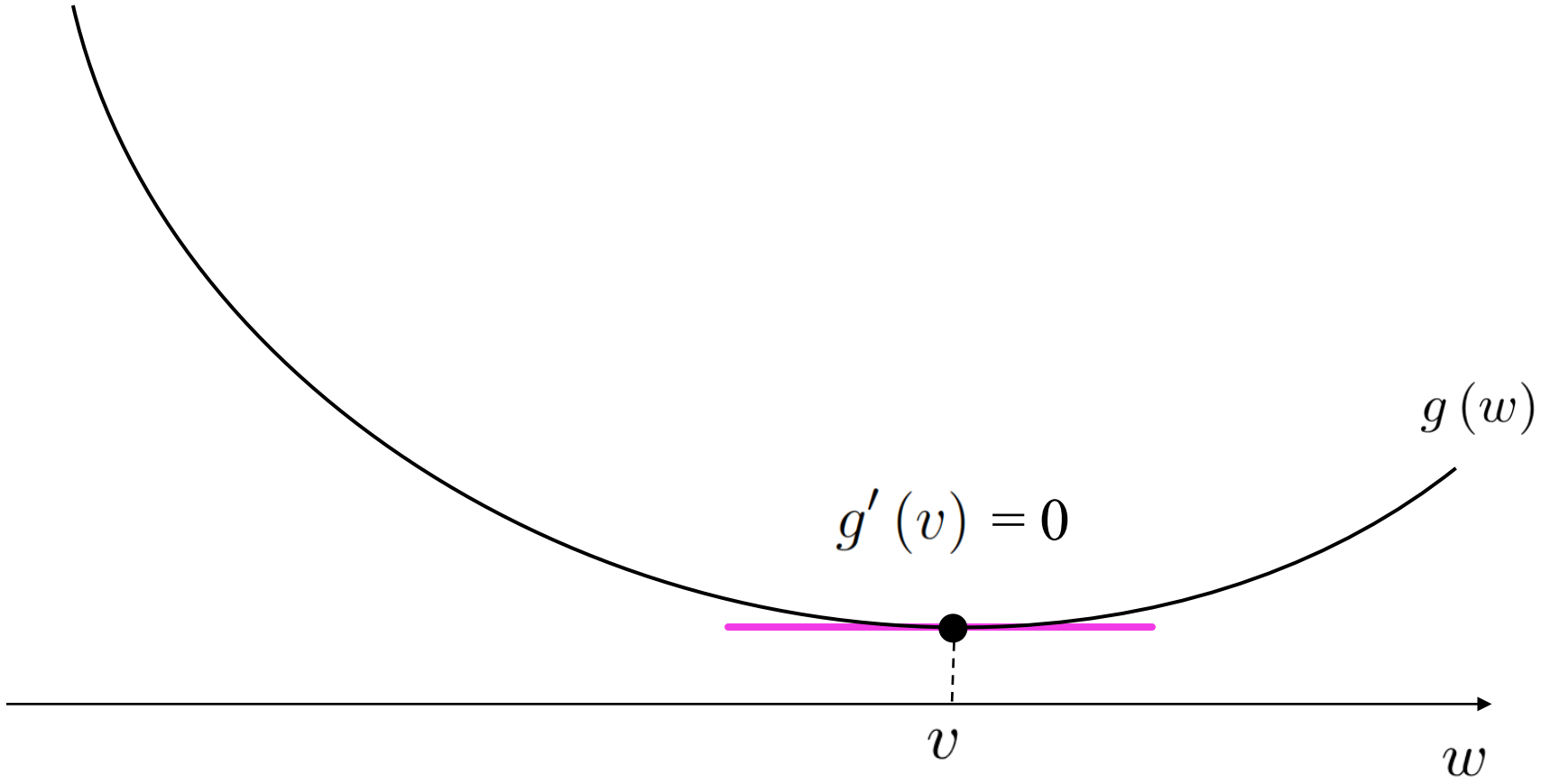
# the derivative

- differentiable function $g\,(w)$ with single input $w$
- derivative $g'\,(w)$ gives the slope of tangent line at $w$



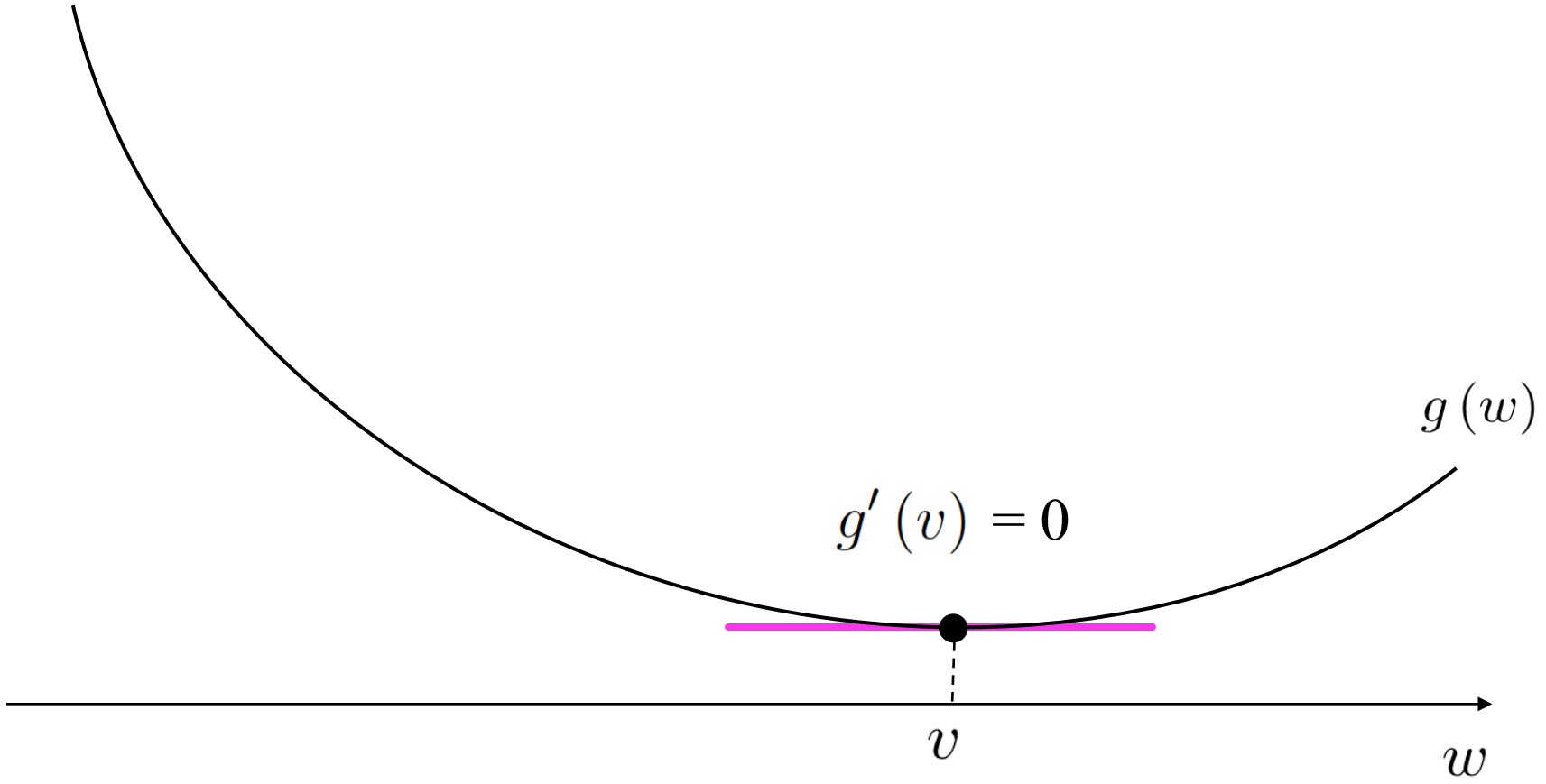
# the derivative

- differentiable function $g\left(w\right)$ with single input $w$
- derivative $g'\left(w\right)$ gives the slope of tangent line at $w$
- **fact**: minimum points *always* have zero derivative



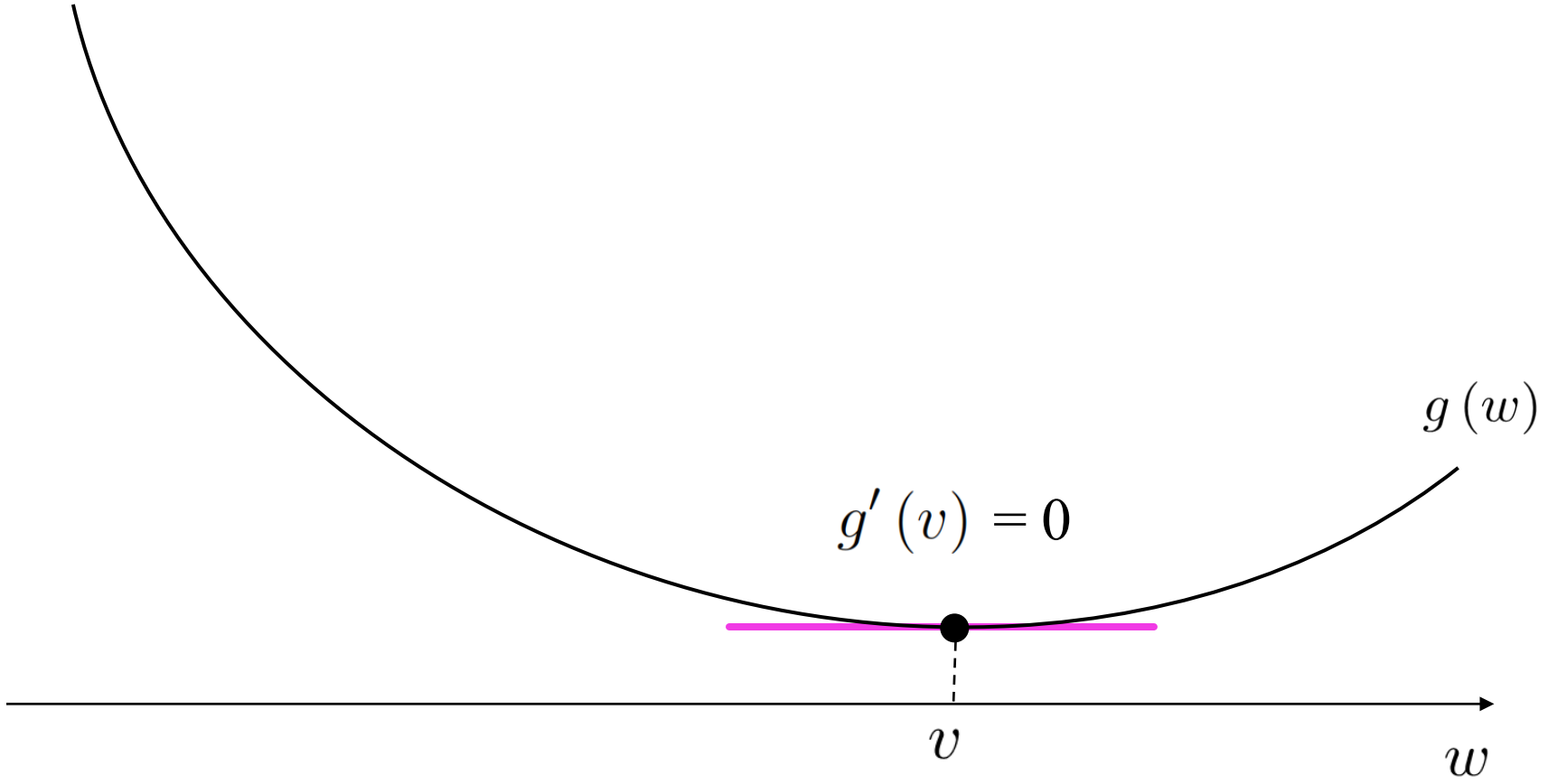
# the derivative

- differentiable function $g\left(w\right)$ with single input $w$
- derivative $g'\left(w\right)$ gives the slope of tangent line at $w$
- **fact**: minimum points *always* have zero derivative



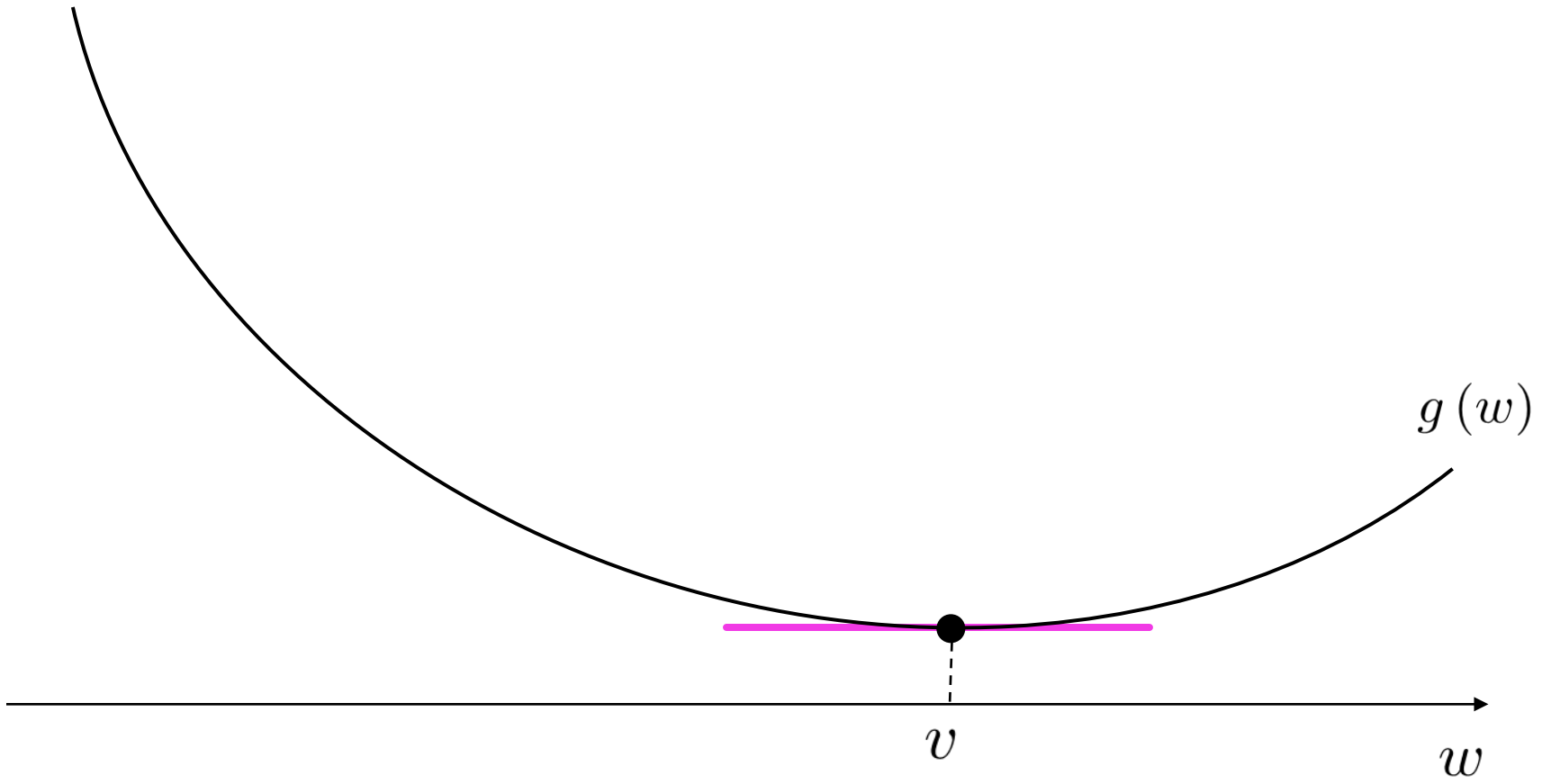
# the derivative

- differentiable function $g\left(w_1, w_2, \ldots, w_N\right)$ with multiple inputs input $w_1, w_2, \ldots, w_N$
- gradient gives the 'slope' of tangent plane at $w_1, w_2, \ldots, w_N$
- **fact**: minimum points *always* have zero gradient



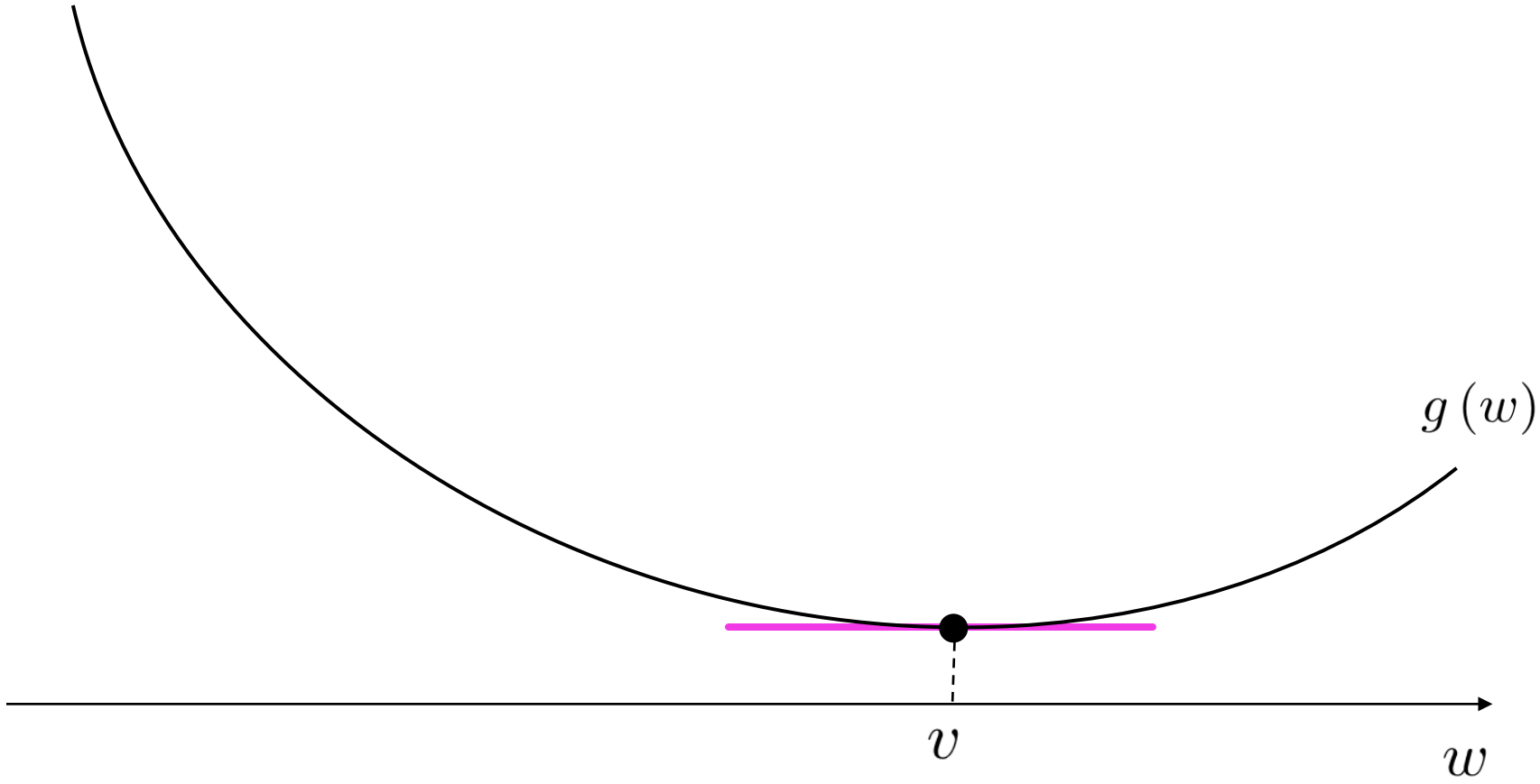
# the derivative

- find minimum of $g\,(w)$ by (algorithmically) determining $w$ with zero derivative



# the derivative

- find minimum of $g\,(w)$ by (algorithmically) determining $w$ with zero derivative
- simple examples can be done by hand, but machine learning costs require algorithm

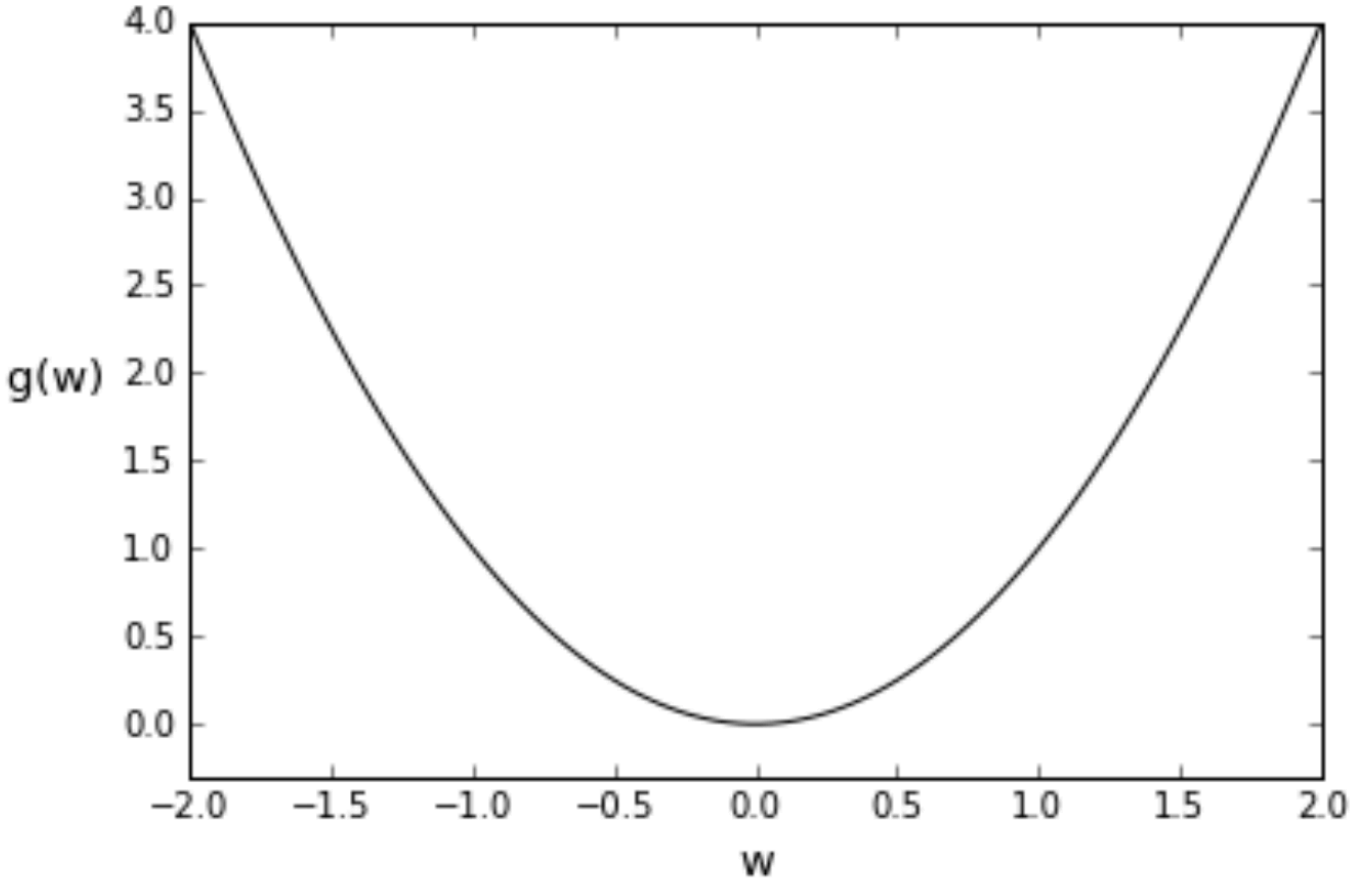


# example

$$g(w) = w^2$$

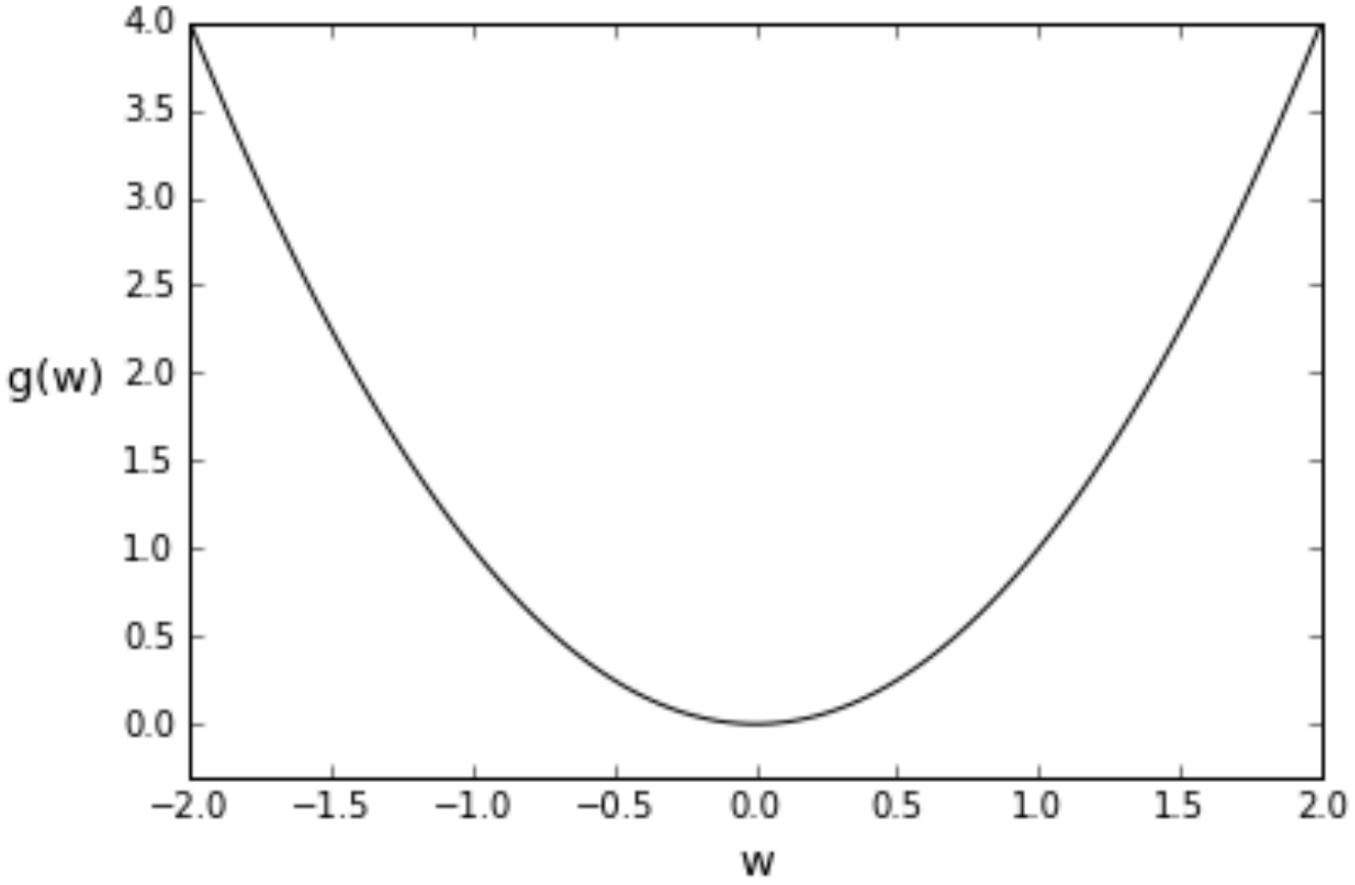
# example

$$g(w) = w^2$$



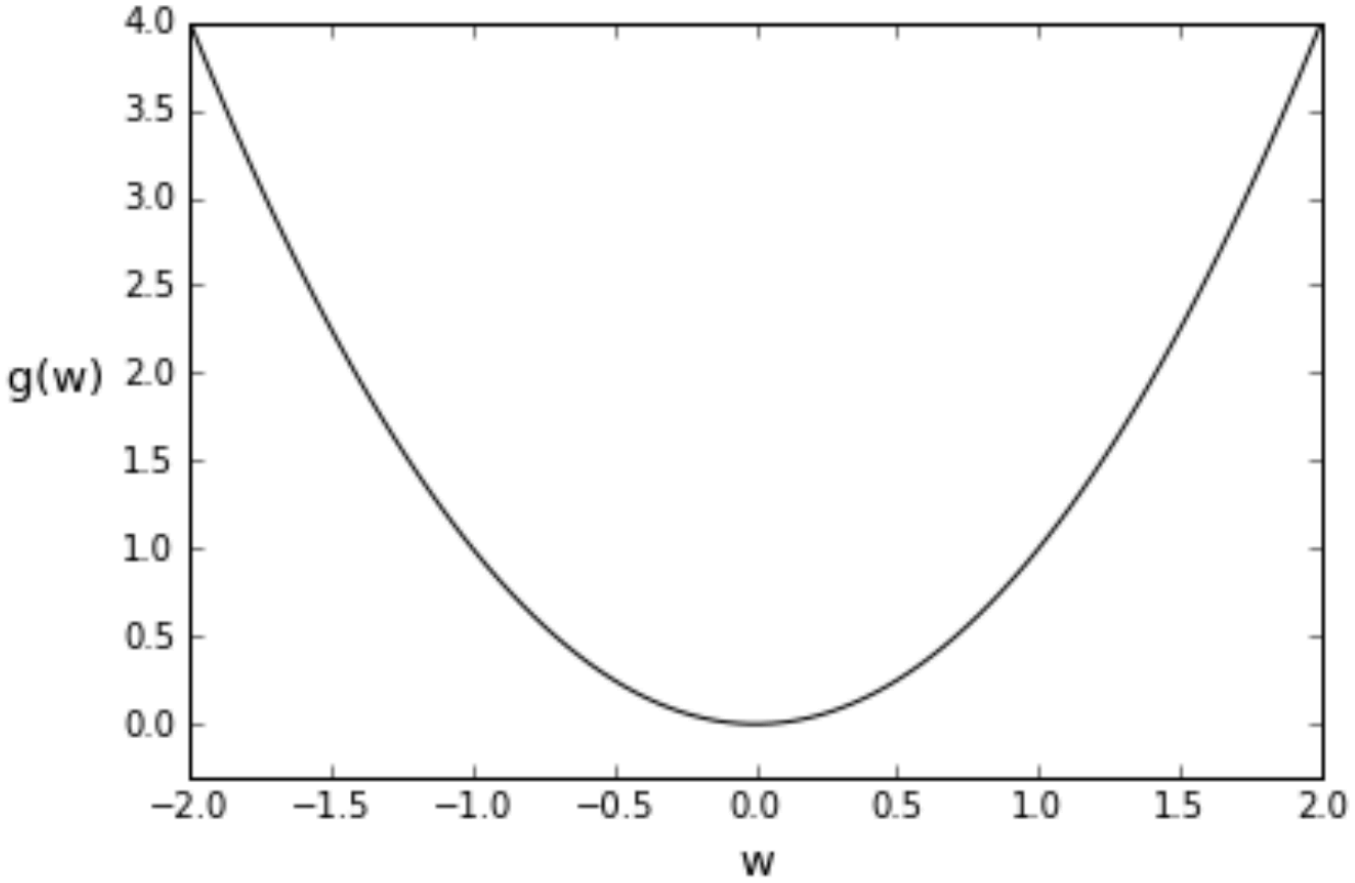
# example

$$g(w) = w^2 \longrightarrow g'(w) = 2w$$



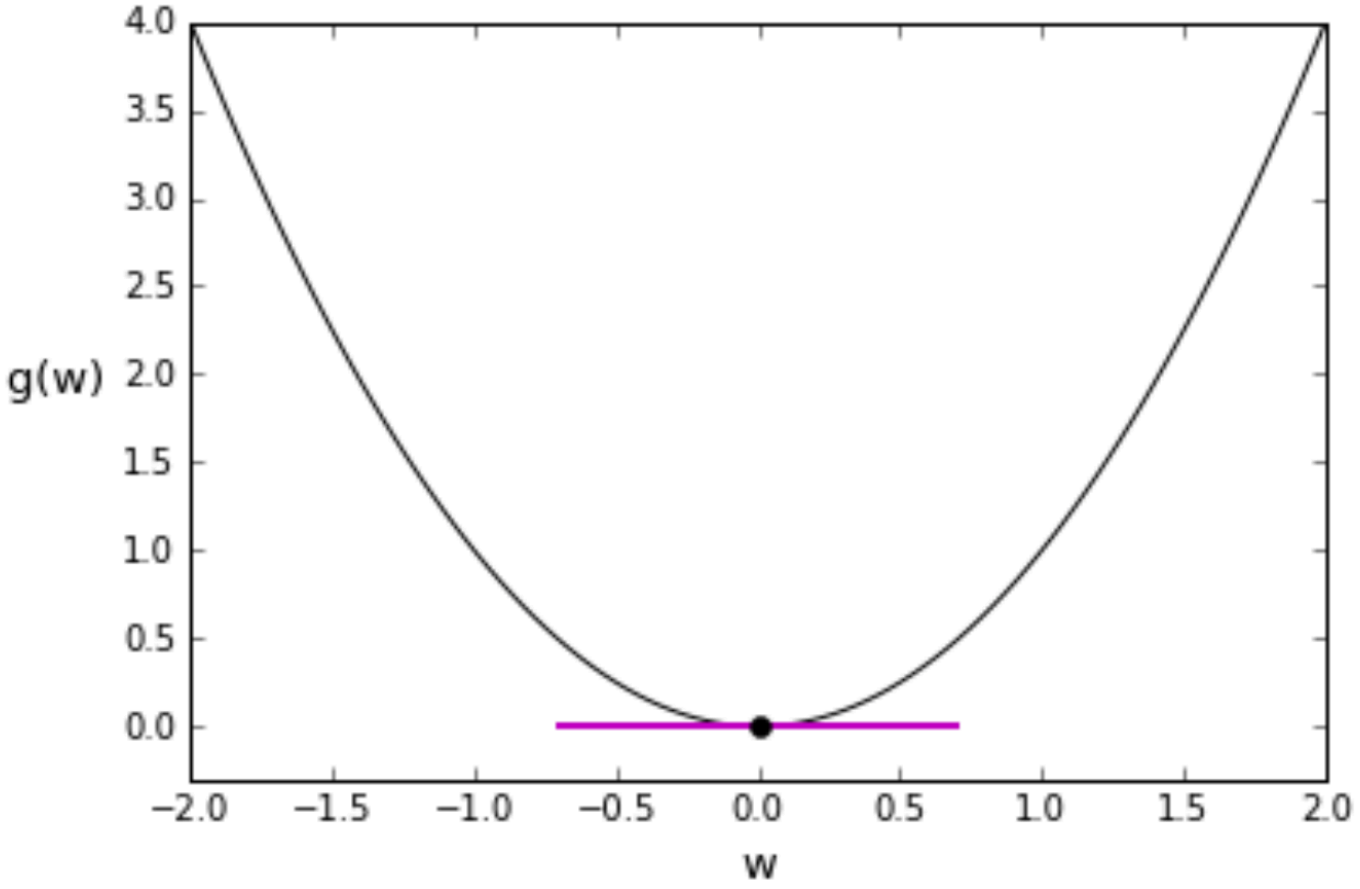
# example

$$g(w) = w^2 \longrightarrow g'(w) = 2w \longrightarrow g'(0) = 0$$



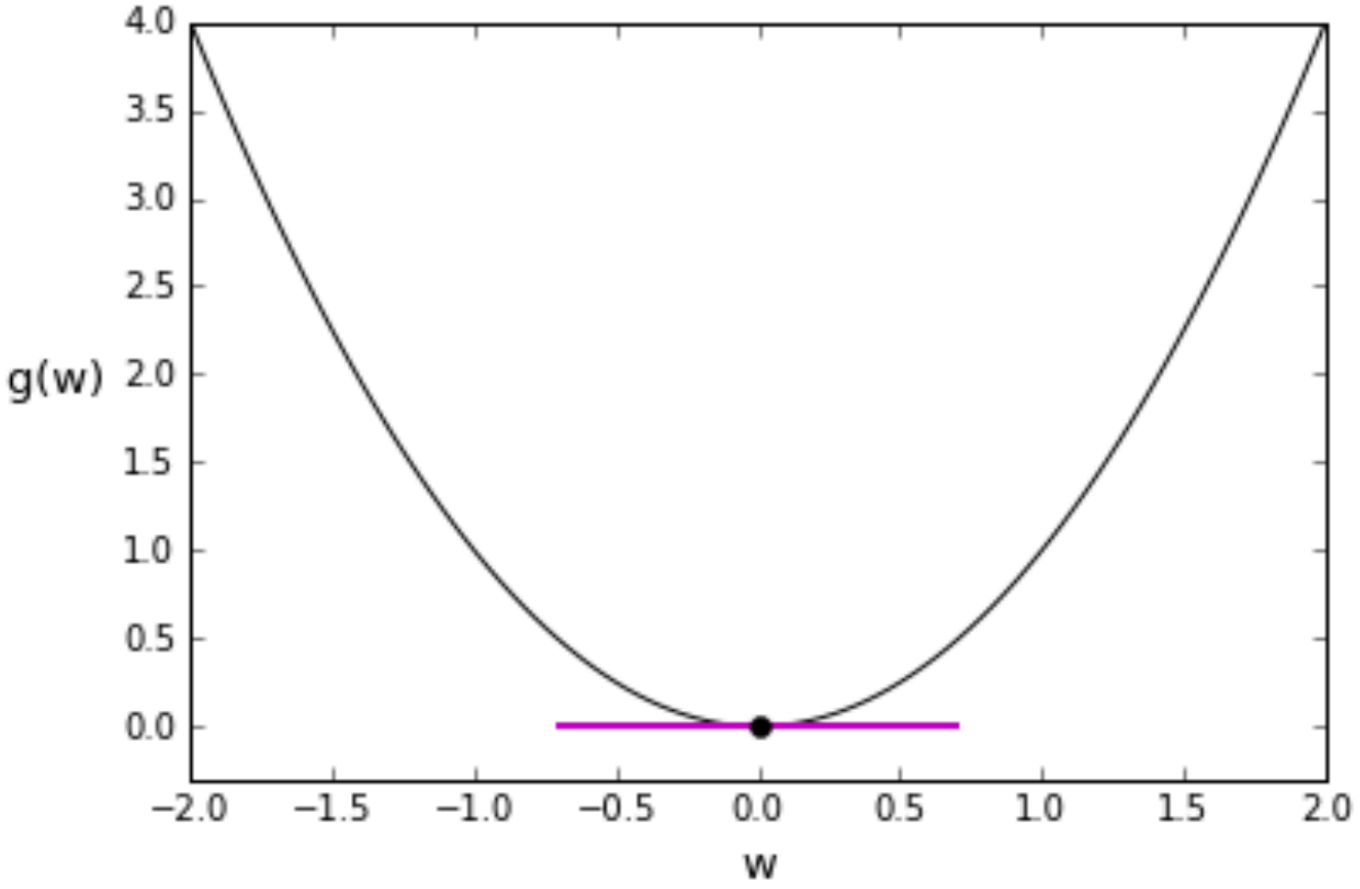
# example

$$g(w) = w^2 \longrightarrow g'(w) = 2w \longrightarrow g'(0) = 0$$



# example

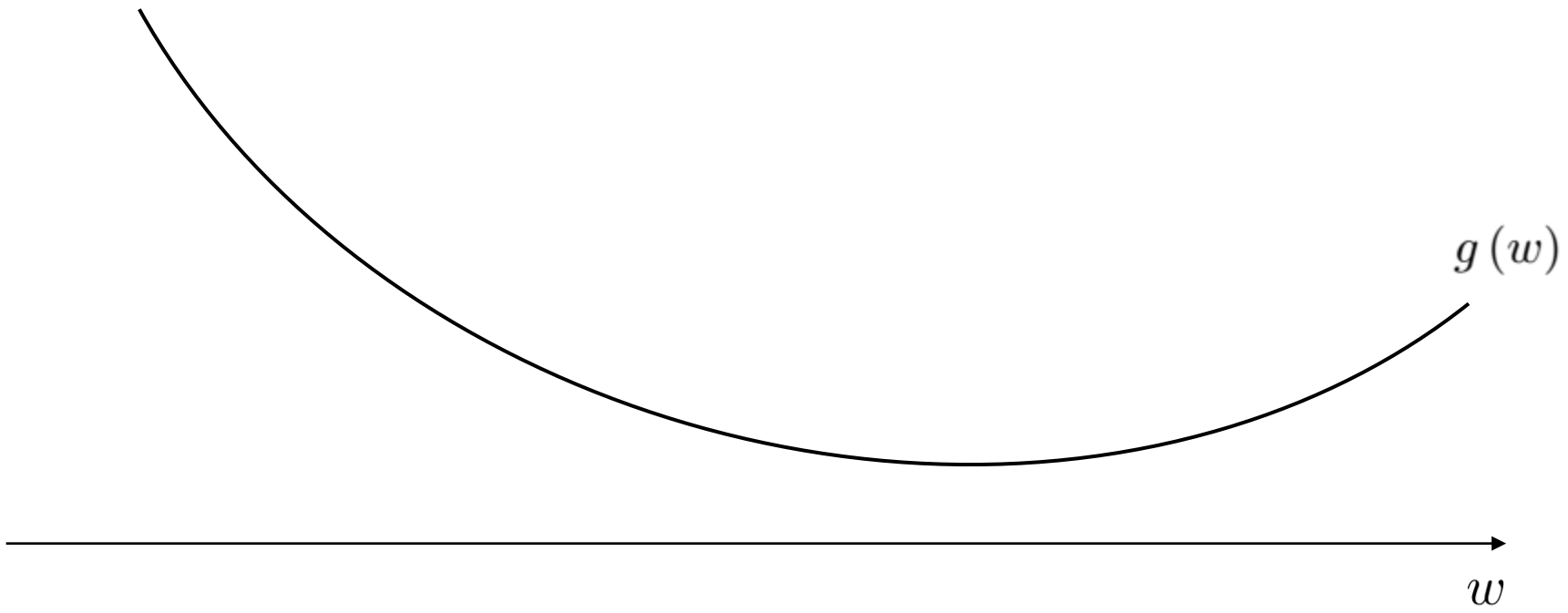
$$g(w) = w^2 \longrightarrow g'(w) = 2w \longrightarrow g'(0) = 0$$



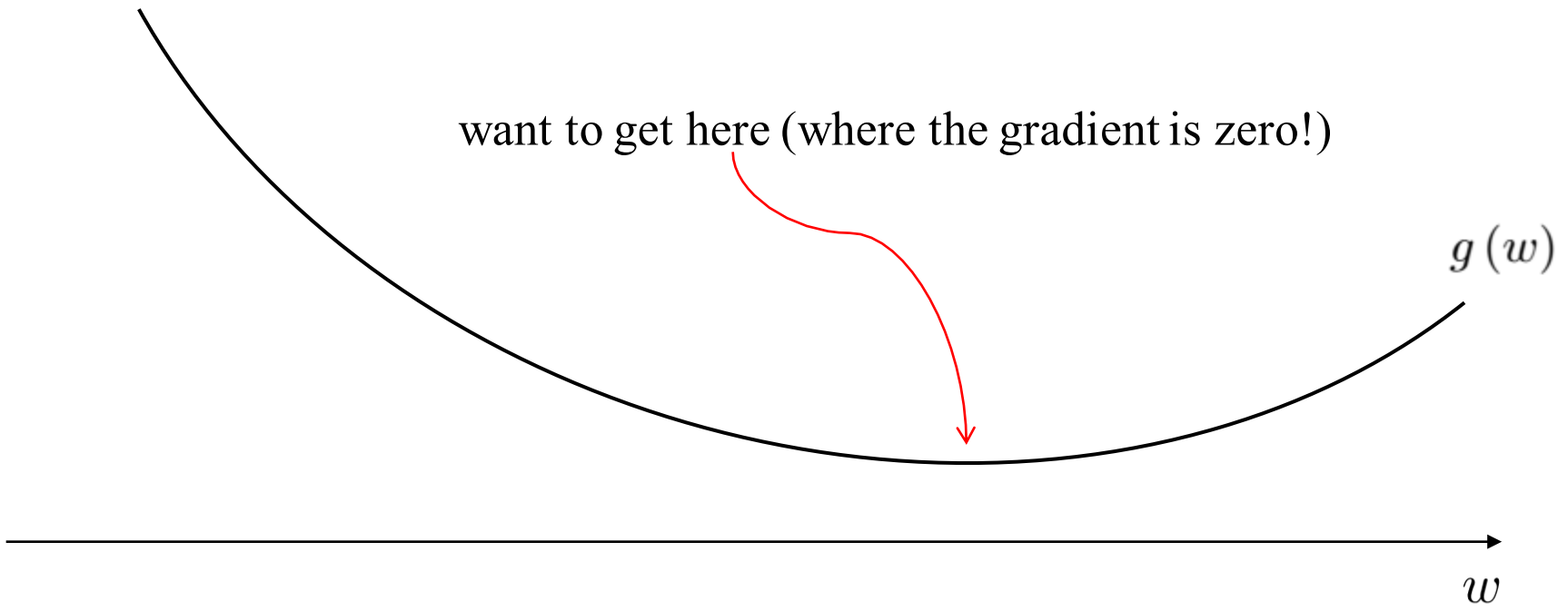
- OK but how about $g(w) = w^4 + w^2 + w$
- Looks simple enough...
- (go to notebook)

# optimization algorithms 101

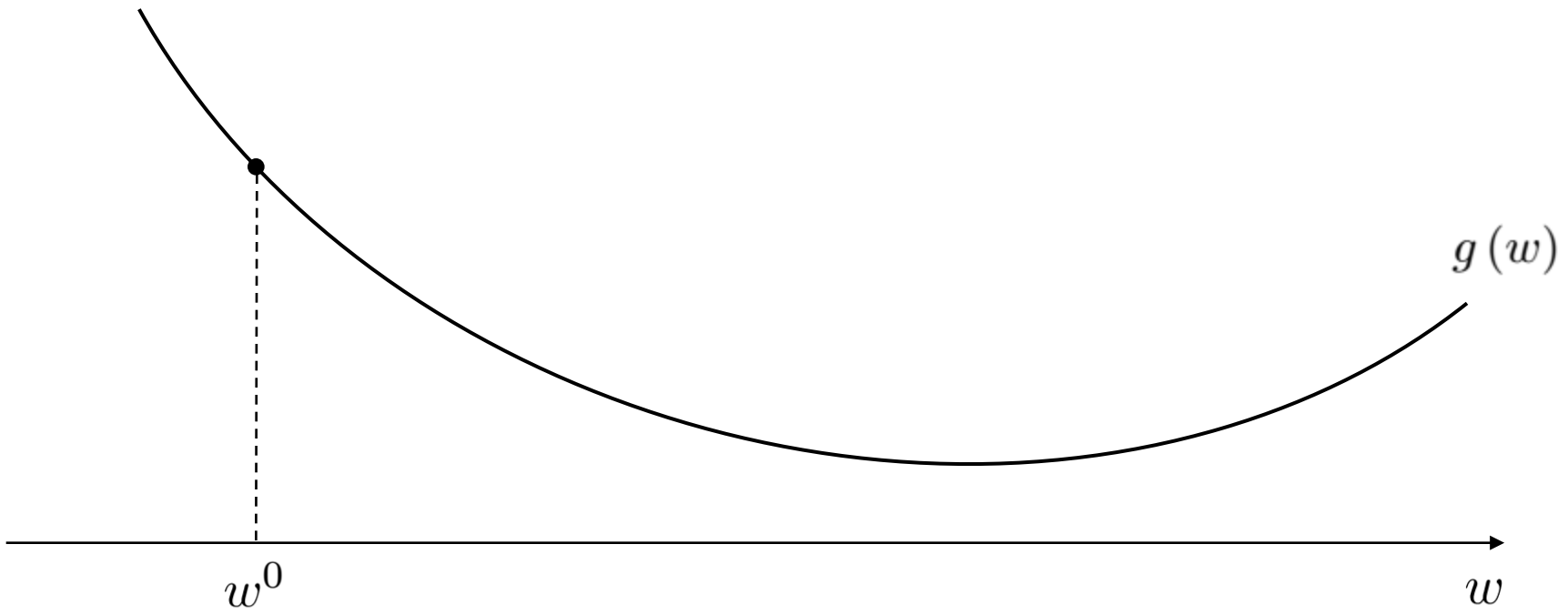
# Iterative methods



# Iterative methods

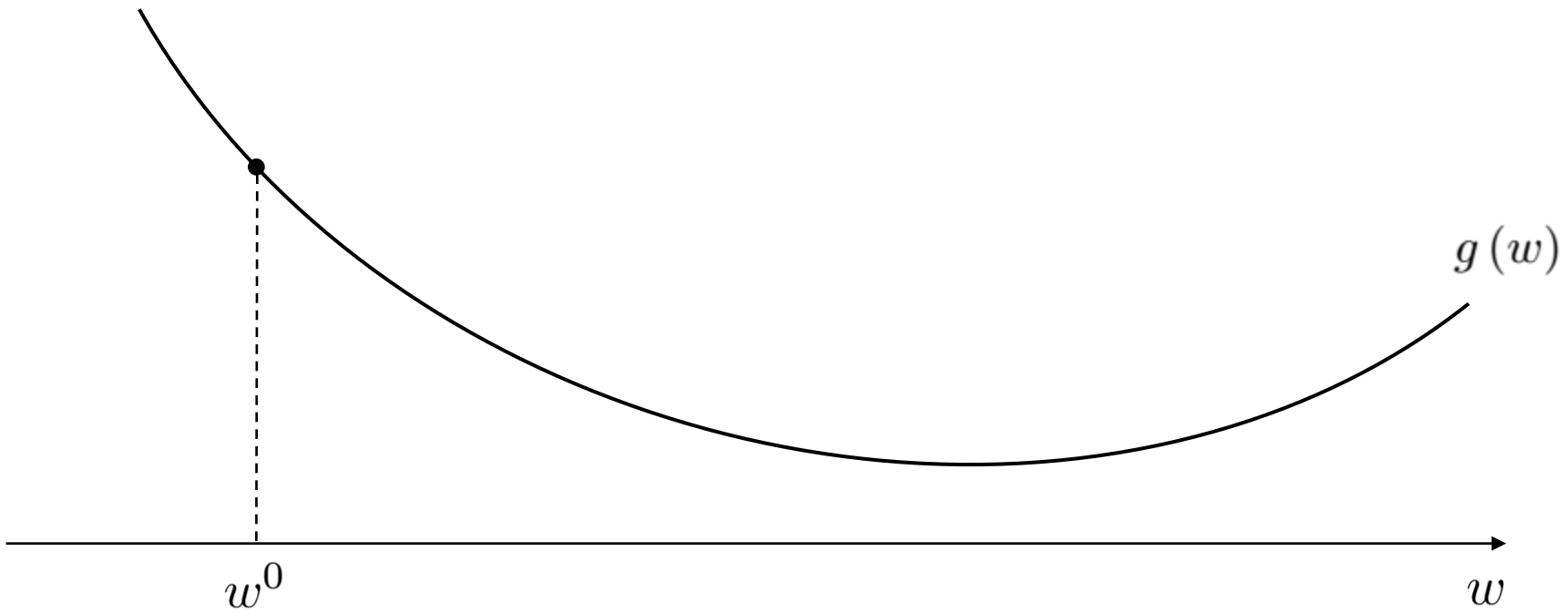


# Iterative methods



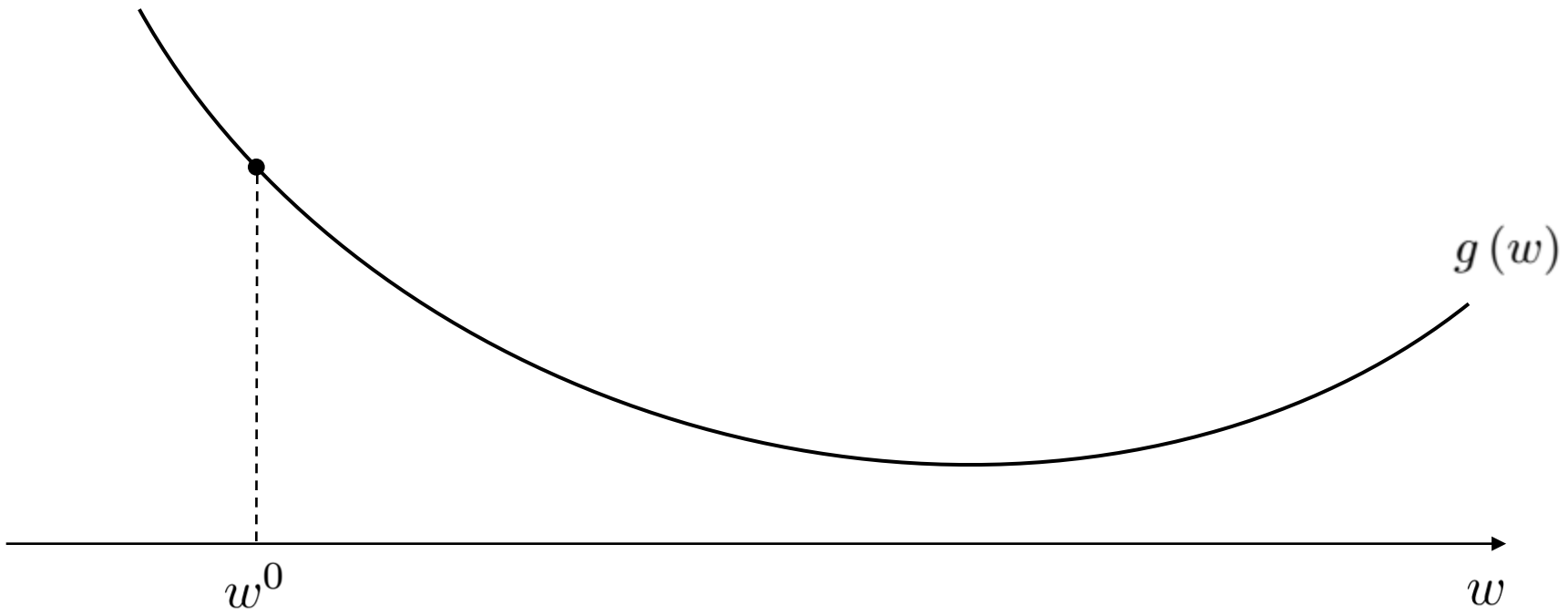
# Iterative methods

1. initialize at a point $w^0$



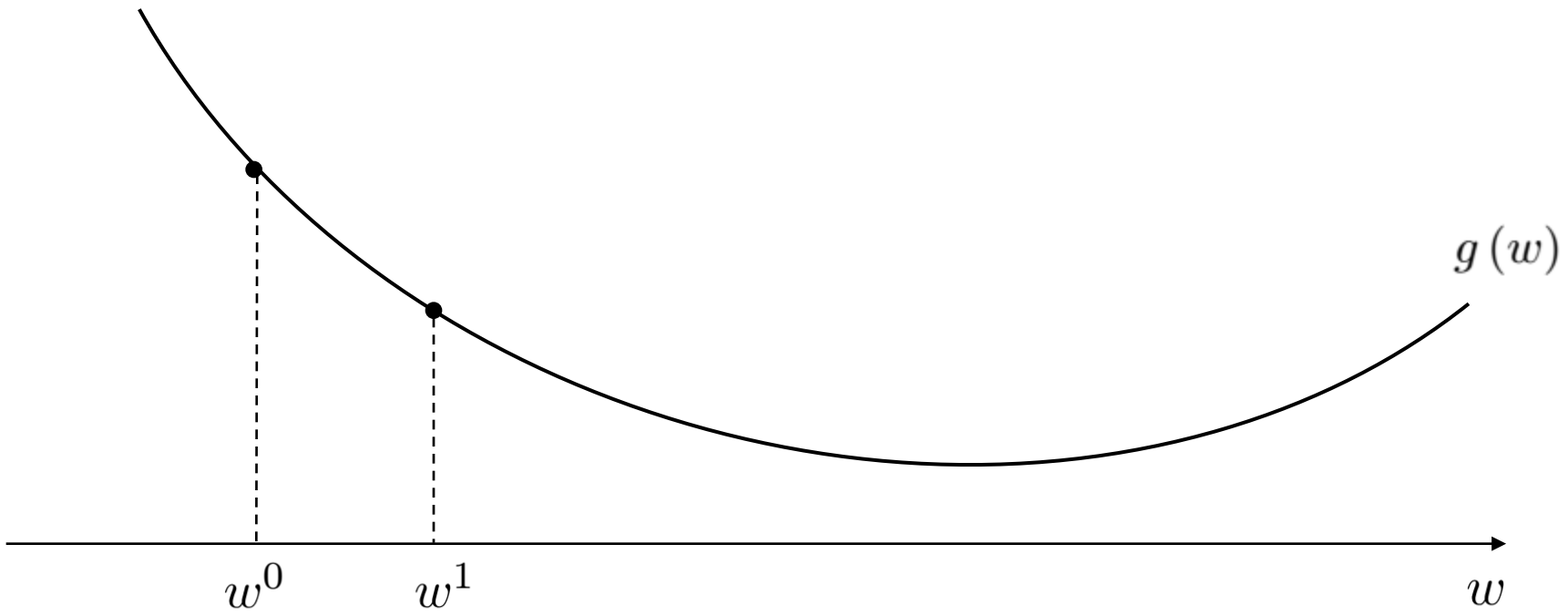
# Iterative methods

1. initialize at a point $w^0$
2. travel in **downward direction** to new point $w^1$



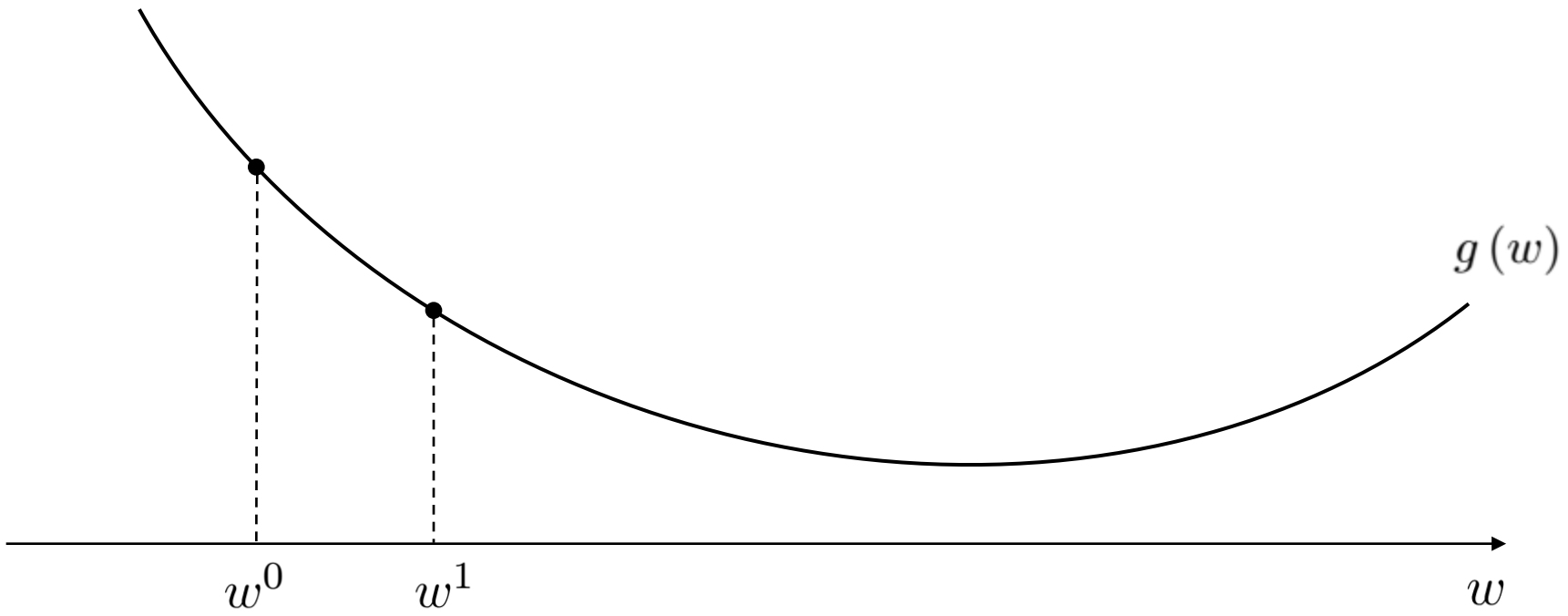
# Iterative methods

1. initialize at a point $w^0$
2. travel in **downward direction** to new point $w^1$



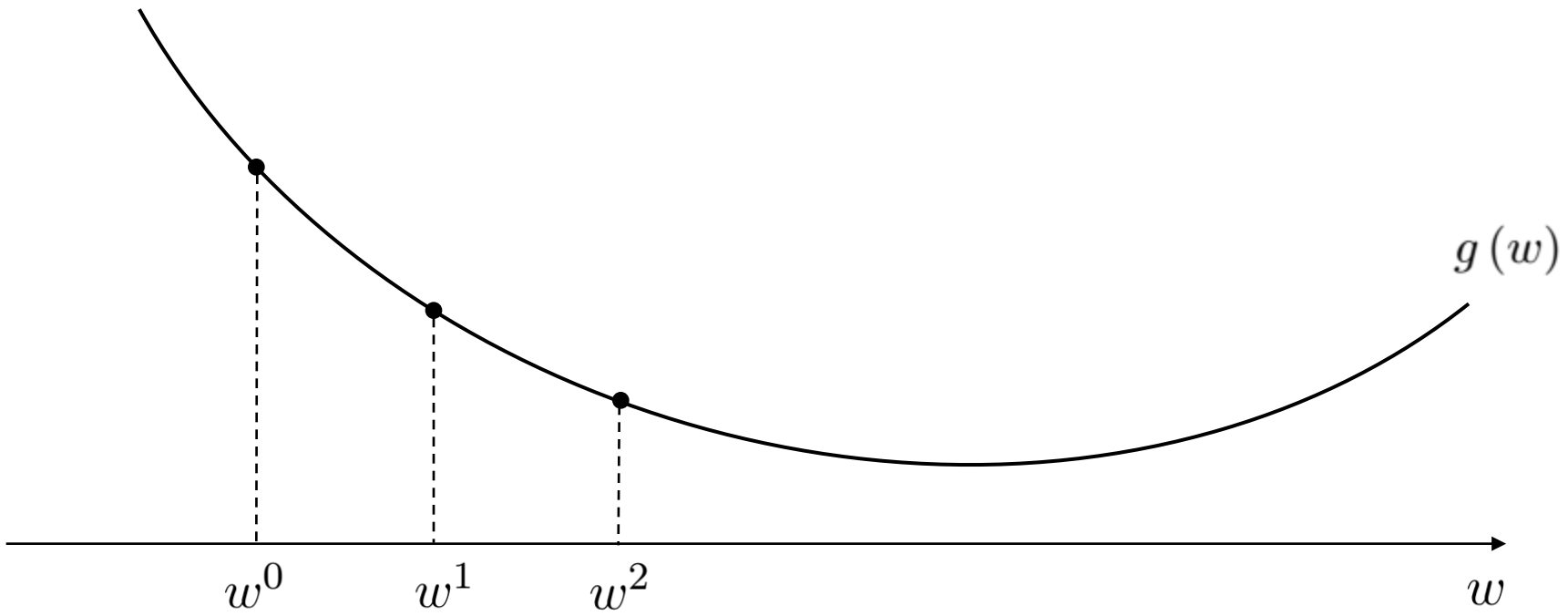
# Iterative methods

1. initialize at a point $w^0$
2. travel in **downward direction** to new point $w^1$
3. repeat 2. until convergence



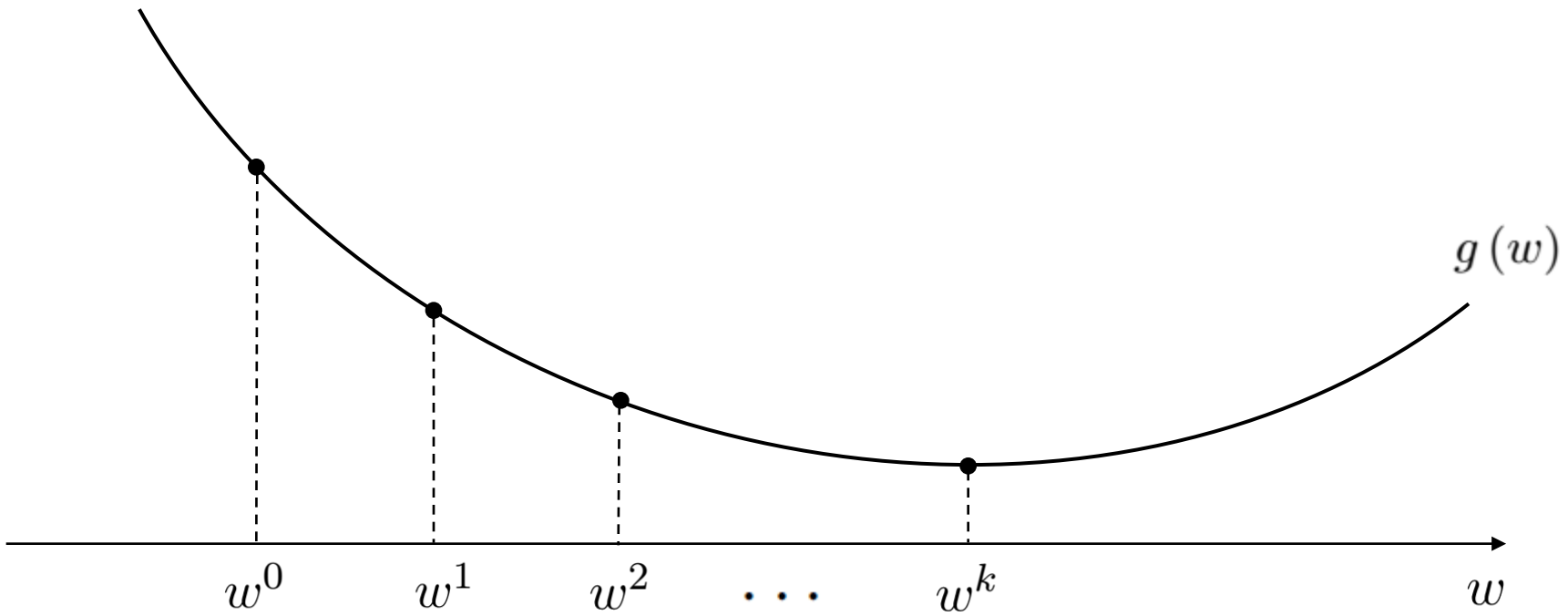
# Iterative methods

1. initialize at a point $w^0$
2. travel in **downward direction** to new point $w^1$
3. repeat 2. until convergence



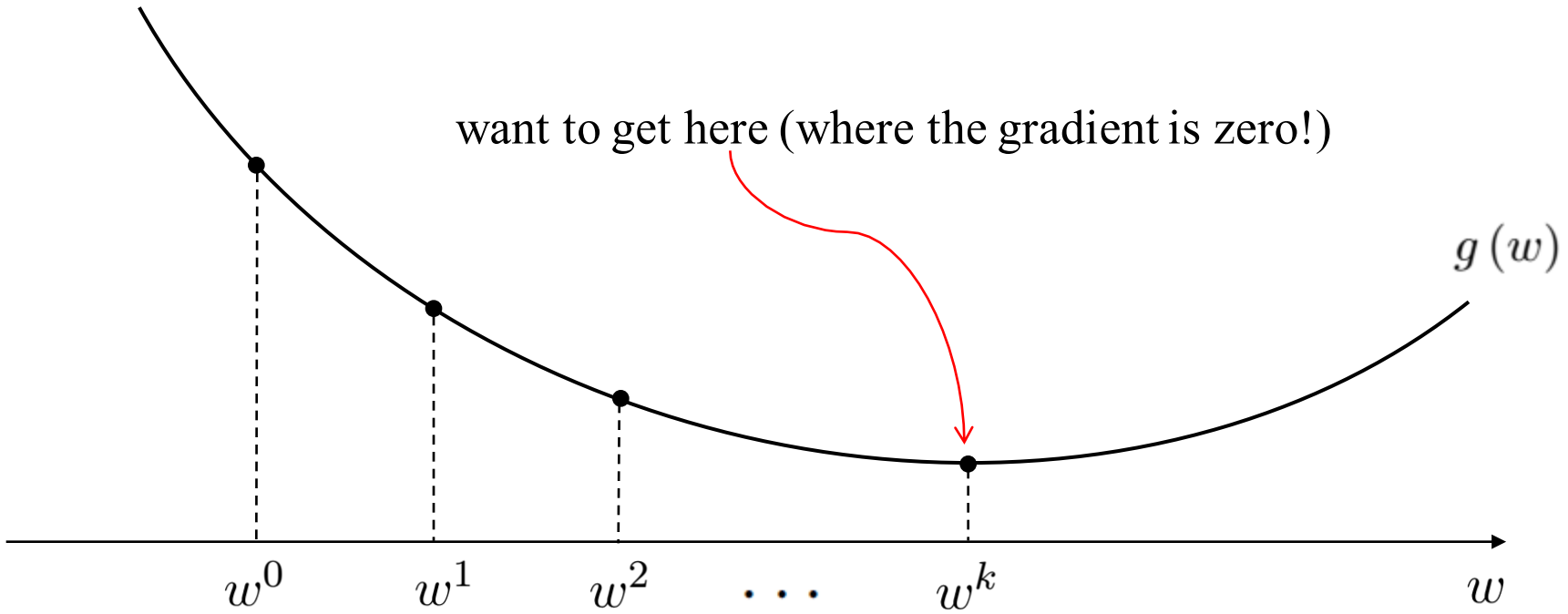
# Iterative methods

1. initialize at a point $w^0$
2. travel in **downward direction** to new point $w^1$
3. repeat 2. until convergence



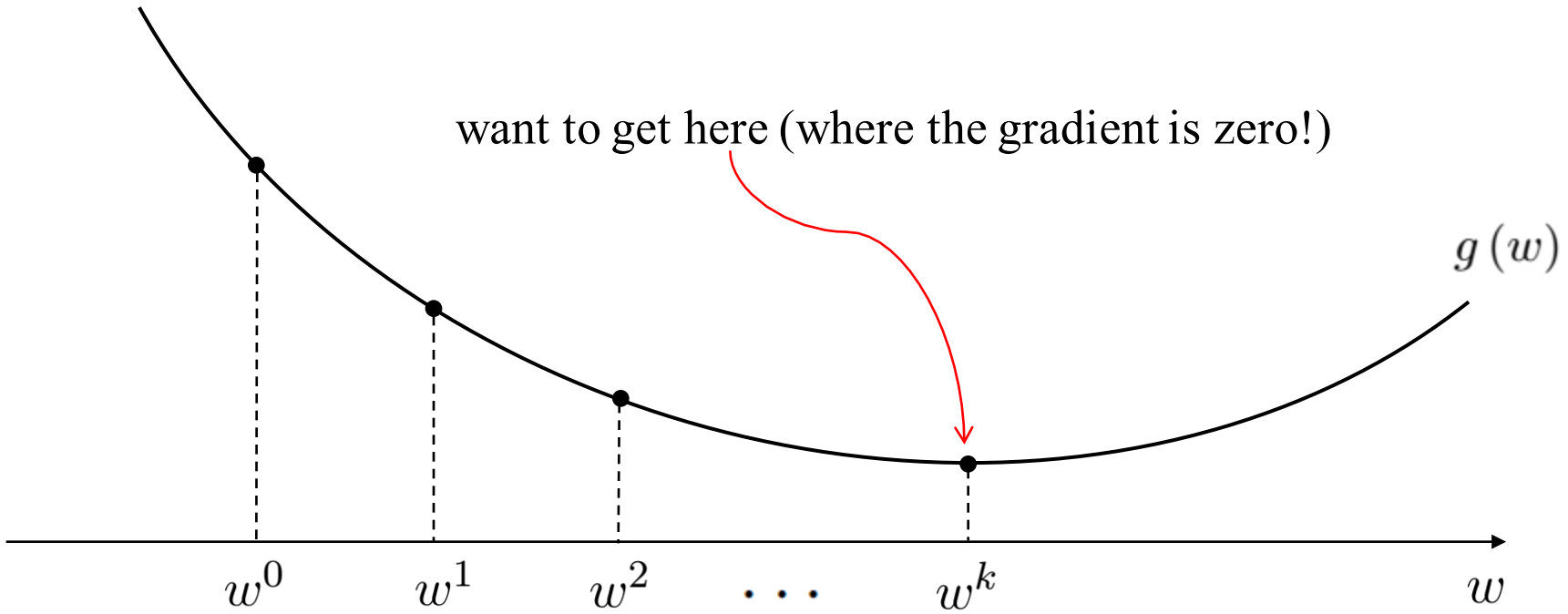
# Iterative methods

1. initialize at a point $w^0$
2. travel in **downward direction** to new point $w^1$
3. repeat 2. until convergence



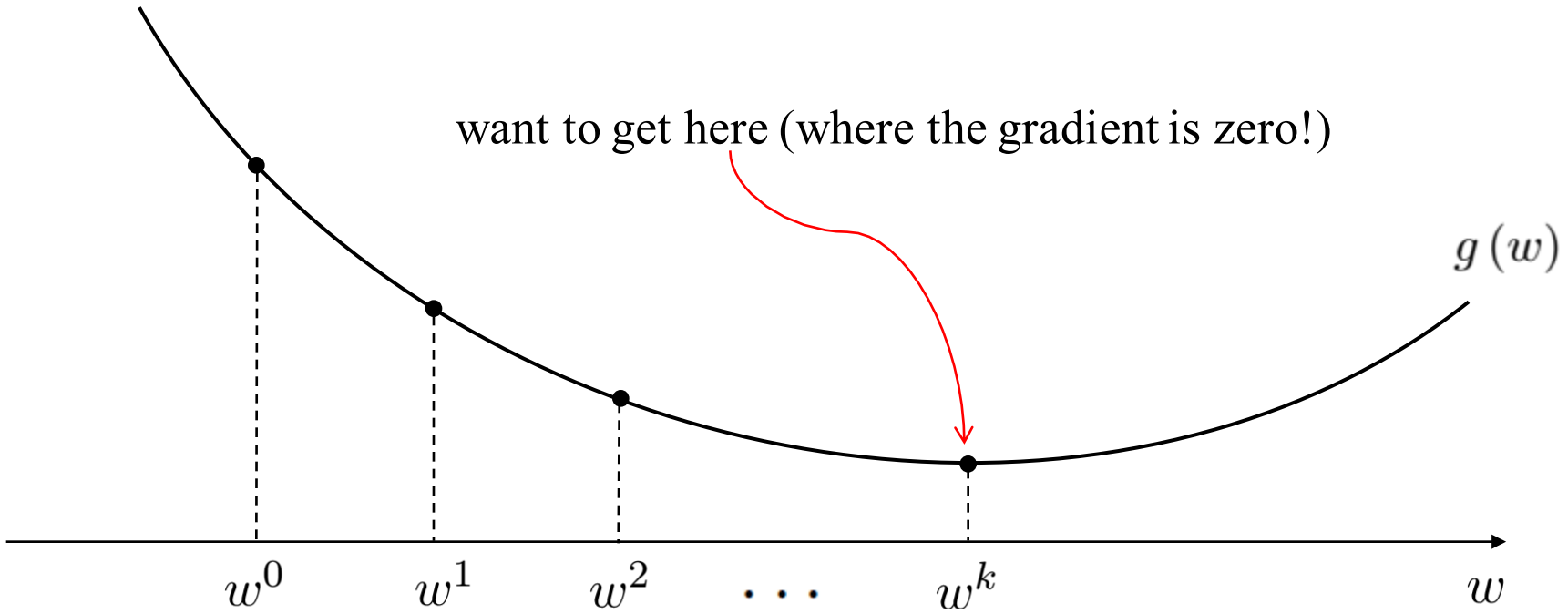
# Iterative methods

1. initialize at a point $w^0$
2. travel in **downward direction** to new point $w^1$
3. repeat 2. until convergence



# Iterative methods

1. initialize at a point $w^0$
2. travel in **downward direction** to new point $w^1$ Q1: which way is 'down'?
3. repeat 2. until convergence

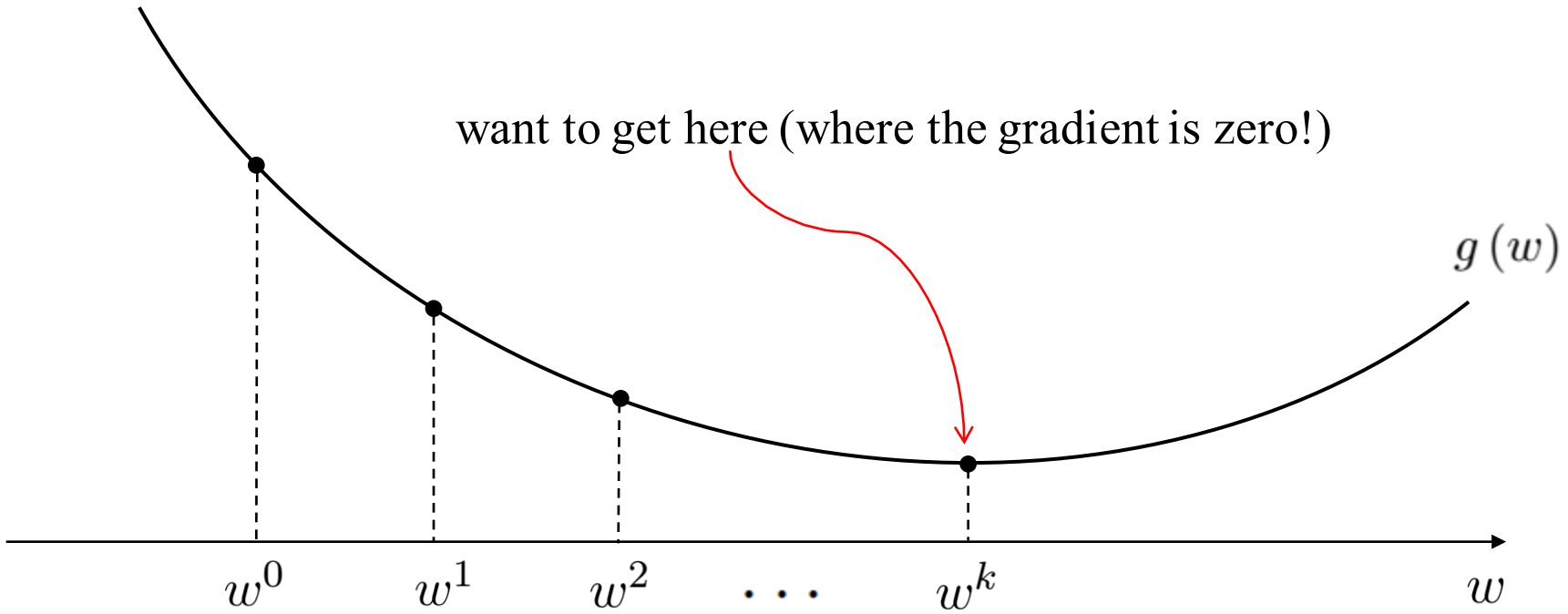


# Iterative methods

1. initialize at a point $w^0$
2. travel in **downward direction** to new point $w^1$
3. repeat 2. until convergence

Q1: which way is 'down'?

Q2: how do we stop?



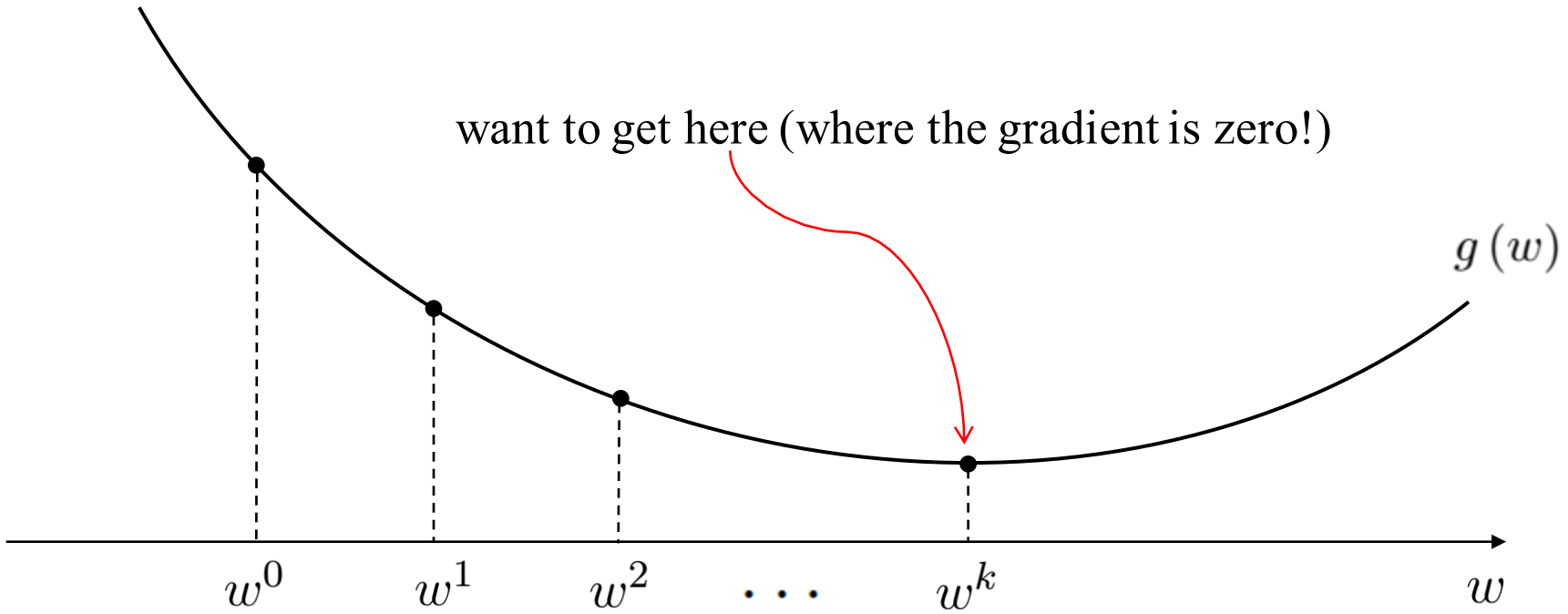
# Iterative methods

1. initialize at a point $w^0$
2. travel in **downward direction** to new point $w^1$
3. repeat 2. until convergence

Q1: which way is 'down'?

Q2: how do we stop?

Short answer: we use the derivative / gradient



# Demo 3 - tuning linear regression

- to the notebook!

# Gradient descent

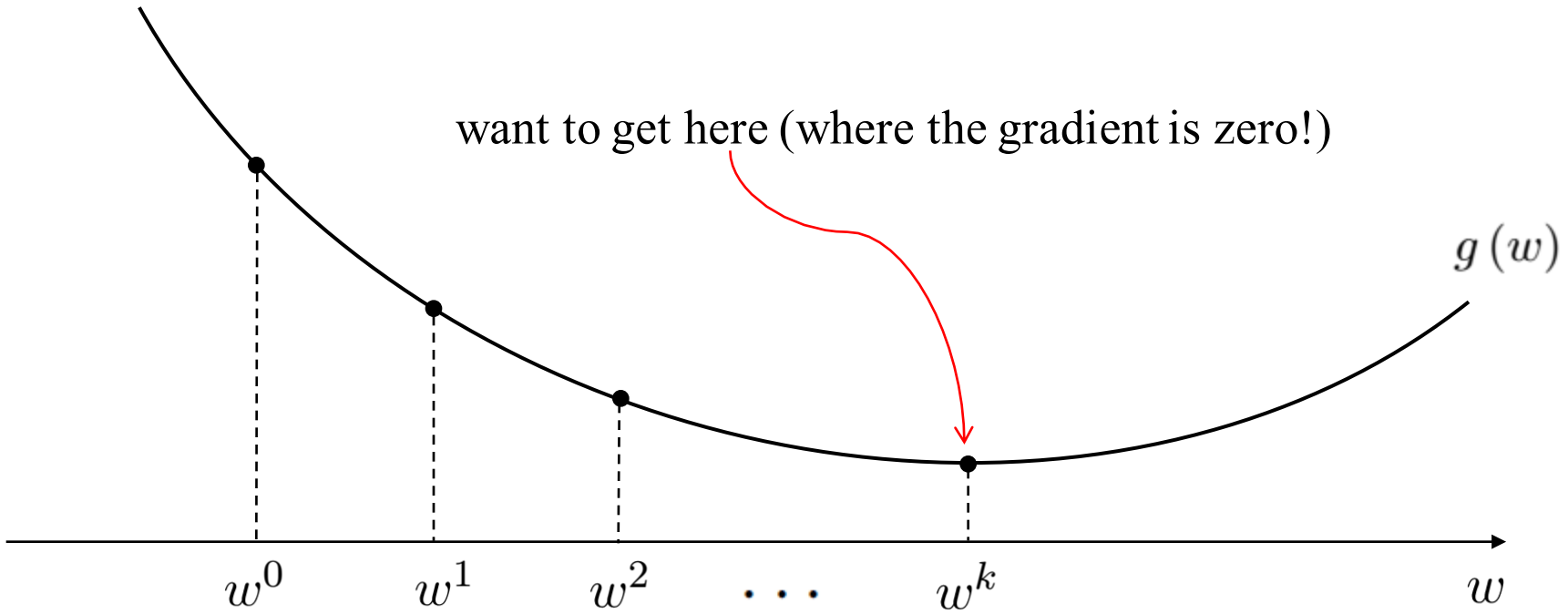
# gradient descent

1. initialize at a point $w^0$
2. travel in **downward direction** to new point $w^1$
3. repeat 2. until convergence

Q1: which way is 'down'?

Q2: how do we stop?

Short answer: we use the derivative / gradient

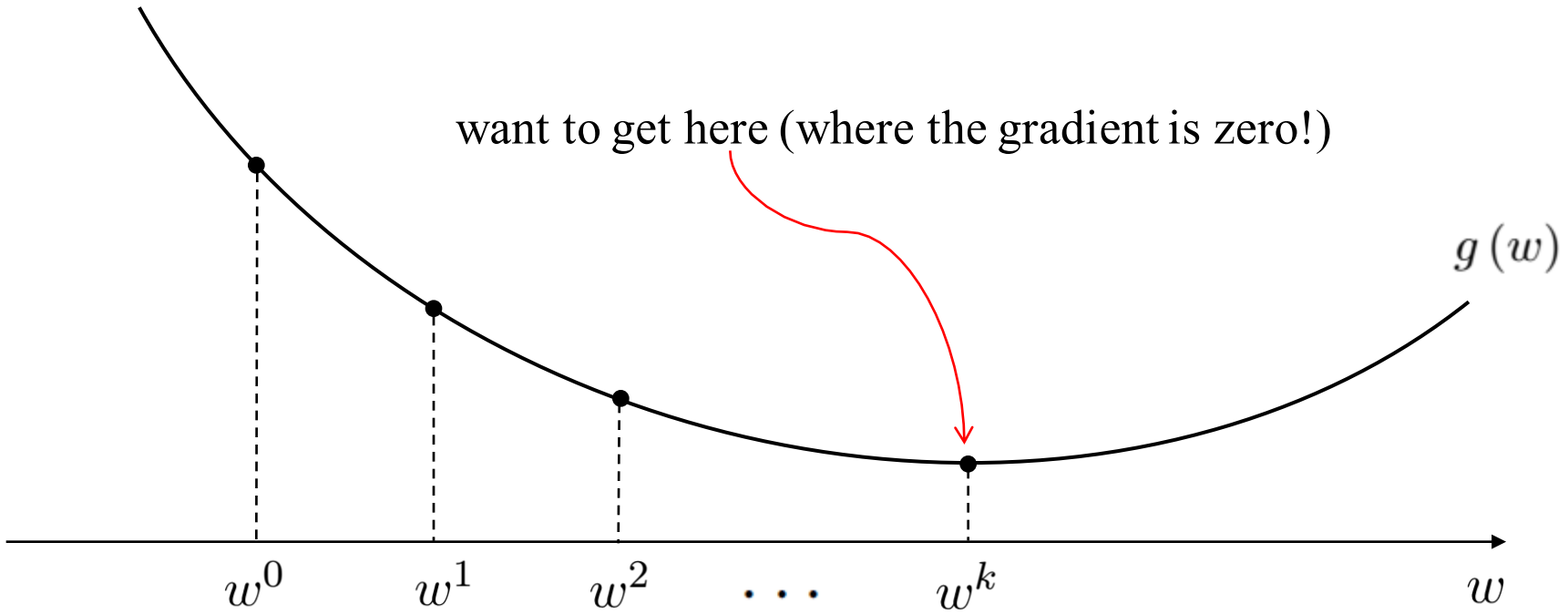


# gradient descent

1. initialize at a point $w^0$
2. travel in **downward direction** to new point $w^1$
3. repeat 2. until convergence

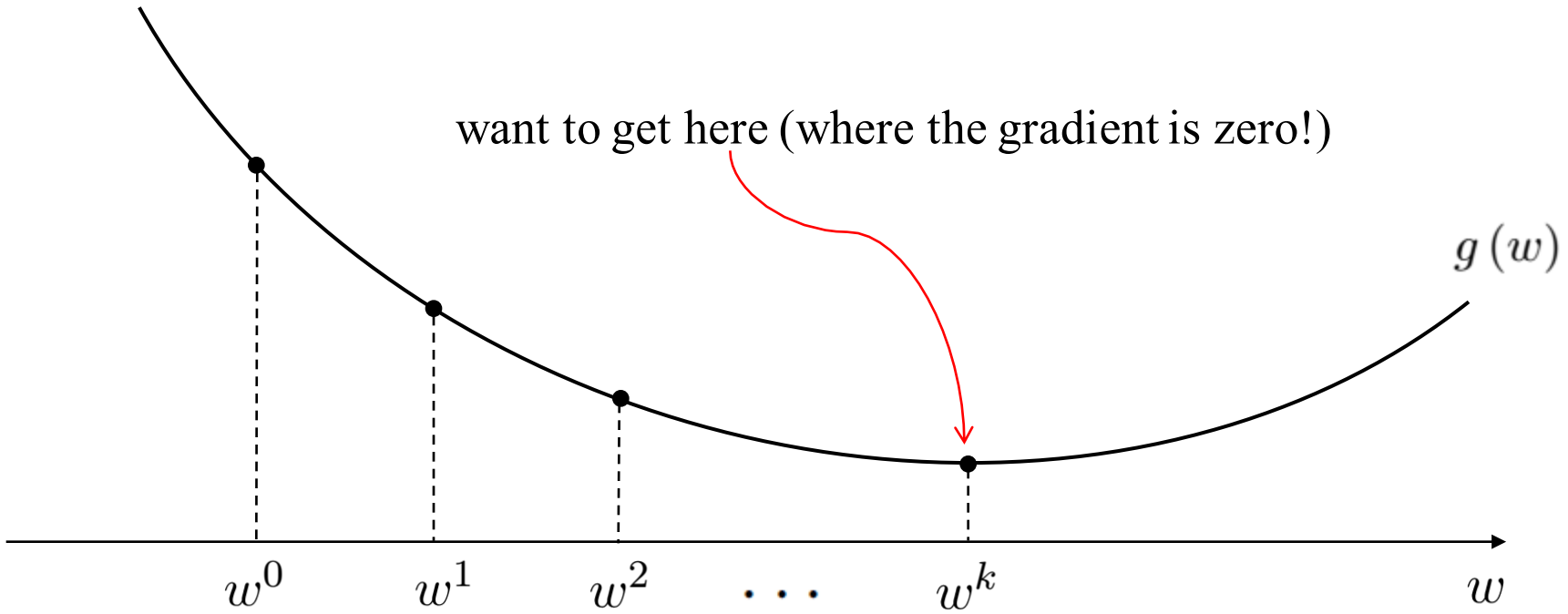
Q1: which way is 'down'?

Q2: how do we stop?



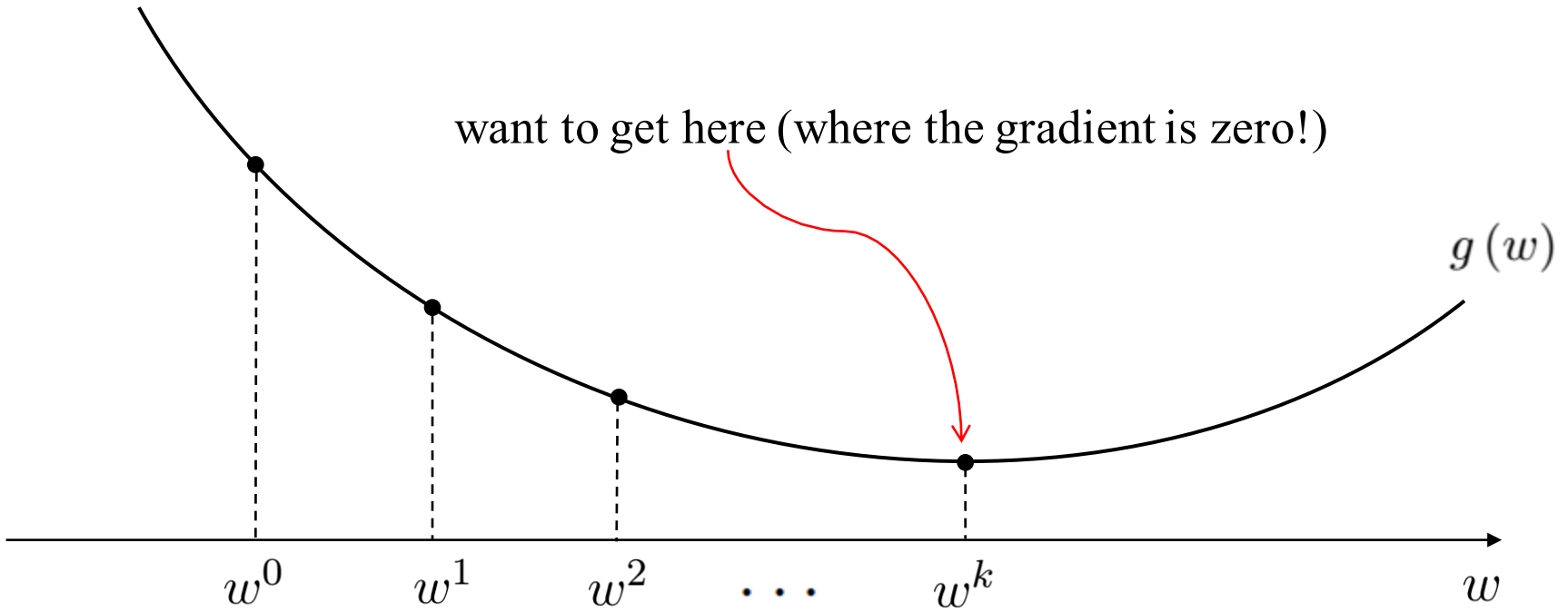
# gradient descent

1. initialize at a point $w^0$
2. travel in **downward direction** to new point $w^1$ Q1: which way is 'down'?
3. repeat 2. until convergence



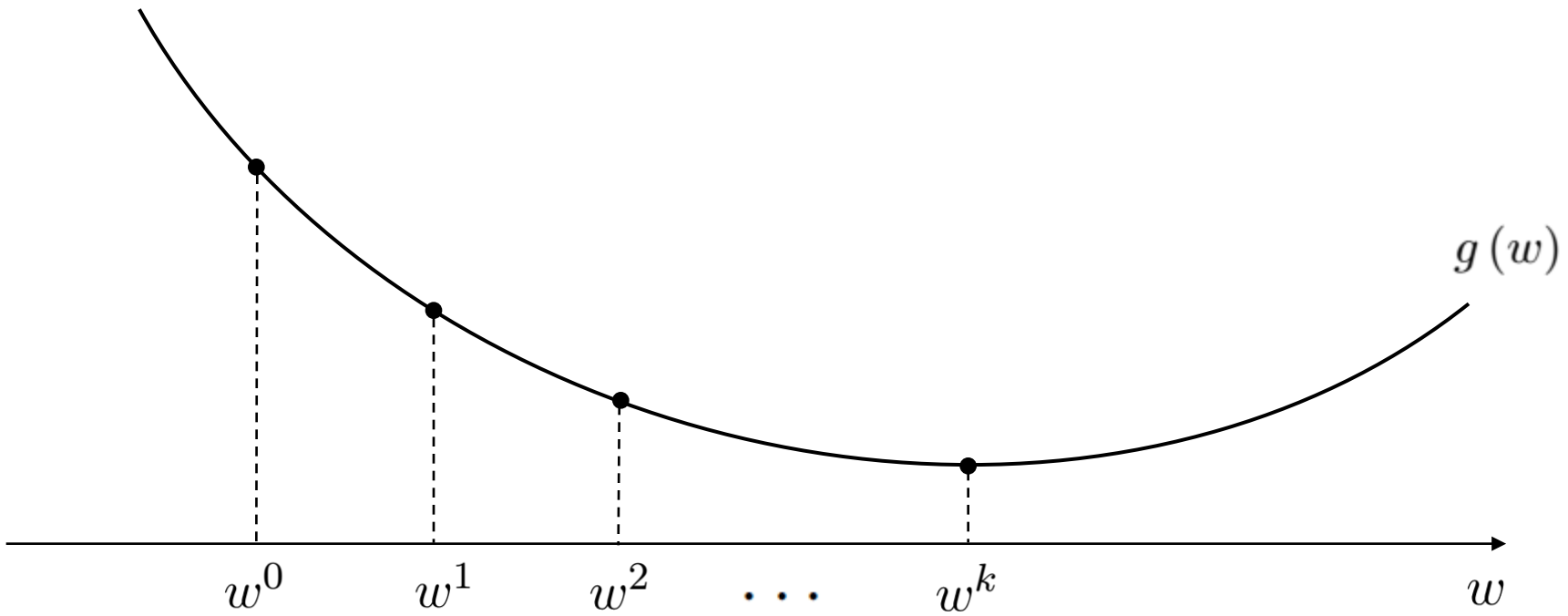
# gradient descent

1. initialize at a point $w^0$
2. travel in **downward direction** to new point $w^1$
3. repeat 2. until convergence



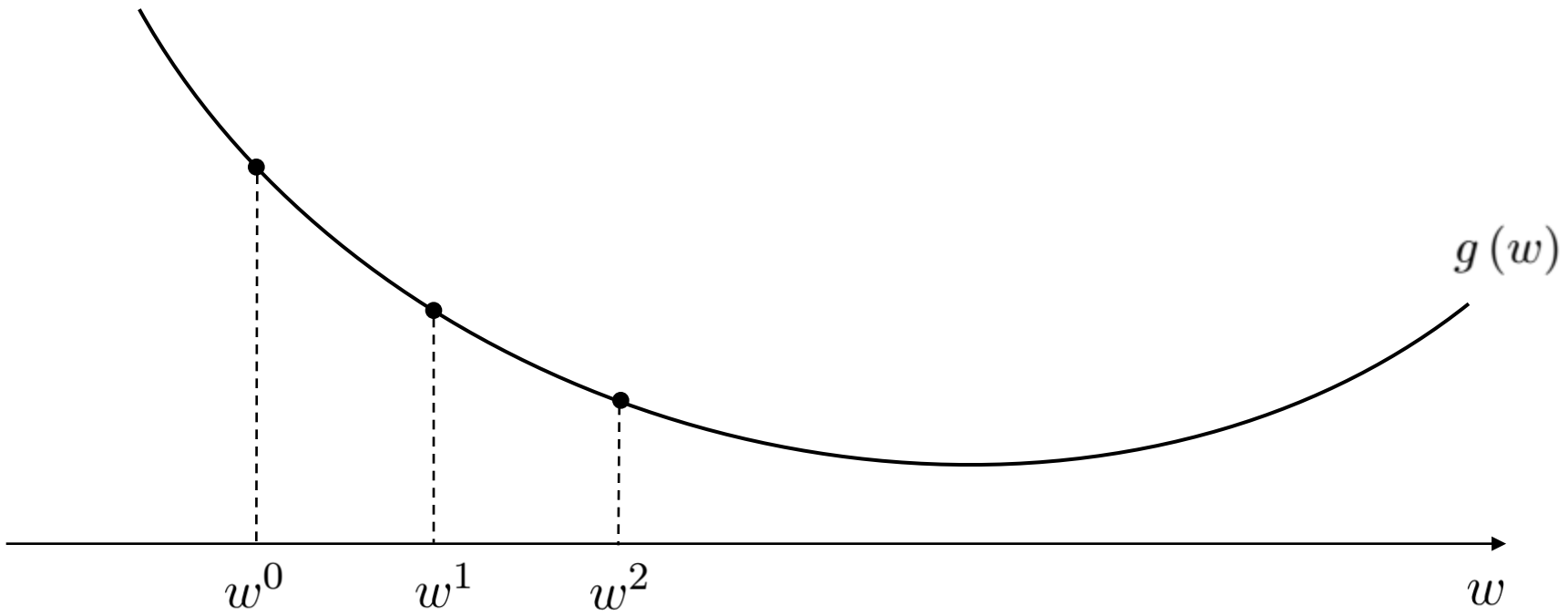
# gradient descent

1. initialize at a point $w^0$
2. travel in **downward direction** to new point $w^1$
3. repeat 2. until convergence



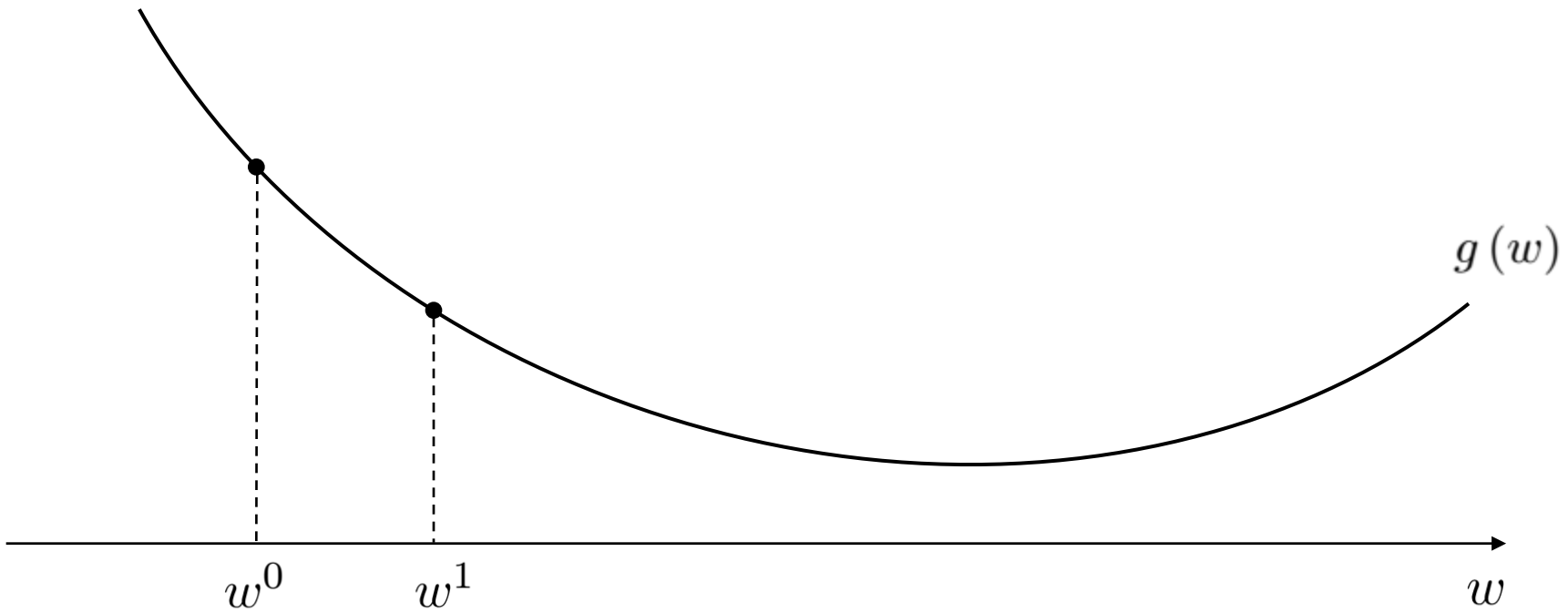
# gradient descent

1. initialize at a point $w^0$
2. travel in **downward direction** to new point $w^1$
3. repeat 2. until convergence



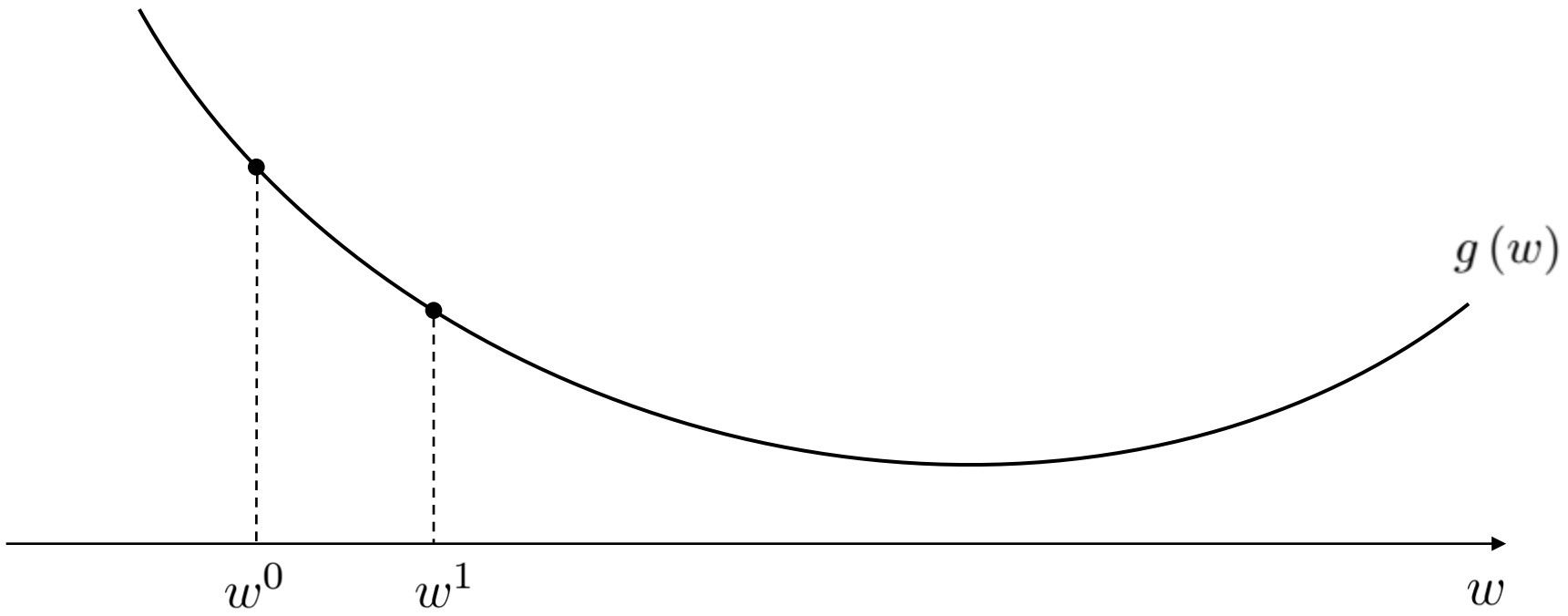
# gradient descent

1. initialize at a point $w^0$
2. travel in **downward direction** to new point $w^1$
3. repeat 2. until convergence



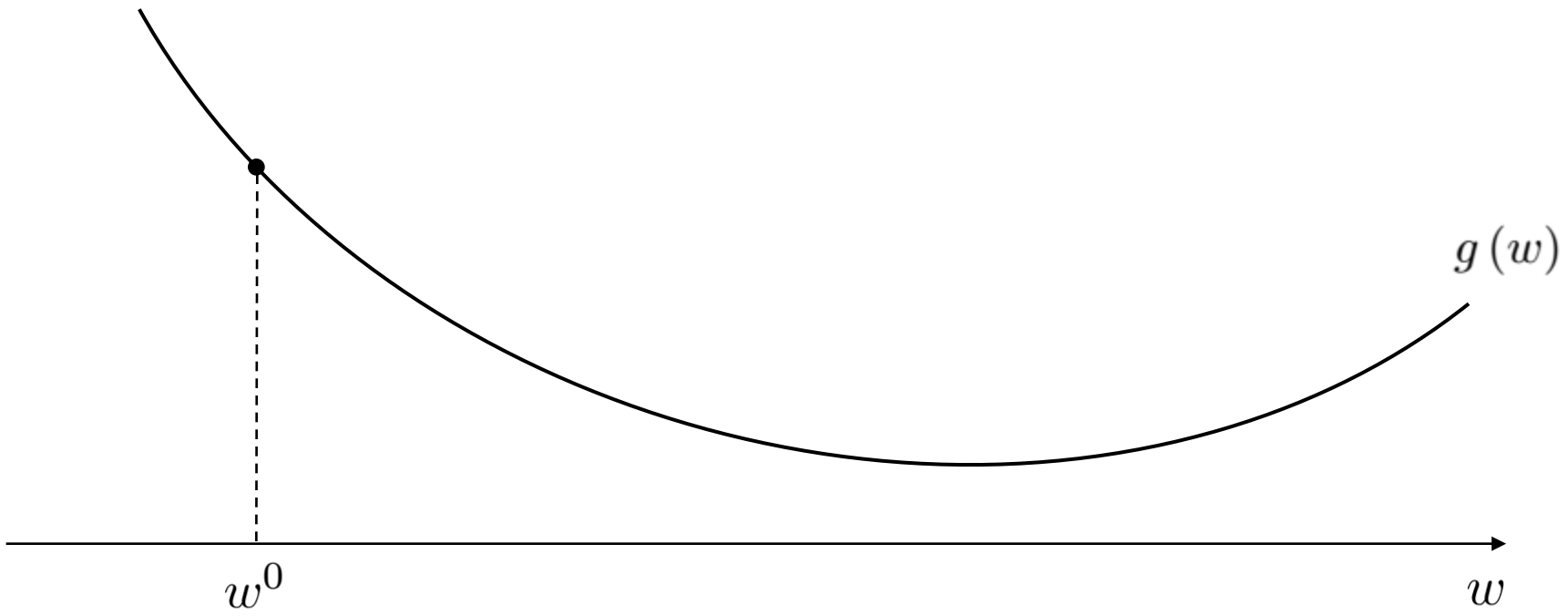
# gradient descent

1. initialize at a point $w^0$
2. travel in **downward direction** to new point $w^1$



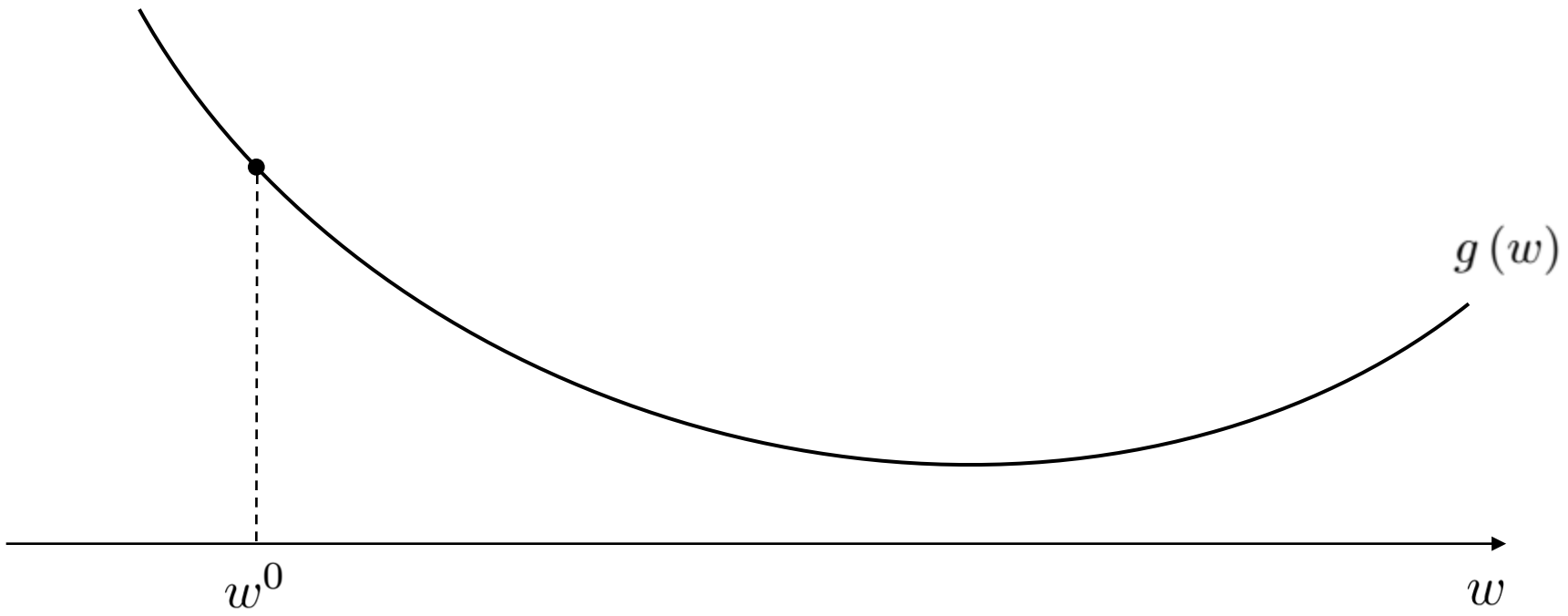
# gradient descent

1. initialize at a point $w^0$
2. travel in **downward direction** to new point $w^1$



# gradient descent

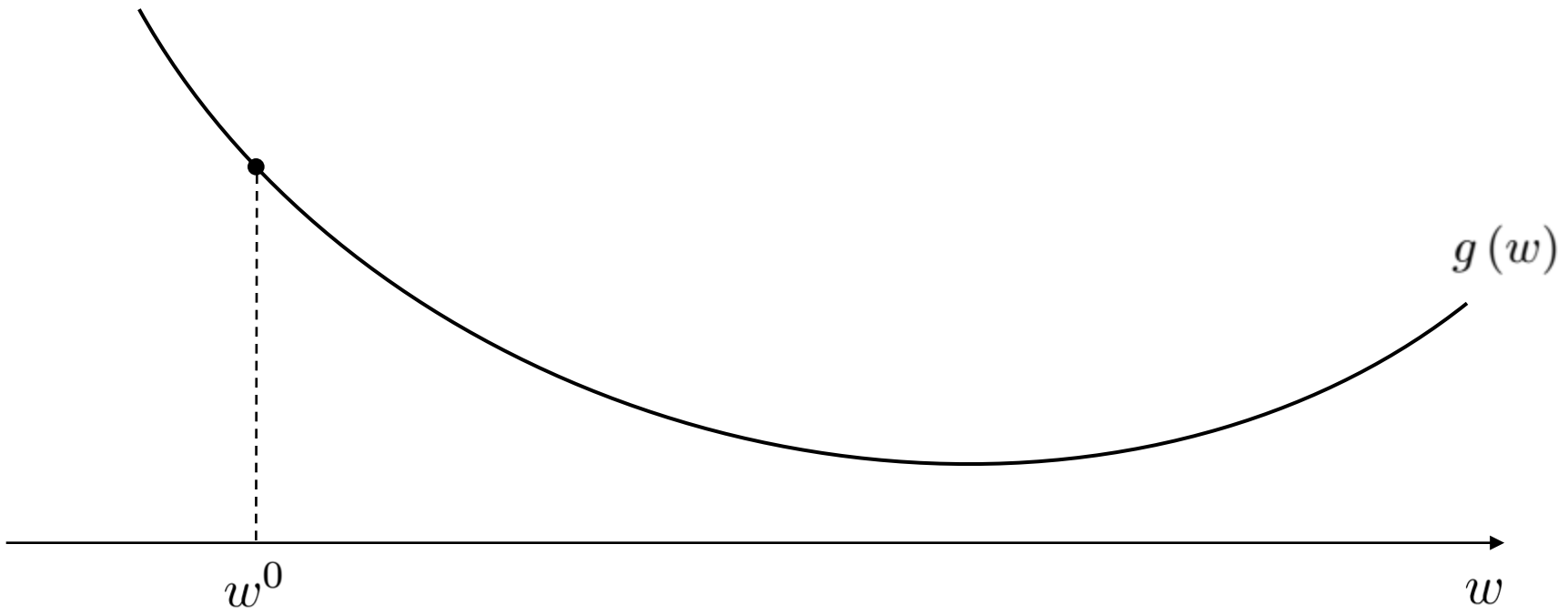
1. initialize at a point $w^0$
2. travel in **<u>downward direction</u>** to new point $w^1$



# gradient descent

1. initialize at a point $w^0$
2. travel in **<u>downward direction</u>** to new point $w^1$

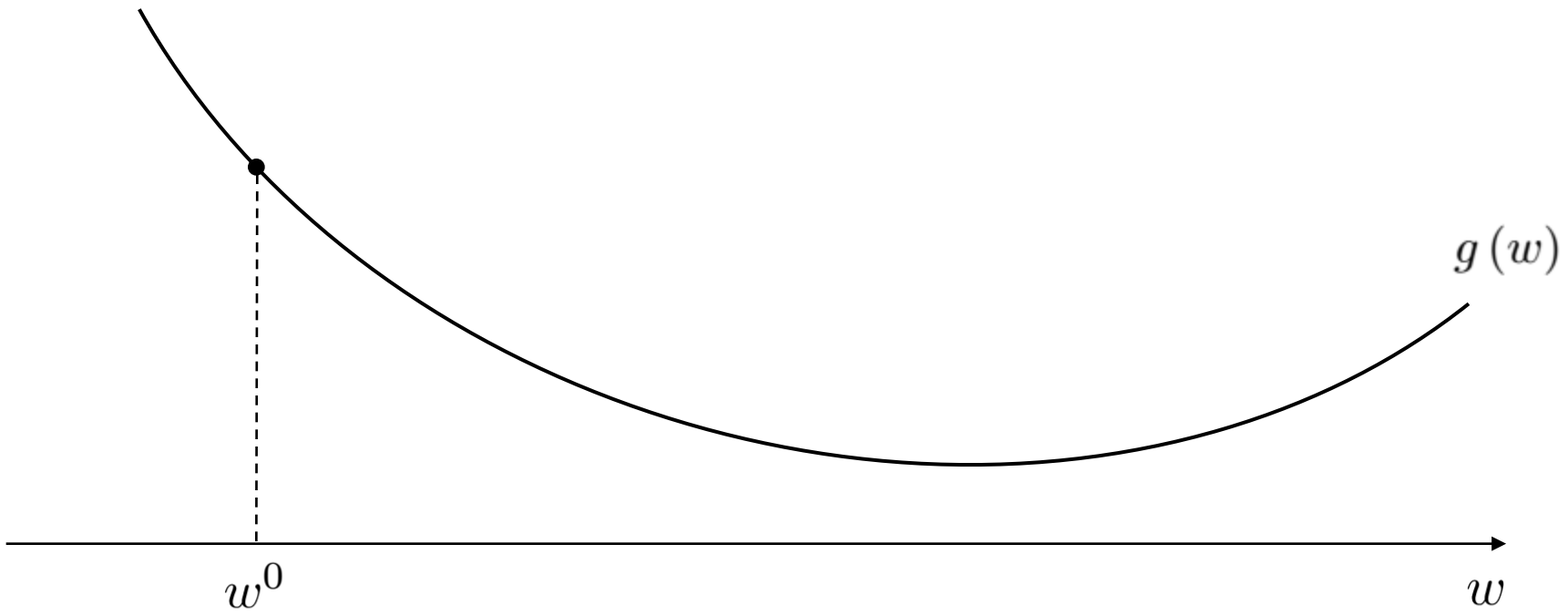
which way is down?



# gradient descent

1. initialize at a point $w^0$
2. travel in **<u>downward direction</u>** to new point $w^1$

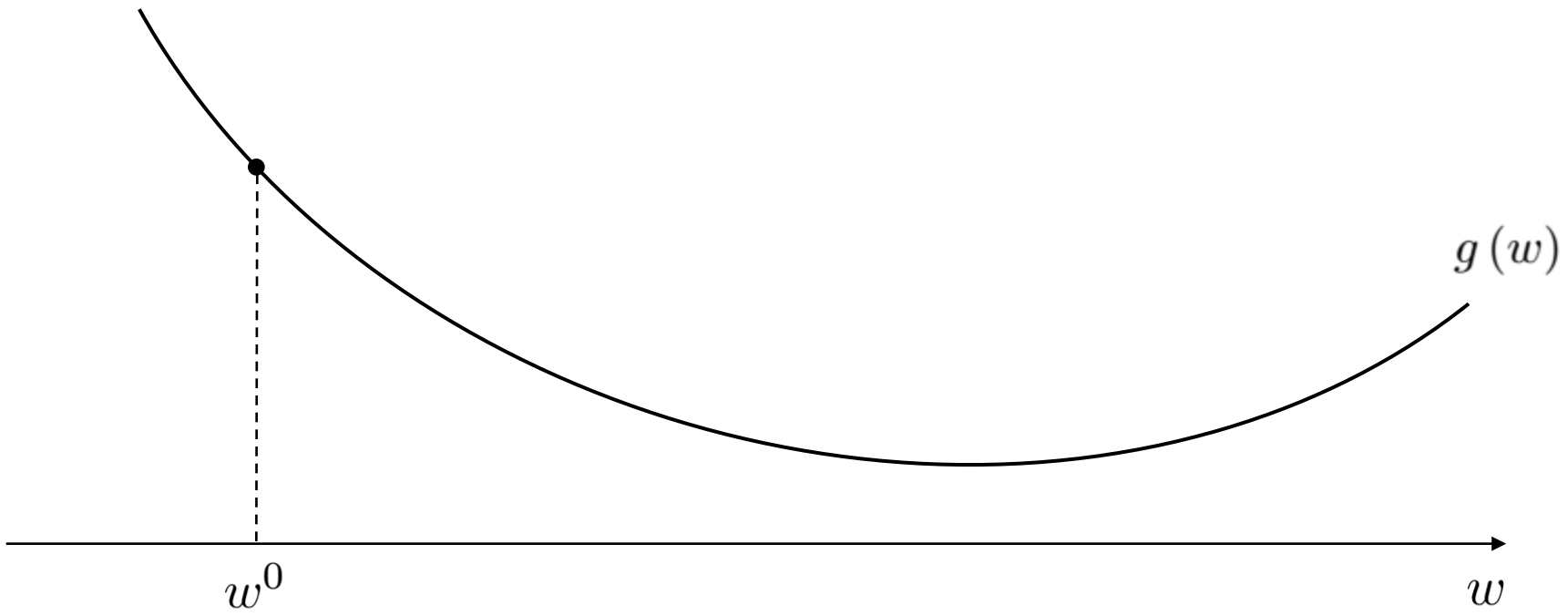
**Q**: What do we know about $g(w)$ at $w^0$?



# gradient descent

1. initialize at a point $w^0$
2. travel in **<u>downward direction</u>** to new point $w^1$

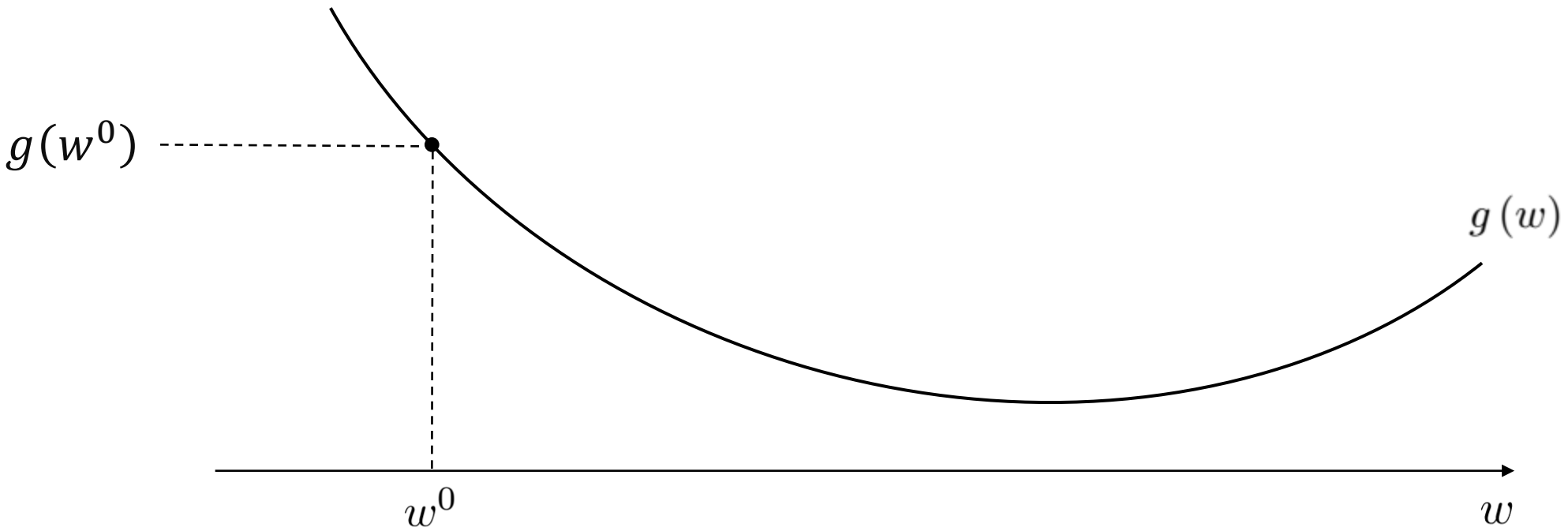
**Q**: What do we know about $g(w)$ at $w^0$?
- value of cost $g(w^0)$



# gradient descent

1. initialize at a point $w^0$
2. travel in **<u>downward direction</u>** to new point $w^1$

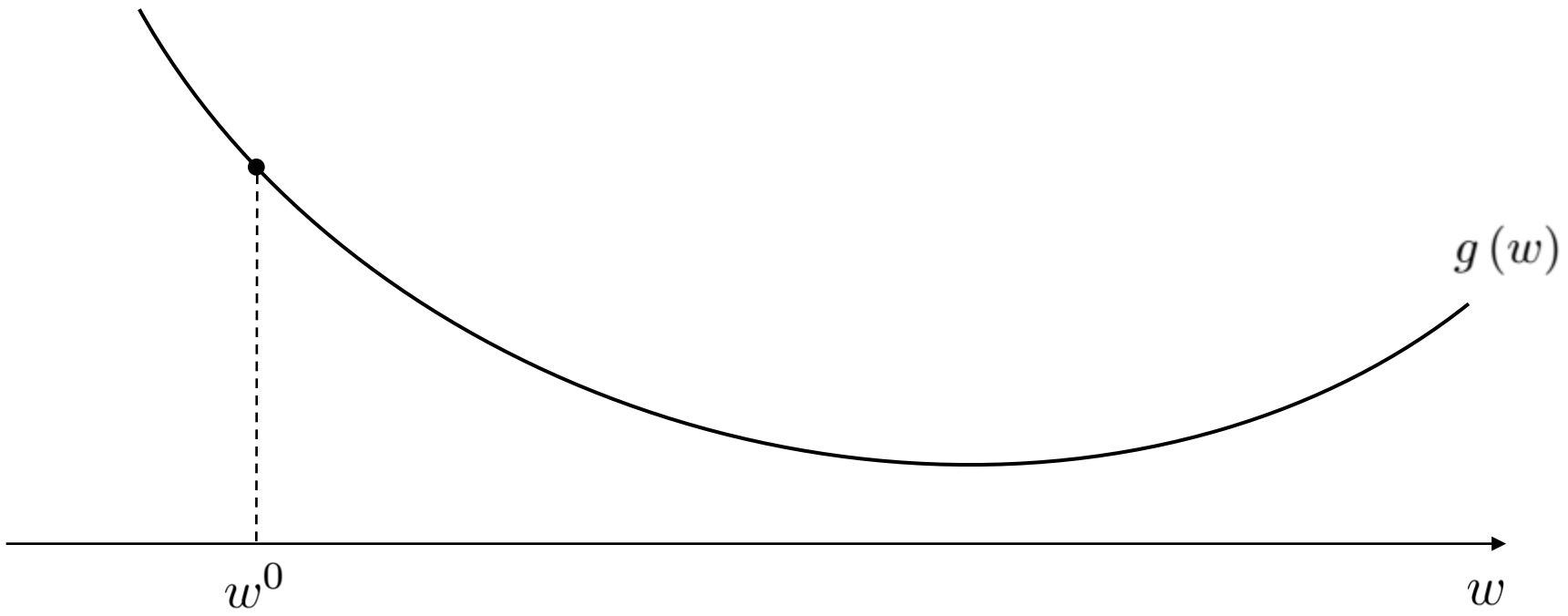
**Q**: What do we know about $g(w)$ at $w^0$?
- value of cost $g(w^0)$



# gradient descent

1. initialize at a point $w^0$
2. travel in **<u>downward direction</u>** to new point $w^1$

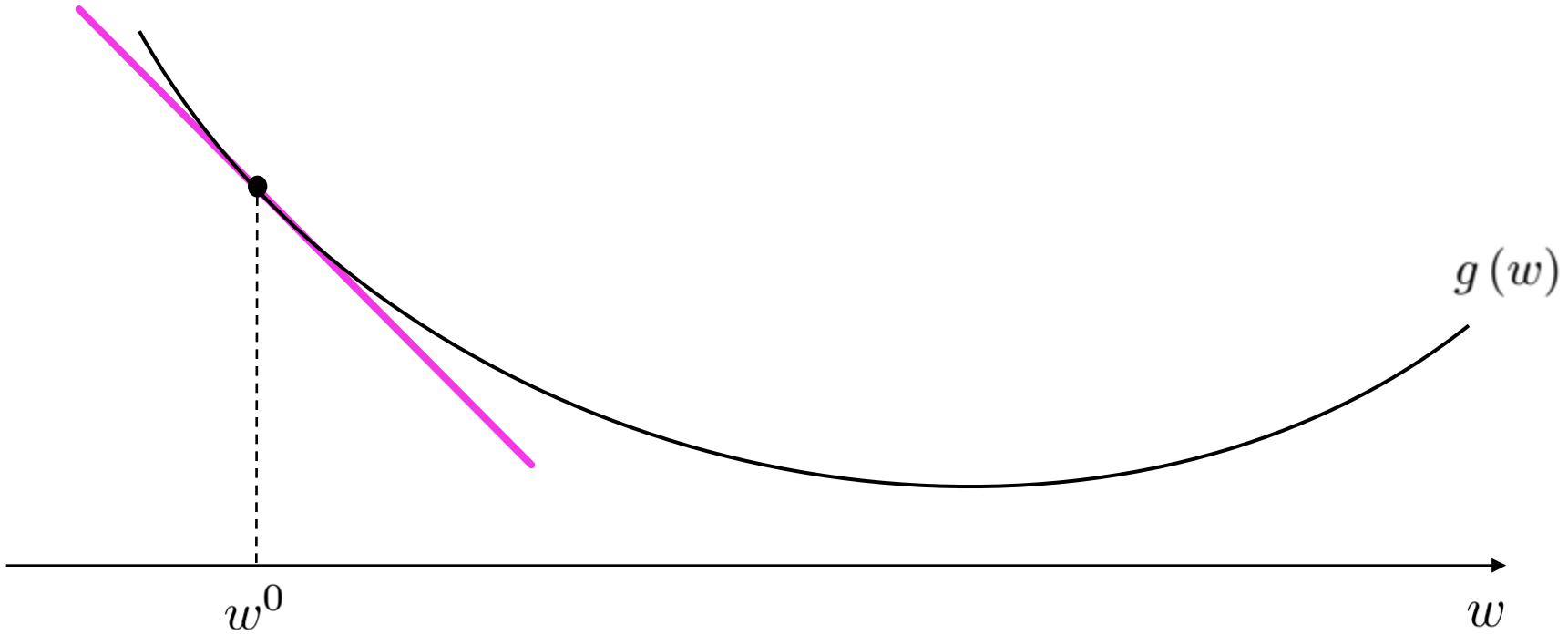
**Q**: What do we know about $g(w)$ at $w^0$?
- value of cost $g(w^0)$
- value of derivative $g'(w^0)$



# gradient descent

1. initialize at a point $w^0$
2. travel in **<u>downward direction</u>** to new point $w^1$

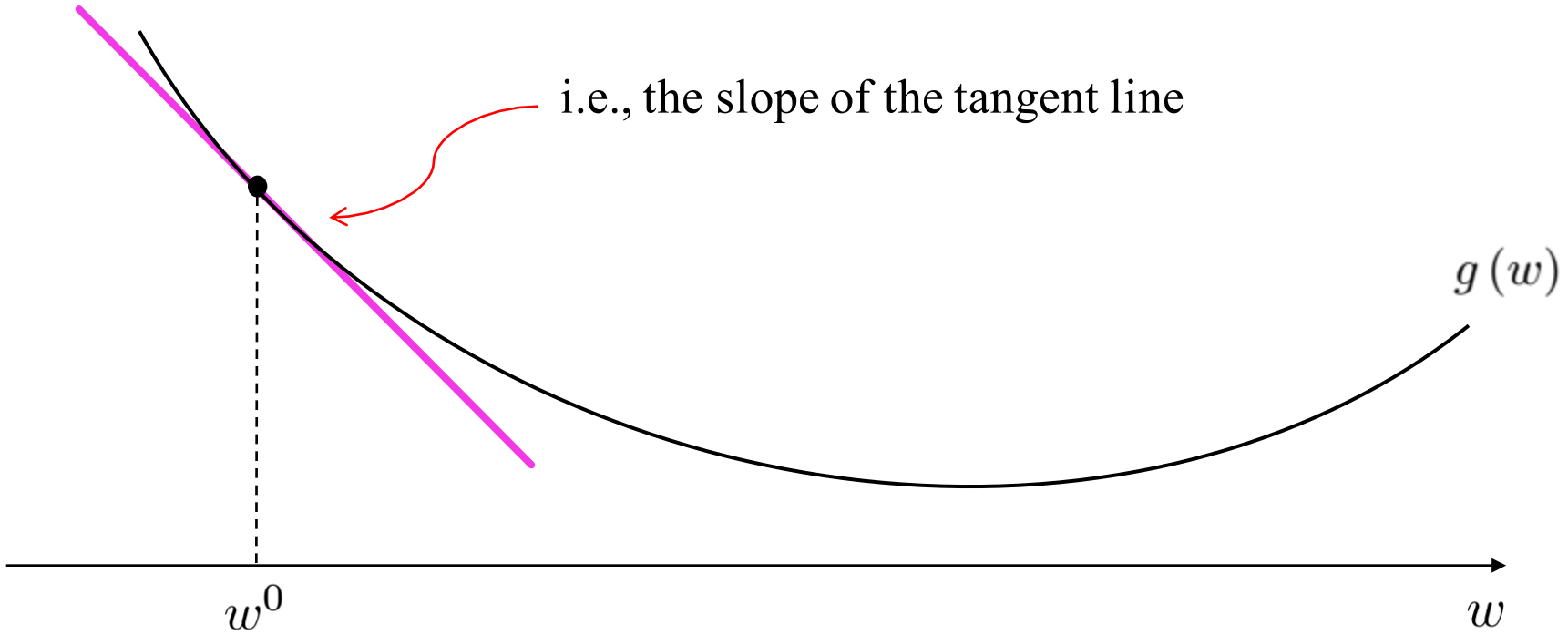
**Q**: What do we know about $g(w)$ at $w^0$?
- value of cost $g(w^0)$
- value of derivative $g'(w^0)$



# gradient descent

1. initialize at a point $w^0$
2. travel in **<u>downward direction</u>** to new point $w^1$

**Q**: What do we know about $g(w)$ at $w^0$?
- value of cost $g(w^0)$
- value of derivative $g'(w^0)$

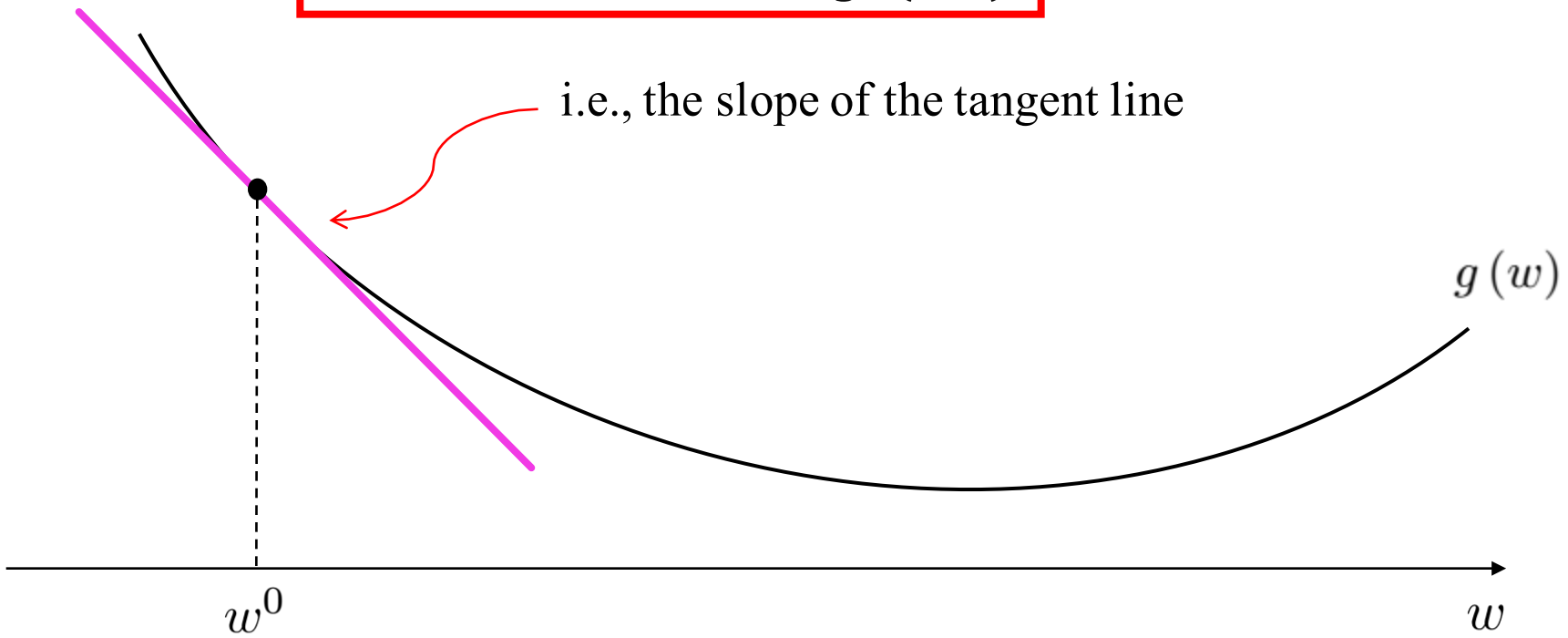


# gradient descent

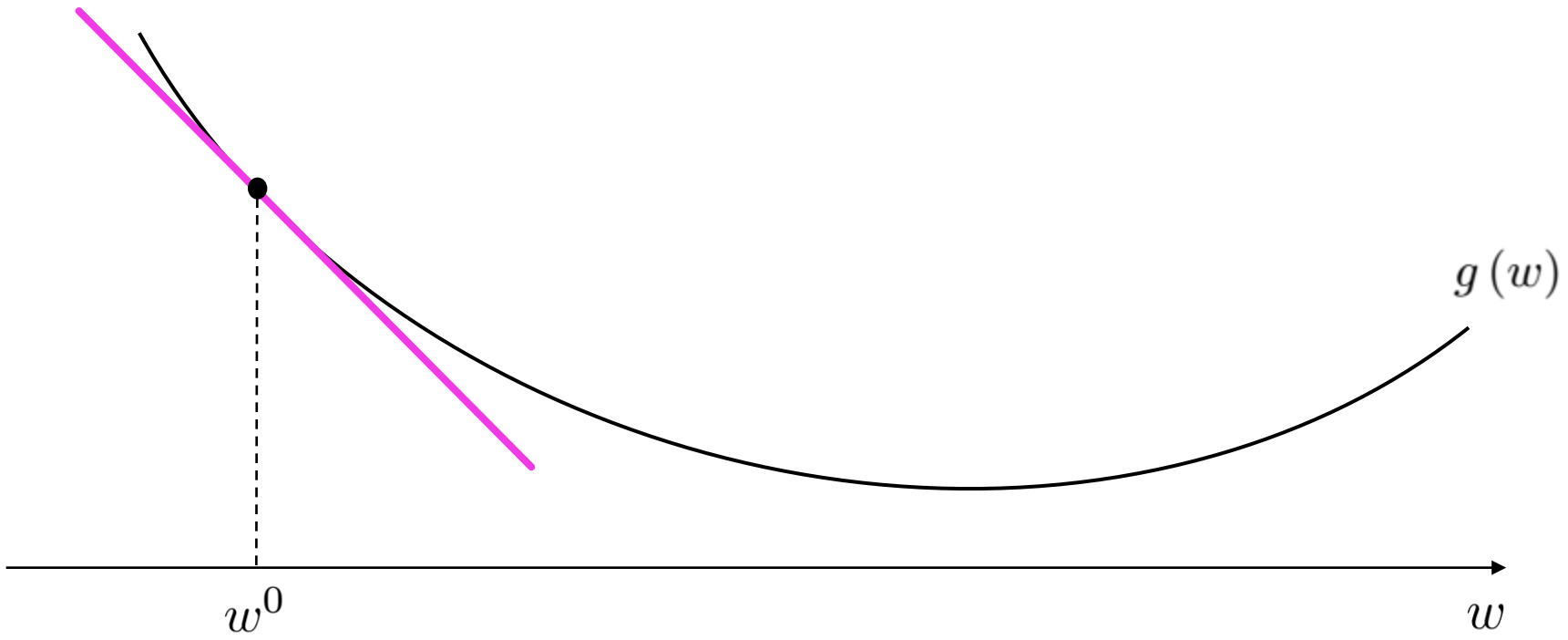
1. initialize at a point $w^0$
2. travel in **<u>downward direction</u>** to new point $w^1$

**Q**: What do we know about $g(w)$ at $w^0$?

- value of cost $g(w^0)$
- value of derivative $g'(w^0)$

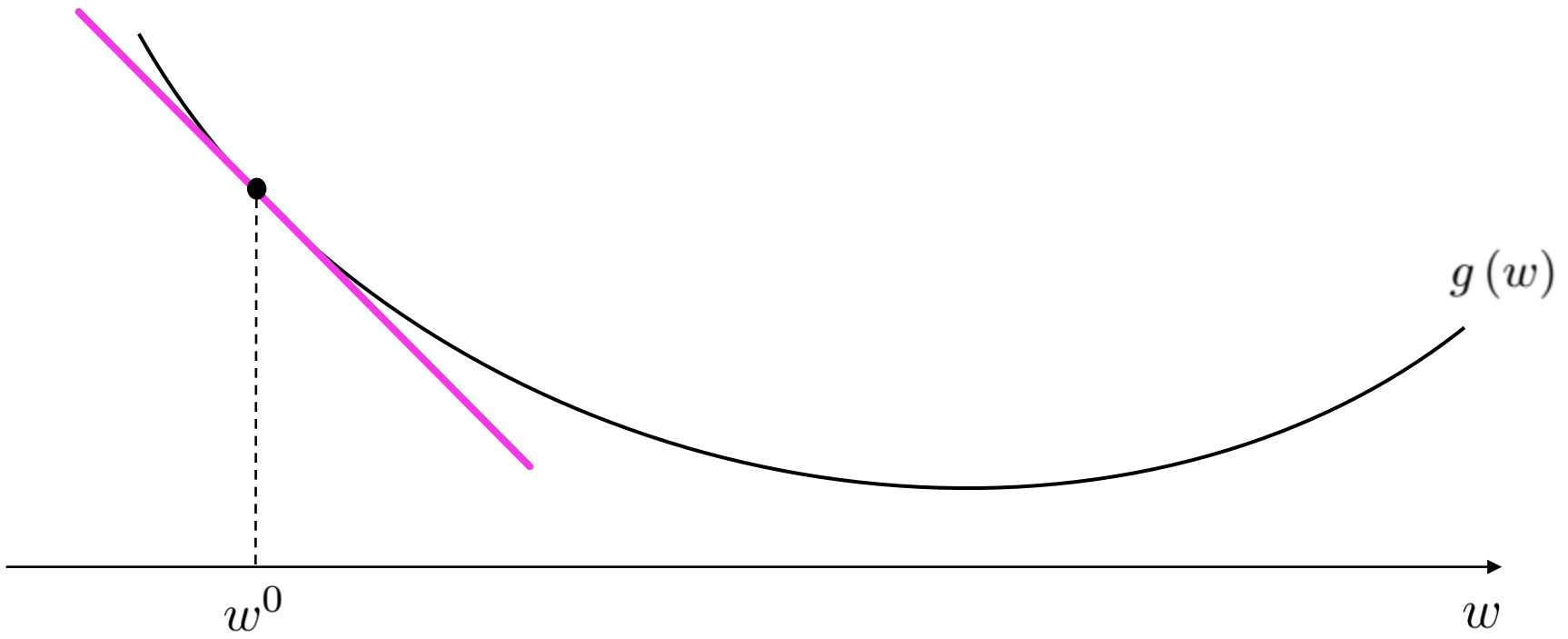


# gradient descent



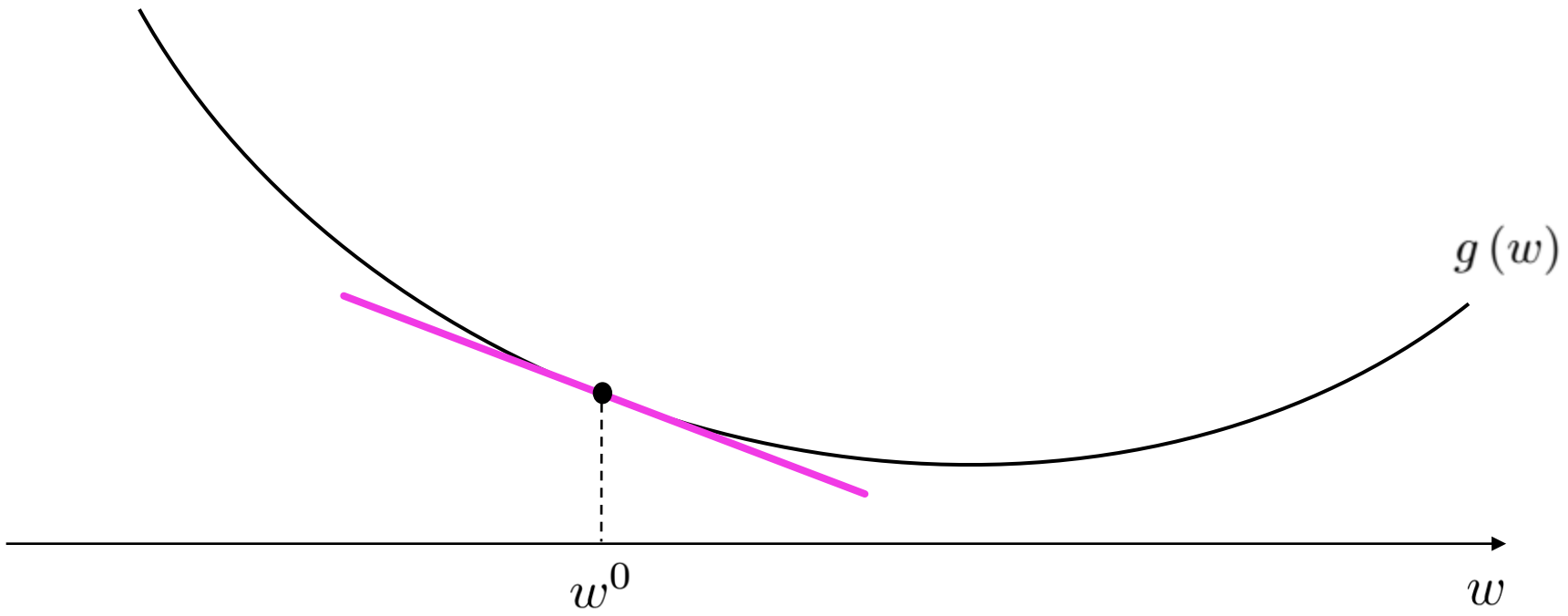
# gradient descent

✓ the tangent line looks a whole lot like $g(w)$ near $w^0$



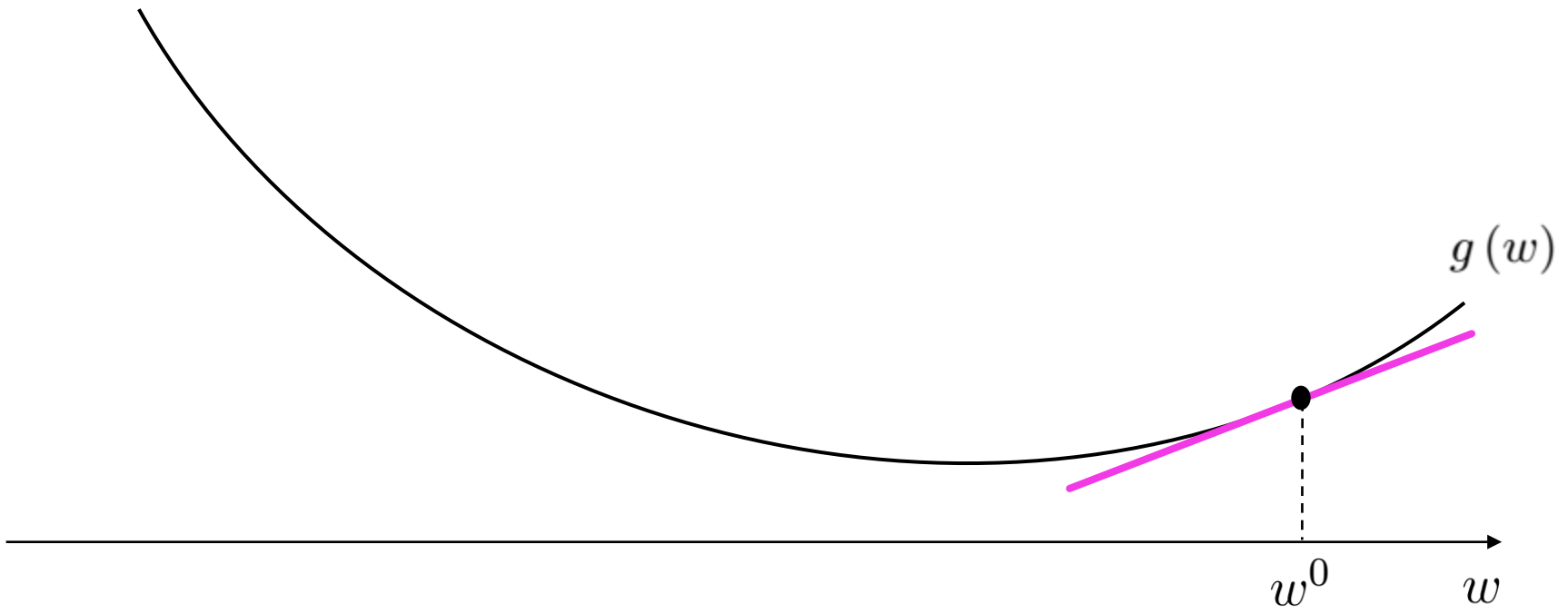
# gradient descent

✓ the tangent line looks a whole lot like $g(w)$ near $w^0$



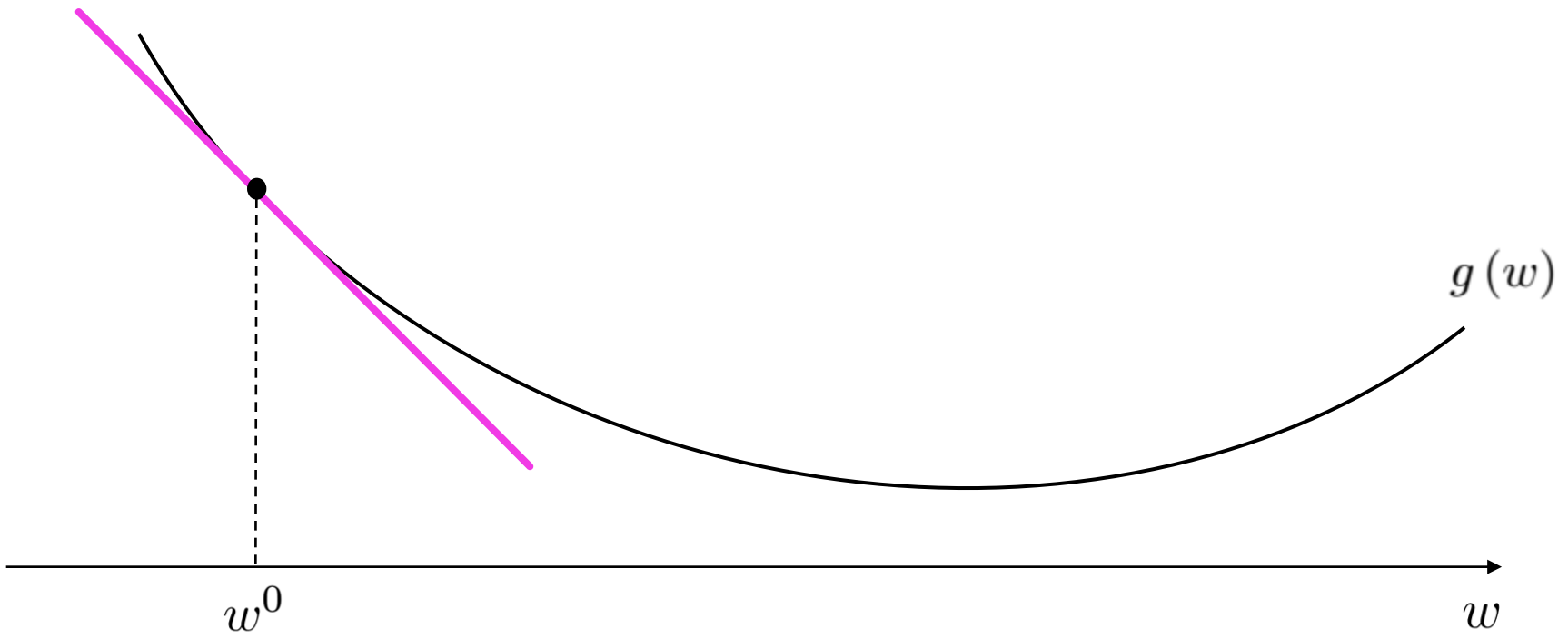
# gradient descent

✓ the tangent line looks a whole lot like $g(w)$ near $w^0$



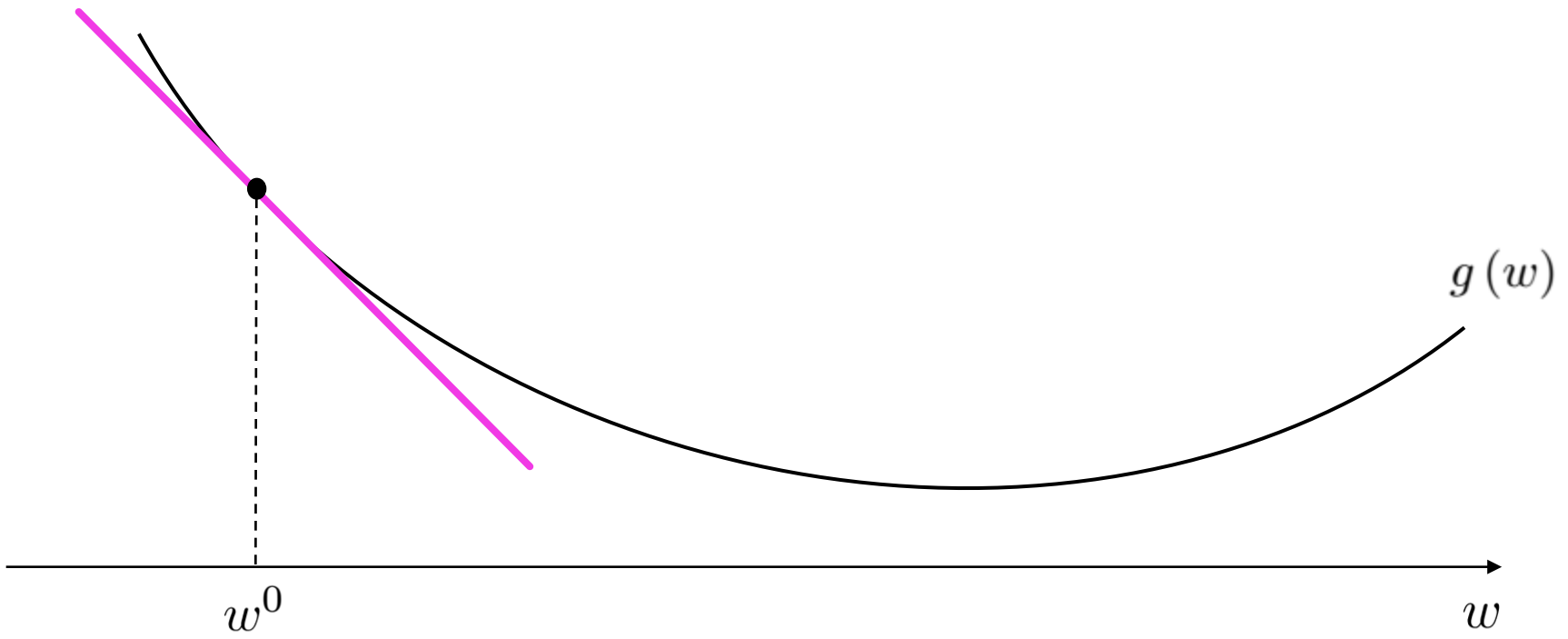
# gradient descent

- ✓ the tangent line looks a whole lot like $g(w)$ near $w^0$
- ✓ traveling downward on line ≈ traveling downward on $g(w)$ near $w^0$



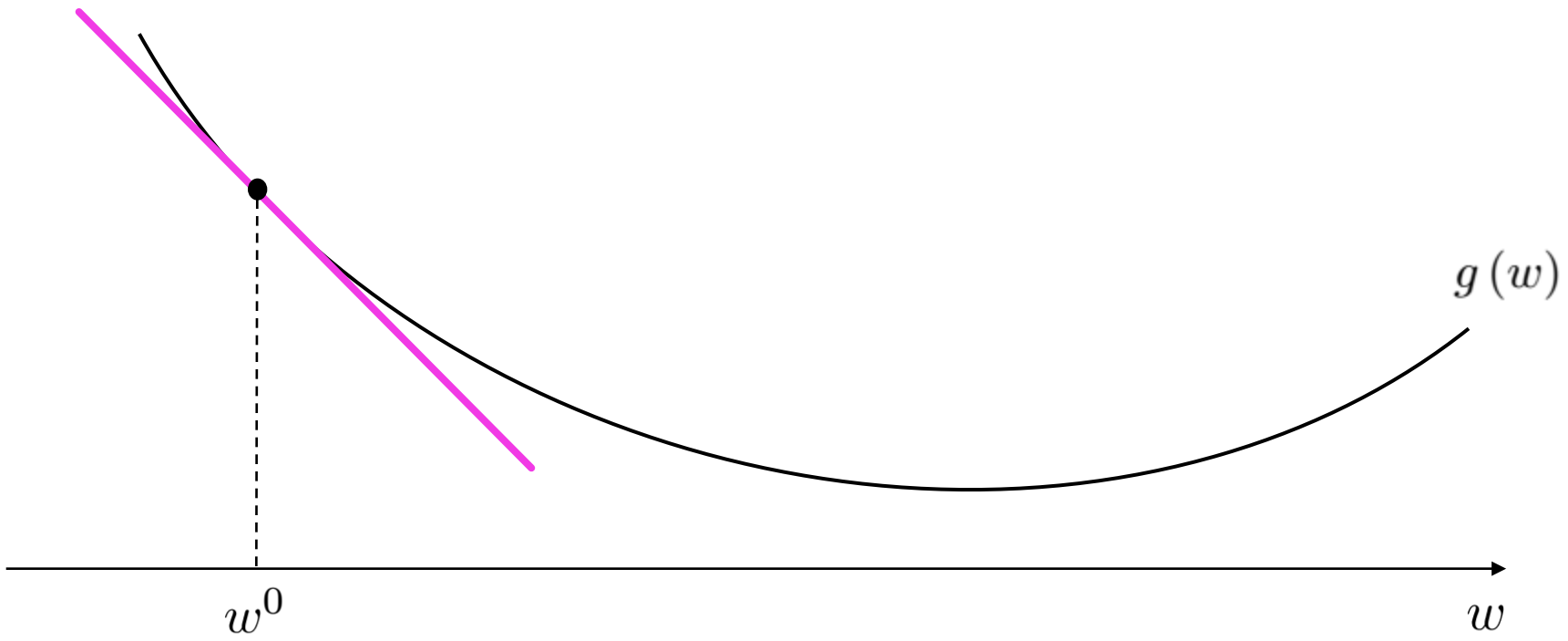
# gradient descent

- ✓ the tangent line looks a whole lot like $g(w)$ near $w^0$
- ✓ traveling downward on line ≈ traveling downward on $g(w)$ near $w^0$

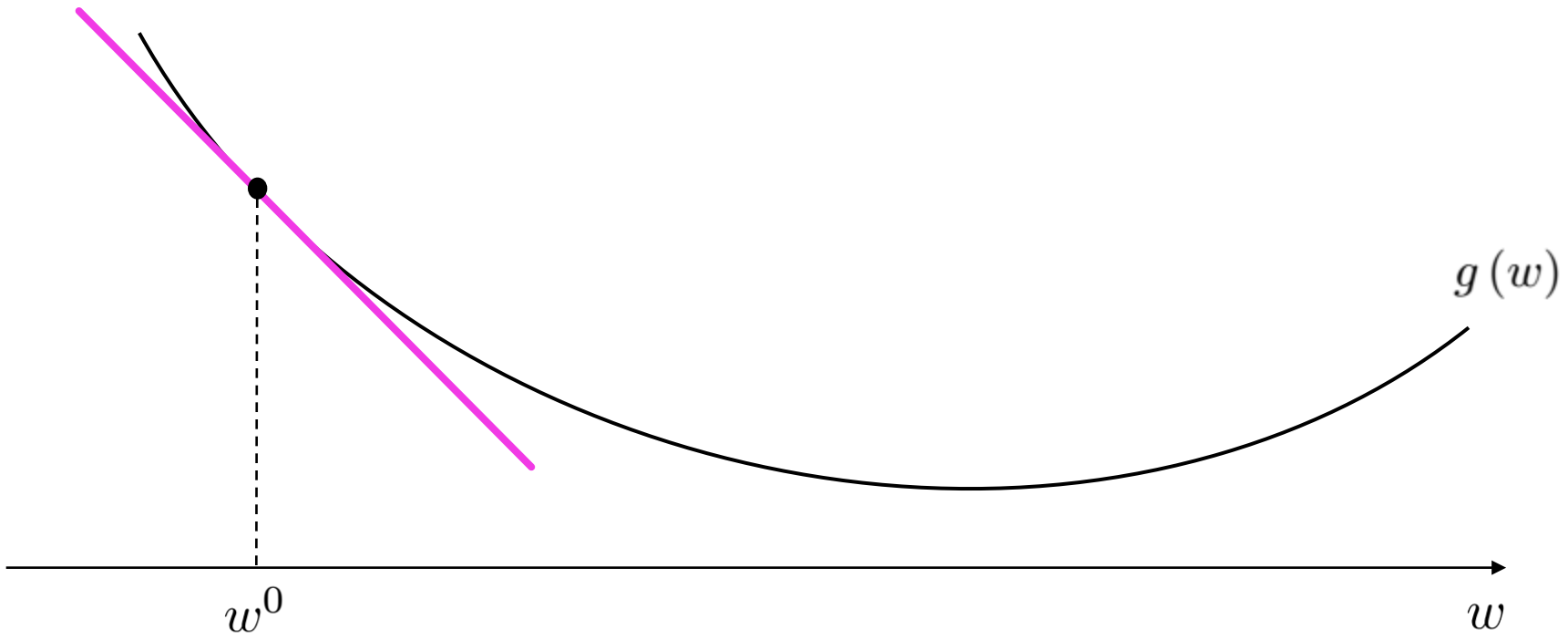


# gradient descent

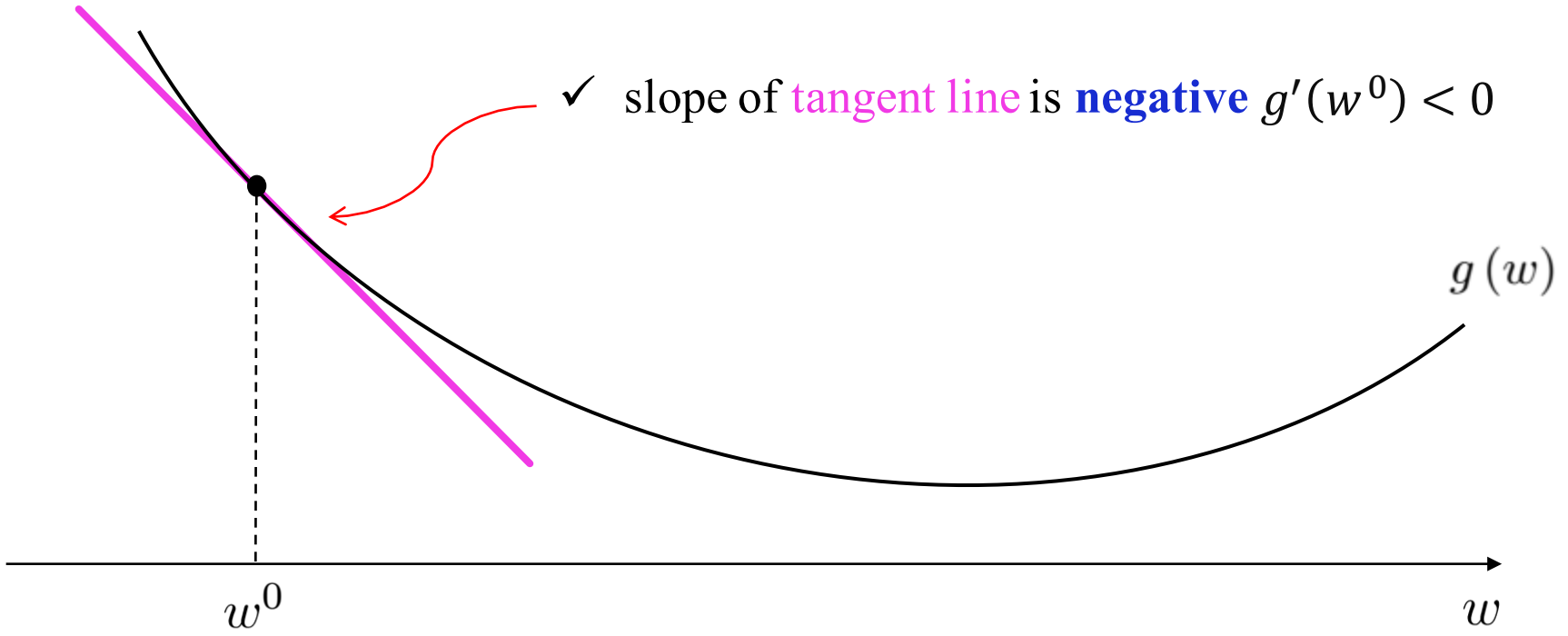
- ✓ the tangent line looks a whole lot like $g(w)$ near $w^0$
- ✓ traveling downward on line ≈ traveling downward on $g(w)$ near $w^0$
- ✓ the downward direction on a line is easily computable



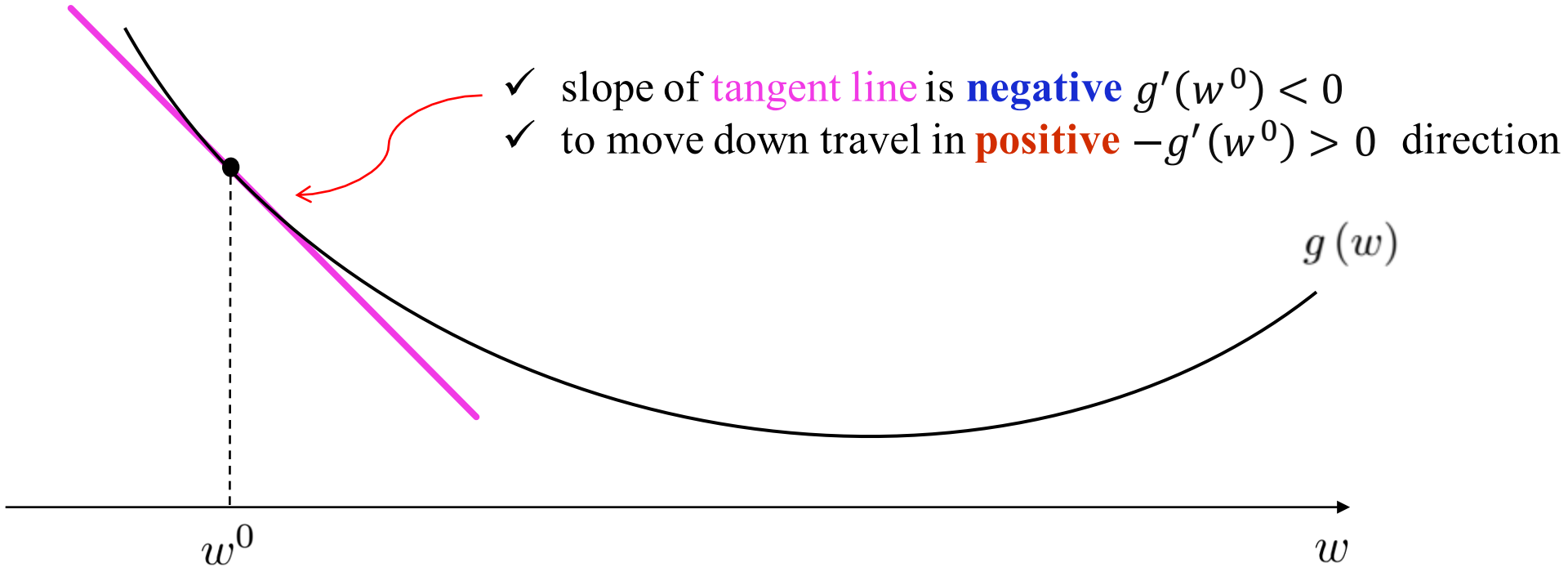
# gradient descent



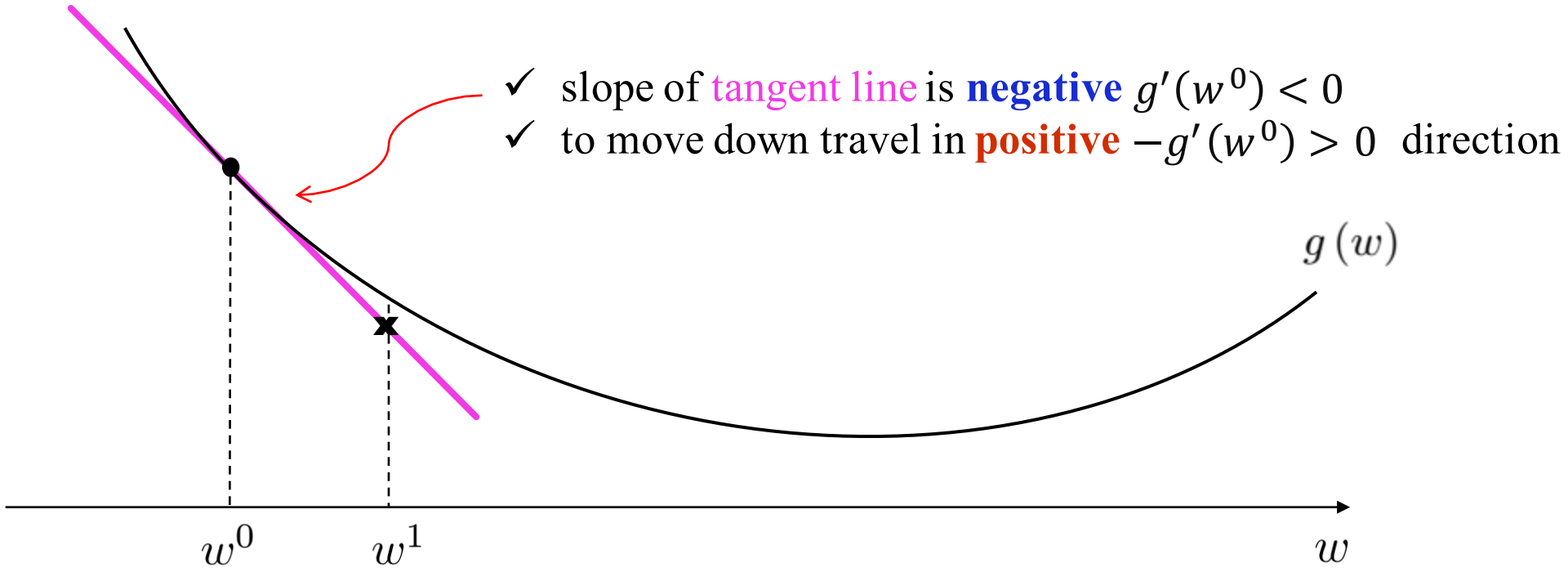
# gradient descent



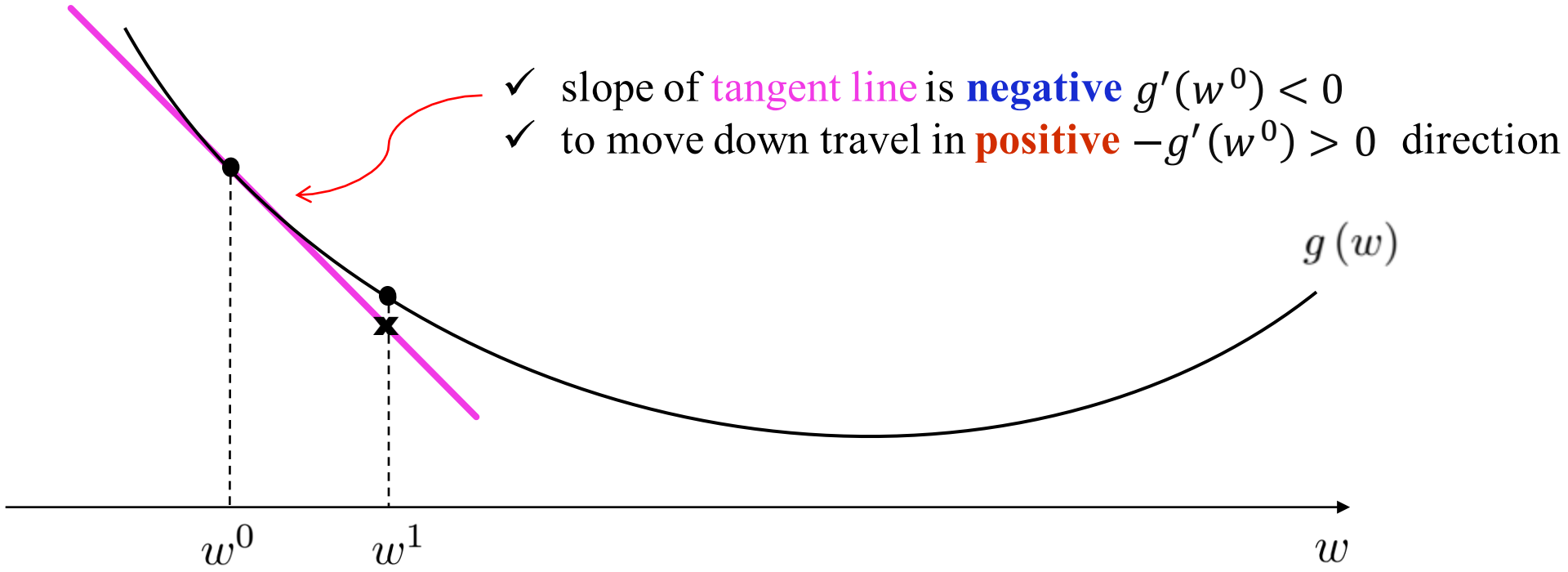
# gradient descent



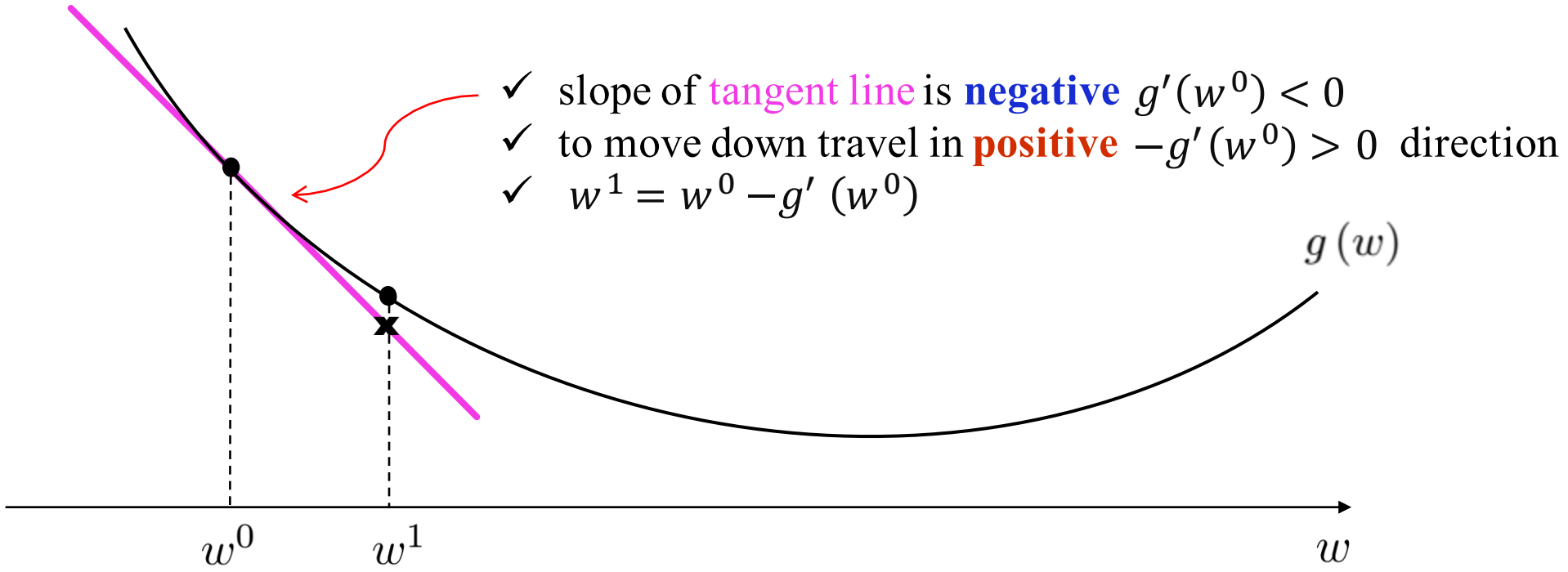
# gradient descent



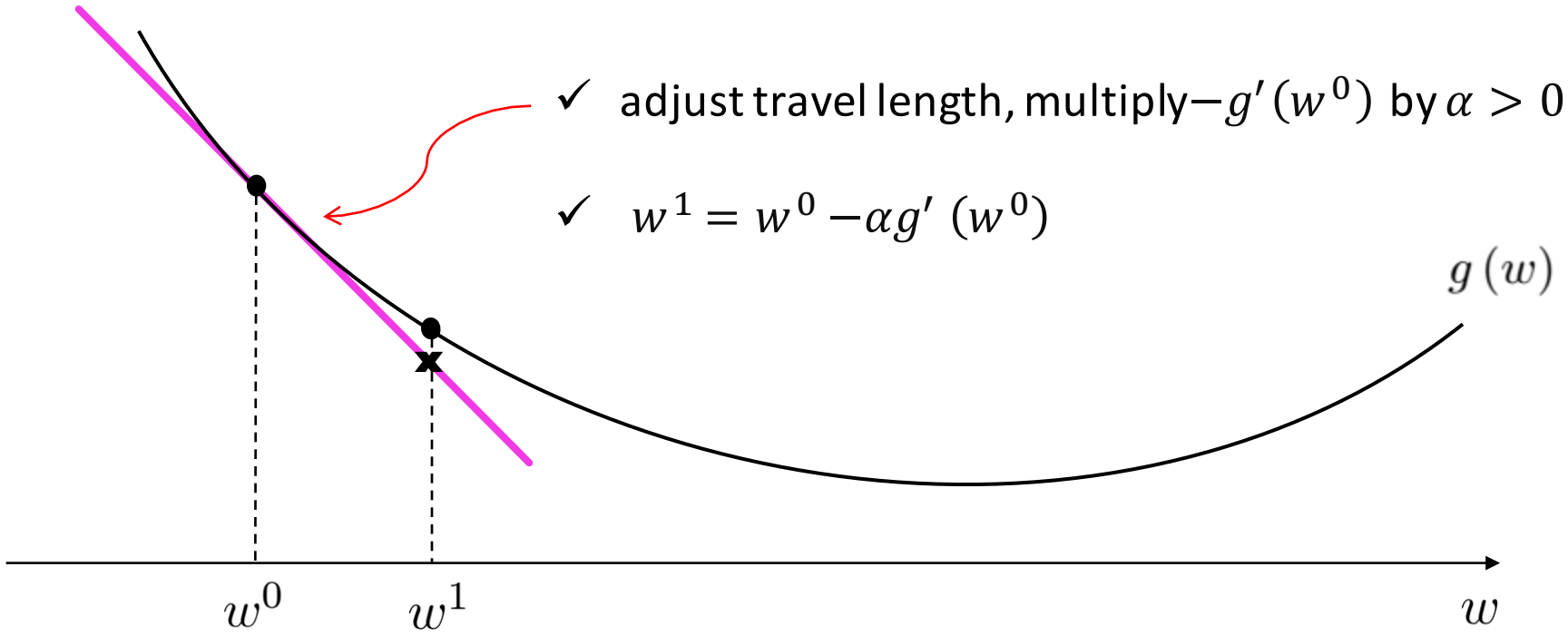
# gradient descent



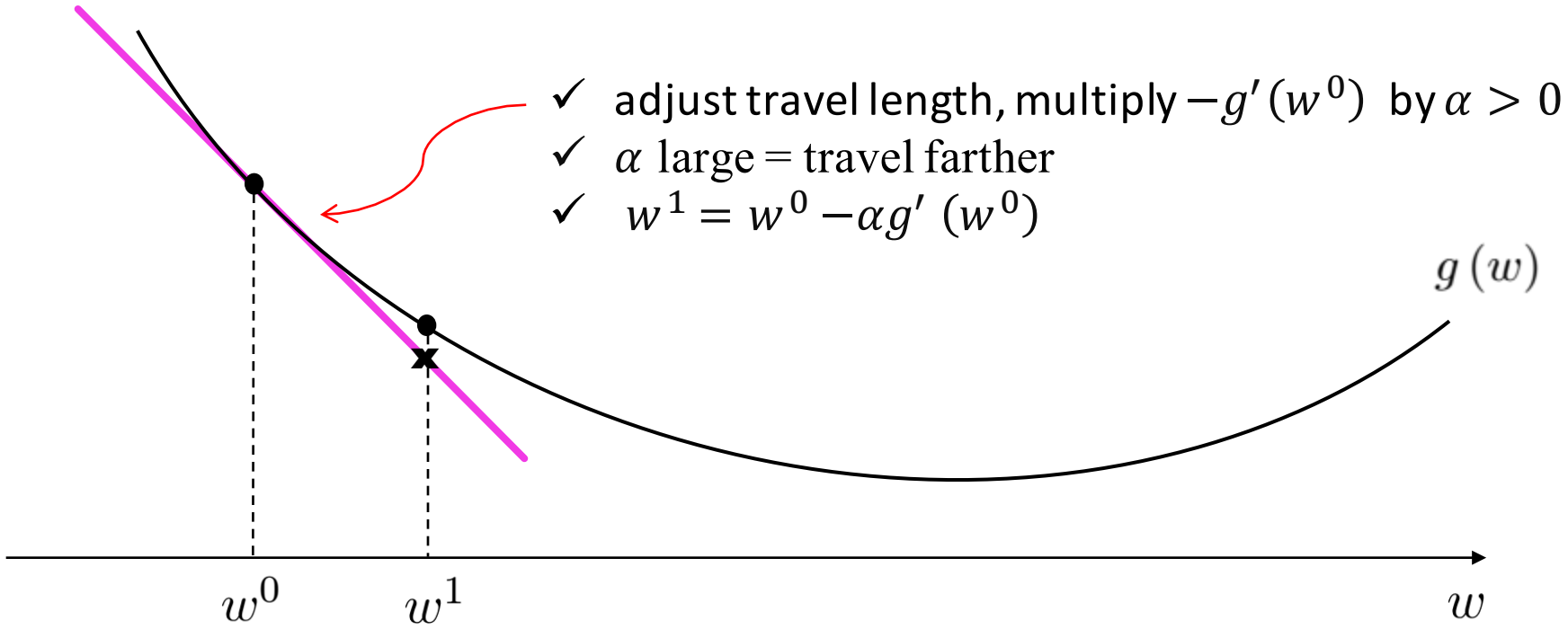
# gradient descent



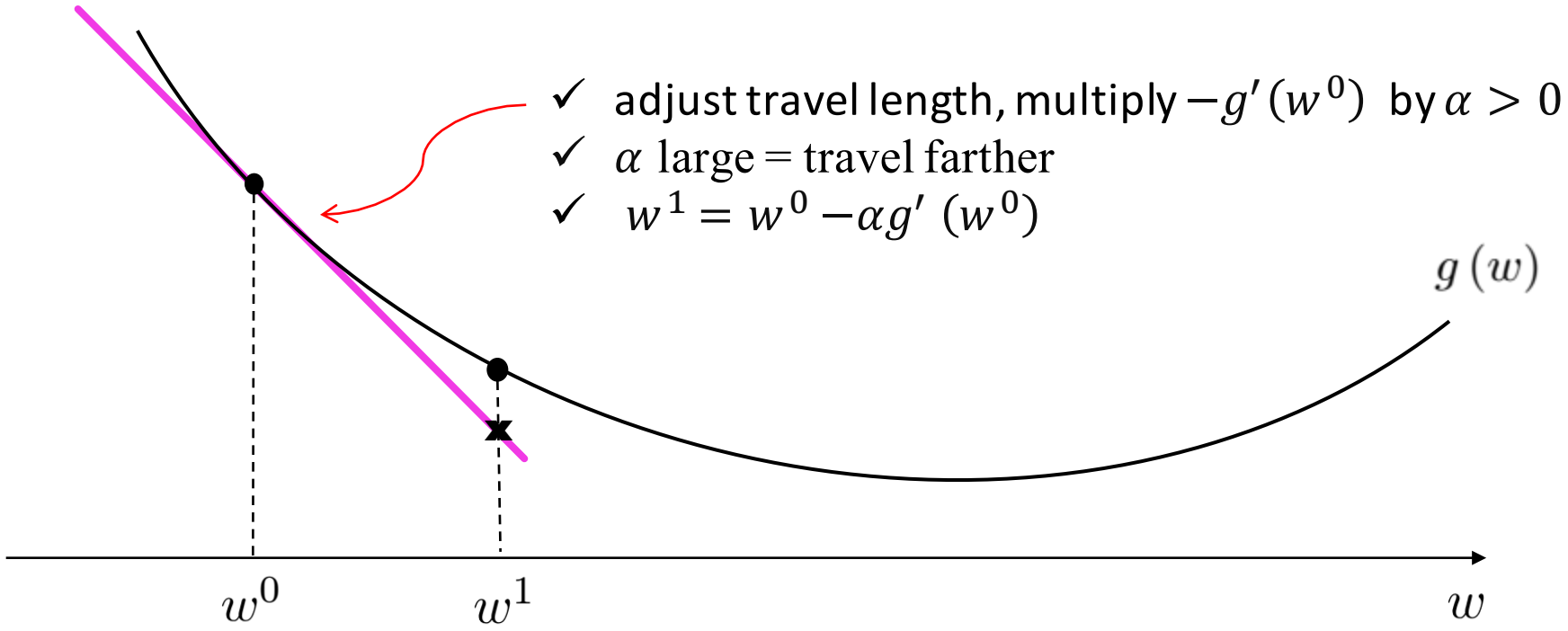
# gradient descent



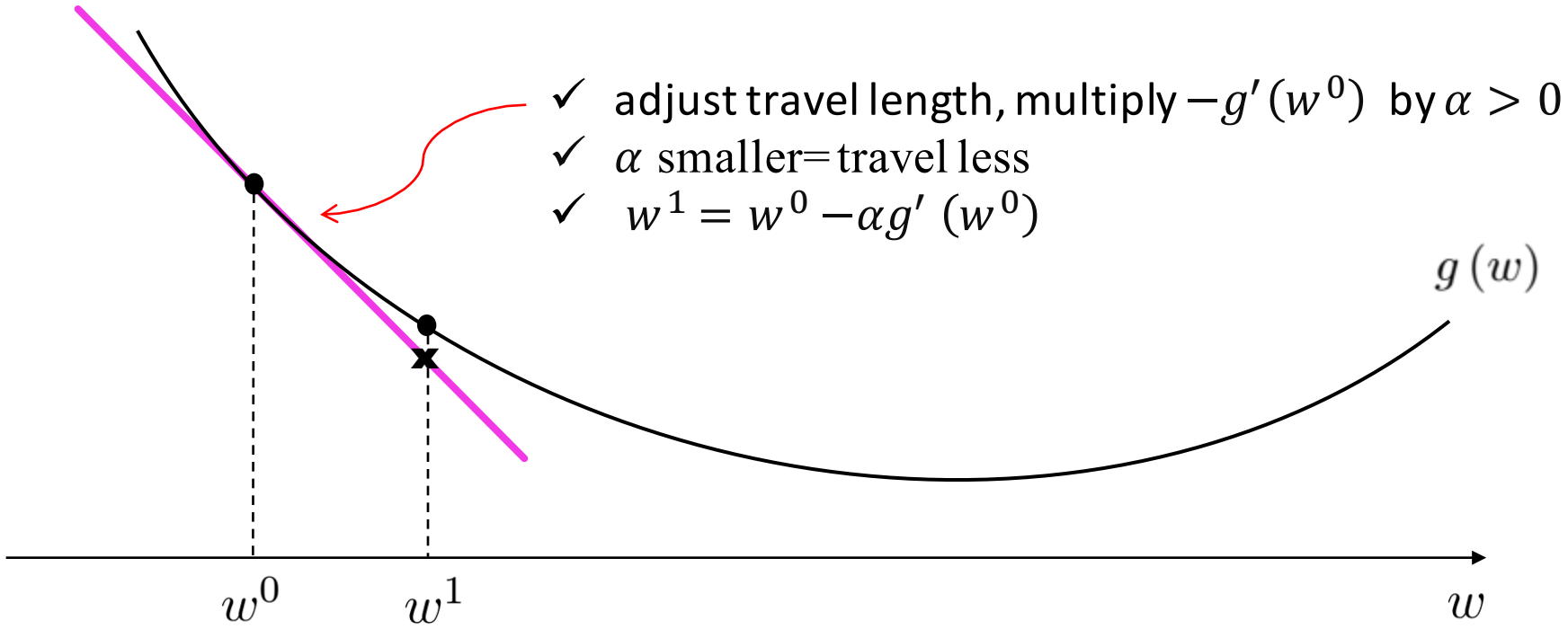
# gradient descent



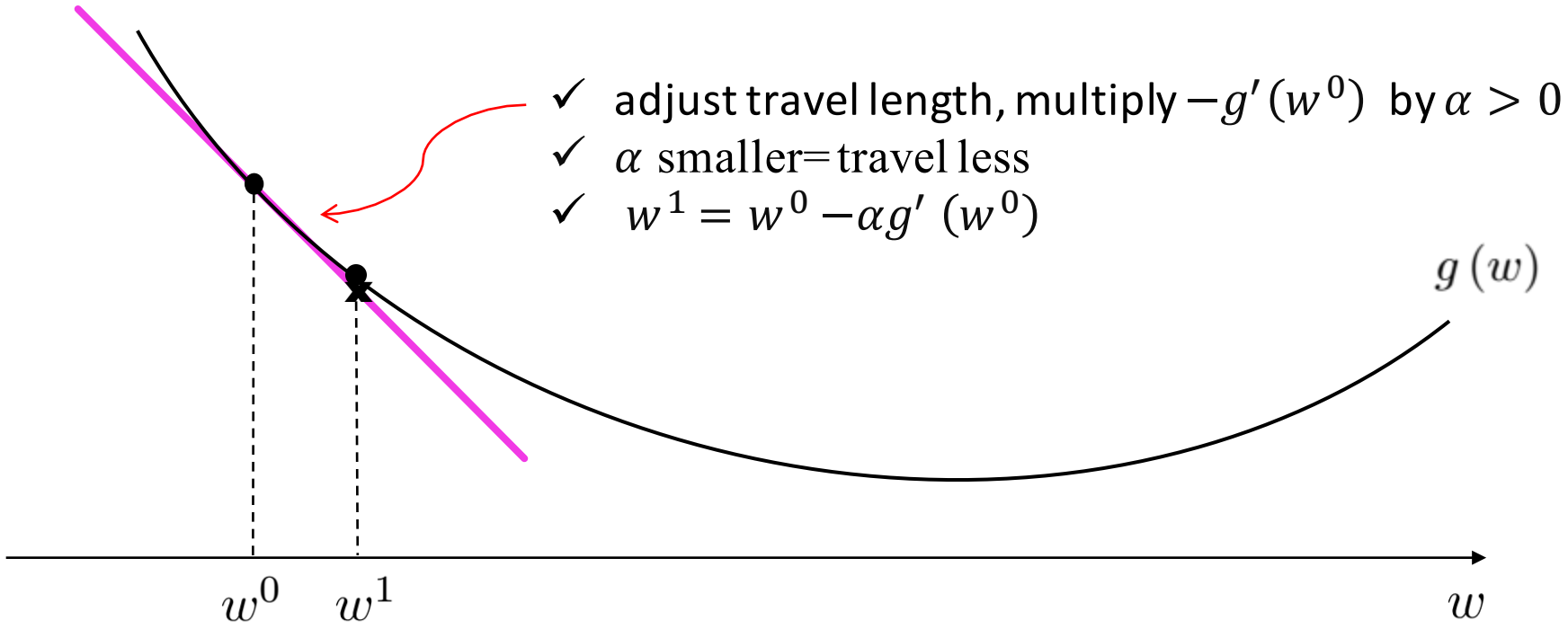
# gradient descent



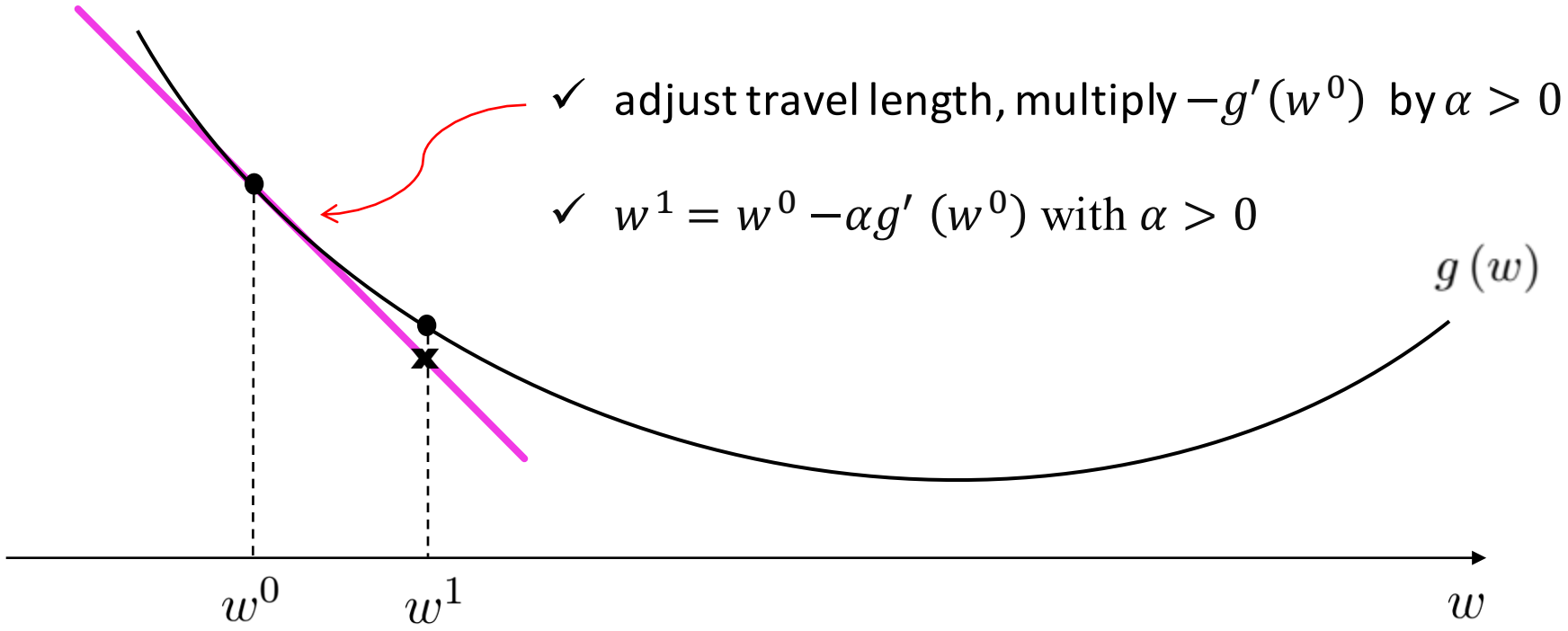
# gradient descent



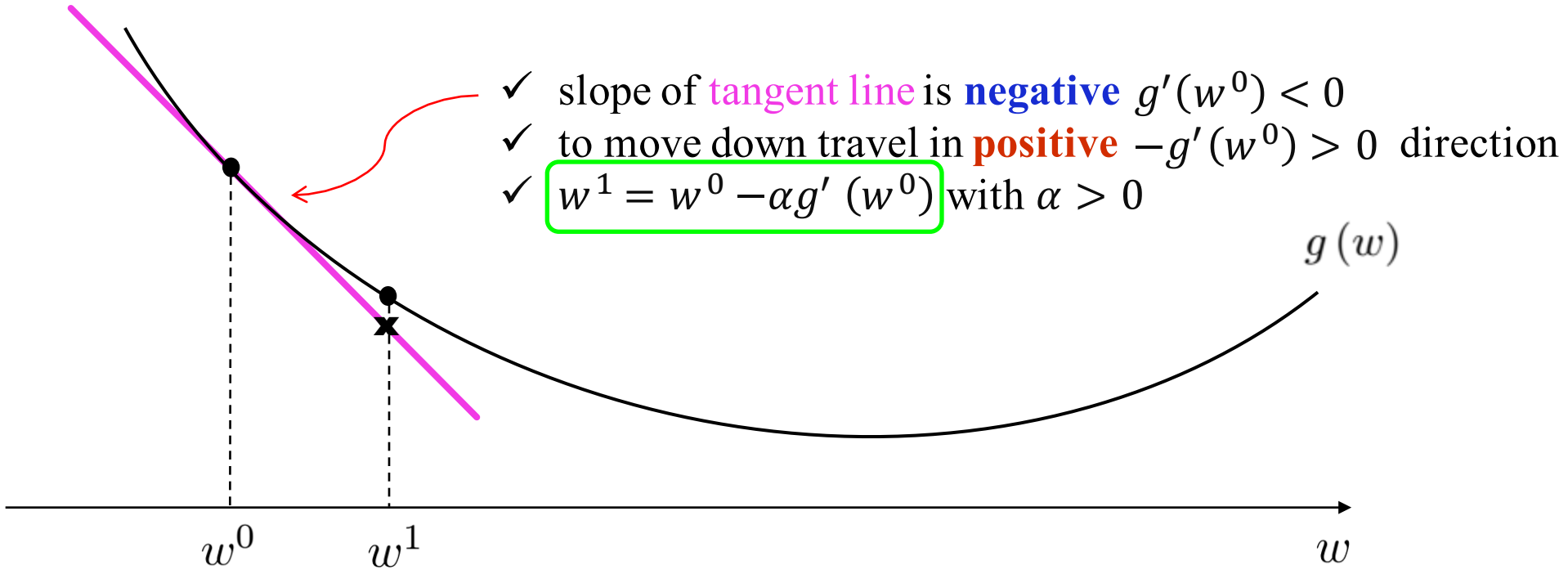
# gradient descent



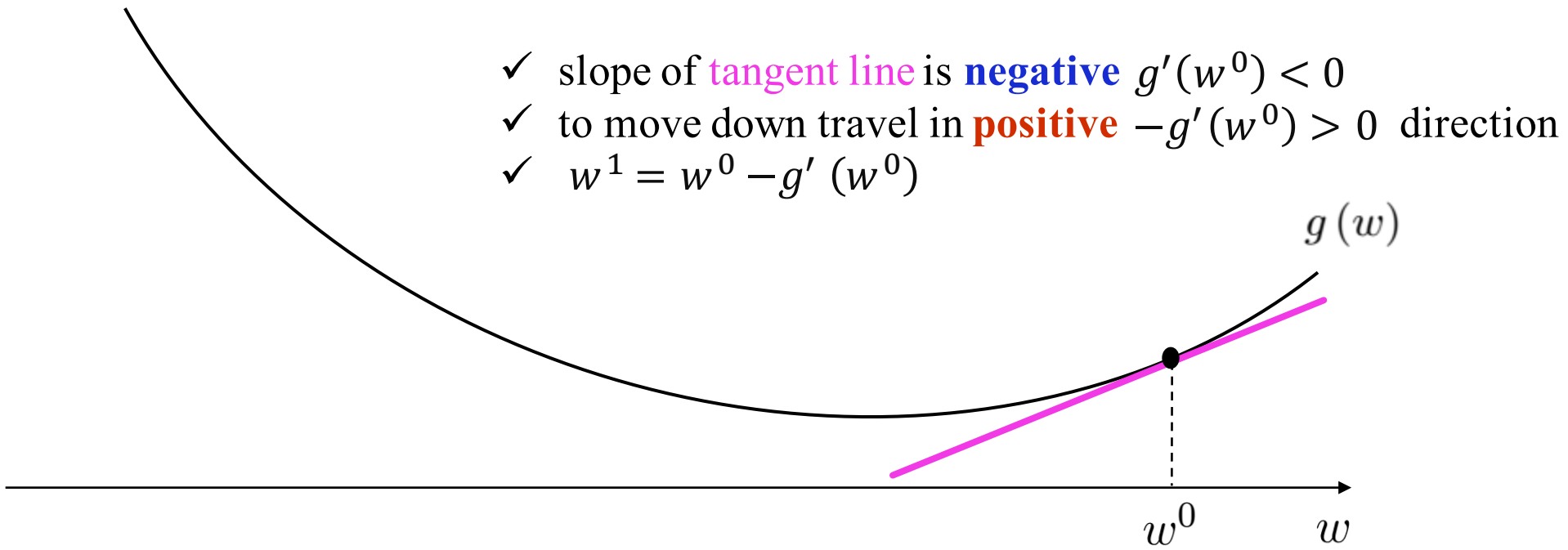
# gradient descent



# gradient descent

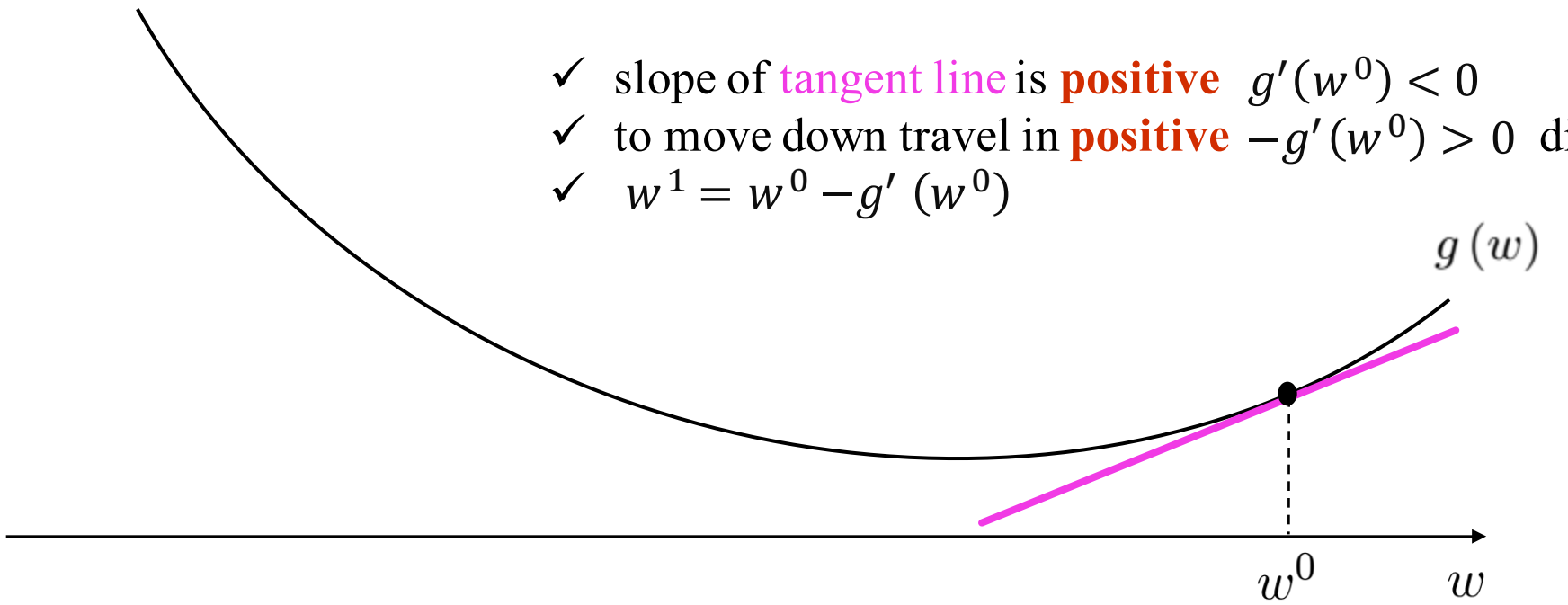


# gradient descent



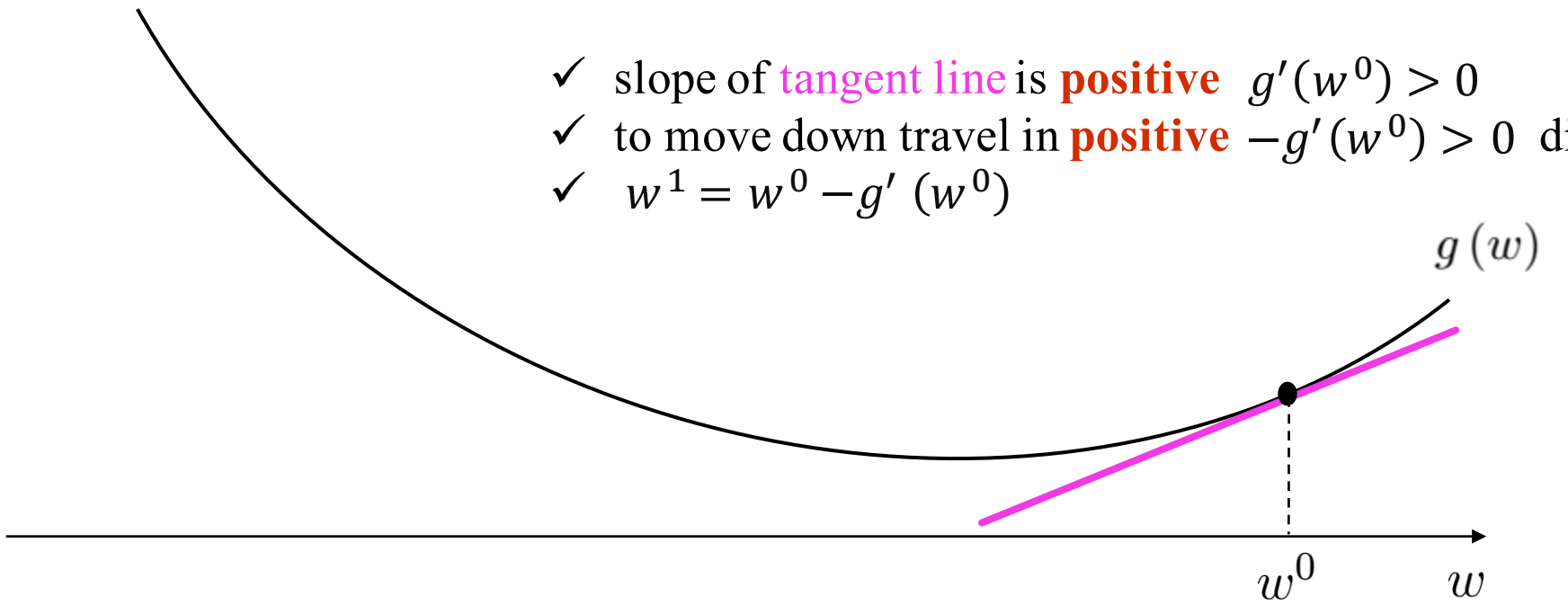
# gradient descent

- ✓ slope of tangent line is **positive** $g'(w^0) < 0$
- ✓ to move down travel in **positive** $-g'(w^0) > 0$ direction
- ✓ $w^1 = w^0 - g'(w^0)$

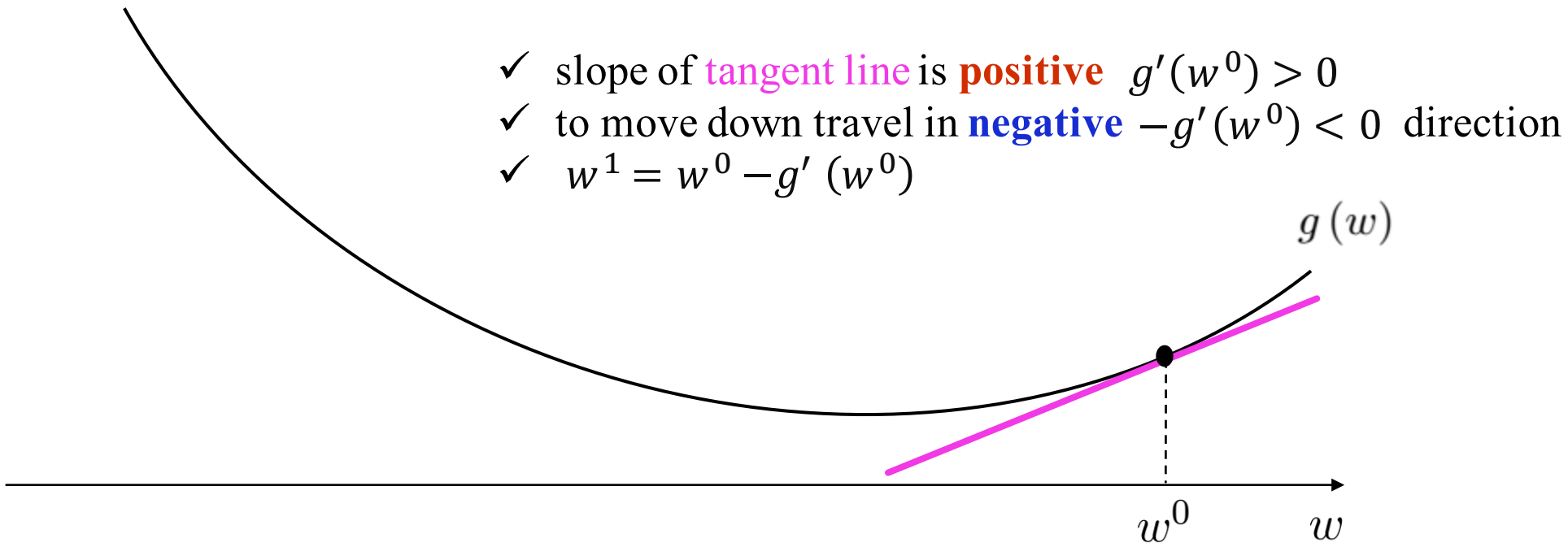


# gradient descent

- ✓ slope of tangent line is **positive** $g'(w^0) > 0$
- ✓ to move down travel in **positive** $-g'(w^0) > 0$ direction
- ✓ $w^1 = w^0 - g'(w^0)$

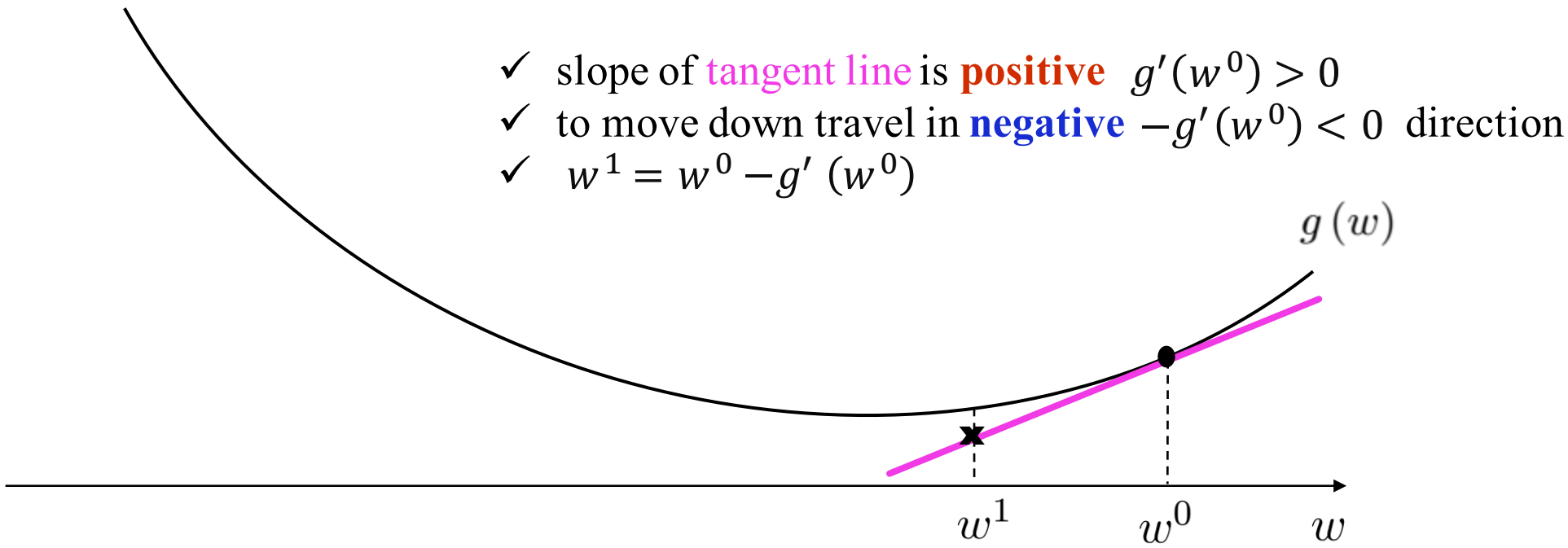


# gradient descent



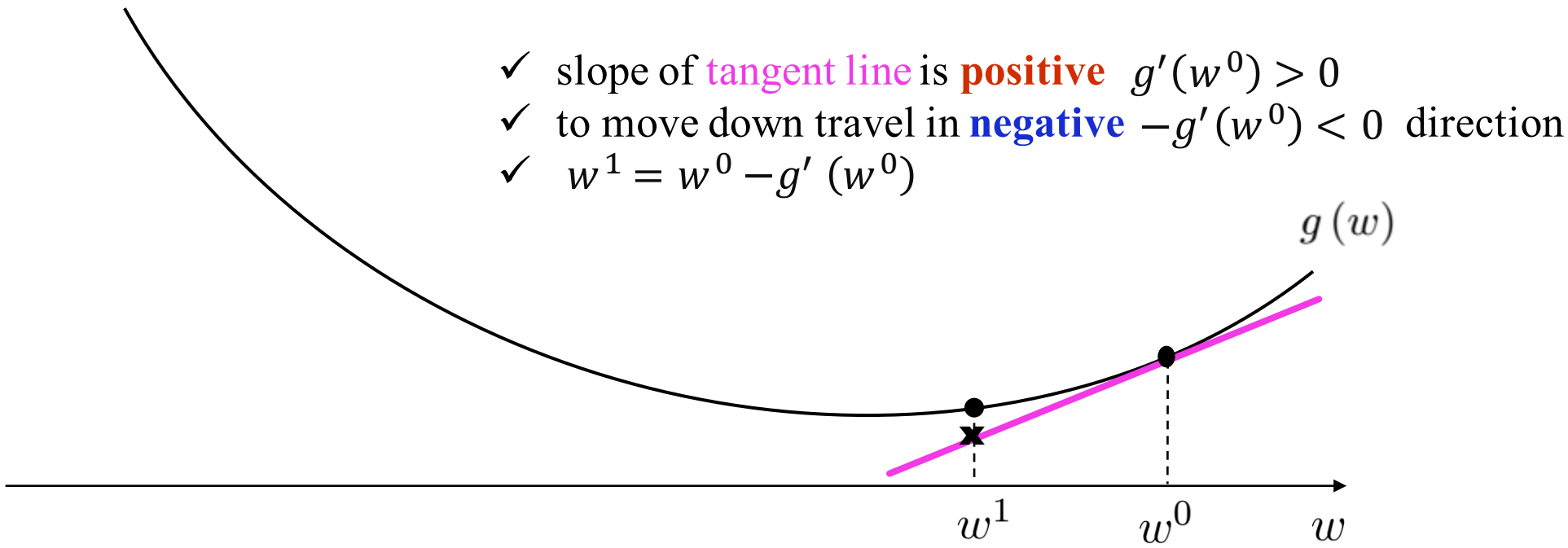
# gradient descent

- ✓ slope of tangent line is **positive** $g'(w^0) > 0$
- ✓ to move down travel in **negative** $-g'(w^0) < 0$ direction
- ✓ $w^1 = w^0 - g'(w^0)$



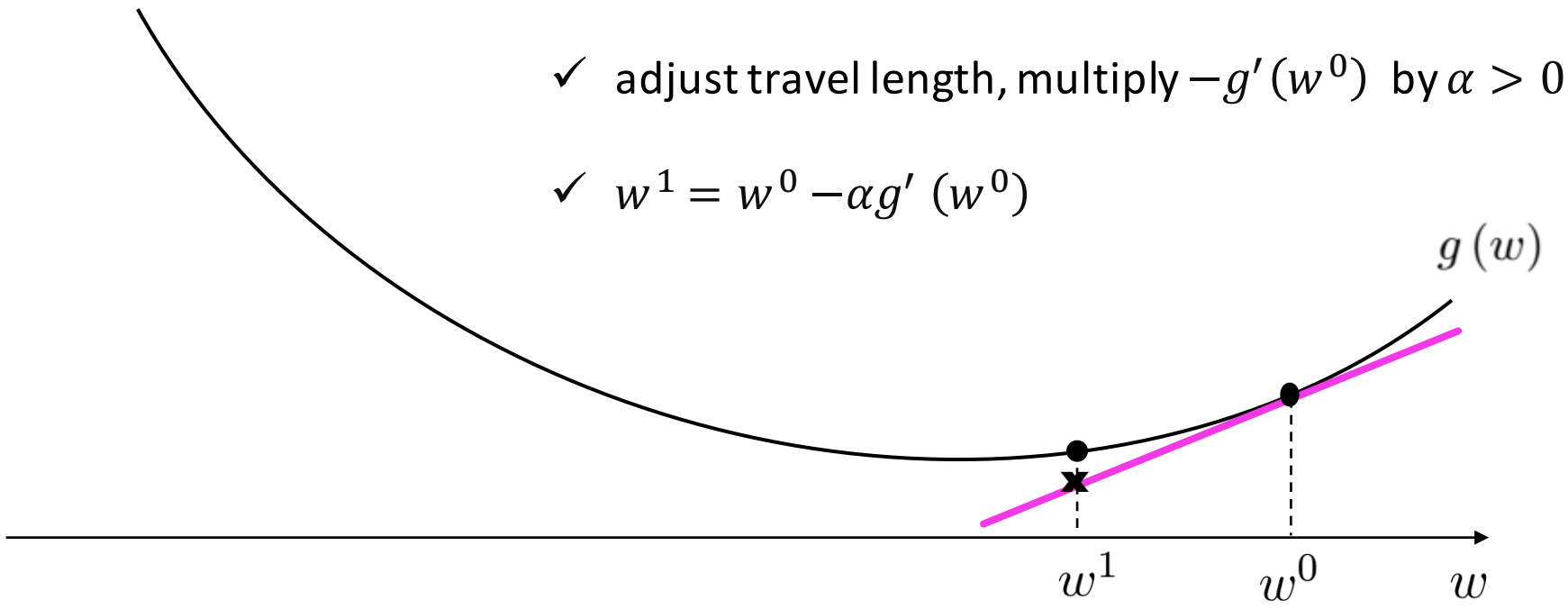
# gradient descent

- ✓ slope of tangent line is **positive** $g'(w^0) > 0$
- ✓ to move down travel in **negative** $-g'(w^0) < 0$ direction
- ✓ $w^1 = w^0 - g'(w^0)$



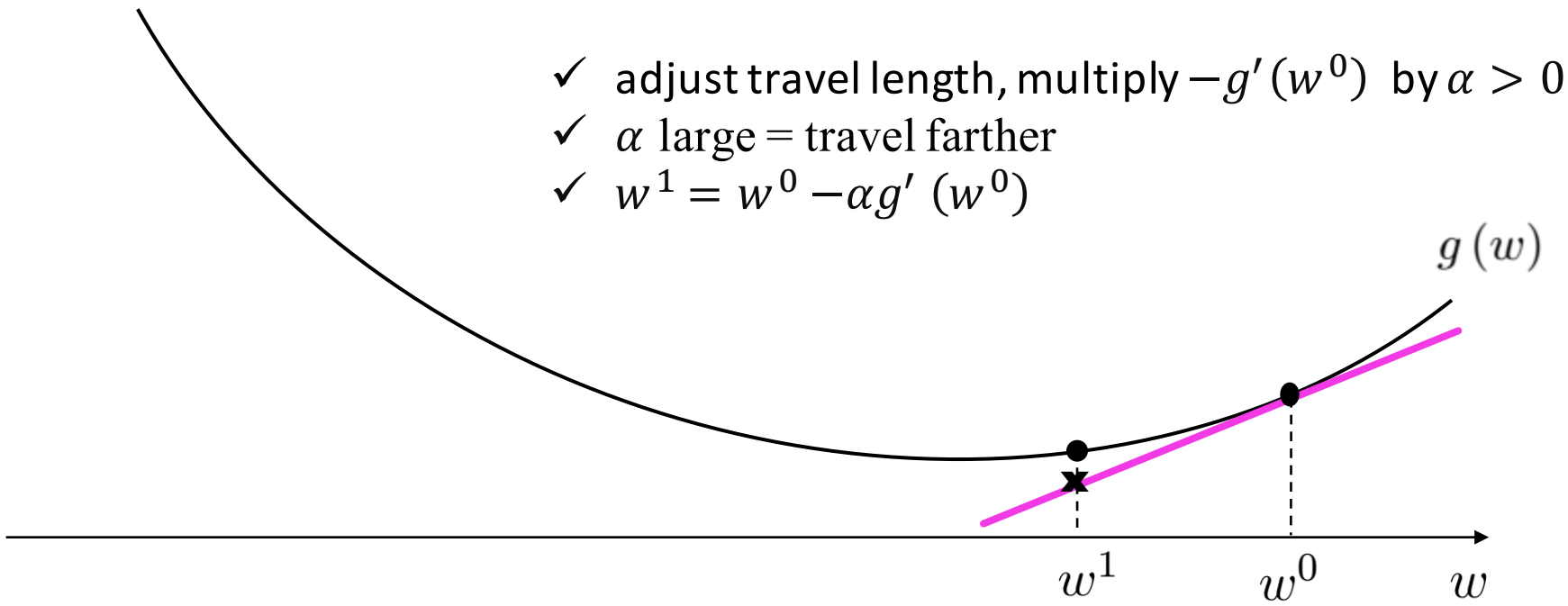
# gradient descent

- ✓ adjust travel length, multiply $-g'(w^0)$ by $\alpha > 0$
- ✓ $w^1 = w^0 - \alpha g'(w^0)$



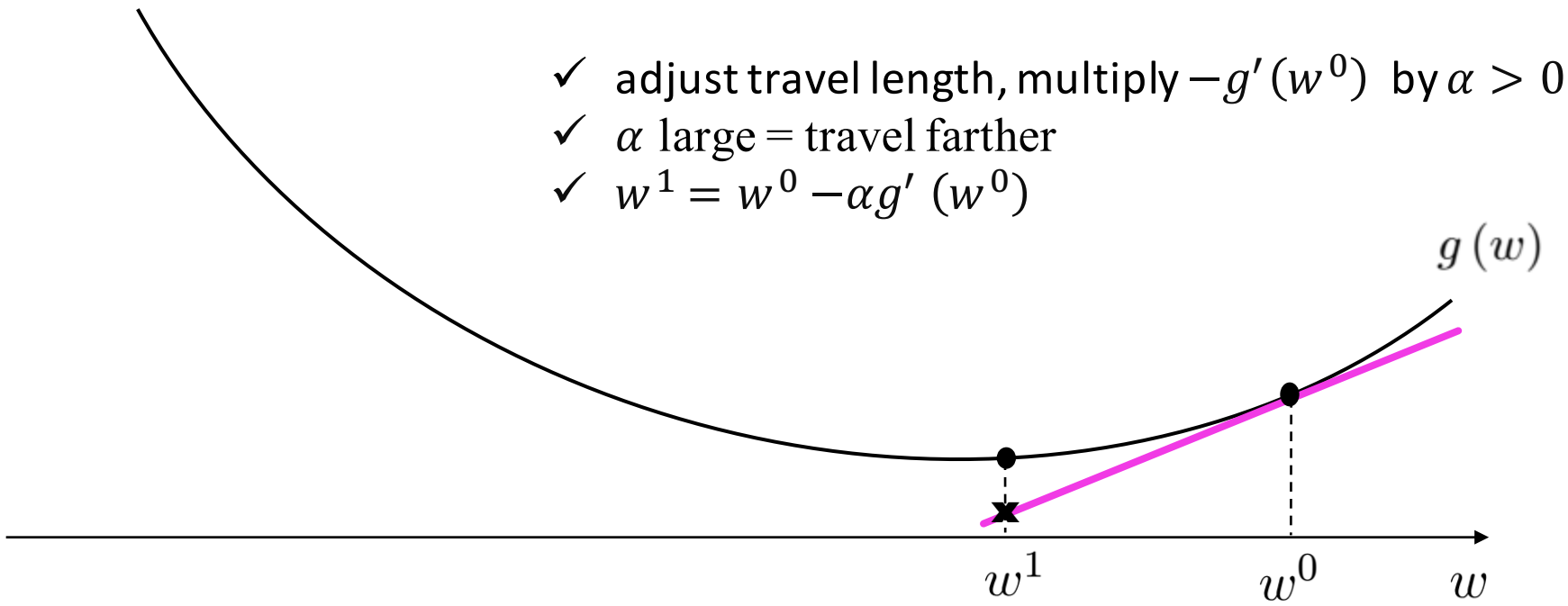
# gradient descent

- ✓ adjust travel length, multiply $-g'(w^0)$ by $\alpha > 0$
- ✓ $\alpha$ large = travel farther
- ✓ $w^1 = w^0 - \alpha g'(w^0)$



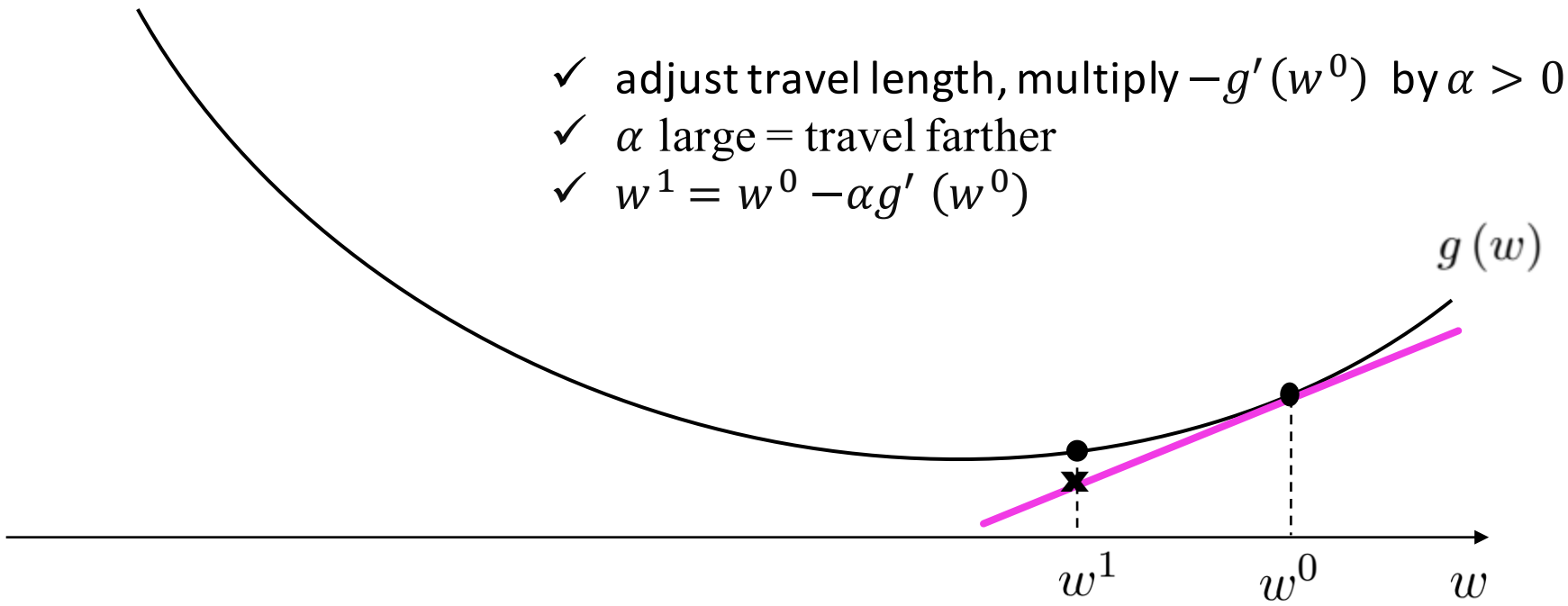
# gradient descent

- ✓ adjust travel length, multiply $-g'(w^0)$ by $\alpha > 0$
- ✓ $\alpha$ large = travel farther
- ✓ $w^1 = w^0 - \alpha g'(w^0)$

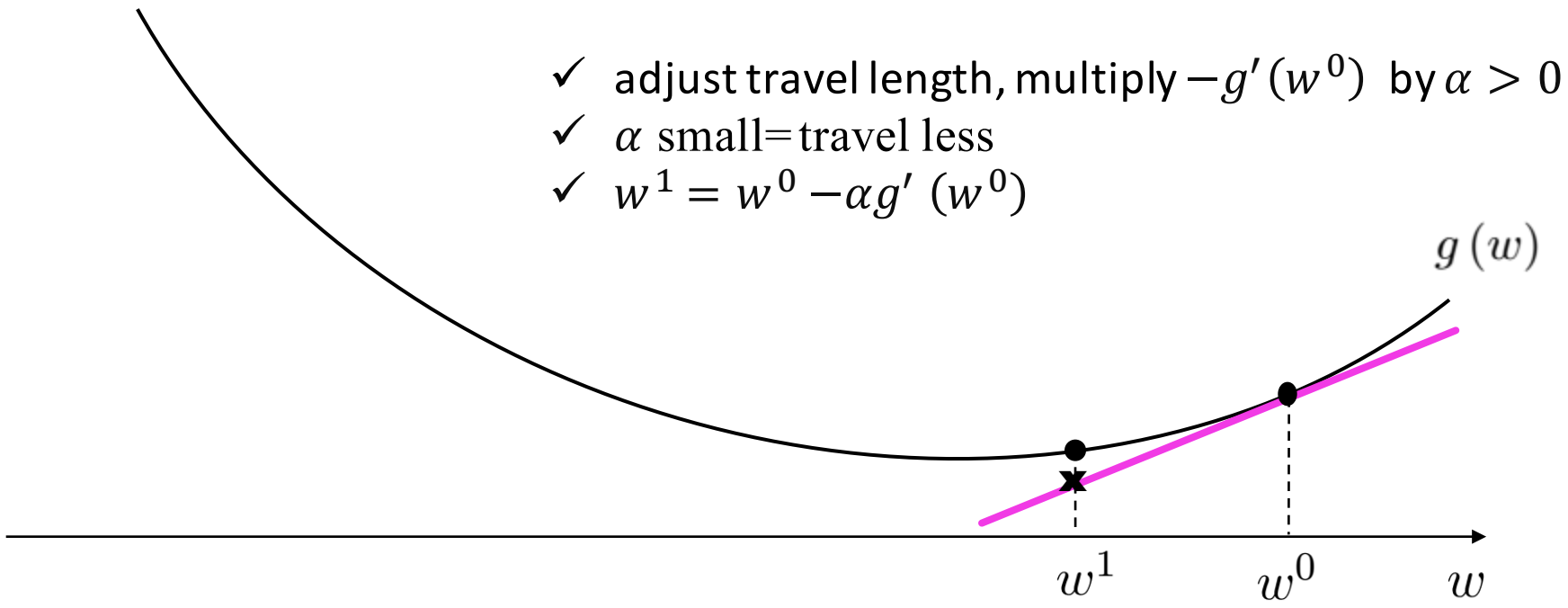


# gradient descent

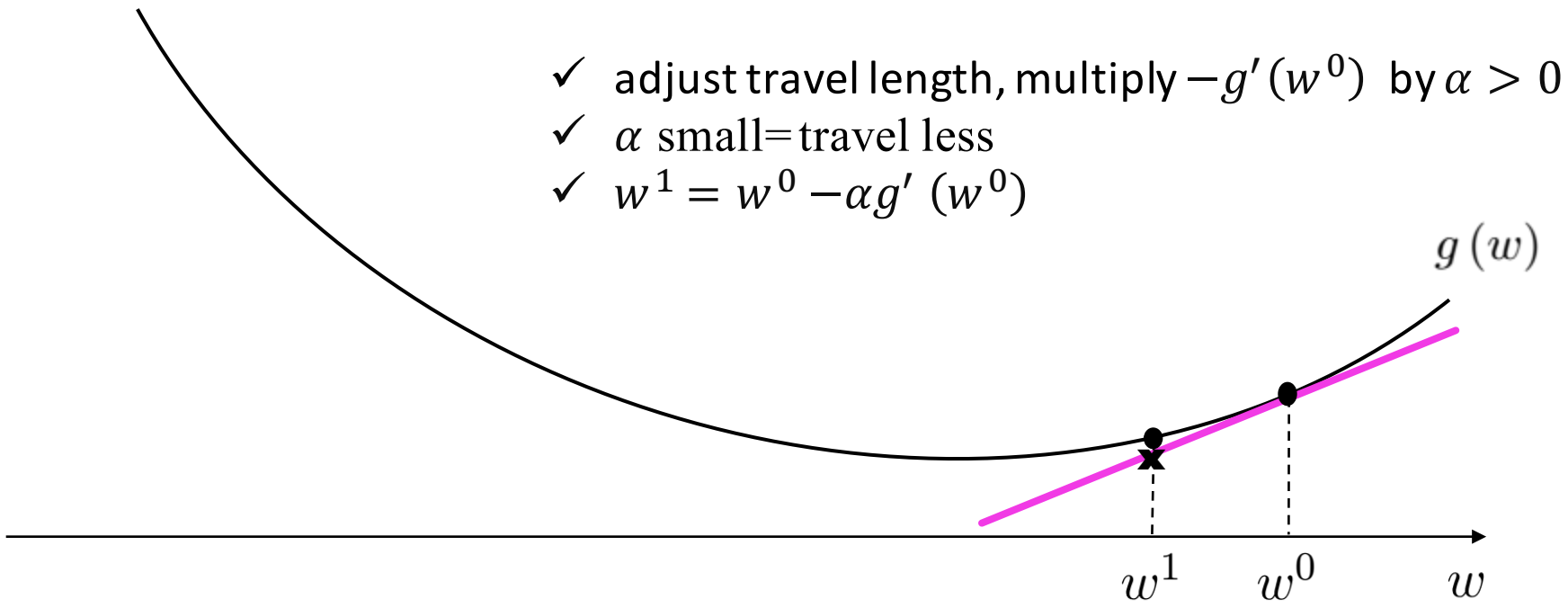
- ✓ adjust travel length, multiply $-g'(w^0)$ by $\alpha > 0$
- ✓ $\alpha$ large = travel farther
- ✓ $w^1 = w^0 - \alpha g'(w^0)$



# gradient descent

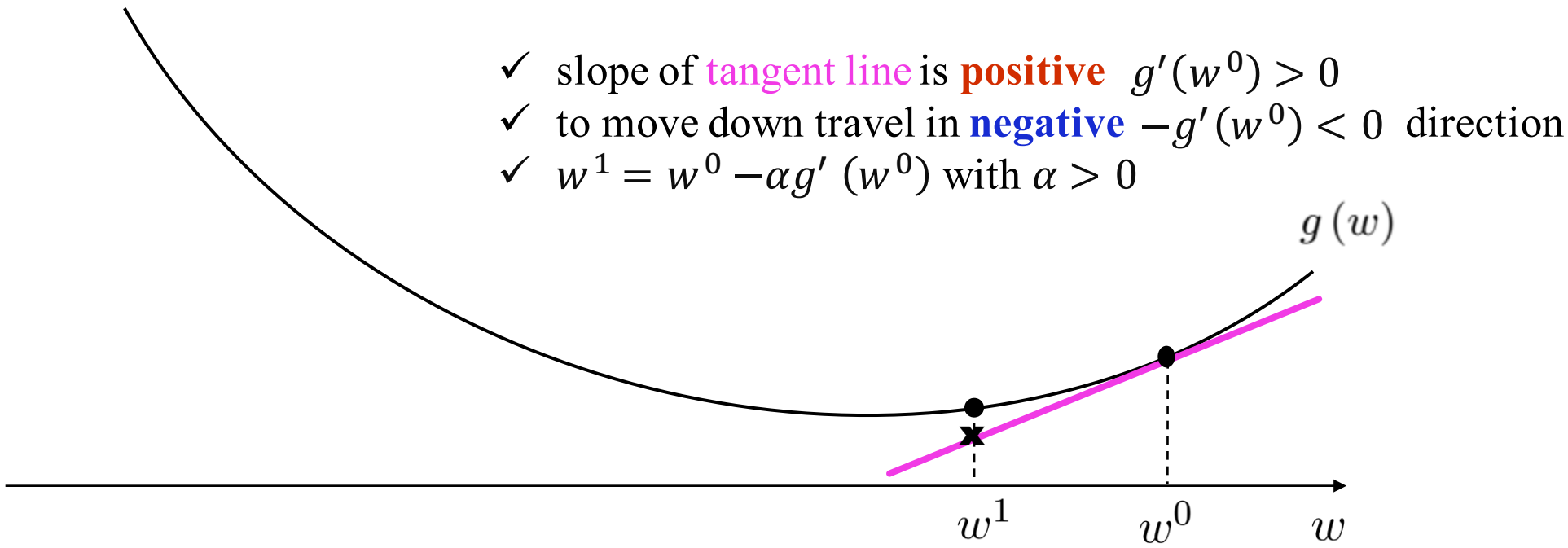


# gradient descent



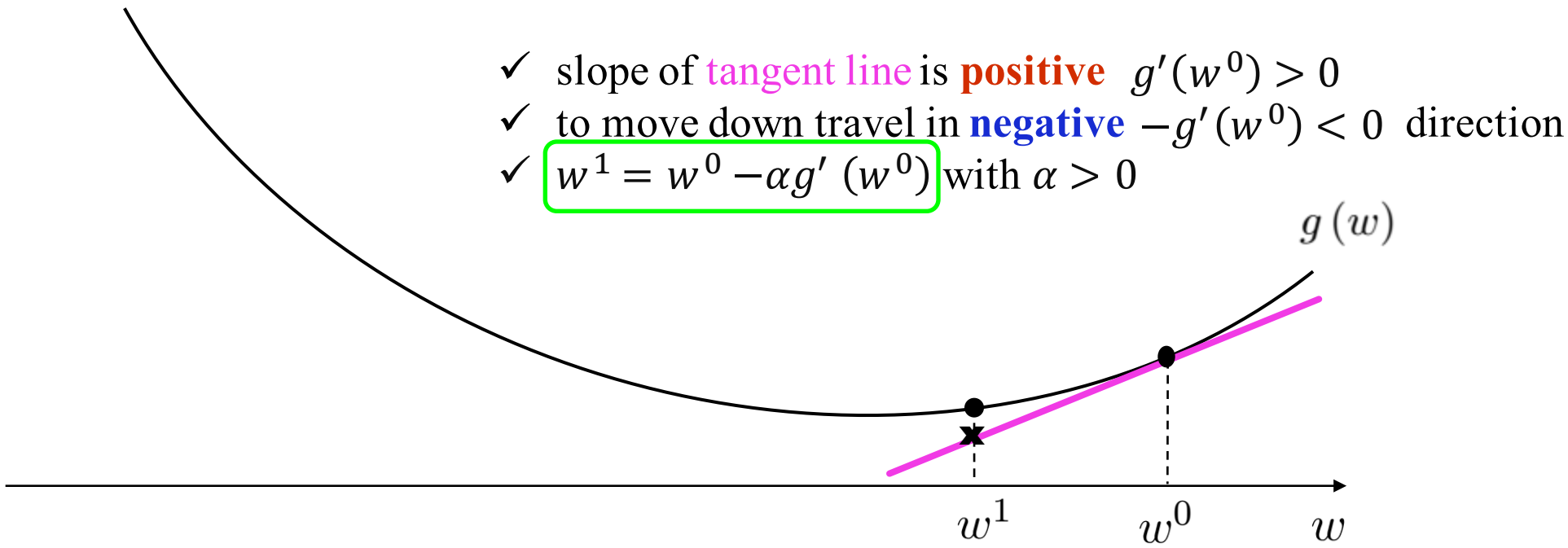
# gradient descent

- ✓ slope of tangent line is **positive** $g'(w^0) > 0$
- ✓ to move down travel in **negative** $-g'(w^0) < 0$ direction
- ✓ $w^1 = w^0 - \alpha g'(w^0)$ with $\alpha > 0$



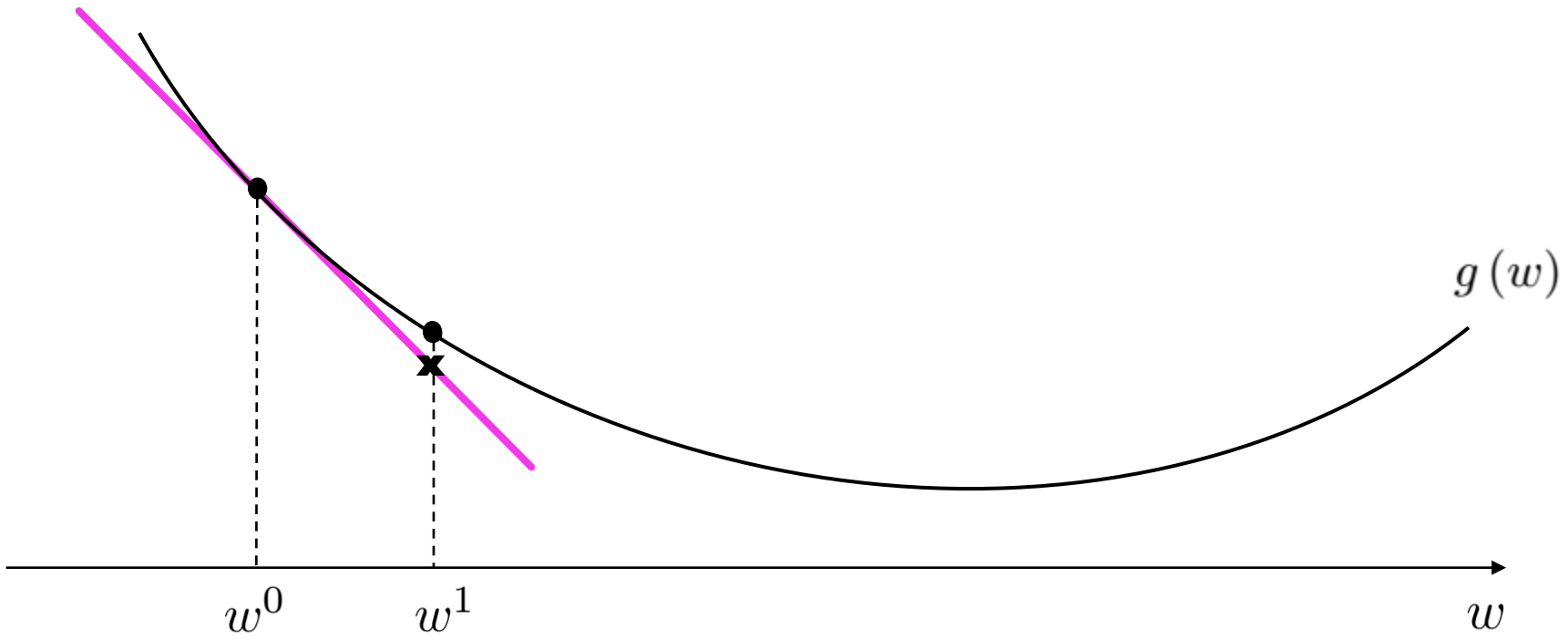
# gradient descent

- ✓ slope of tangent line is **positive** $g'(w^0) > 0$
- ✓ to move down travel in **negative** $-g'(w^0) < 0$ direction
- ✓ $w^1 = w^0 - \alpha g'(w^0)$ with $\alpha > 0$



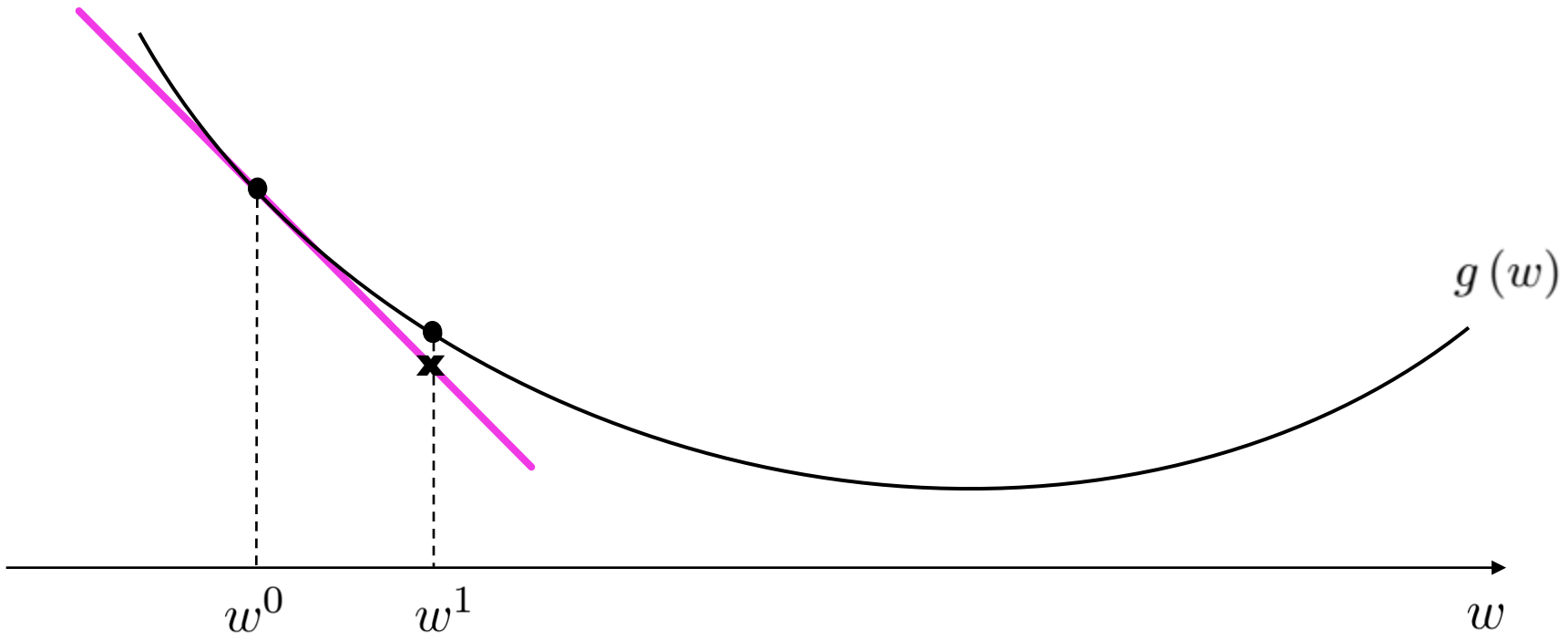
# gradient descent

- ✓ traveling downward on line ≈ traveling downward on $g(w)$ near $w^0$
- ✓ the downward direction on a line is easily computable



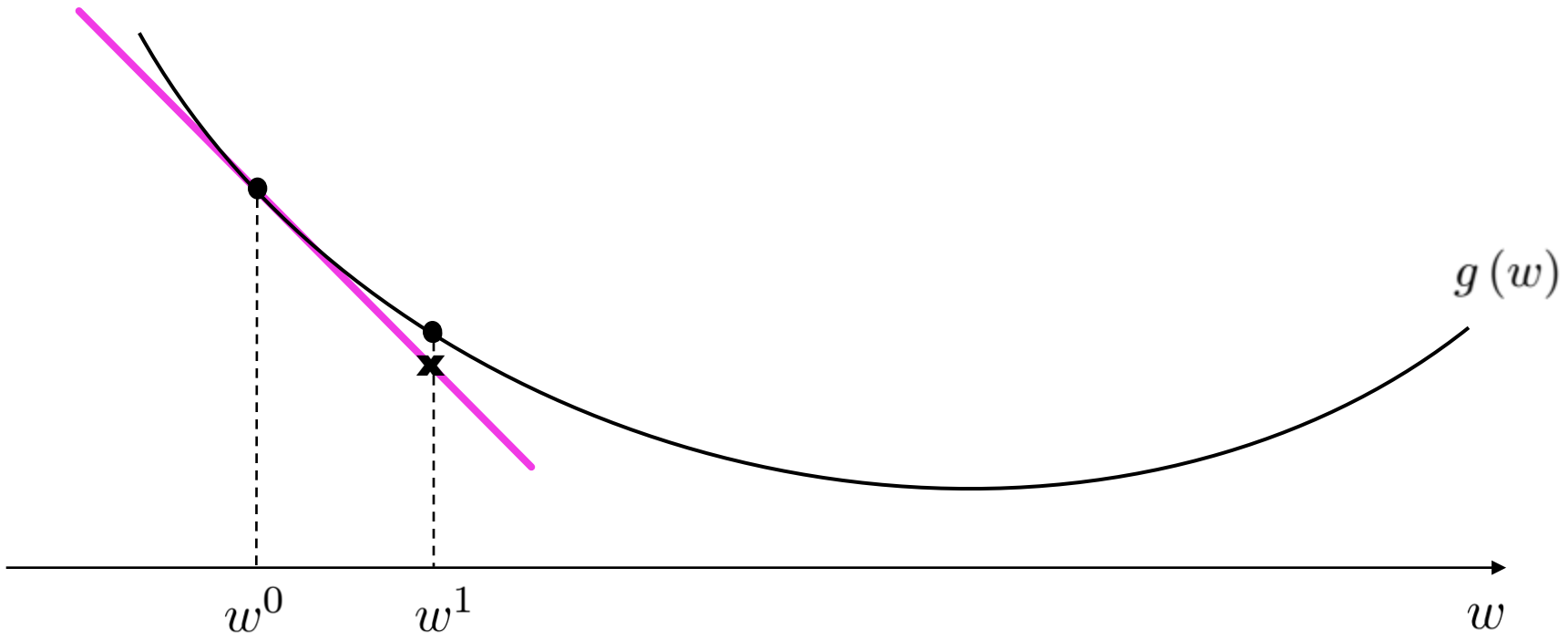
# gradient descent

- ✓ traveling downward on line ≈ traveling downward on $g(w)$ near $w^0$
- ✓ the downward direction on a line is **always** $-g'(w^0)$



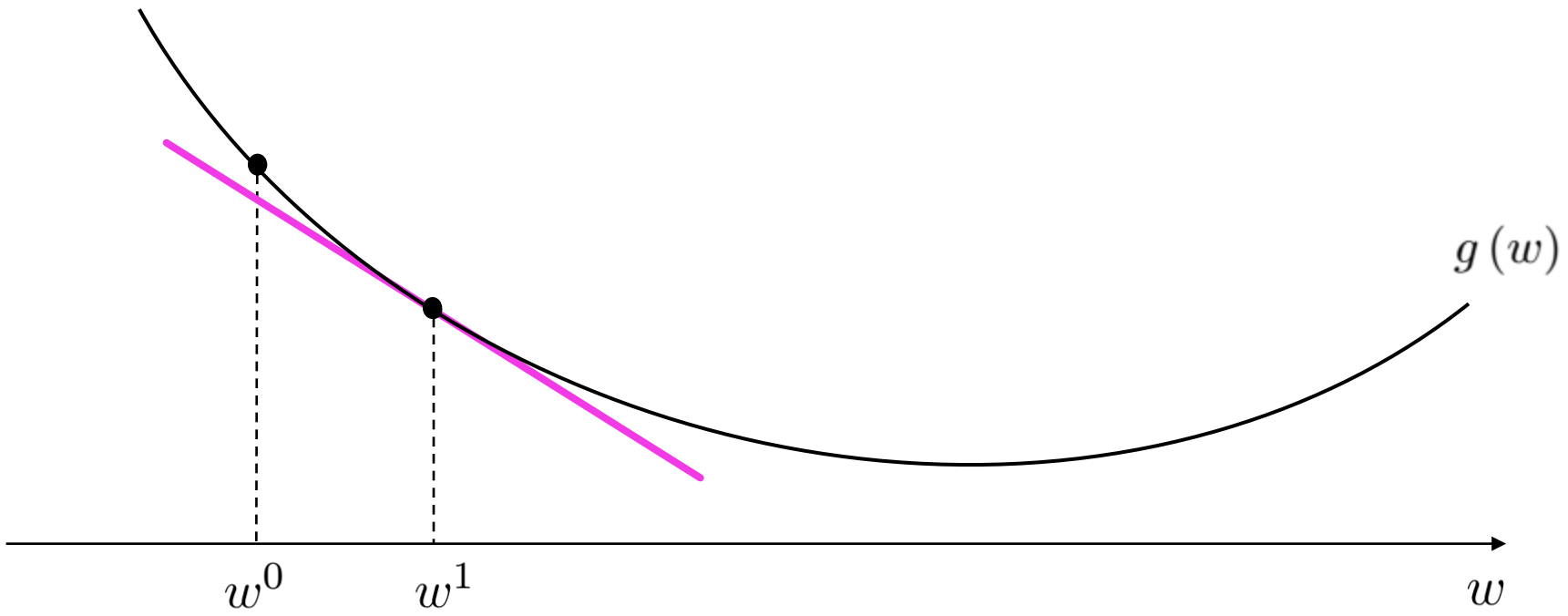
# gradient descent

- ✓ traveling downward on line $\approx$ traveling downward on $g(w)$ near $w^0$
- ✓ the downward direction on a line is **always** $-g'(w^0)$
- ✓ move down with $w^1 = w^0 - \alpha g'(w^0)$ with $\alpha > 0$



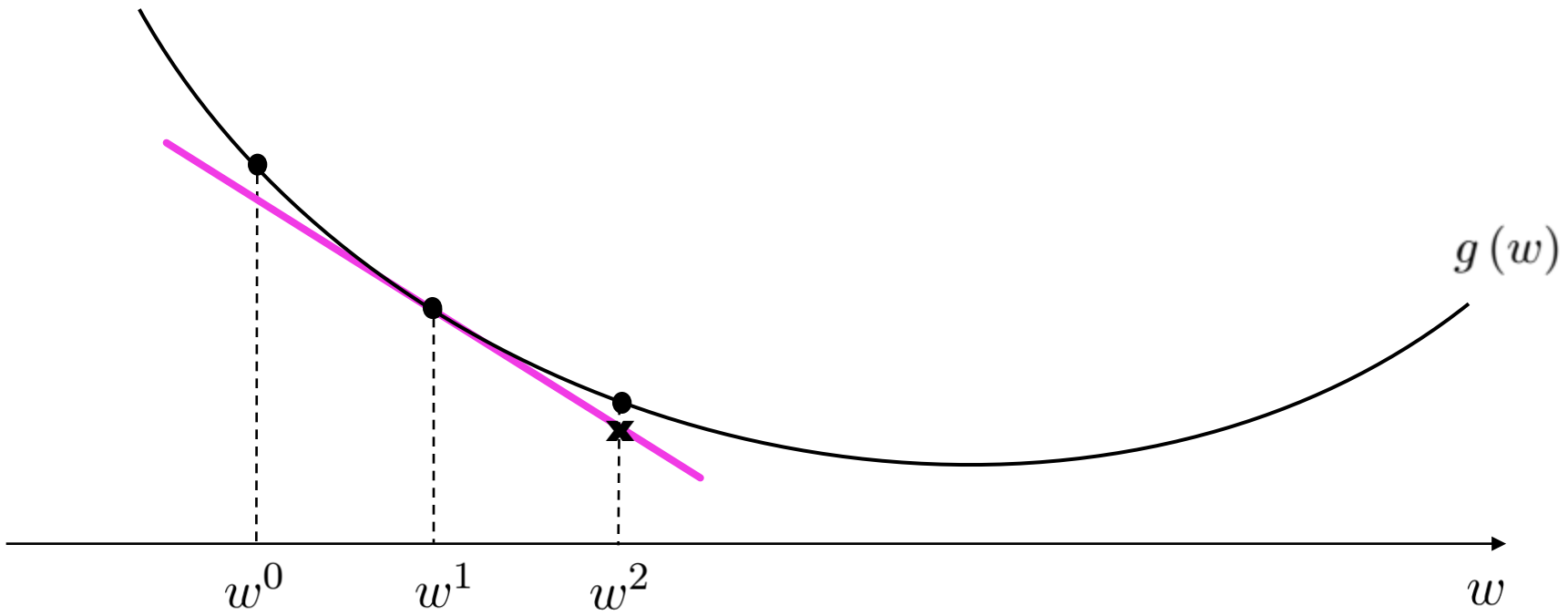
# gradient descent

- ✓ traveling downward on line $\approx$ traveling downward on $g(w)$ near $w^1$
- ✓ the downward direction on a line is **always** $-g'(w^1)$
- ✓ move down with $w^1 = w^0 - \alpha g'(w^0)$ with $\alpha > 0$



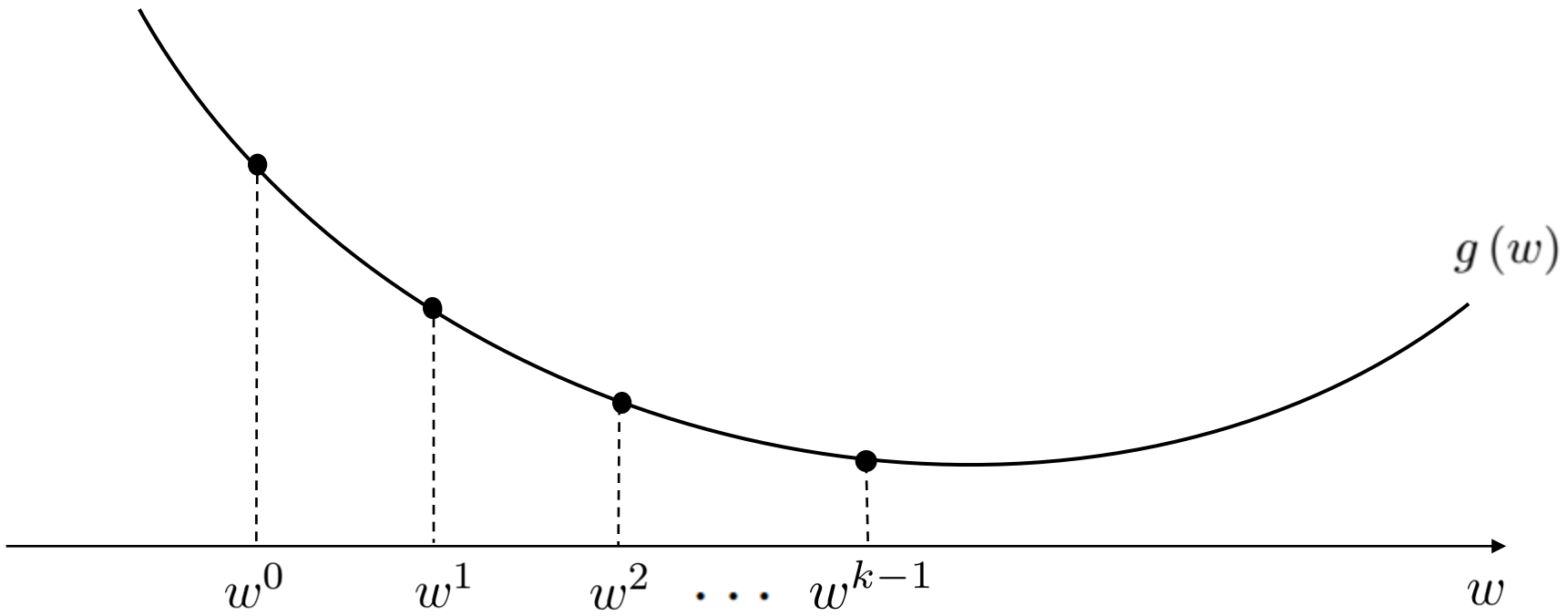
# gradient descent

- ✓ traveling downward on line ≈ traveling downward on $g(w)$ near $w^1$
- ✓ the downward direction on a line is **always** $-g'(w^1)$
- ✓ move down with $w^2 = w^1 - \alpha g'(w^1)$ with $\alpha > 0$



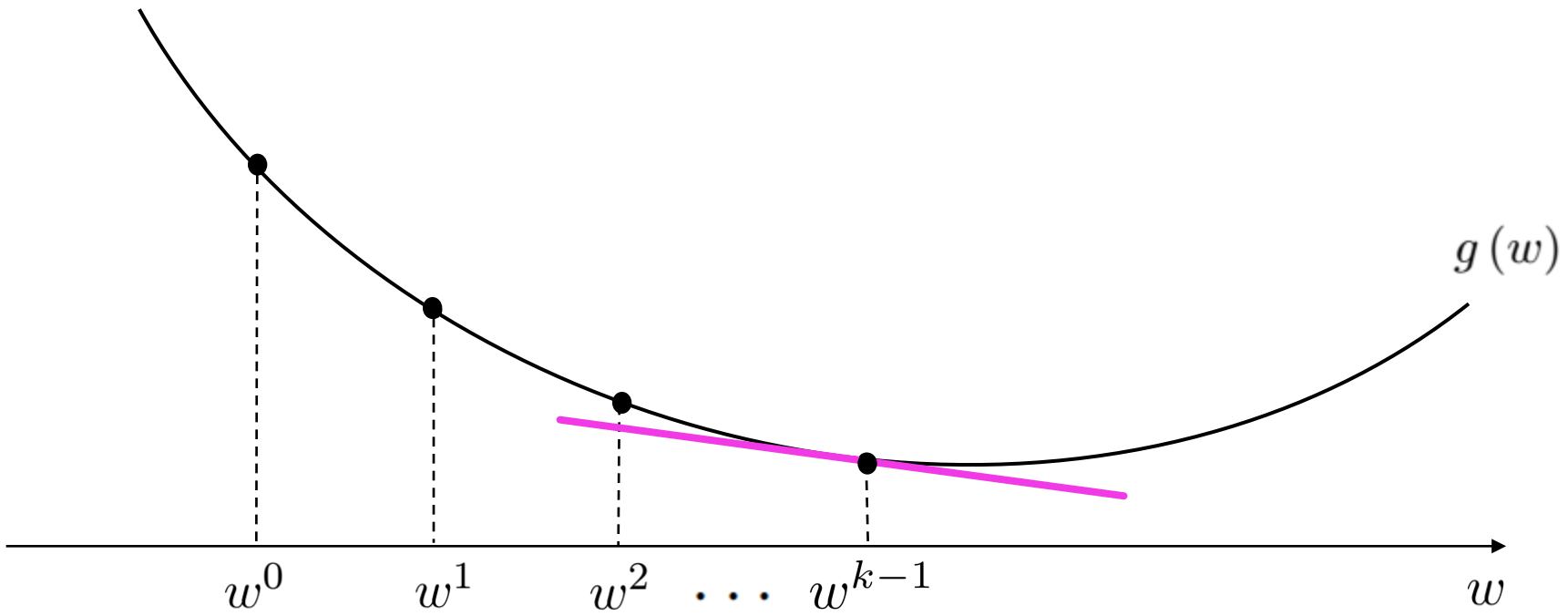
# gradient descent

- ✓ travel downward on line ≈ traveling downward on $g(w)$ near $w^{k-1}$
- ✓ the downward direction on a line is **always** $-g'\left(w^{k-1}\right)$
- ✓ move down with $w^2 = w^1 - \alpha g'\left(w^1\right)$ with $\alpha > 0$



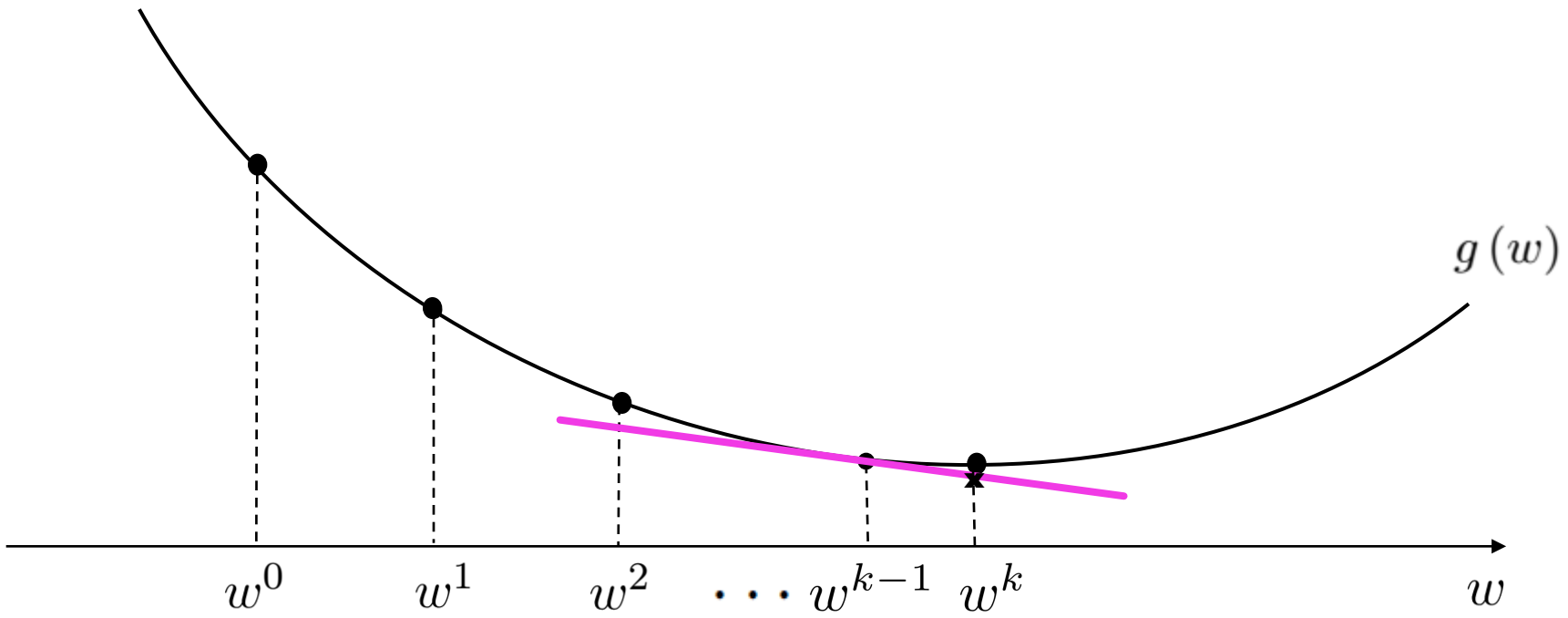
# gradient descent

- ✓ travel downward on line ≈ traveling downward on $g(w)$ near $w^{k-1}$
- ✓ the downward direction on a line is **always** $-g'\left(w^{k-1}\right)$
- ✓ move down with $w^k = w^{k-1} - \alpha g'\left(w^{k-1}\right)$ with $\alpha > 0$



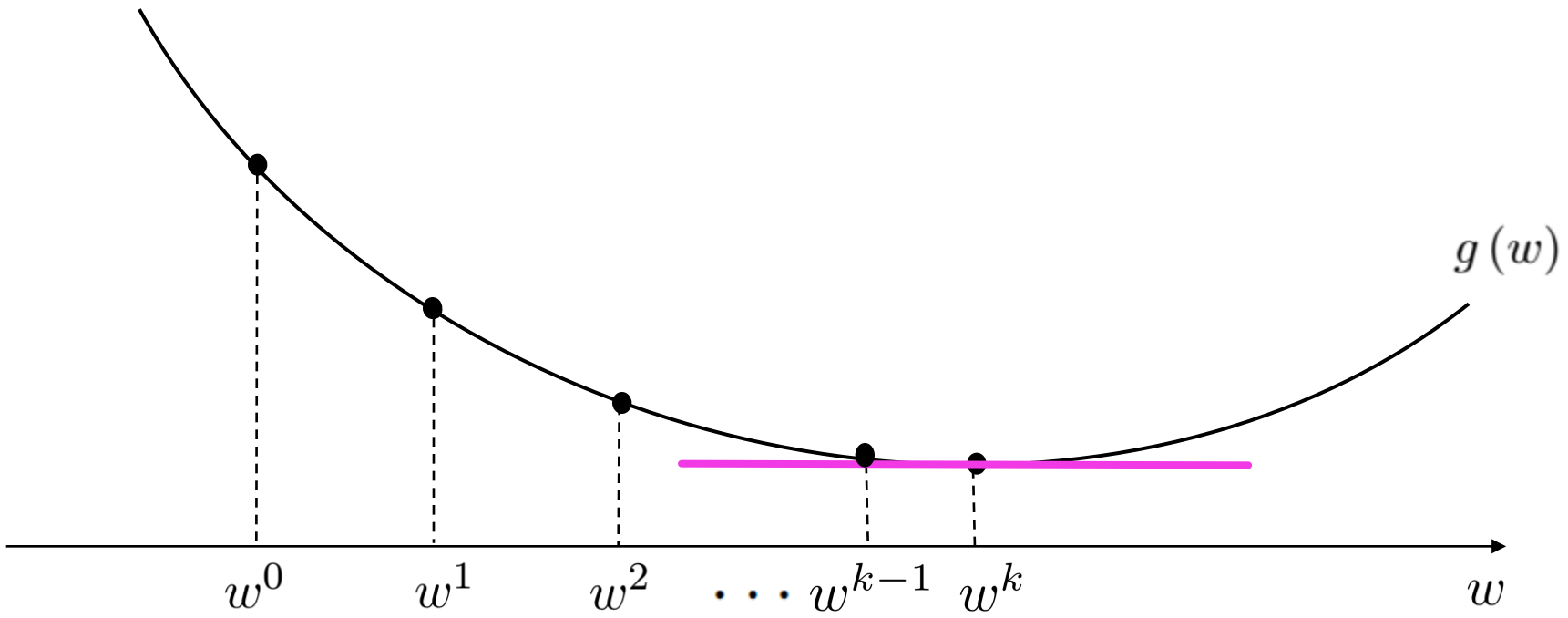
# gradient descent

- ✓ travel downward on line ≈ traveling downward on $g(w)$ near $w^{k-1}$
- ✓ the downward direction on a line is **always** $-g'\left(w^{k-1}\right)$
- ✓ move down with $w^k = w^{k-1} - \alpha g'\left(w^{k-1}\right)$ with $\alpha > 0$



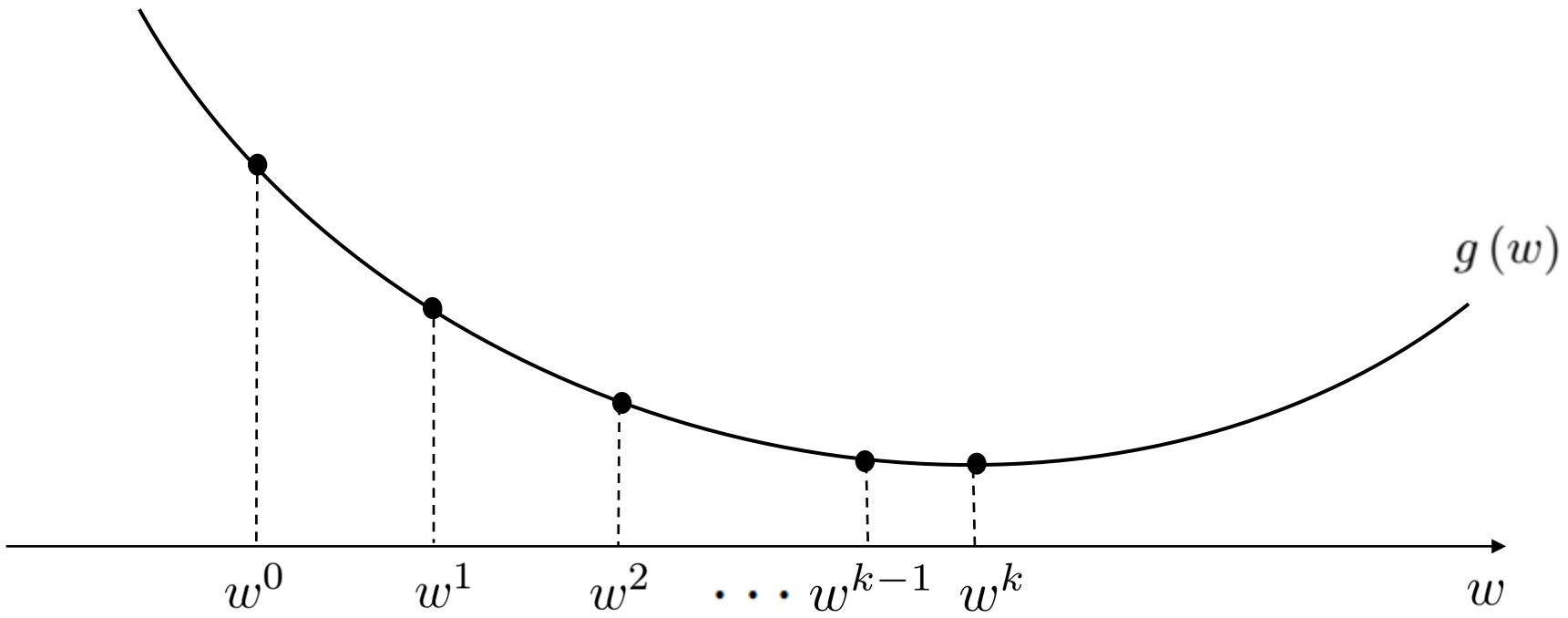
# gradient descent

- ✓ travel downward on line ≈ traveling downward on $g(w)$ near $w^{k-1}$
- ✓ the downward direction on a line is **always** $-g'\left(w^{k-1}\right)$
- ✓ move down with $w^k = w^{k-1} - \alpha g'\left(w^{k-1}\right)$ with $\alpha > 0$



# gradient descent

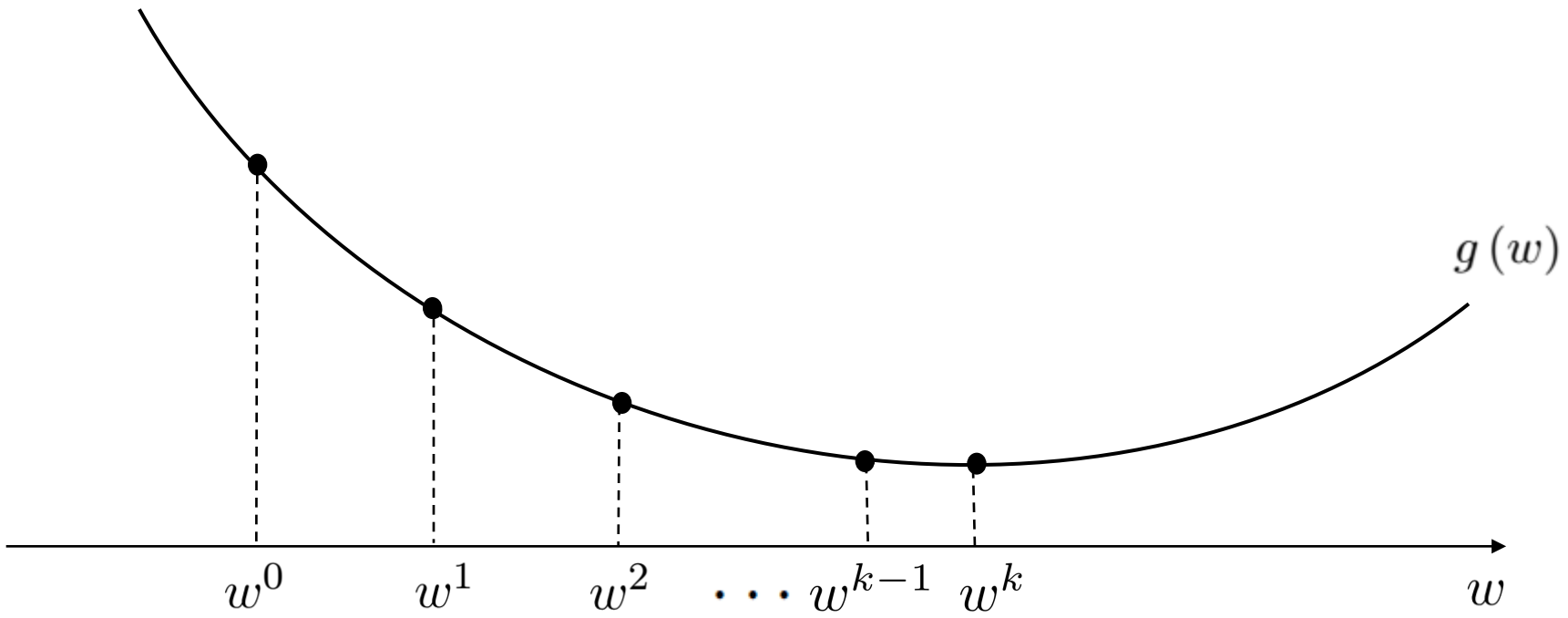
**Full gradient descent algorithm:**



# gradient descent

**Full gradient descent algorithm:**

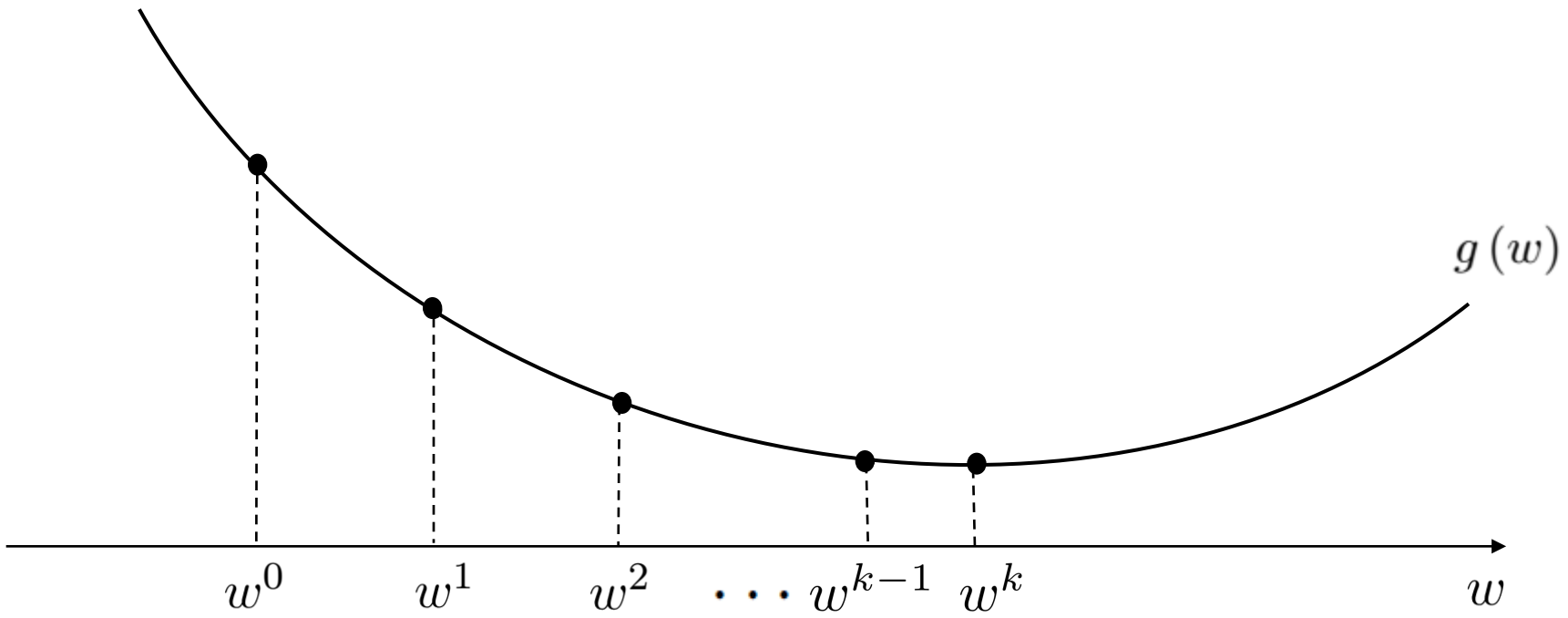
1. start at some $w^0$, choose value for $\alpha > 0$



# gradient descent

**Full gradient descent algorithm:**

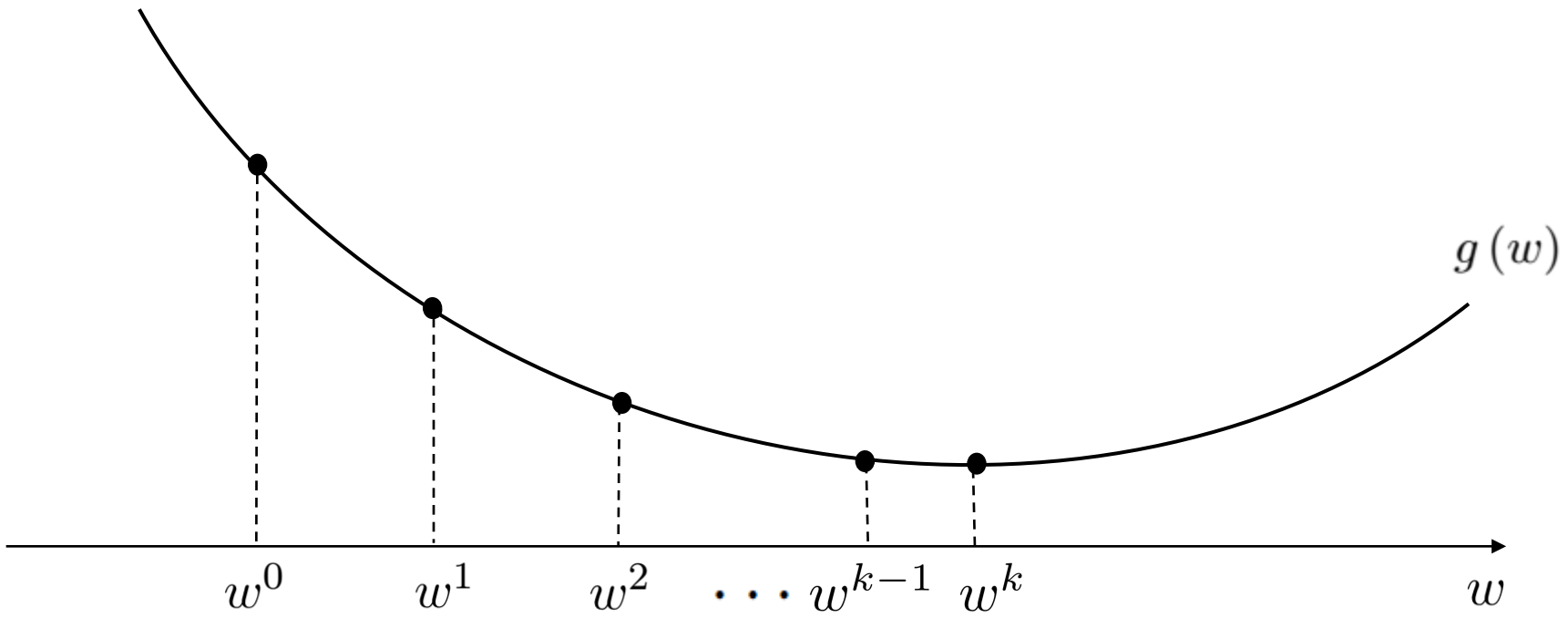
1. start at some $w^0$, choose value for $\alpha > 0$
2. move down with $w^k = w^{k-1} - \alpha g'\left(w^{k-1}\right)$ with $\alpha > 0$



# gradient descent

**Full gradient descent algorithm:**

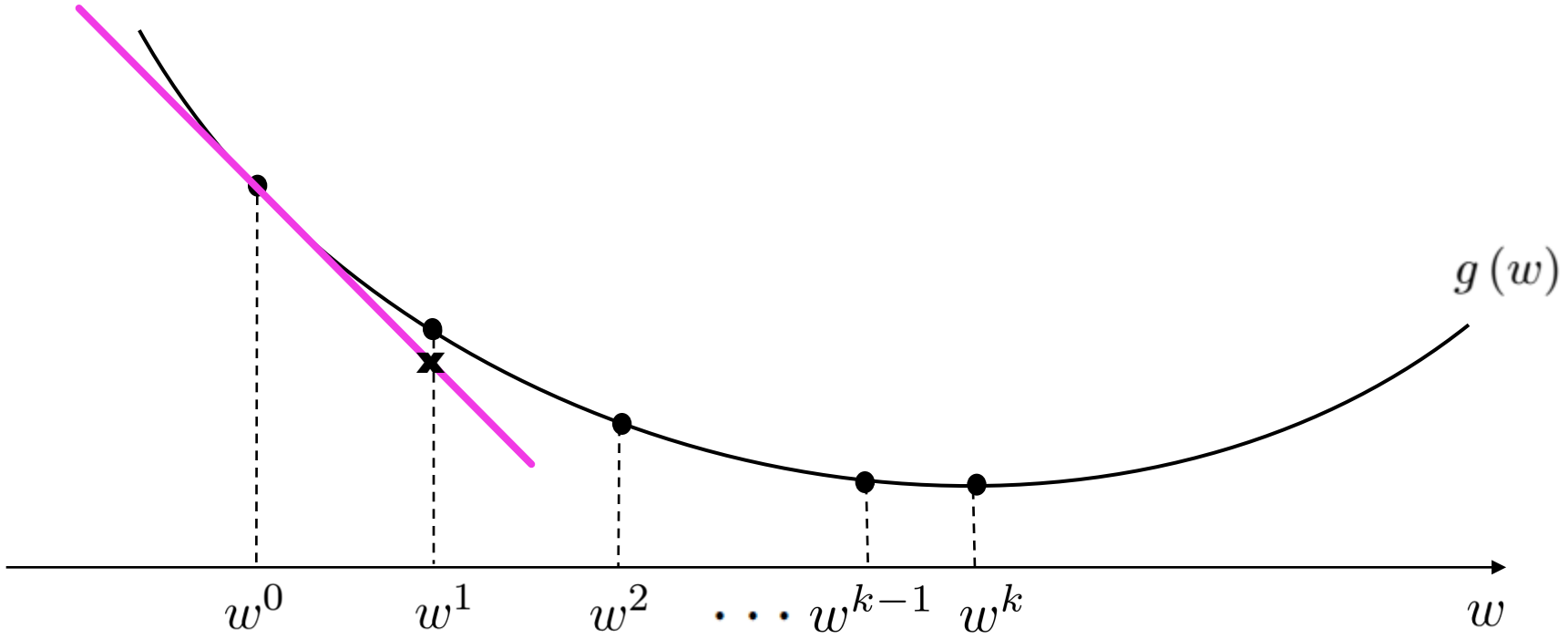
1. start at some $w^0$, choose value for $\alpha > 0$
2. move down with $w^k = w^{k-1} - \alpha g'\left(w^{k-1}\right)$ with $\alpha > 0$
3. repeat 2. until convergence



# gradient descent

**Full gradient descent algorithm:**

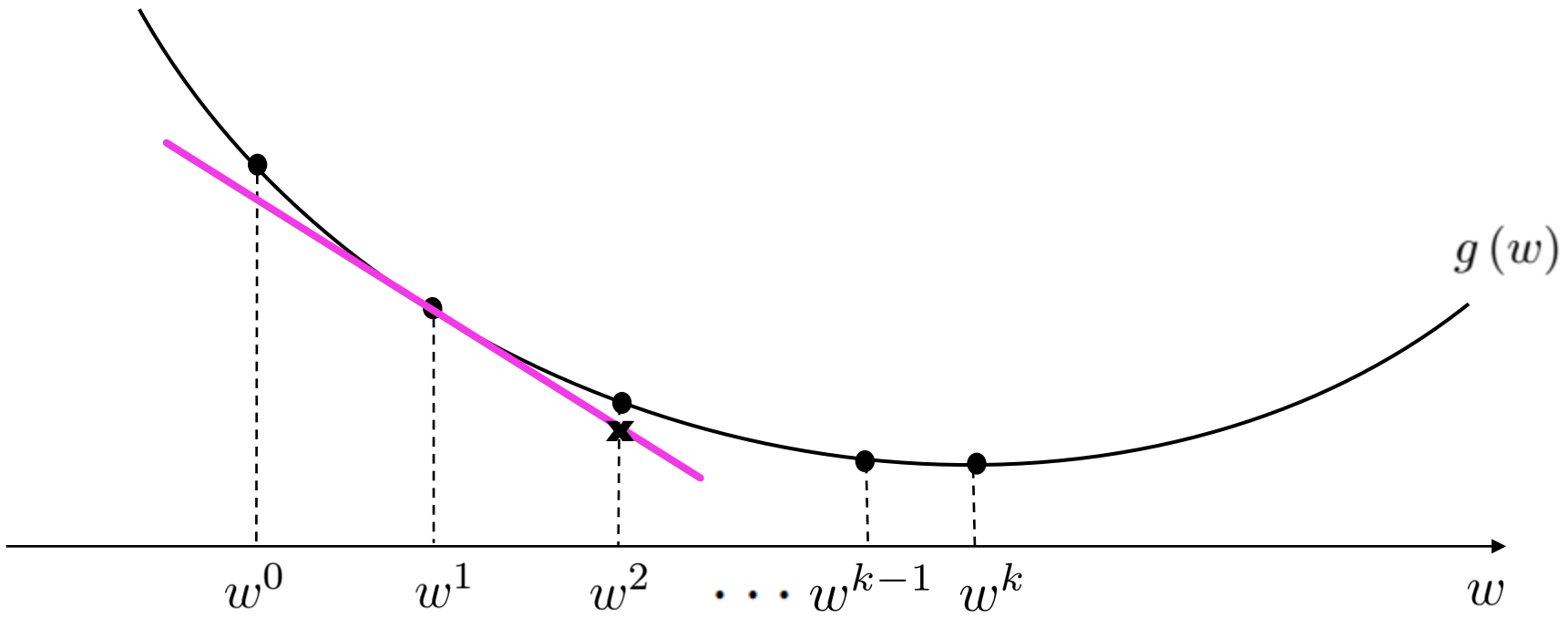
1. start at some $w^0$, choose value for $\alpha > 0$
2. move down with $w^k = w^{k-1} - \alpha g'\left(w^{k-1}\right)$ with $\alpha > 0$
3. repeat 2. until convergence



# gradient descent

**Full gradient descent algorithm:**

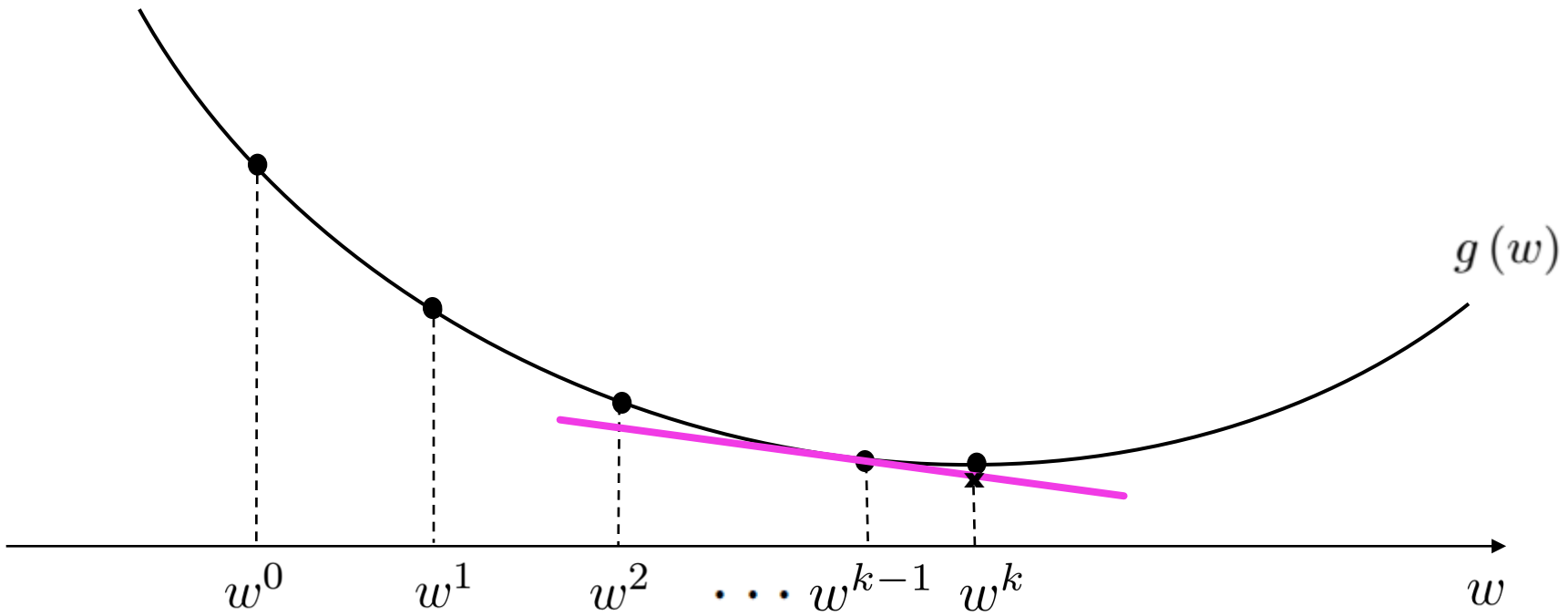
1. start at some $w^0$, choose value for $\alpha > 0$
2. move down with $w^k = w^{k-1} - \alpha g'\left(w^{k-1}\right)$ with $\alpha > 0$
3. repeat 2. until convergence



# gradient descent

**Full gradient descent algorithm:**

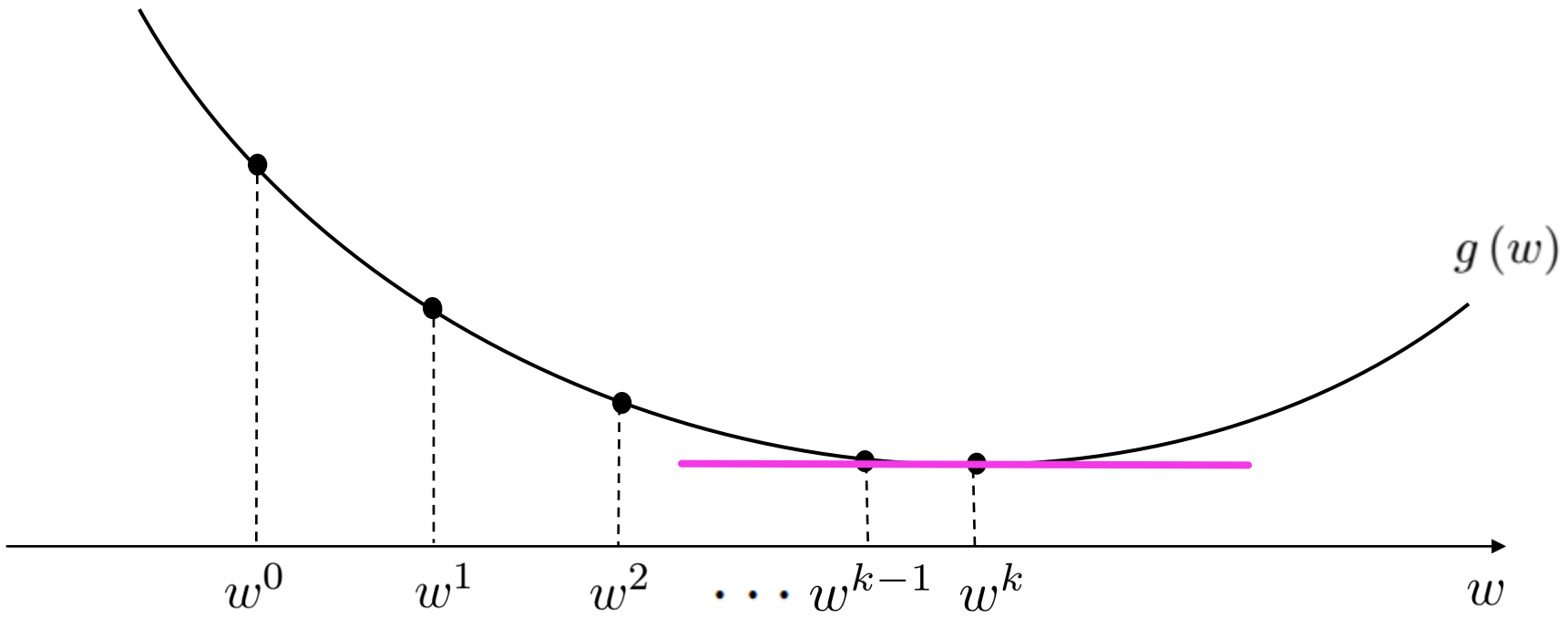
1. start at some $w^0$, choose value for $\alpha > 0$
2. move down with $w^k = w^{k-1} - \alpha g'\left(w^{k-1}\right)$ with $\alpha > 0$
3. repeat 2. until convergence



# gradient descent

**Full gradient descent algorithm:**

1. start at some $w^0$, choose value for $\alpha > 0$
2. move down with $w^k = w^{k-1} - \alpha g'\left(w^{k-1}\right)$ with $\alpha > 0$
3. repeat 2. until convergence



# Demo # 3 convex grad descent

- to the notebook!

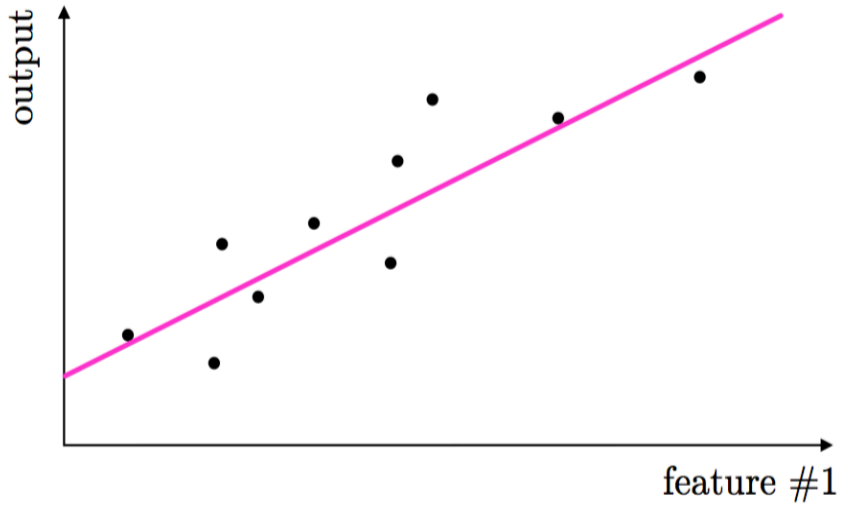
# Demo # 4 linear regression

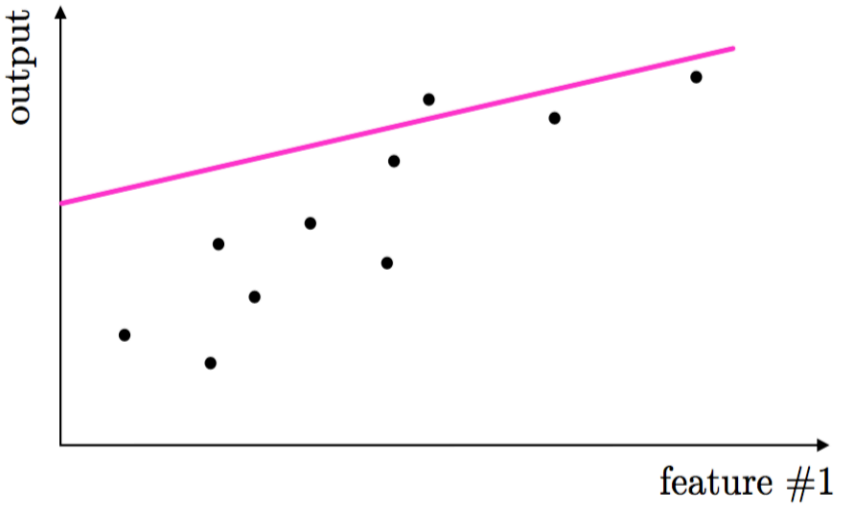
- to the notebook!

# Parameter tuning for optimal learning

# parameter tuning: regression

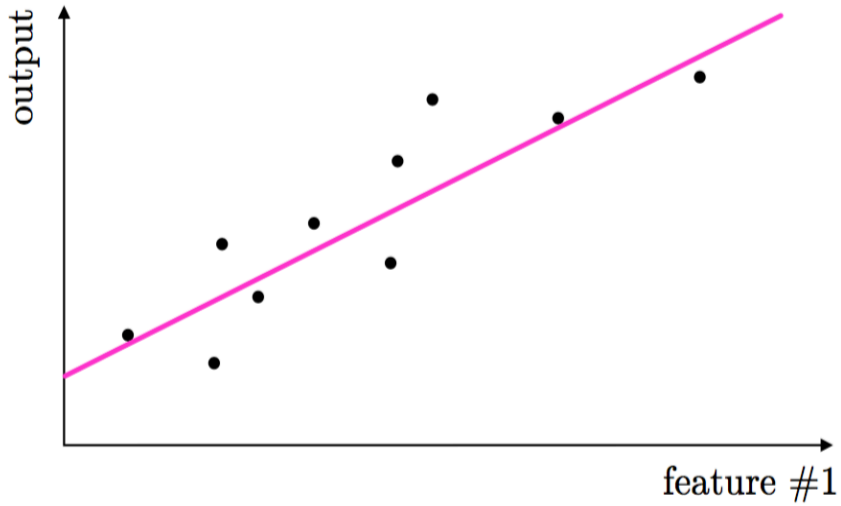
- parameters must be tuned properly to ensure optimal learning

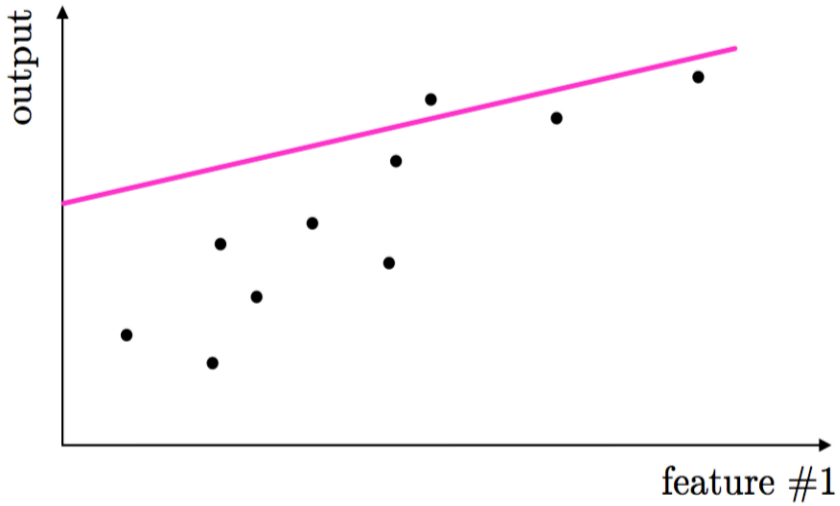


# parameter tuning: regression

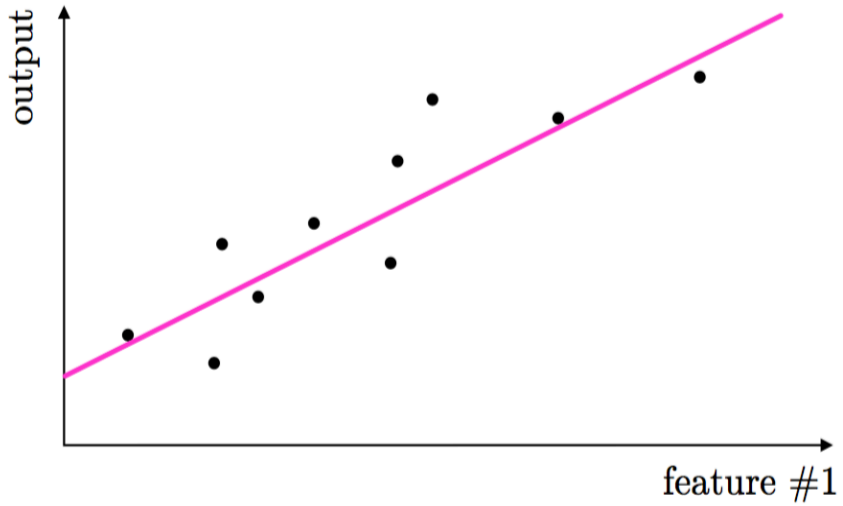
- parameters must be tuned properly to ensure optimal learning
- a 'cost function' measures how well a model ***fits*** data given a set of parameters

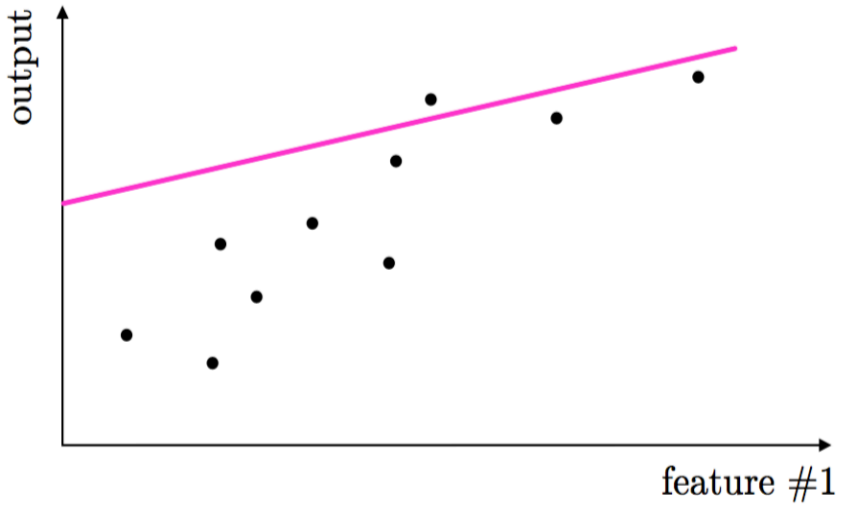


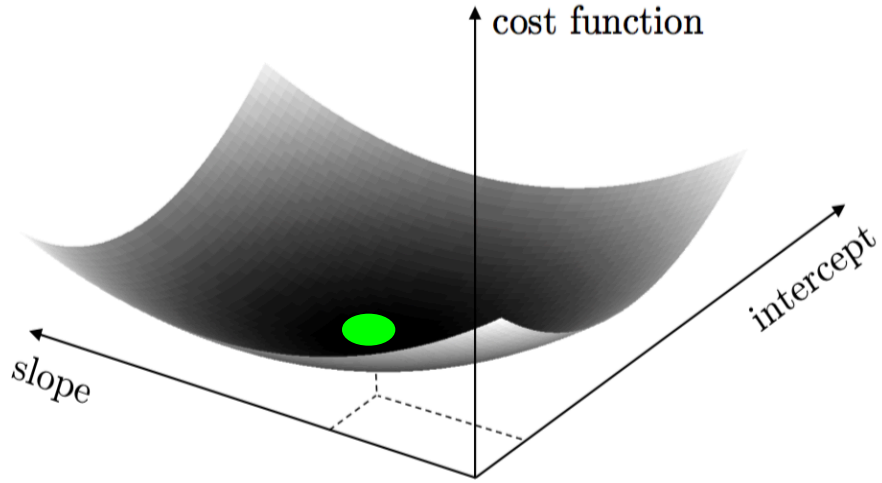
# parameter tuning: regression

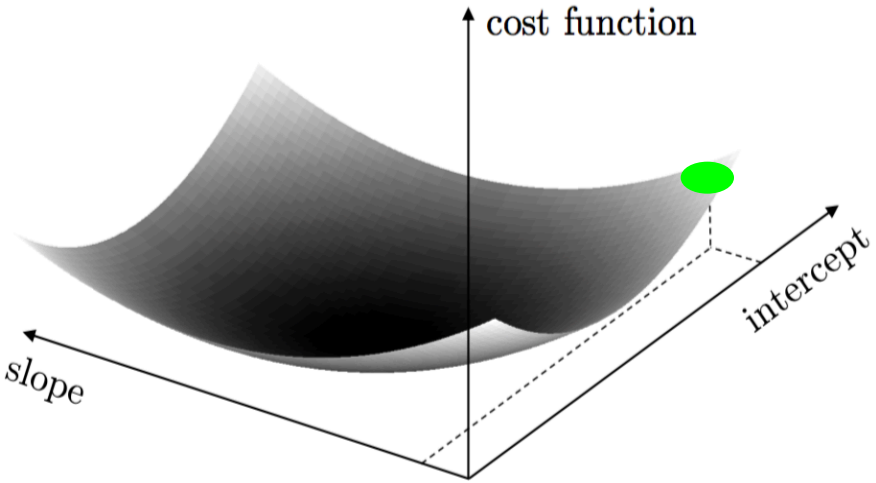
- parameters must be tuned properly to ensure optimal learning
- a 'cost function' measures how well a model ***fits*** data given a set of parameters
- better parameters provide a better fit, and ***lower*** value of cost function

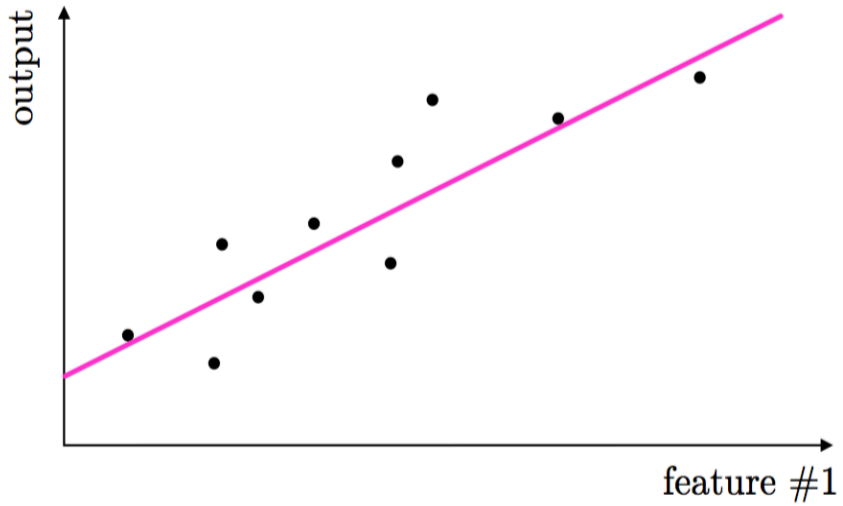


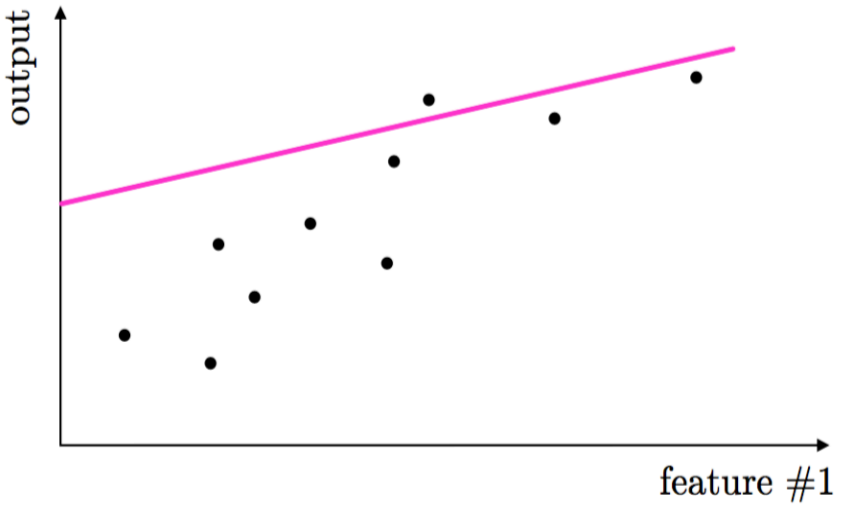



# parameter tuning: regression



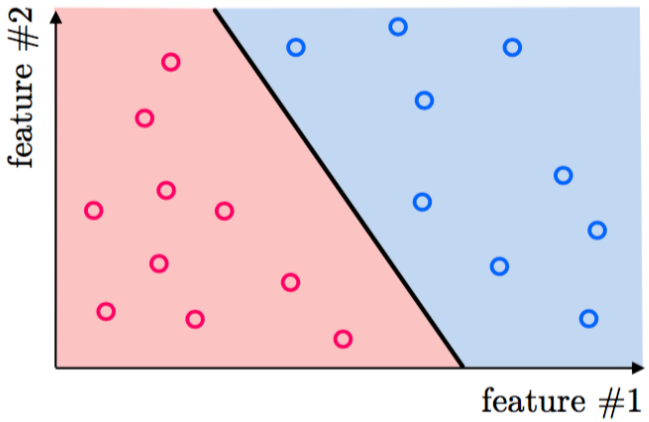



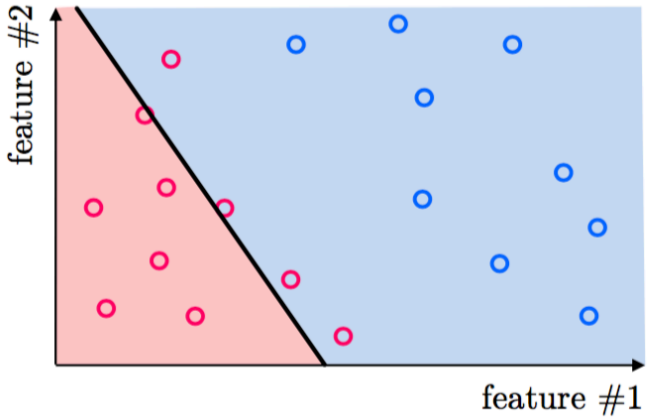



# parameter tuning: classification

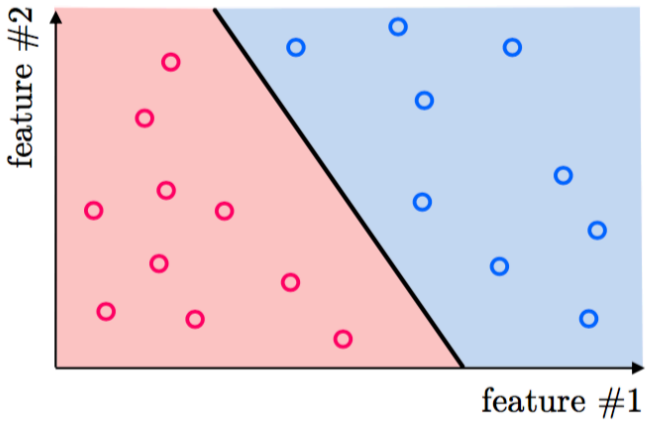
- precisely the same situation with classification

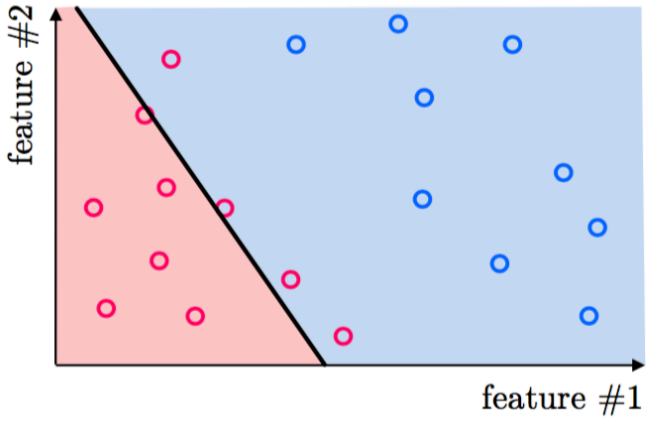


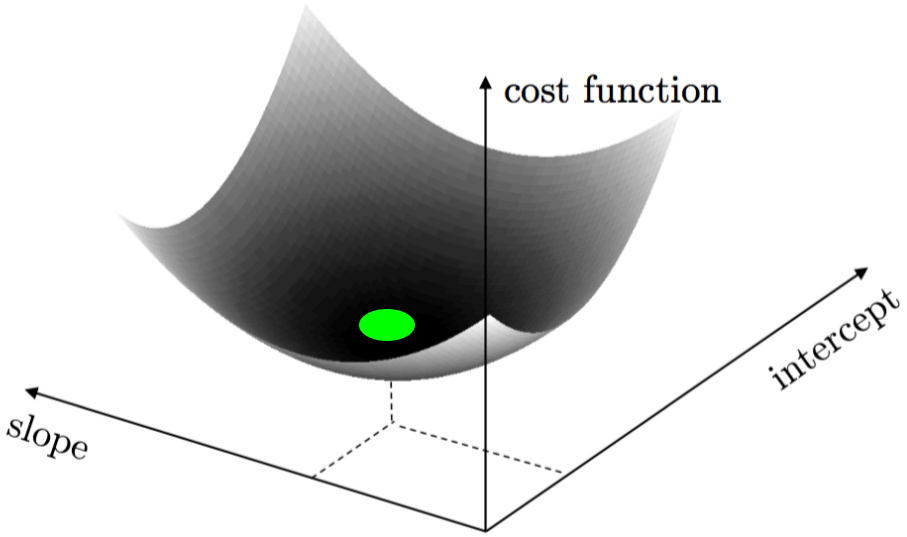
# parameter tuning: classification

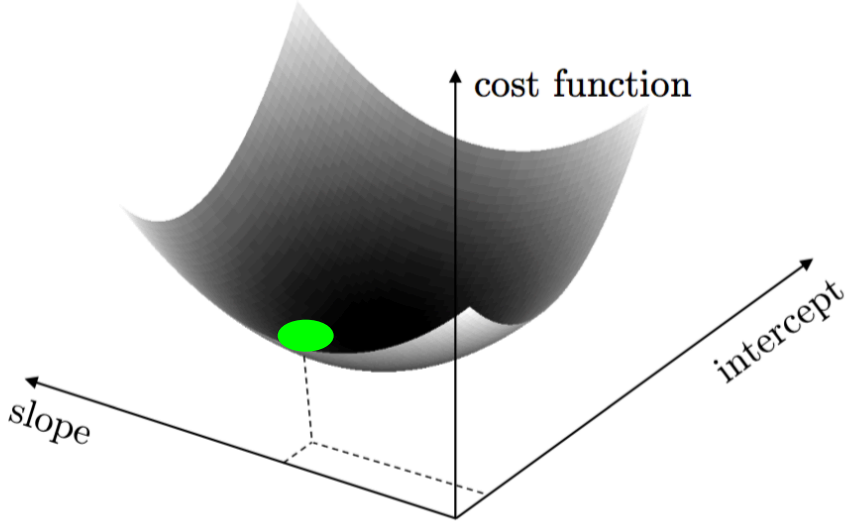
- precisely the same situation with classification
- a 'cost function' measures how well a model ***splits*** data given a set of parameters
- better parameters provide a better fit, and ***lower*** value of cost function

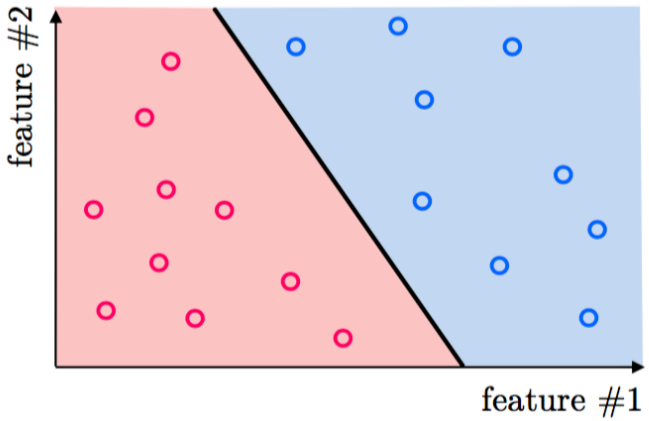


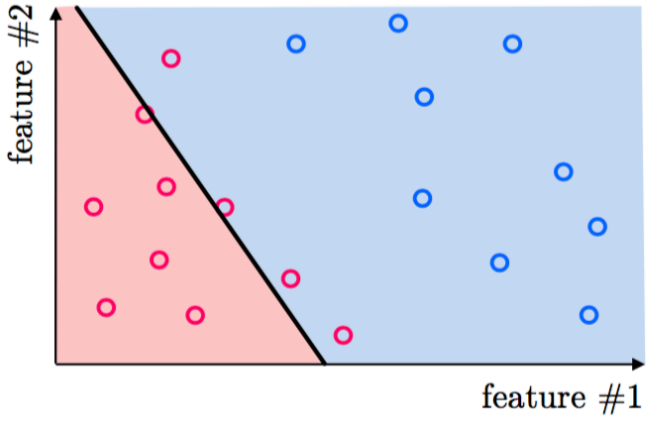



# parameter tuning



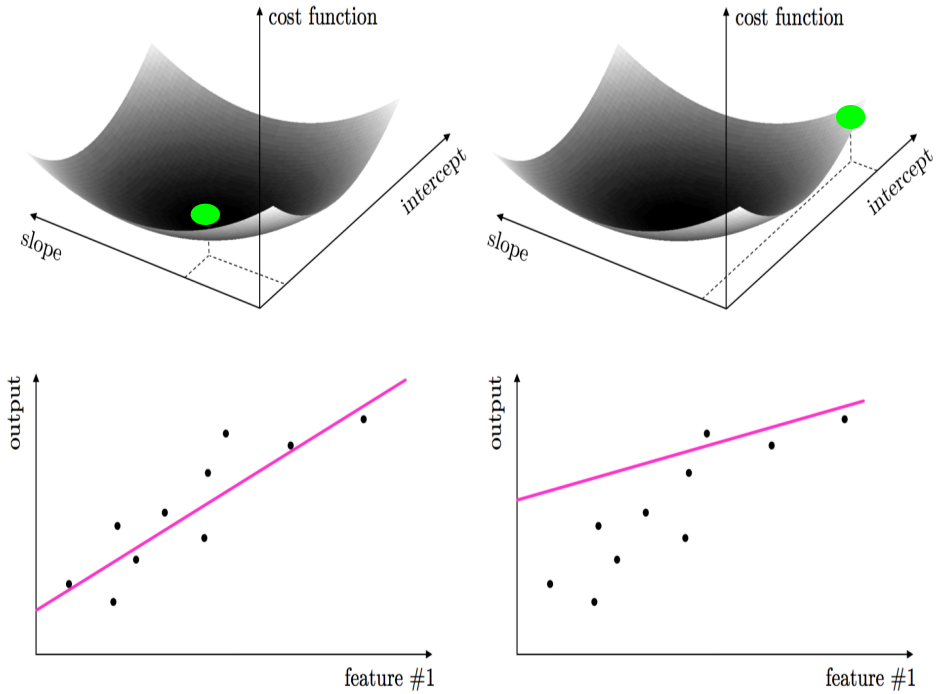



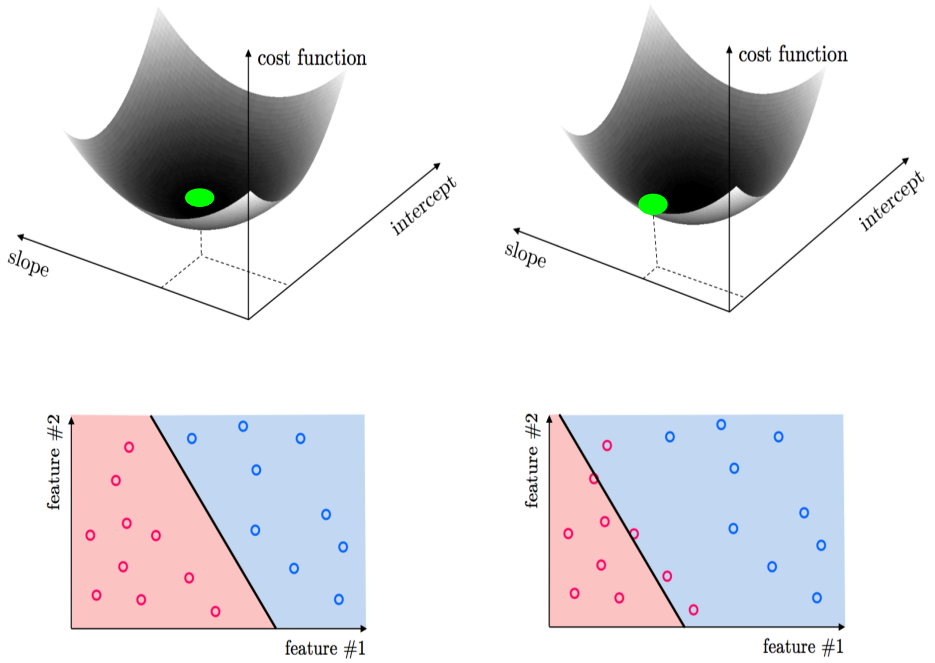



# parameter tuning: in general

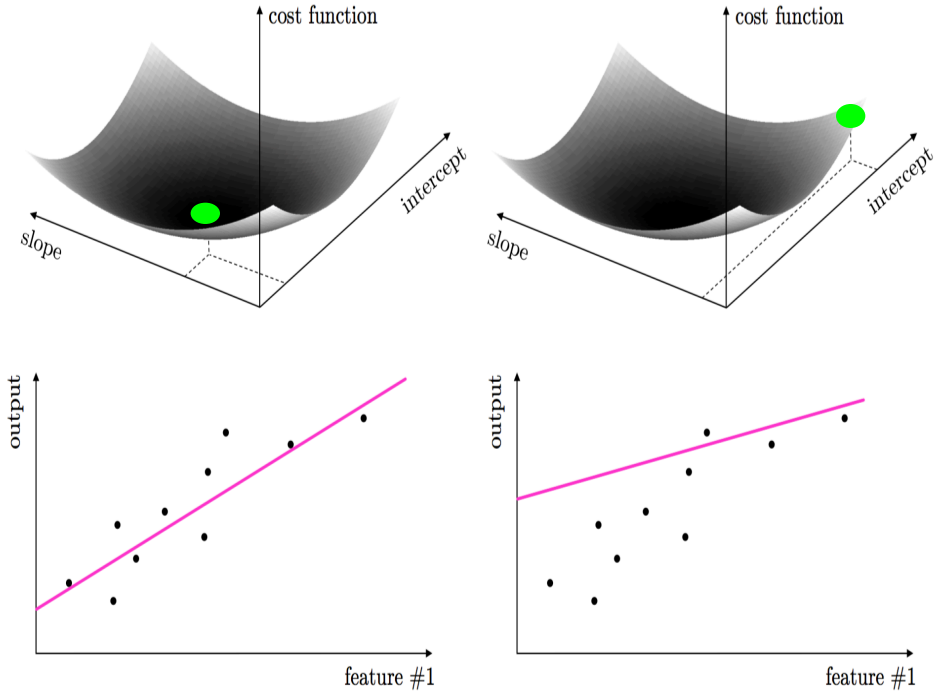
- regardless if regression/classification, tuning parameters is the same issue



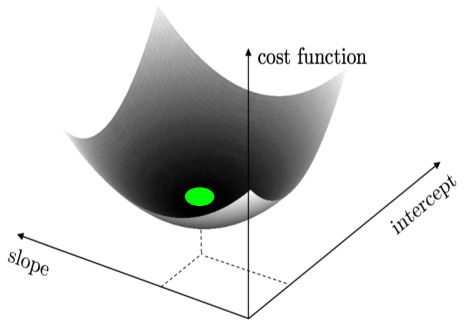
# parameter tuning: in general

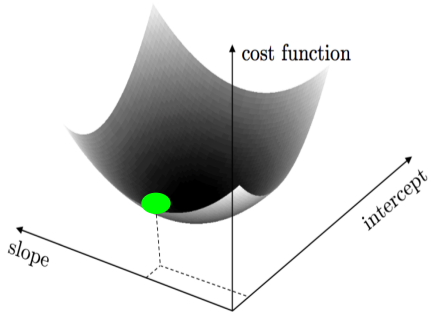
- regardless if regression/classification, tuning parameters is the same issue
- how to find the set of parameters that ***minimizes*** a differentiable cost function

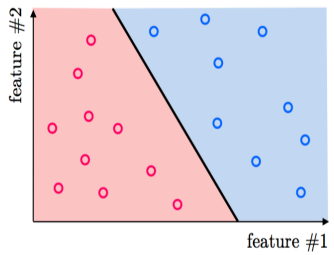


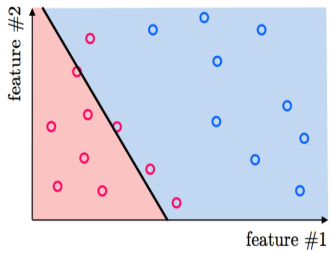






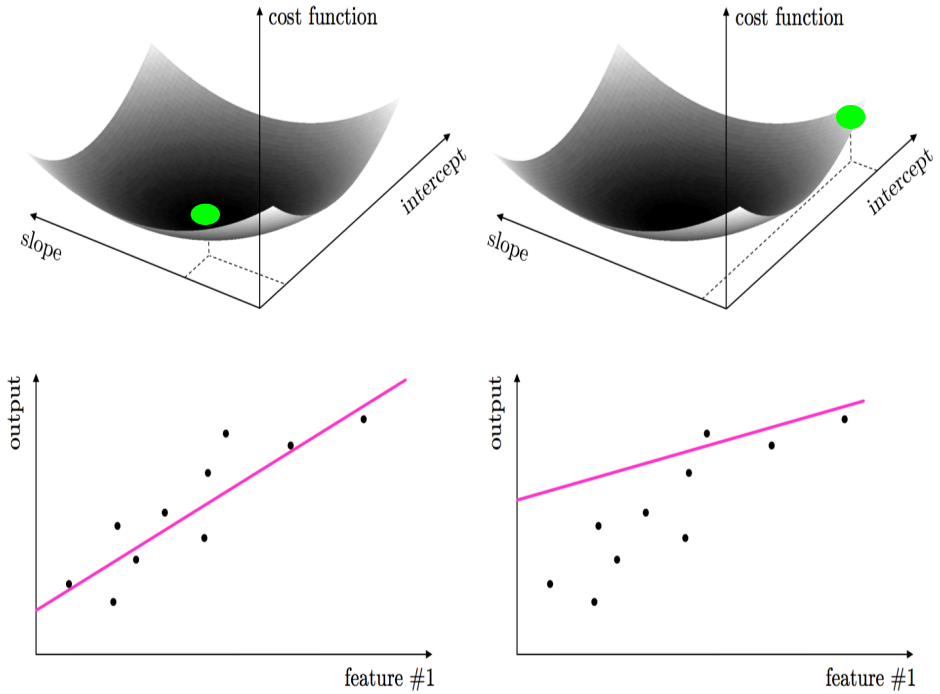



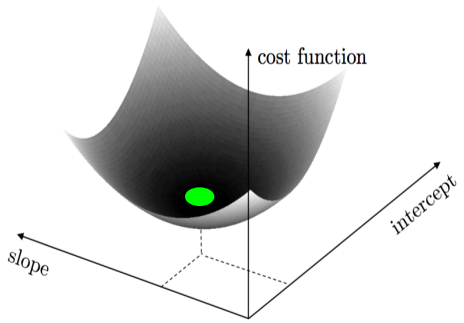
# parameter tuning: in general

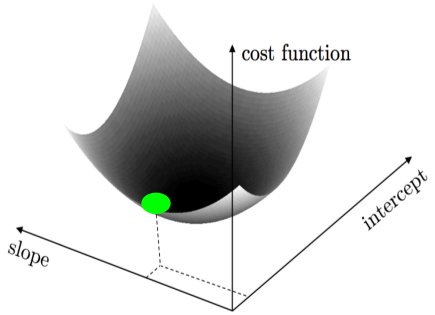
- **problem**: typically cannot visualize cost function, number of parameters > 2

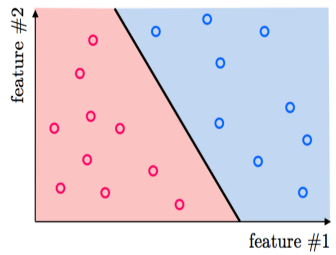


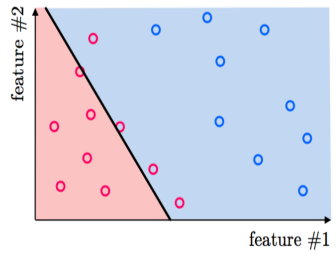






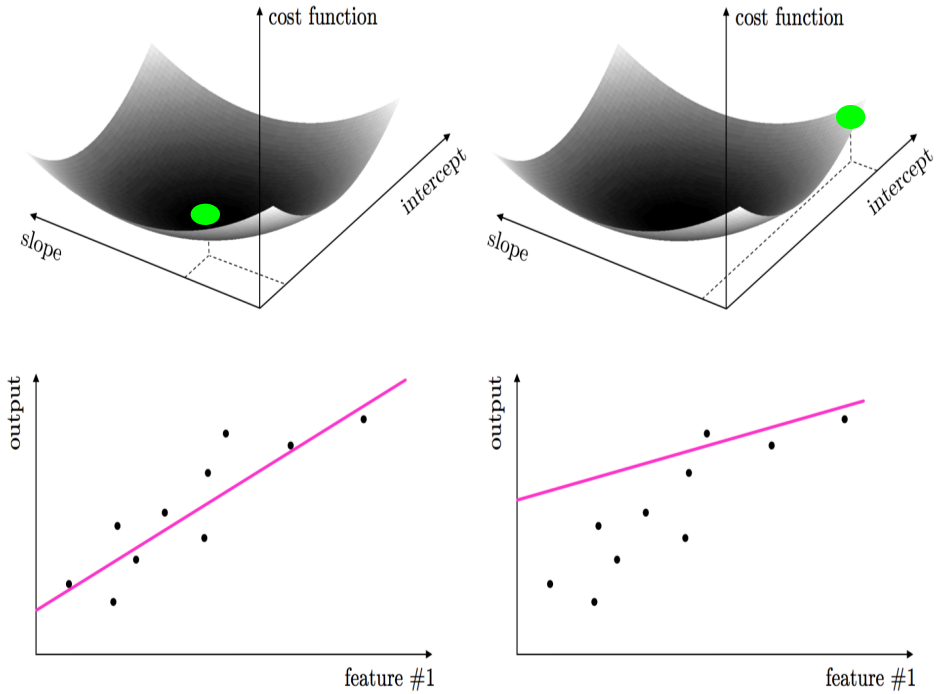



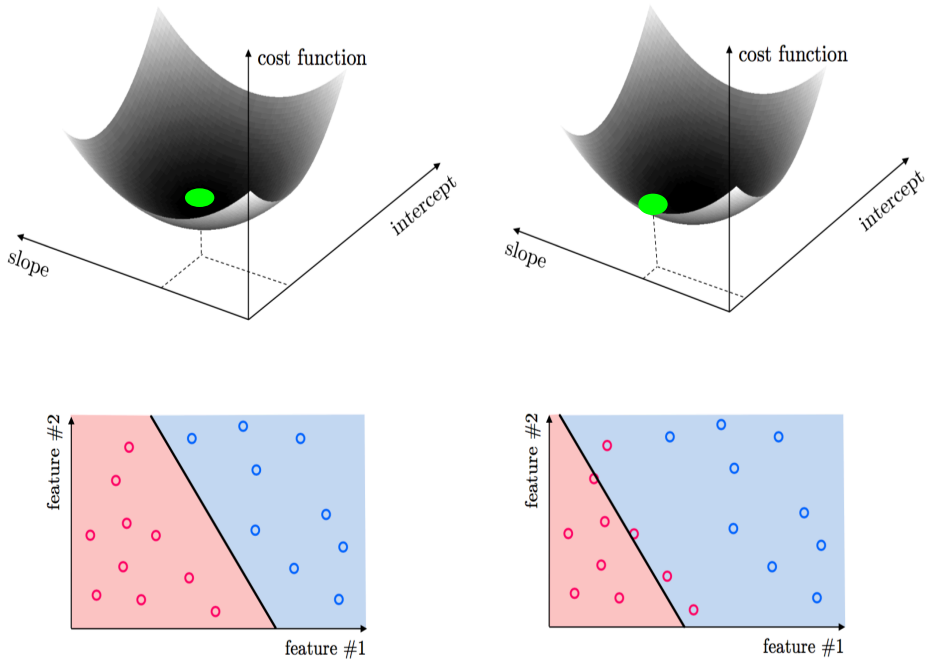



# parameter tuning: in general

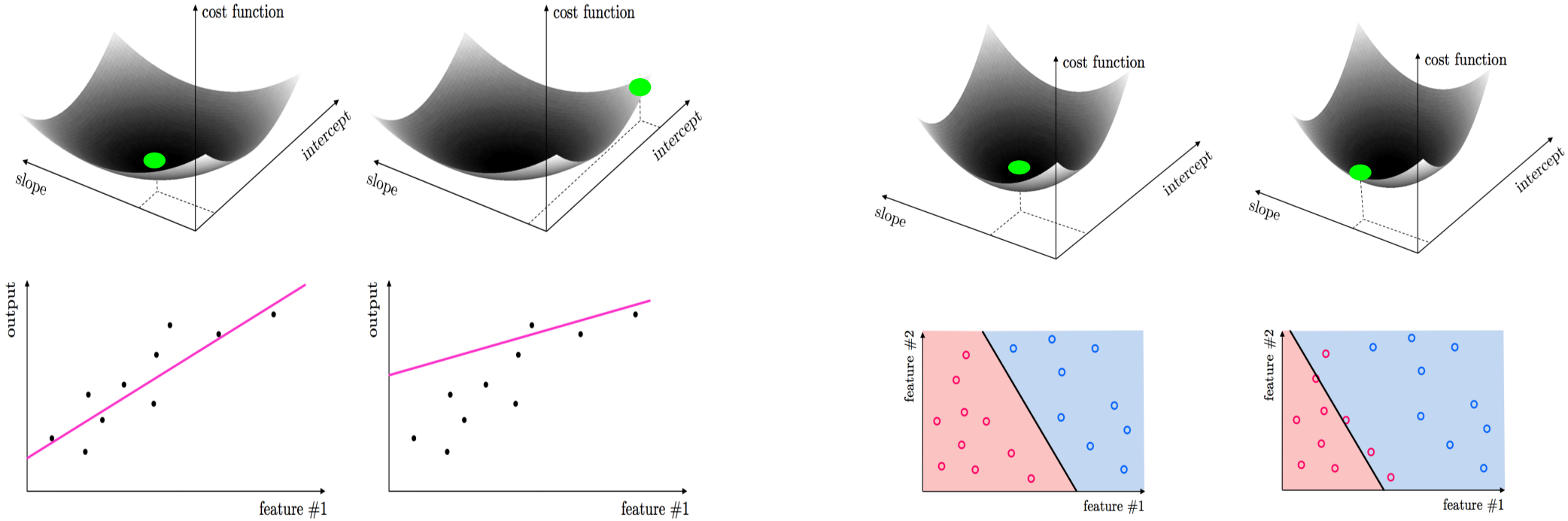
- **problem**: typically cannot visualize cost function, number of parameters > 2
- **solution**: host of algorithms based on simple calculus



# parameter tuning: in general

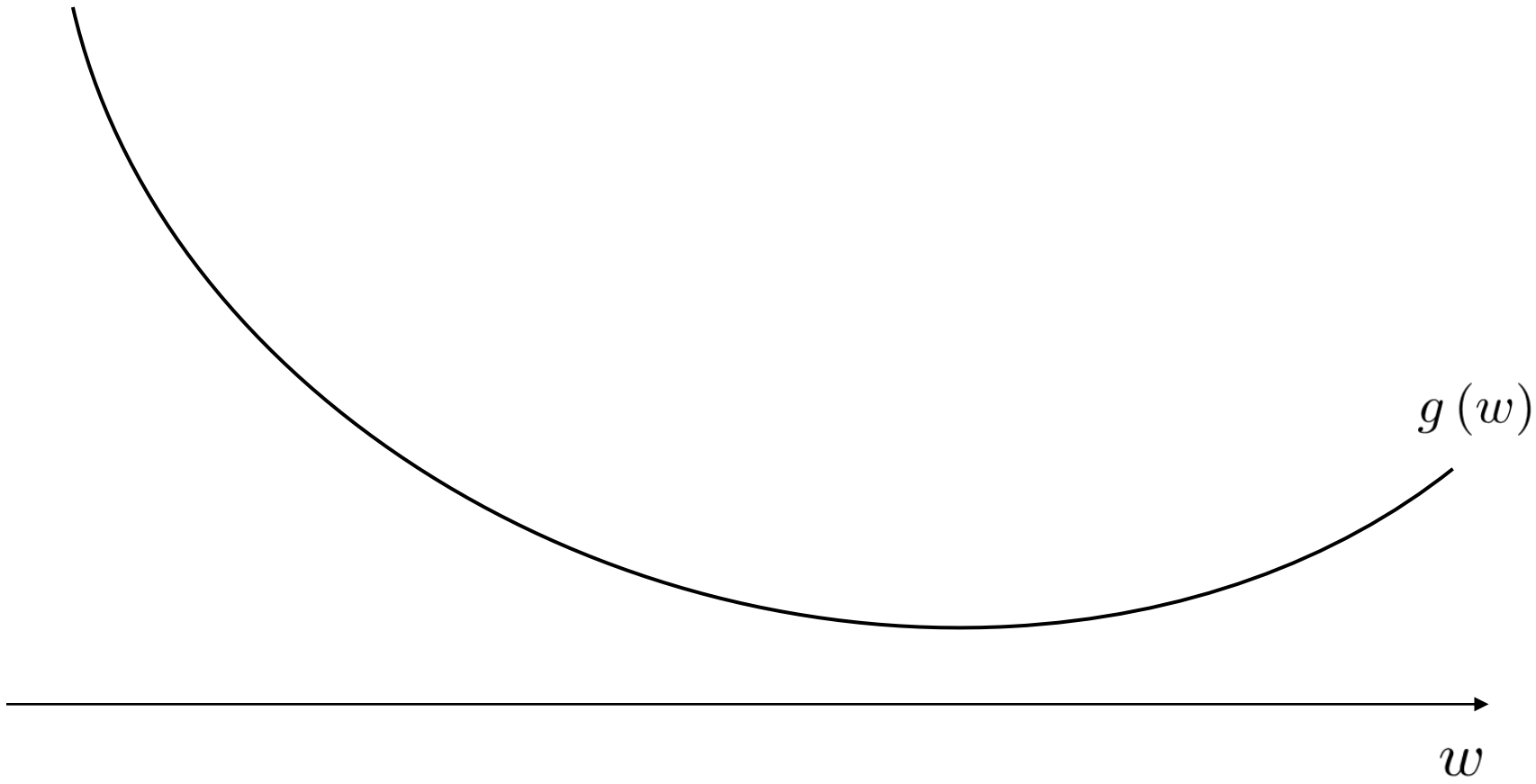
- **problem**: typically cannot visualize cost function, number of parameters > 2
- **solution**: host of algorithms based on simple calculus fact
- **gradient descent** the most popular among such algorithms



# Demo # 6 – linear classification

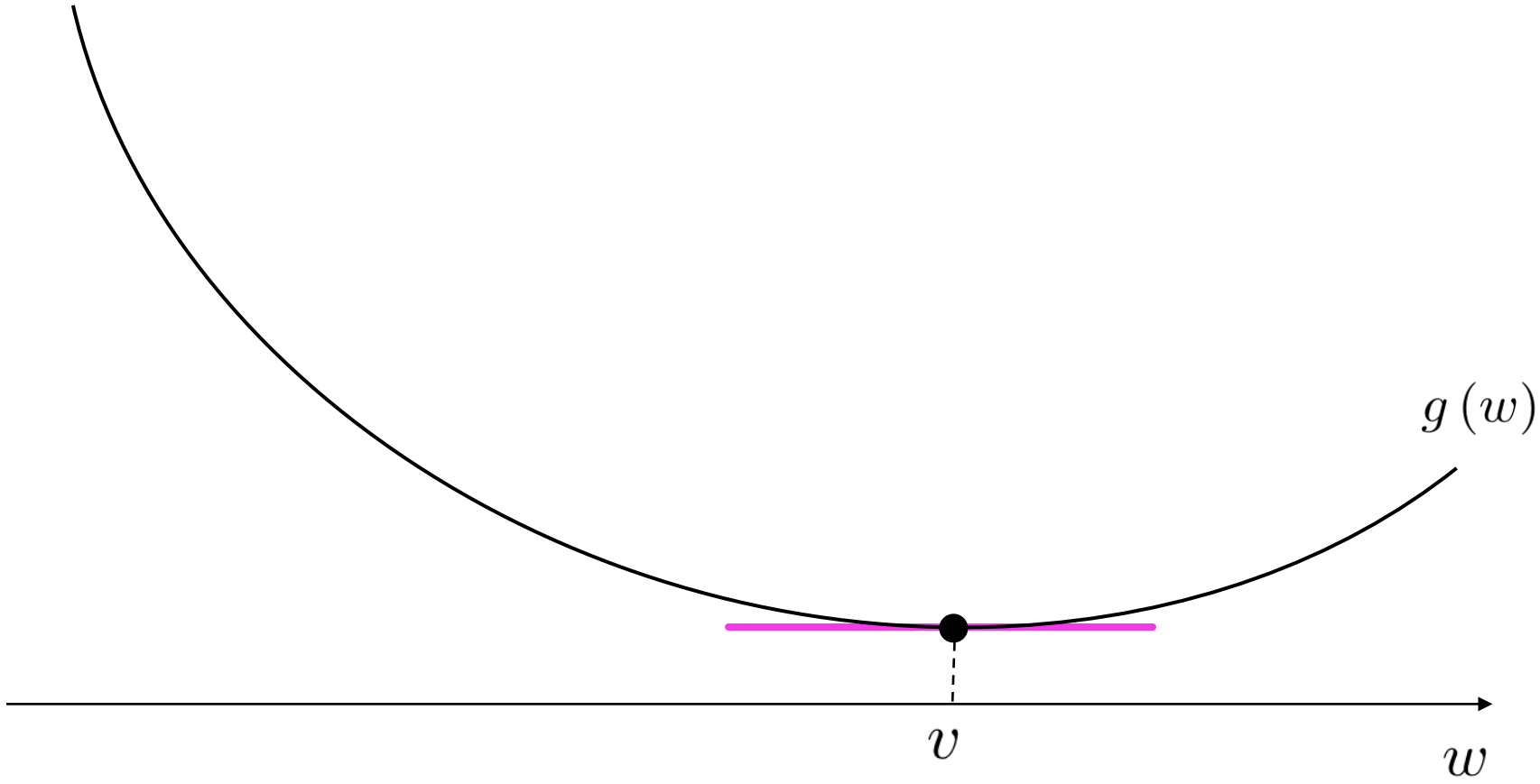
- to the notebook!

# the derivative



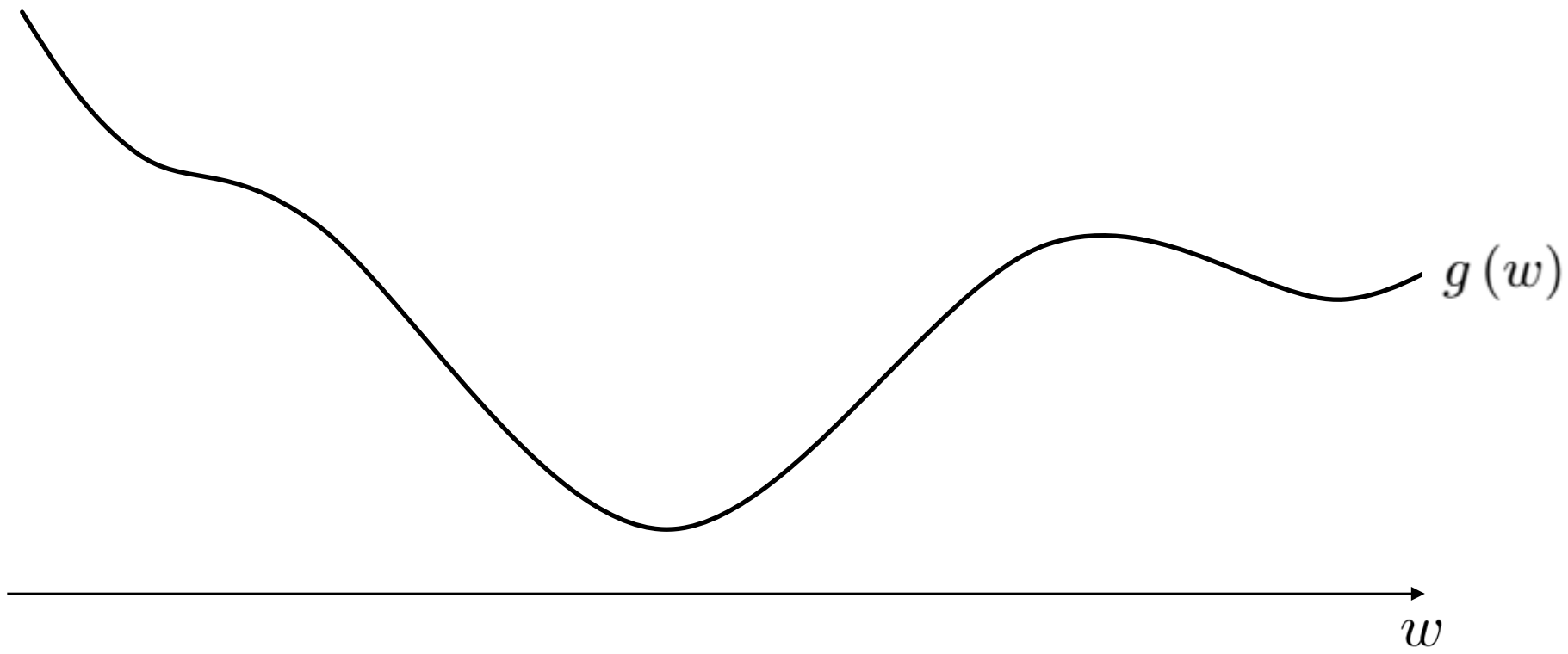
# the derivative

- differentiable function $g\left(w\right)$ with single input $w$
- derivative $g'\left(w\right)$ gives the slope of tangent line at $w$



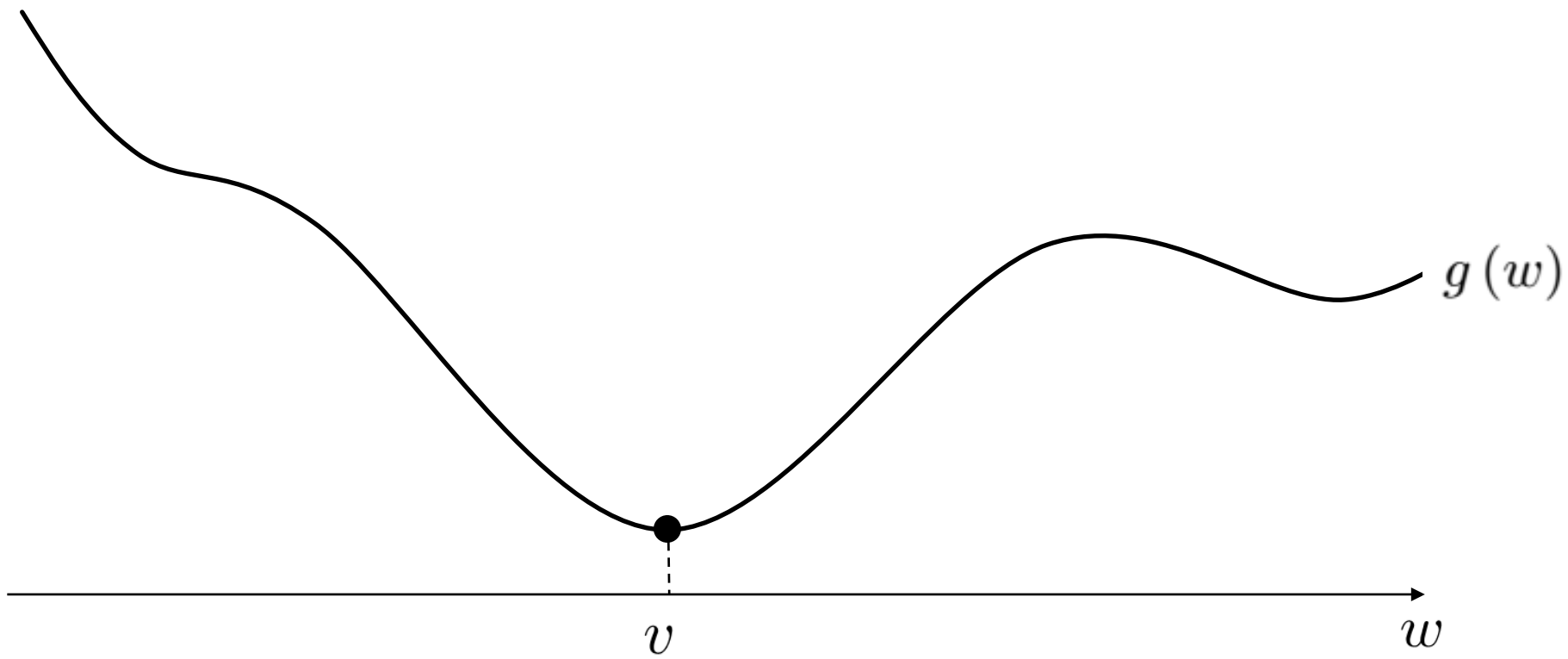
# the derivative

- differentiable function $g\,(w)$ with single input $w$
- derivative $g'\,(w)$ gives the slope of tangent line at $w$



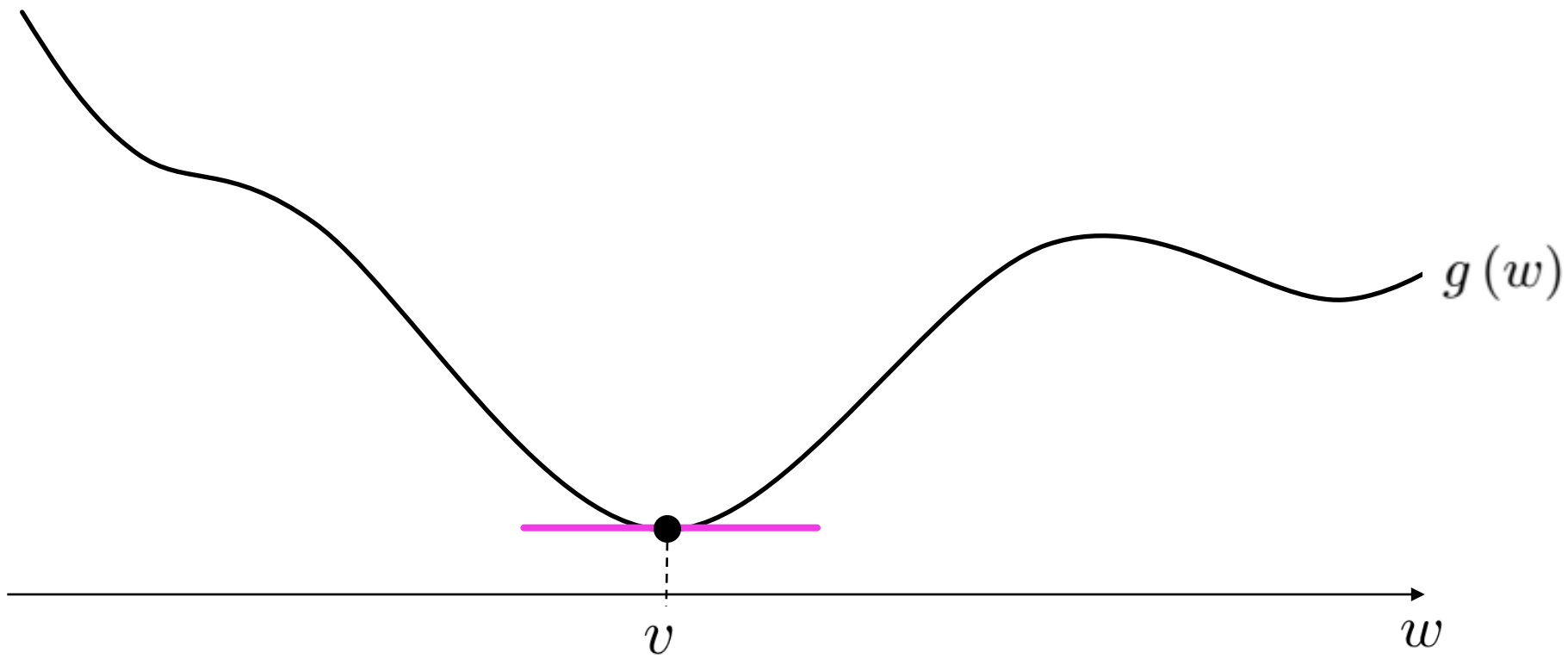
# the derivative

- differentiable function $g\left(w\right)$ with single input $w$
- derivative $g'\left(w\right)$ gives the slope of tangent line at $w$



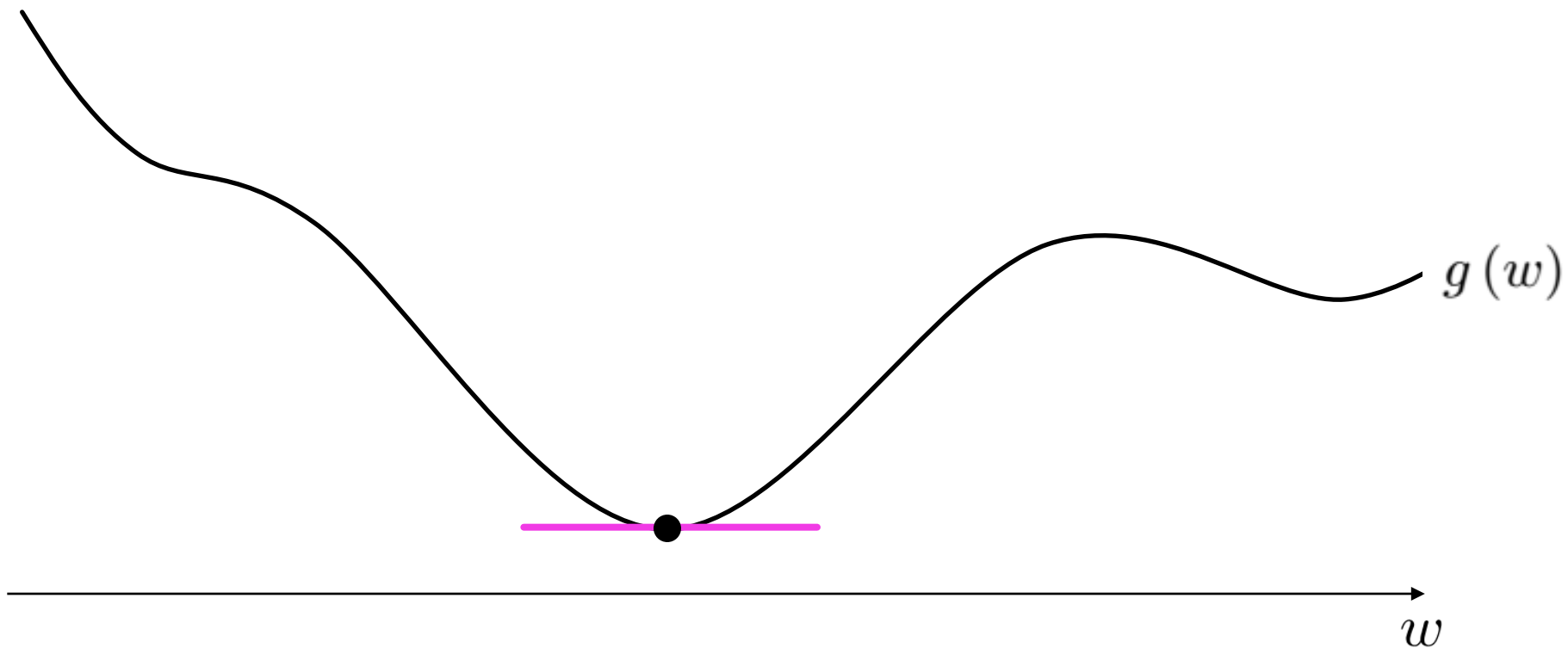
# the derivative

- differentiable function $g\left(w\right)$ with single input $w$
- derivative $g'\left(w\right)$ gives the slope of tangent line at $w$



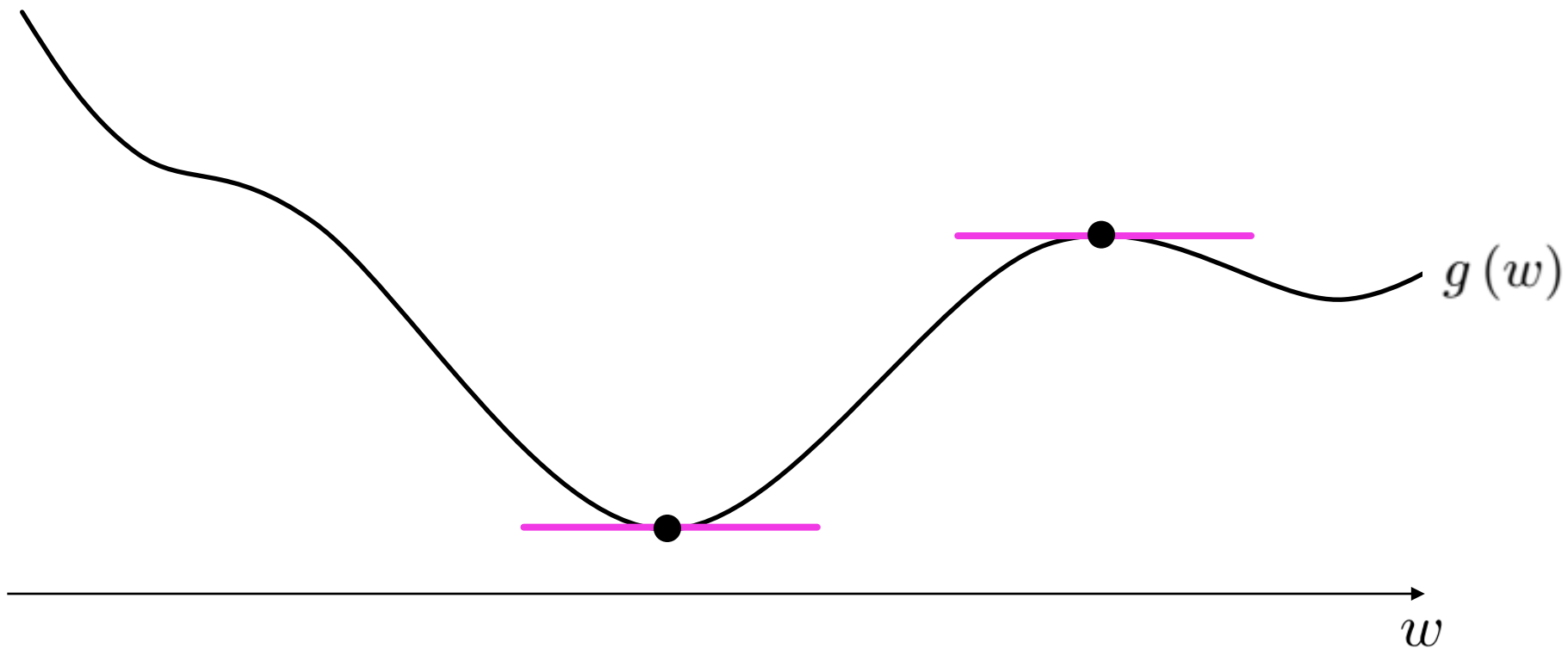
# the derivative

- differentiable function $g\left(w\right)$ with single input $w$
- derivative $g'\left(w\right)$ gives the slope of tangent line at $w$



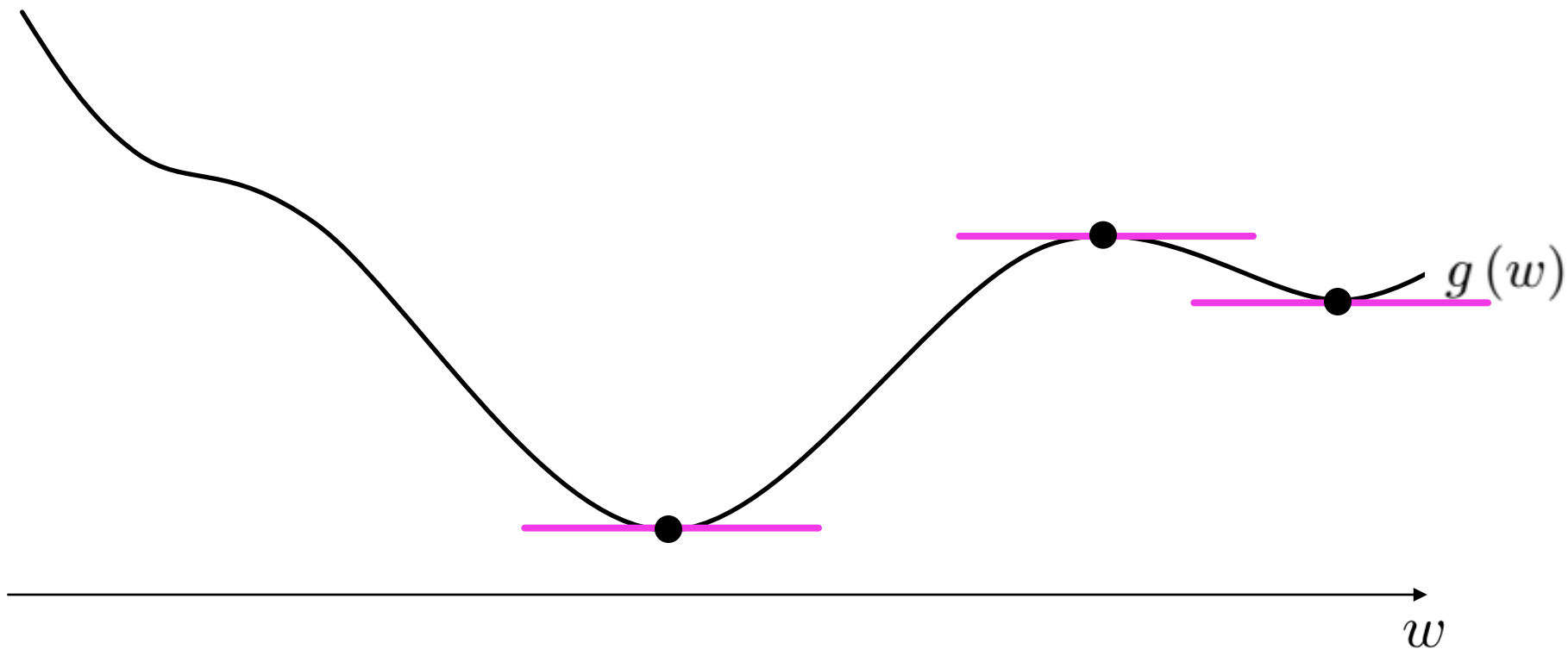
# the derivative

- differentiable function $g\,(w)$ with single input $w$
- derivative $g'\,(w)$ gives the slope of tangent line at $w$



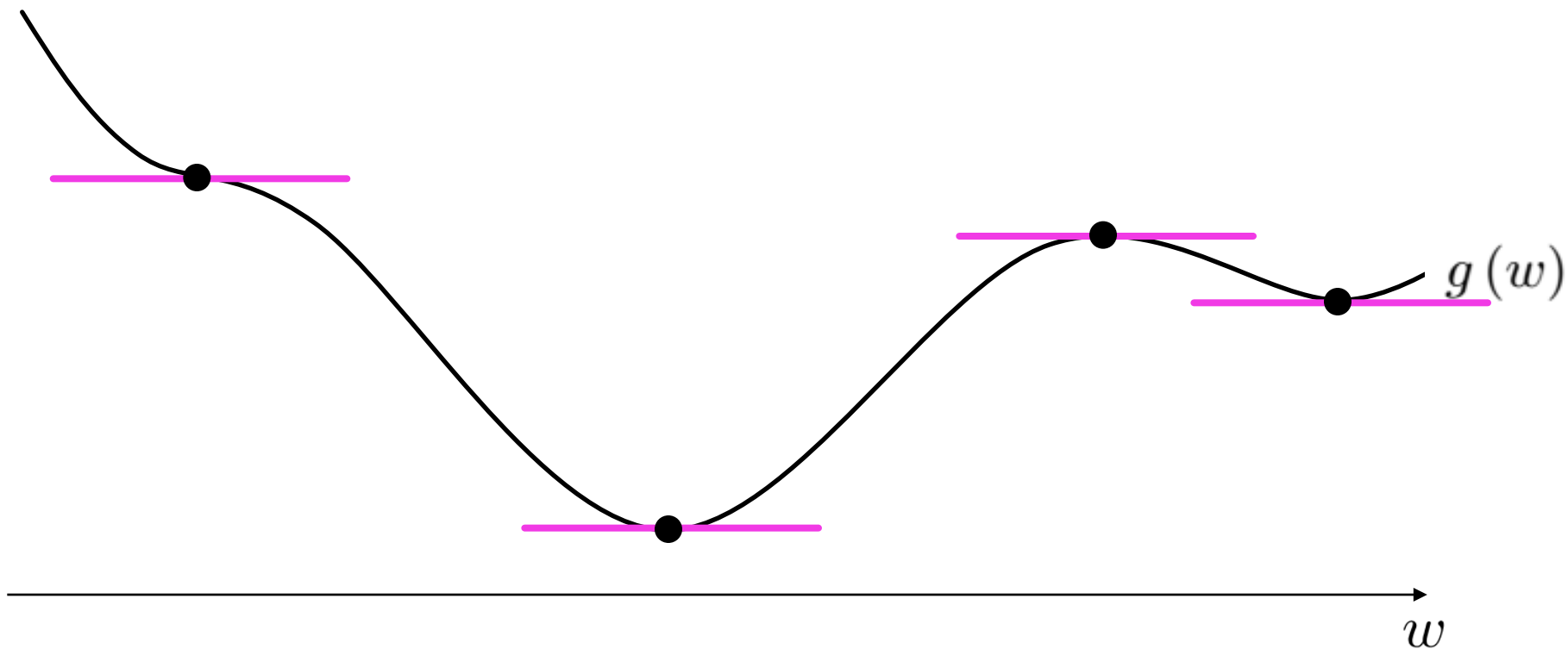
# the derivative

- differentiable function $g\,(w)$ with single input $w$
- derivative $g'\,(w)$ gives the slope of tangent line at $w$



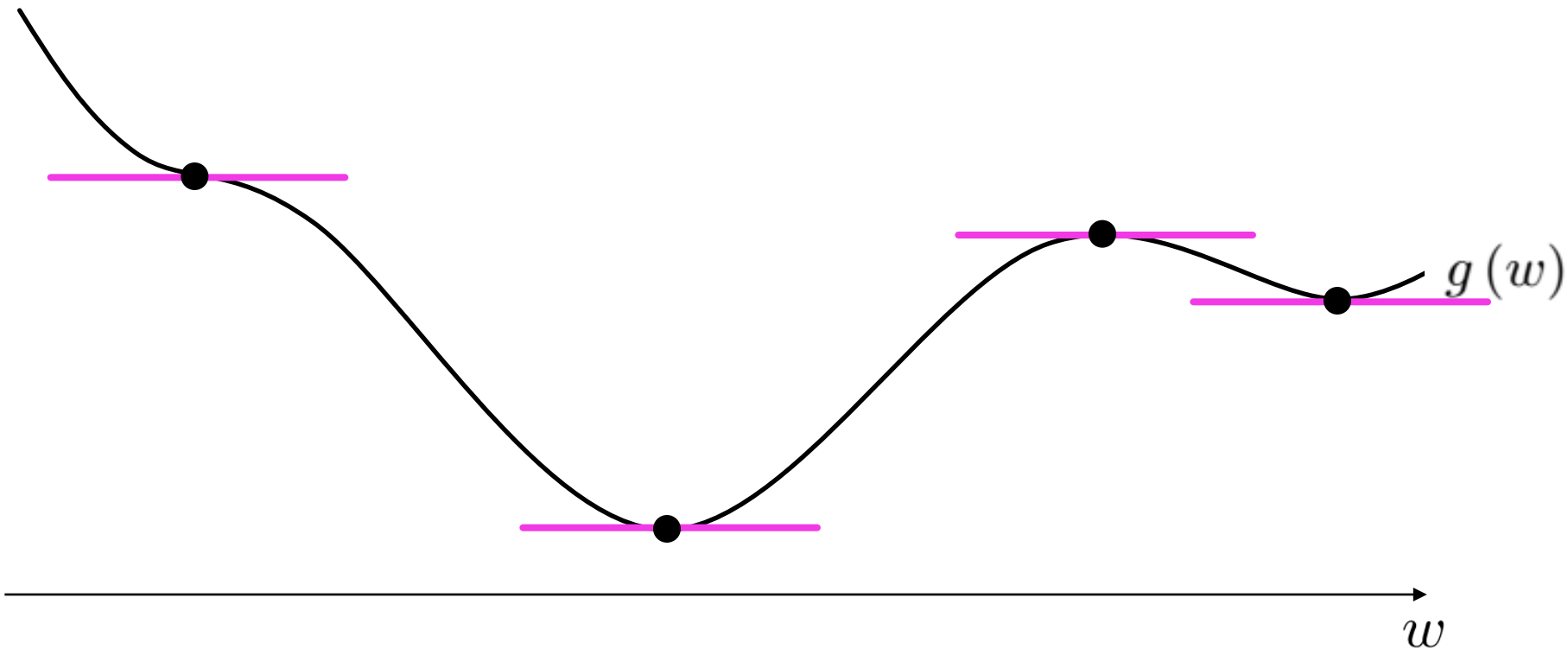
# the derivative

- differentiable function $g\,(w)$ with single input $w$
- derivative $g'\,(w)$ gives the slope of tangent line at $w$



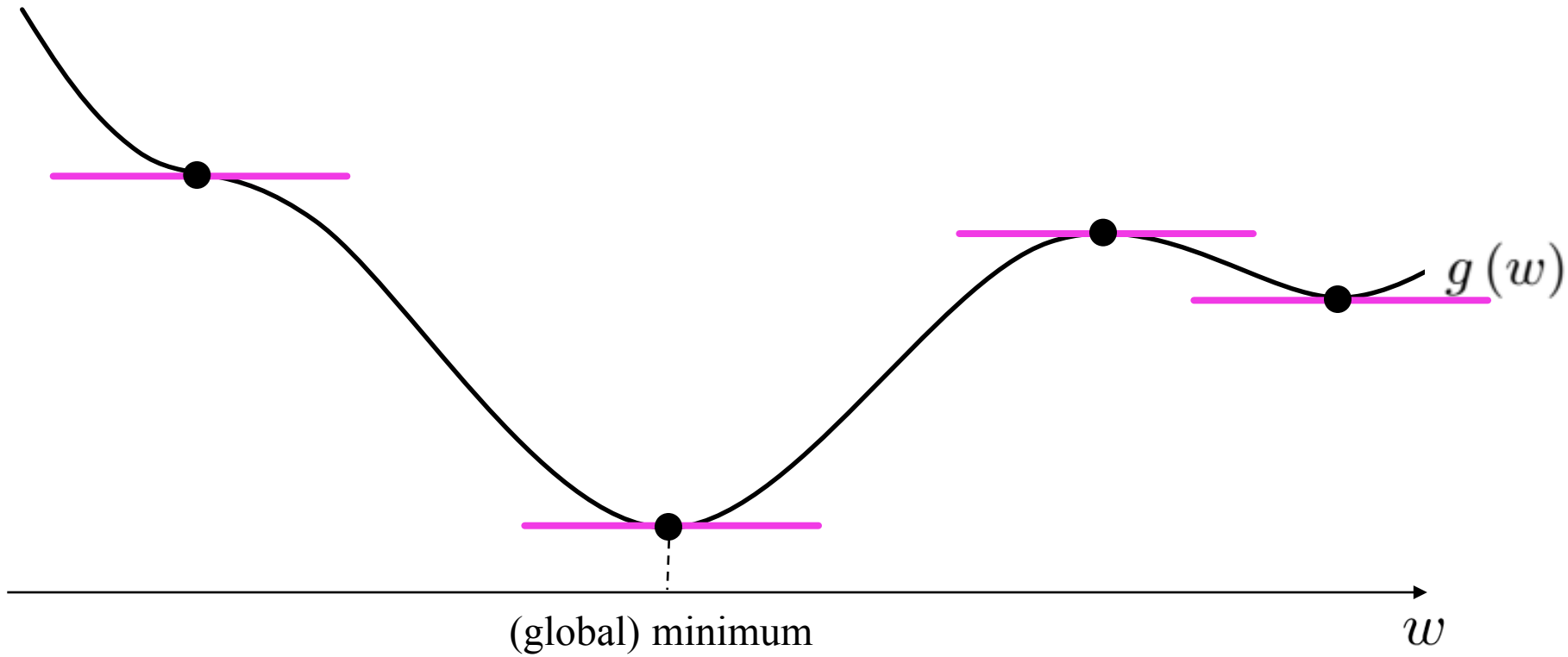
# the derivative

- differentiable function $g\,(w)$ with single input w
- derivative $g'\,(w)$ gives the slope of tangent line
- points *w* where $g'\,(w) = 0$ are referred to as ***stationary points***



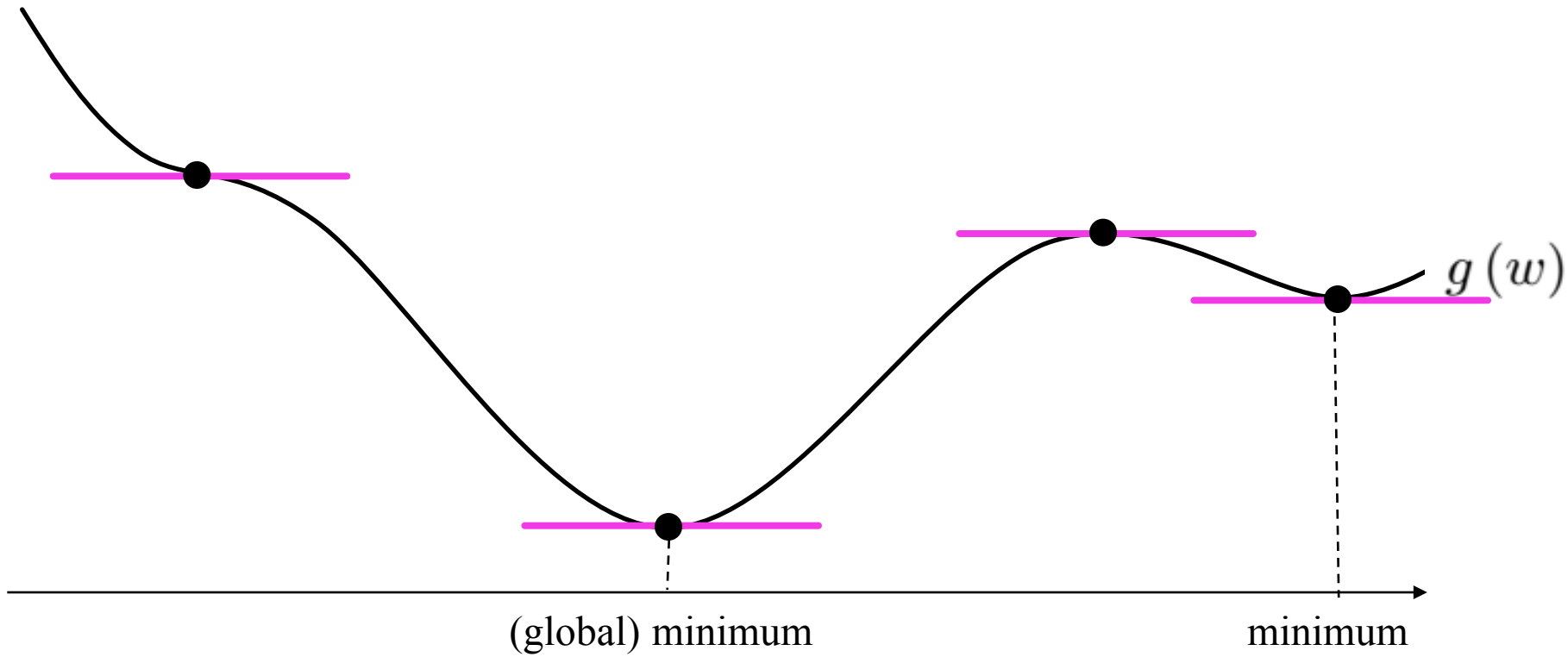
# the derivative

- differentiable function $g\,(w)$ with single input w
- derivative $g'\,(w)$ gives the slope of tangent line
- points *w* where $g'\,(w) = 0$ are referred to as ***stationary points***



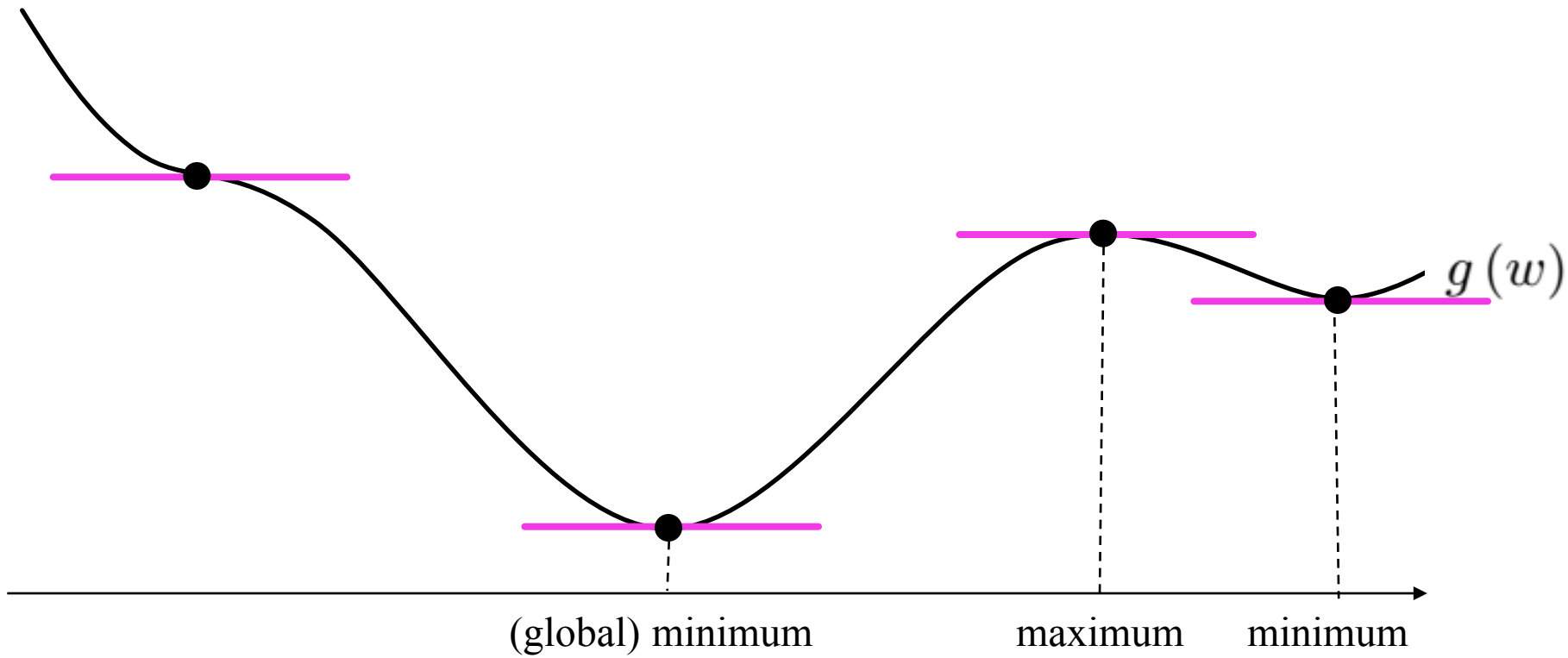
# the derivative

- differentiable function $g\,(w)$ with single input w
- derivative $g'\,(w)$ gives the slope of tangent line
- points *w* where $g'\,(w) = 0$ are referred to as ***stationary points***



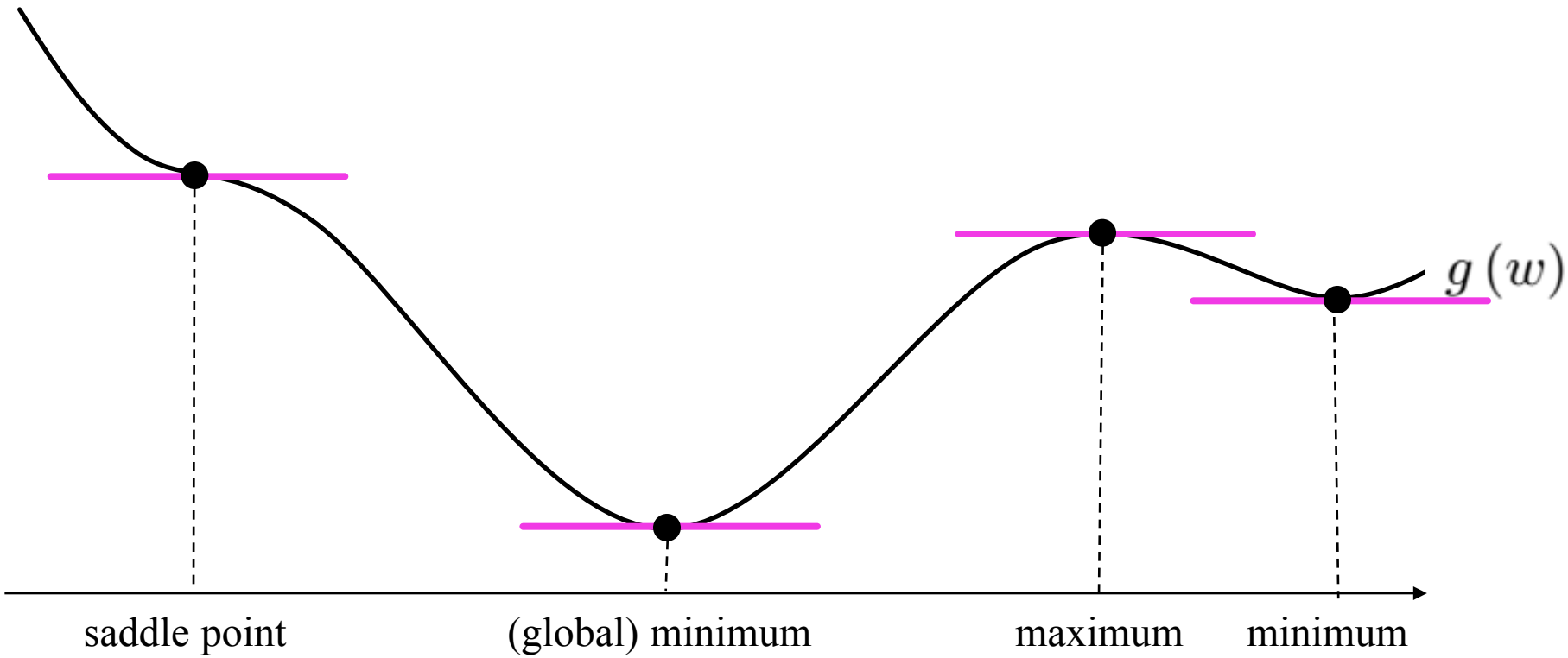
# the derivative

- differentiable function $g\,(w)$ with single input w
- derivative $g'\,(w)$ gives the slope of tangent line
- points *w* where $g'\,(w) = 0$ are referred to as ***stationary points***



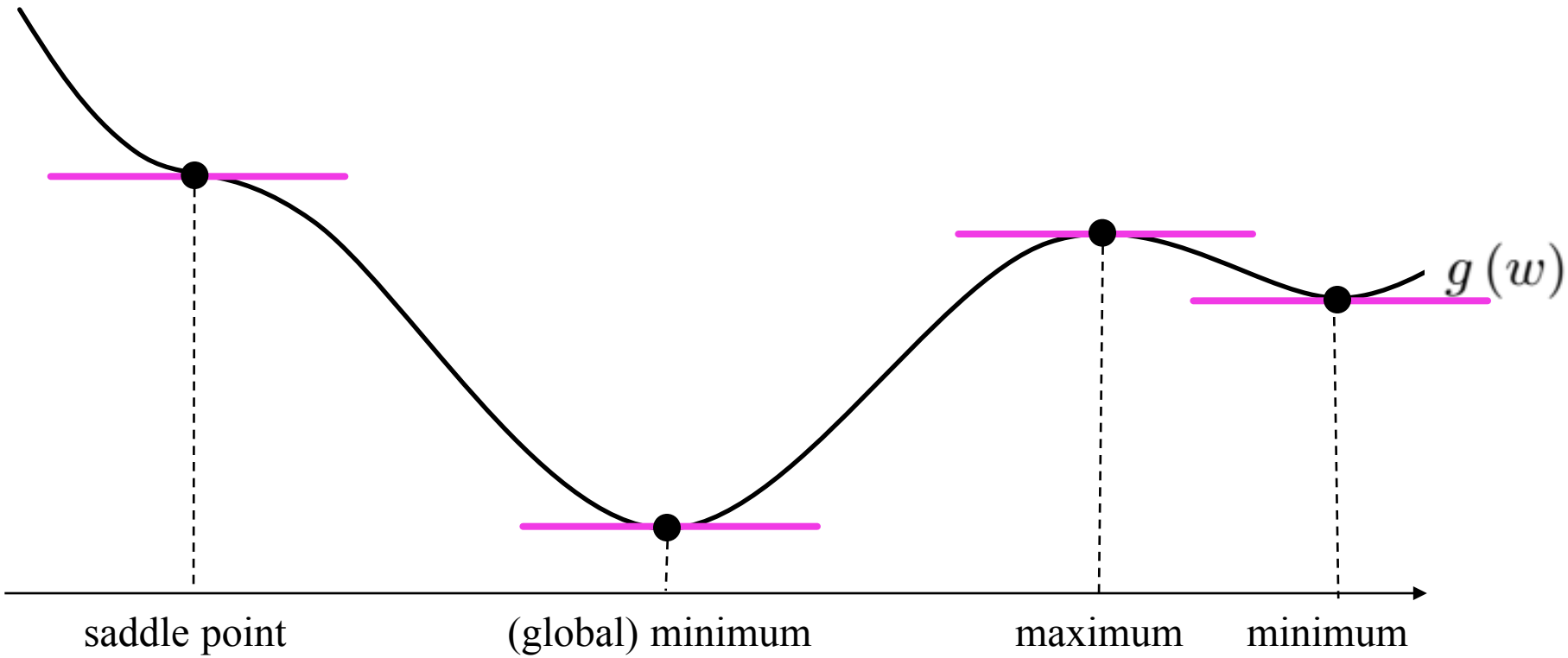
# the derivative

- differentiable function $g\,(w)$ with single input w
- derivative $g'\,(w)$ gives the slope of tangent line
- points *w* where $g'\,(w) = 0$ are referred to as ***stationary points***



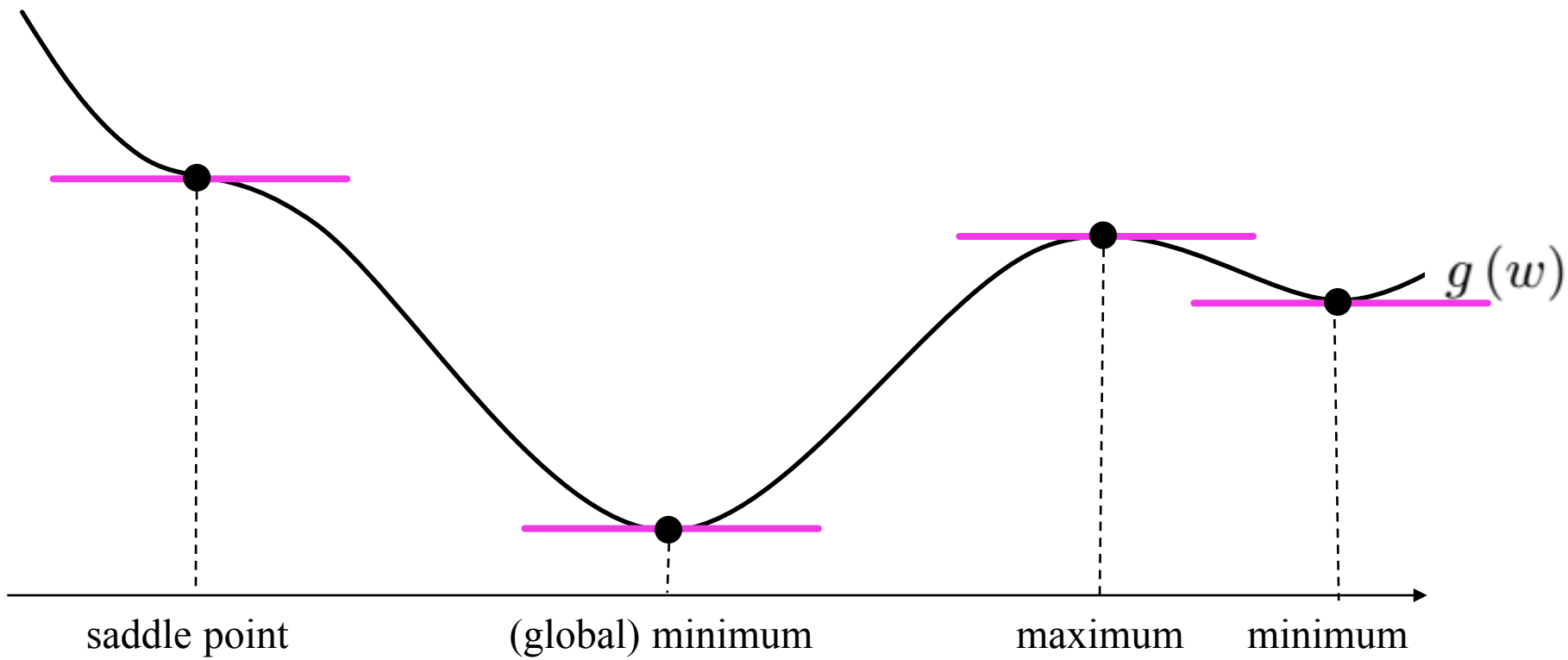
# the derivative

- so not all stationary points are (global) minima for a general cost function



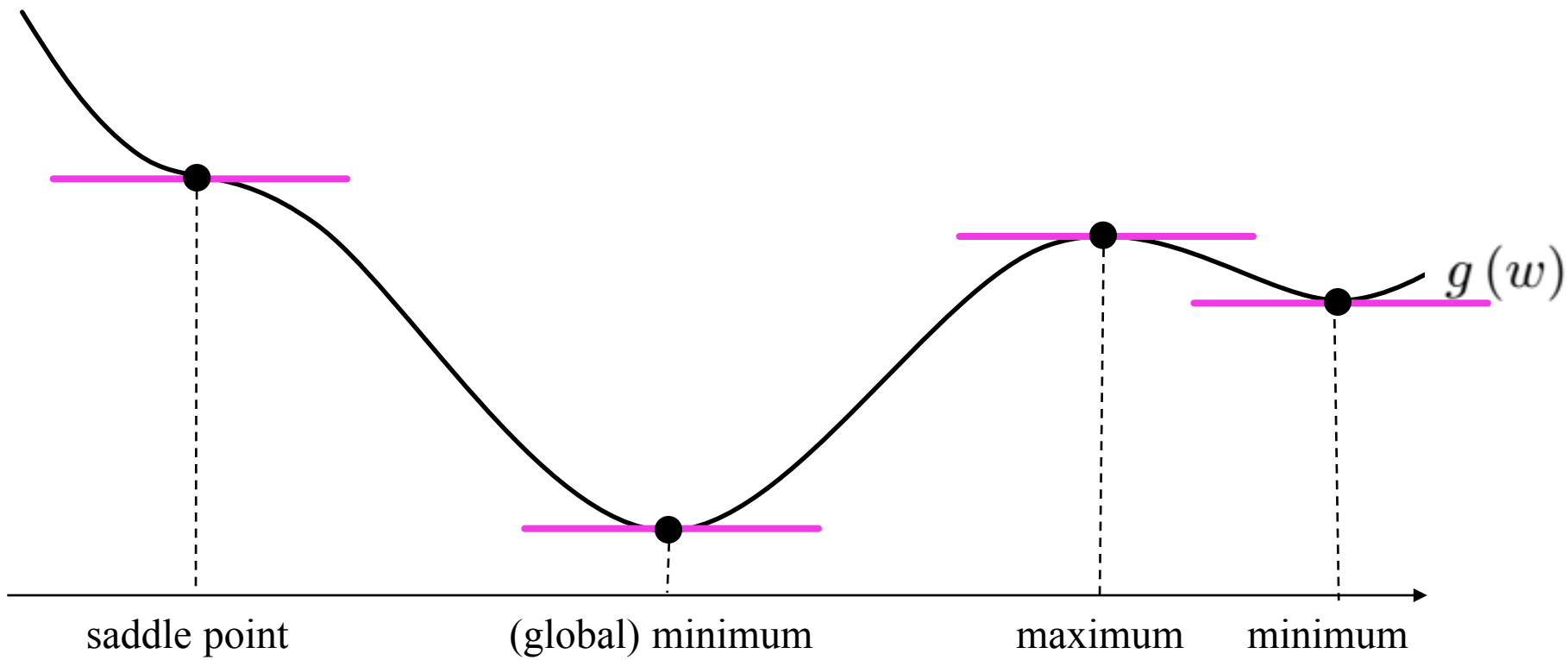
# the derivative

- so not all stationary points are (global) minima for a general cost function
- nonetheless finding them is the most common way to minimize a cost function



# the derivative

- so not all stationary points are (global) minima for a general cost function
- nonetheless gradient descent still effectively used to find global minima



# Demo 7: nonconvex grad

- to the notebook!

# Demo 8: nonlinear classification

- to the notebook!

# Regression and Classification

**Regression** ⇒ summarize a trend or predict future output values

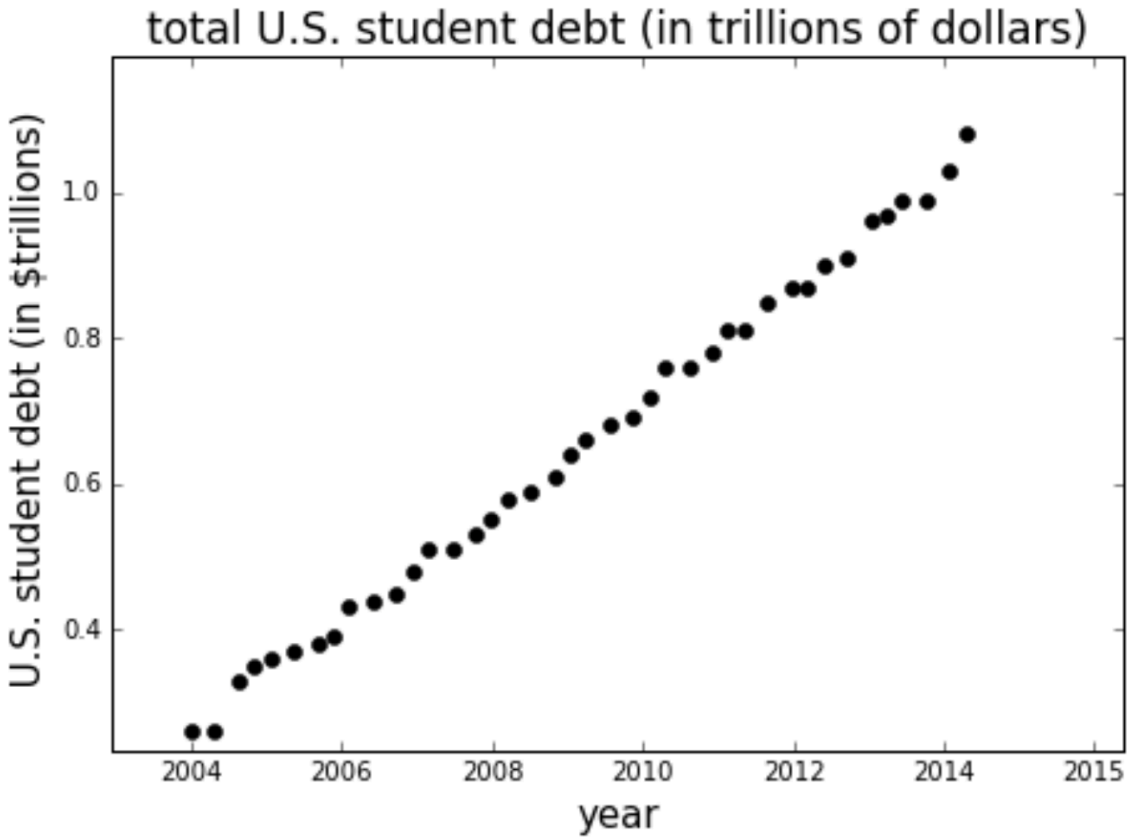



Image taken from [29]

**Regression** ⇒ summarize a trend or predict future output values

Image taken from [29]

**Classification** ⇒ distinguish between different classes of data

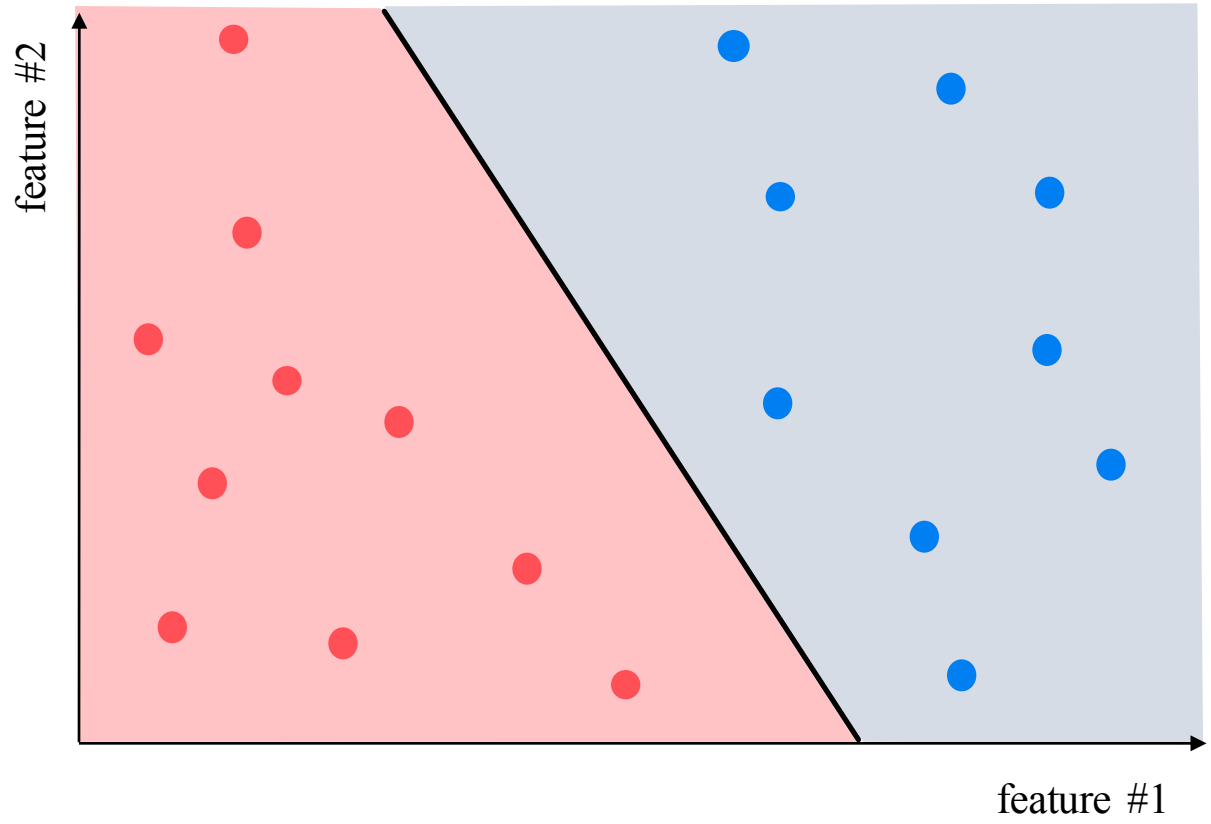



Image taken from [29]

**Classification** ⇒ distinguish between different classes of data

- **Object detection and recognition**:
  - for identification, organizing/taking better photos

**Classification** ⇒ distinguish between different classes of data

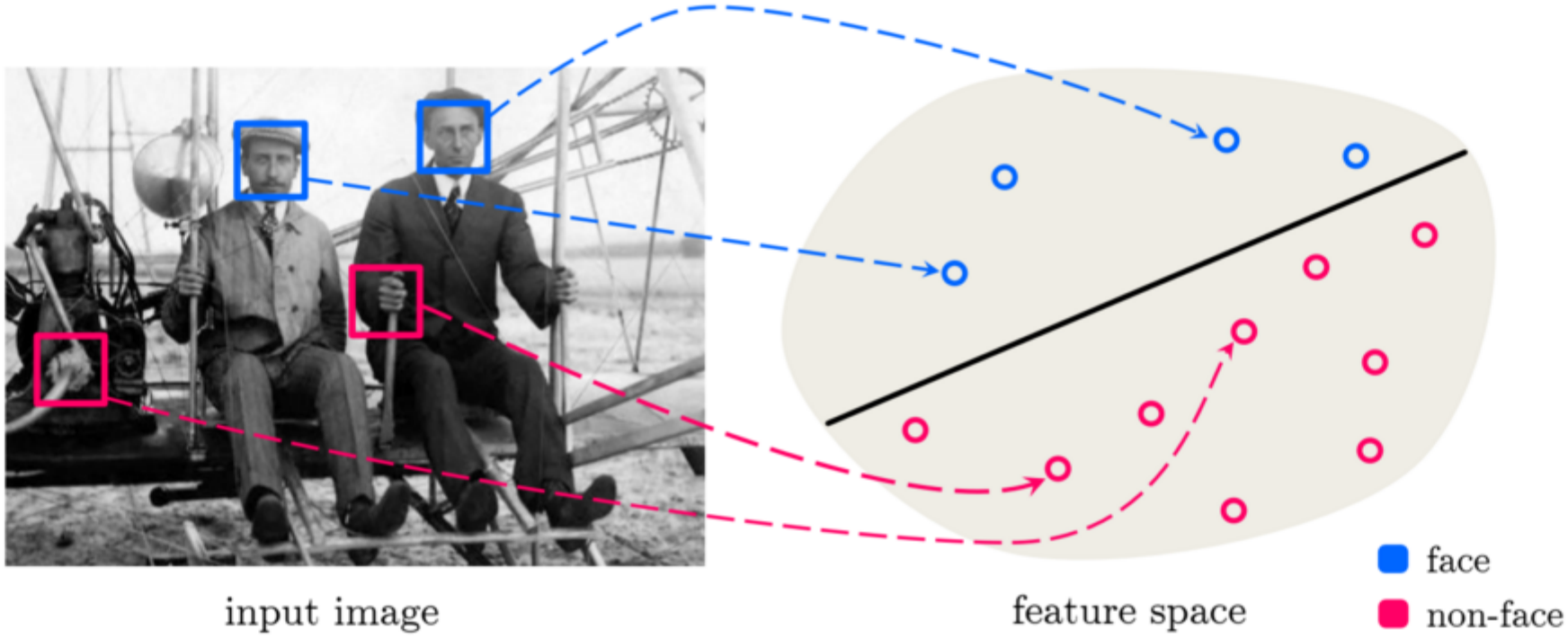



Image taken from [29]

**Classification** ⇒ distinguish between different classes of data

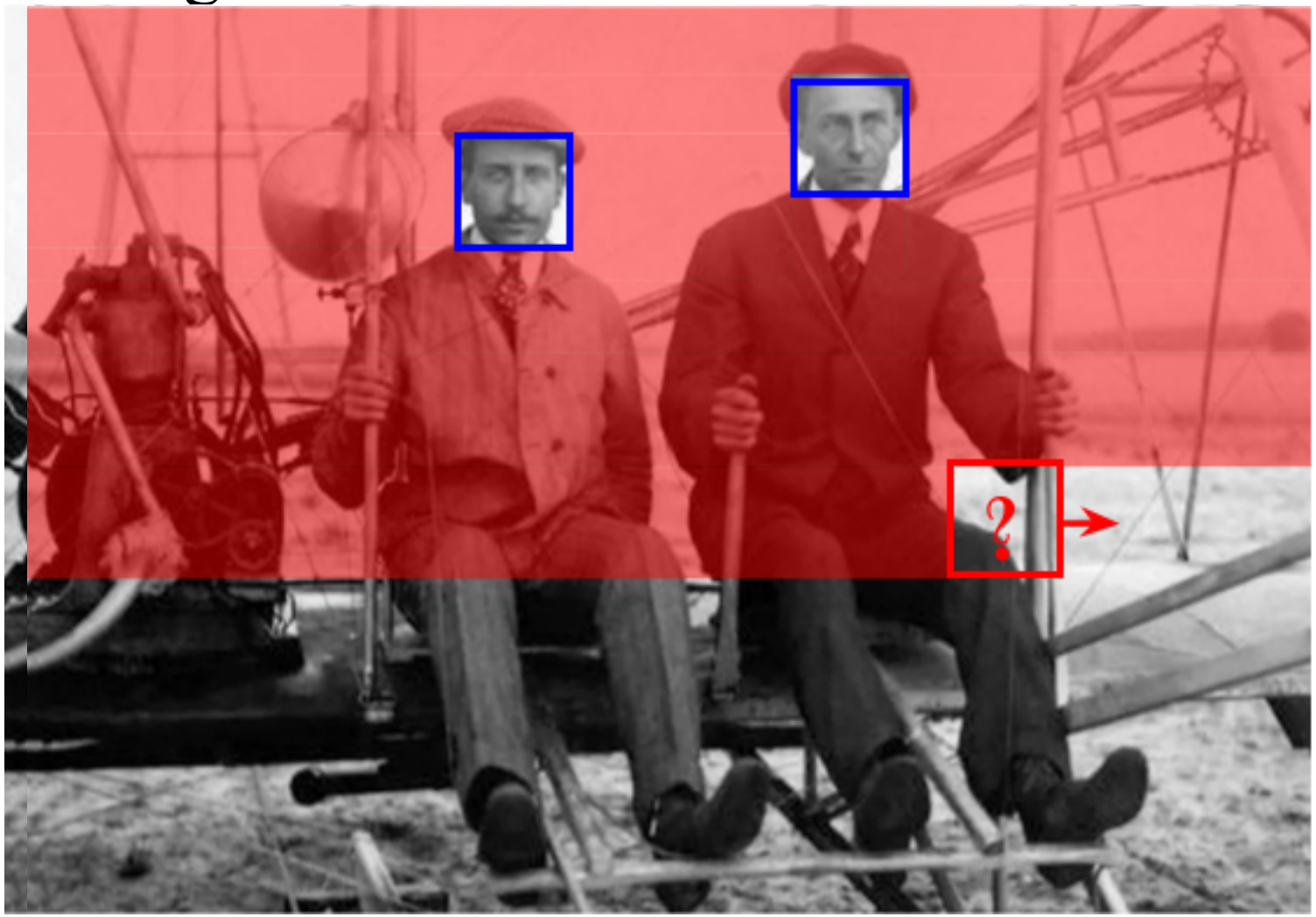



Image taken from [29]

**Classification** ⇒ distinguish between different classes of data

- **Object detection and recognition**:
  - for human-computer interaction

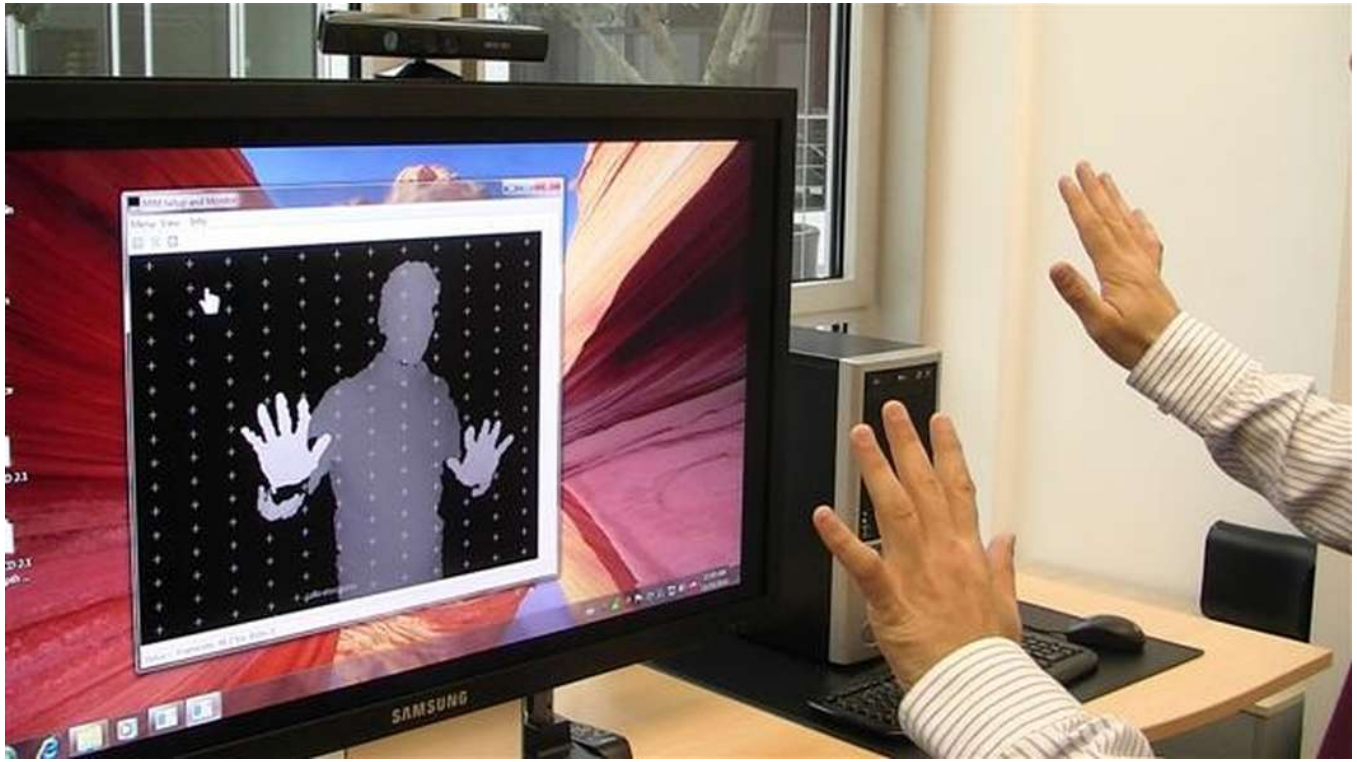



Image taken from http://www.gizmag.com/kinect-used-to-control-windows-7/16997/

**Classification** ⇒ distinguish between different classes of data

- **Speech recognition**:
  - for human-computer interaction



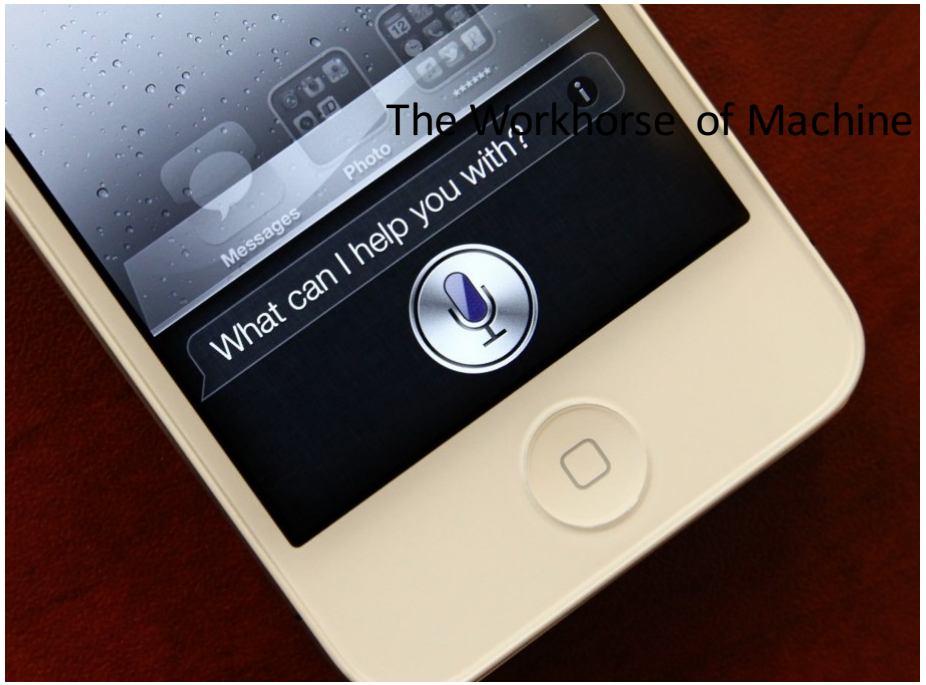




Image taken from (left) http://tiny.cc/n25z8x and (right) www.amazon.com

# References

[1] Jerem Watt, Reza Borani, and Aggelos Katsaggelos. Machine Learning Refined: Foundations, Algorithms, and Applications. Cambridge University Press, 2016.

[2] Donghoon Lee, Wilbert Van der Klaauw, Andrew Haughwout, Meta Brown, and Joelle Scally. Measuring student debt and its performance. *FRB of New York Staff Report*, (668), 2014.