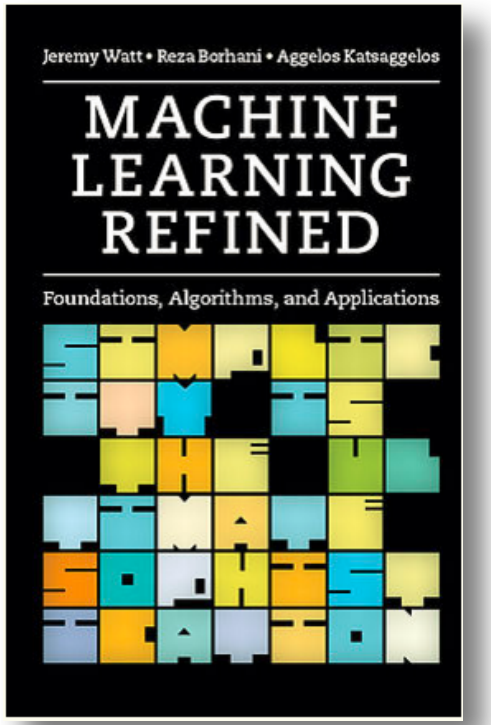
Gradient Descent: Workhorse (Algorithm) of Machine Learning

What is this talk based on?

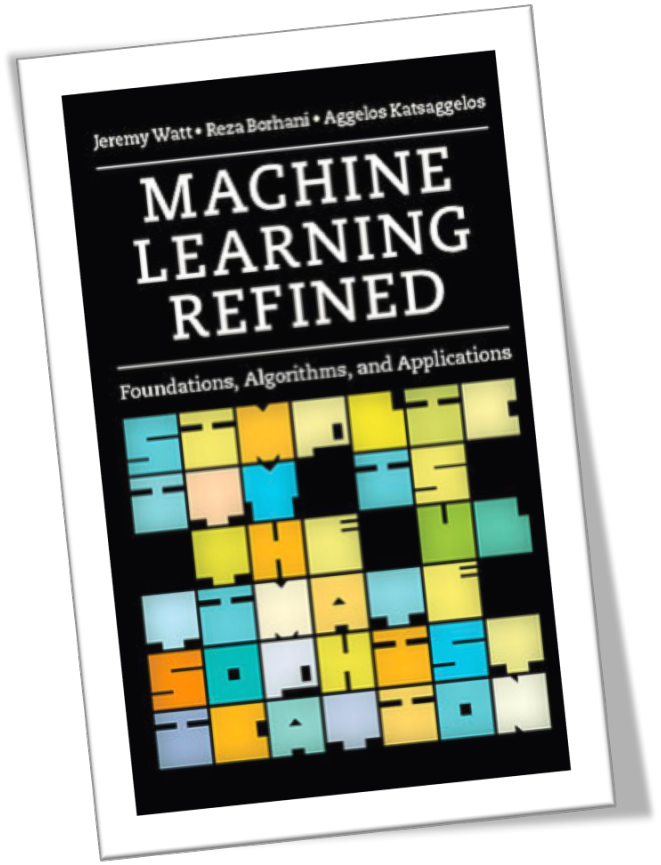


A new ML textbook!
Cambridge University Press
July 2016

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)

MACHINE LEARNING: ALGORITHMS AND APPLICATIONS

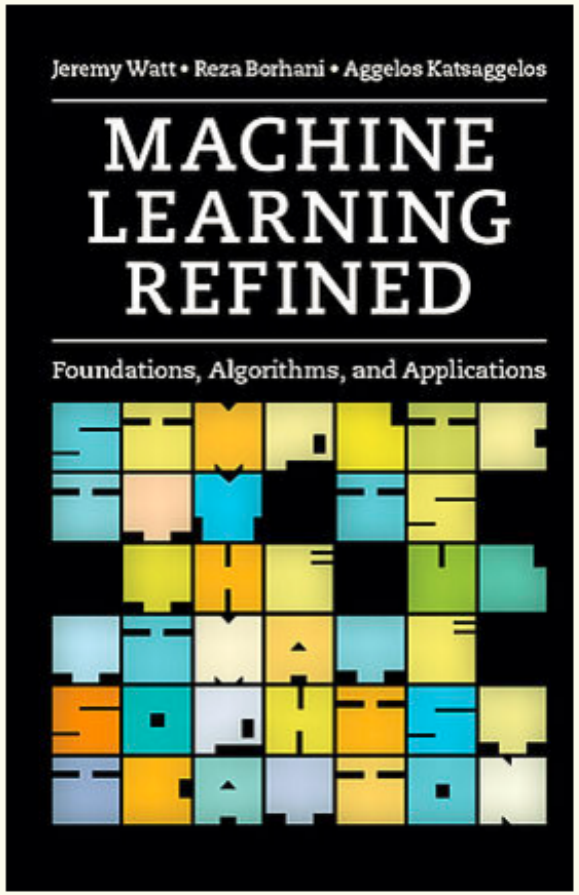
CHICAGO

July 11 - August 17
Mondays & Wednesdays
6:30pm - 9:30pm
$2,500

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.

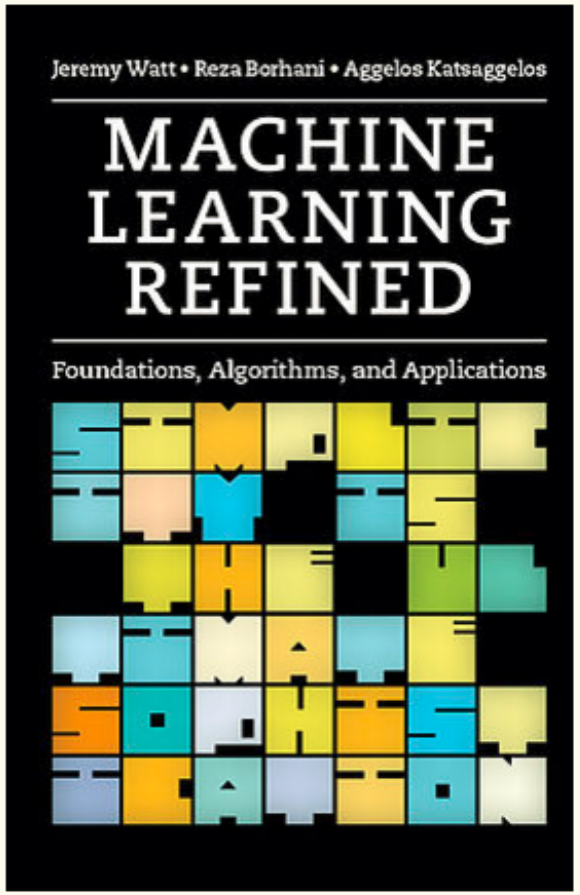
To learn more about what makes our textbook so great click here, and then see for yourself by downloading free sample chapters via the link below!

click here

Where to download the code?

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 familiar with

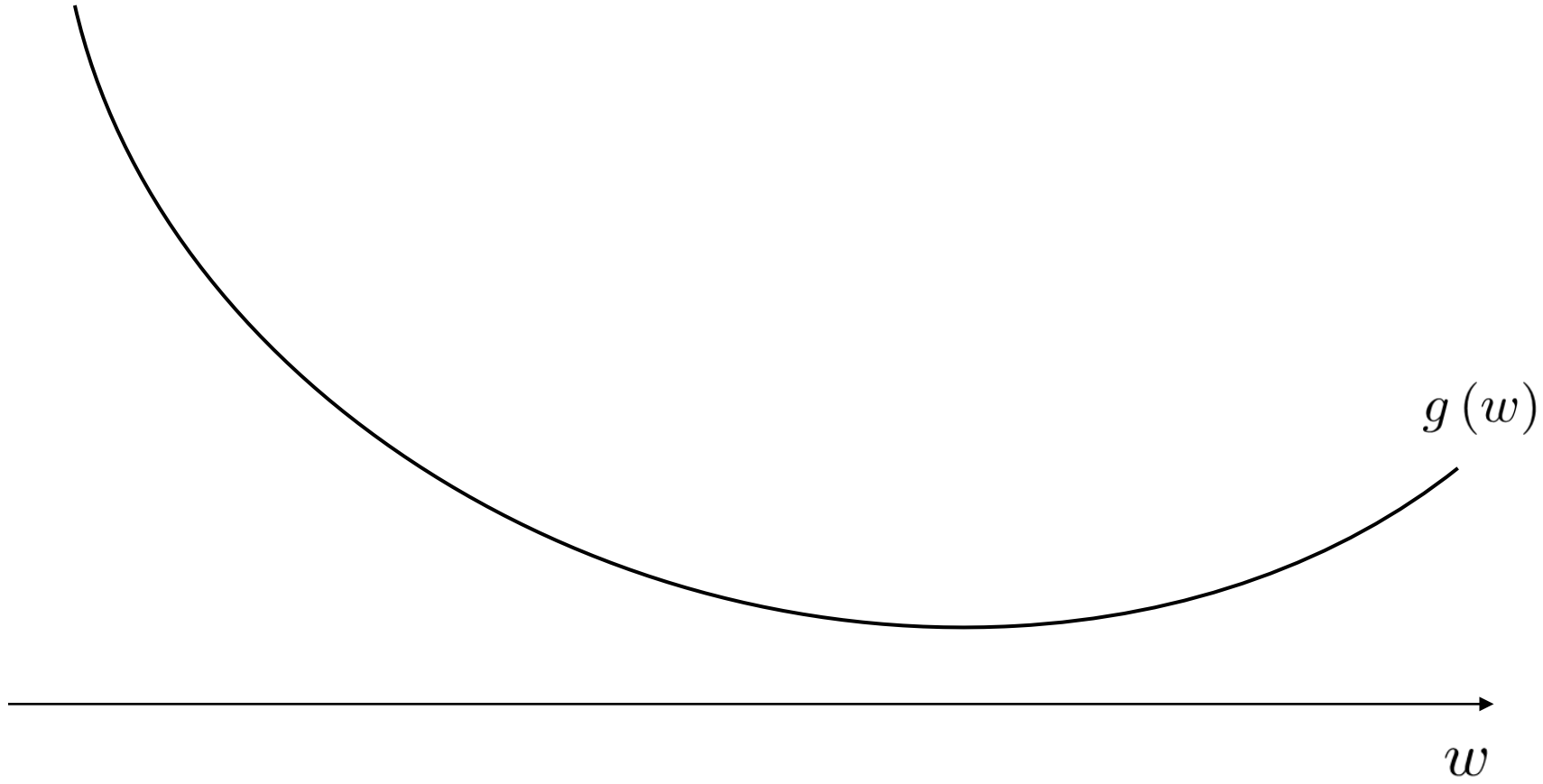
1. basic python programming and some experience with the IPython interactive notebook

2. understand basic mathematical notation (like “f(x)” notation for denoting a function)

3. the notion of a scalar valued derivative (from calculus)

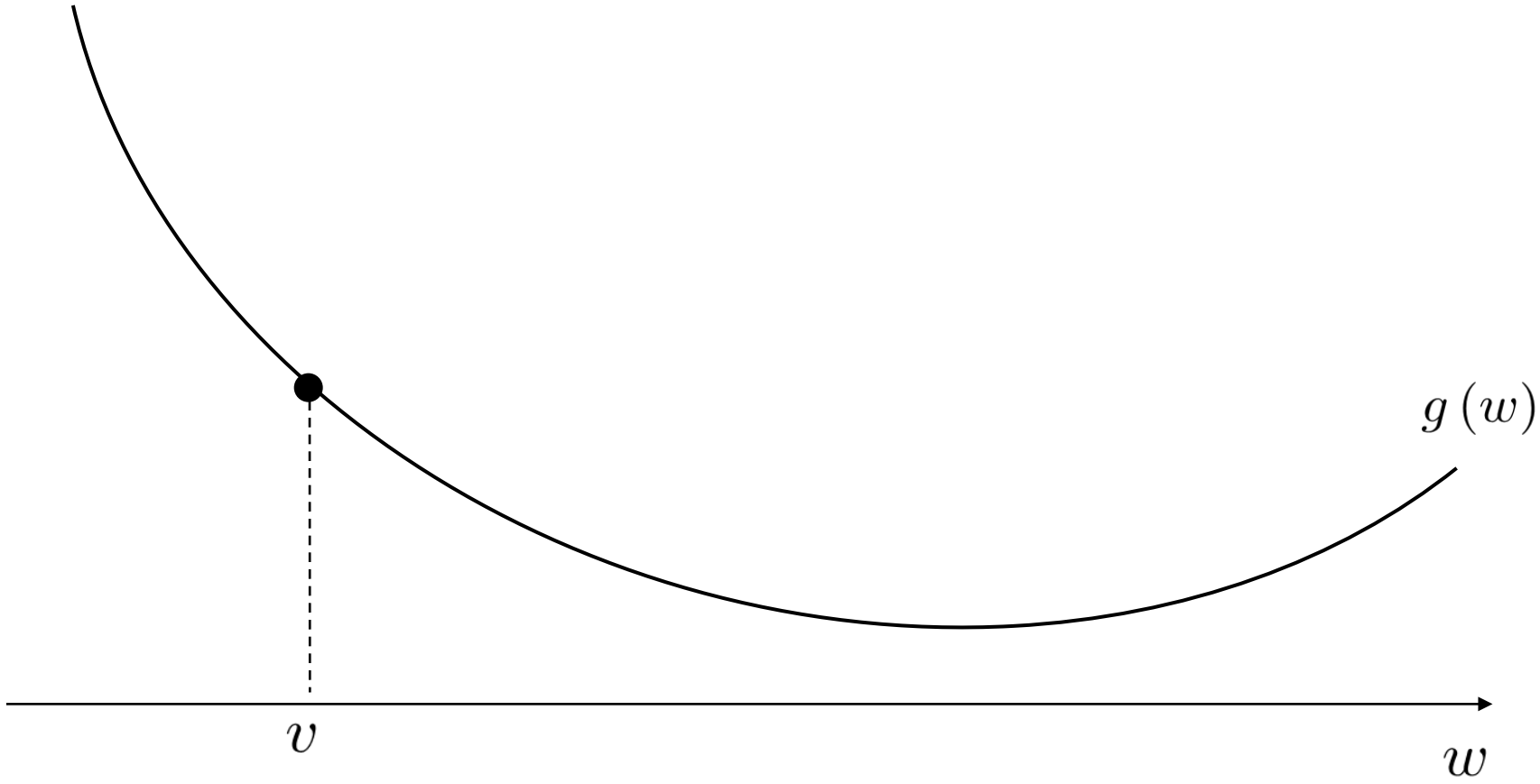
the derivative

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



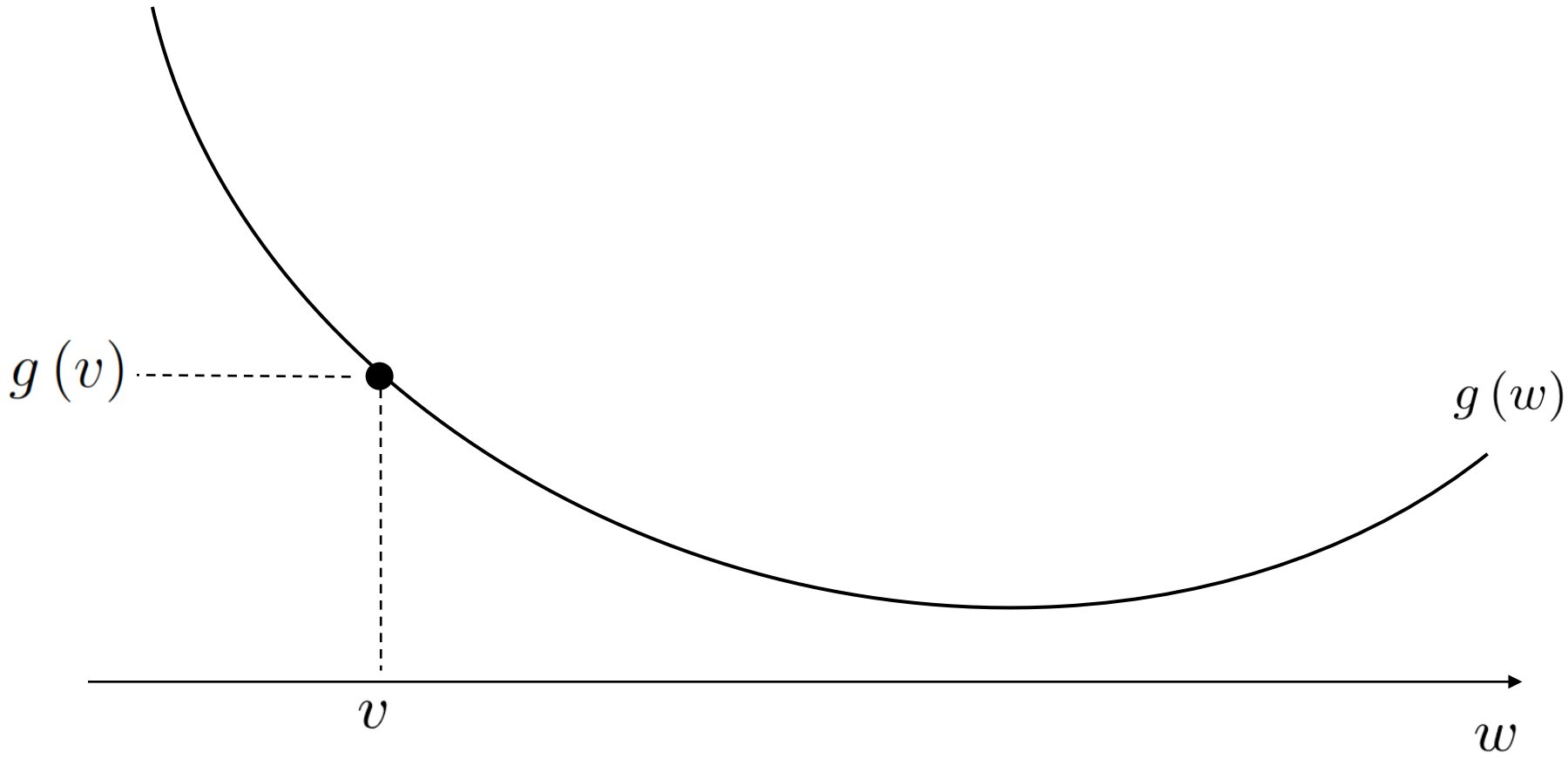
the derivative

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



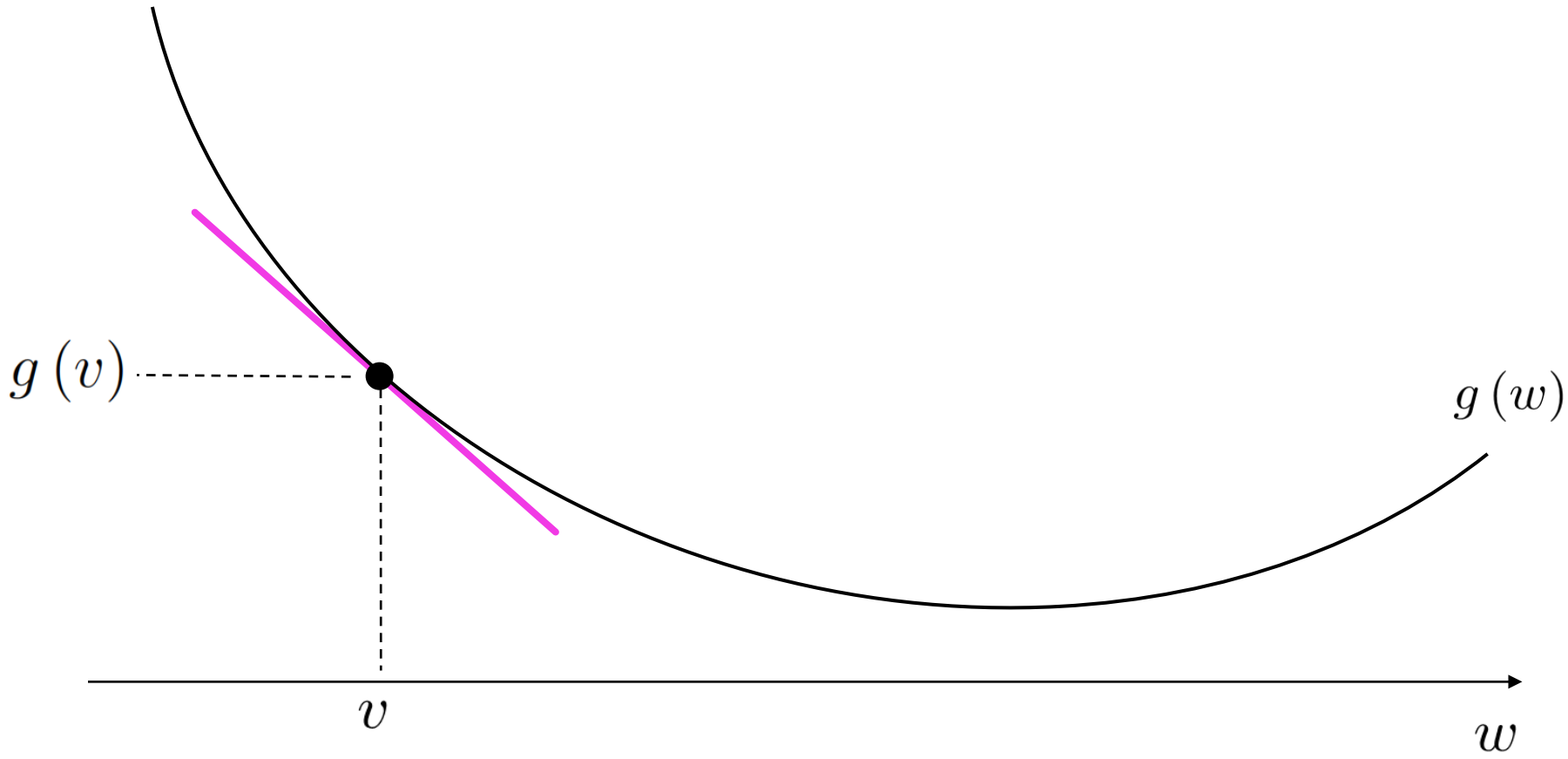
the derivative

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



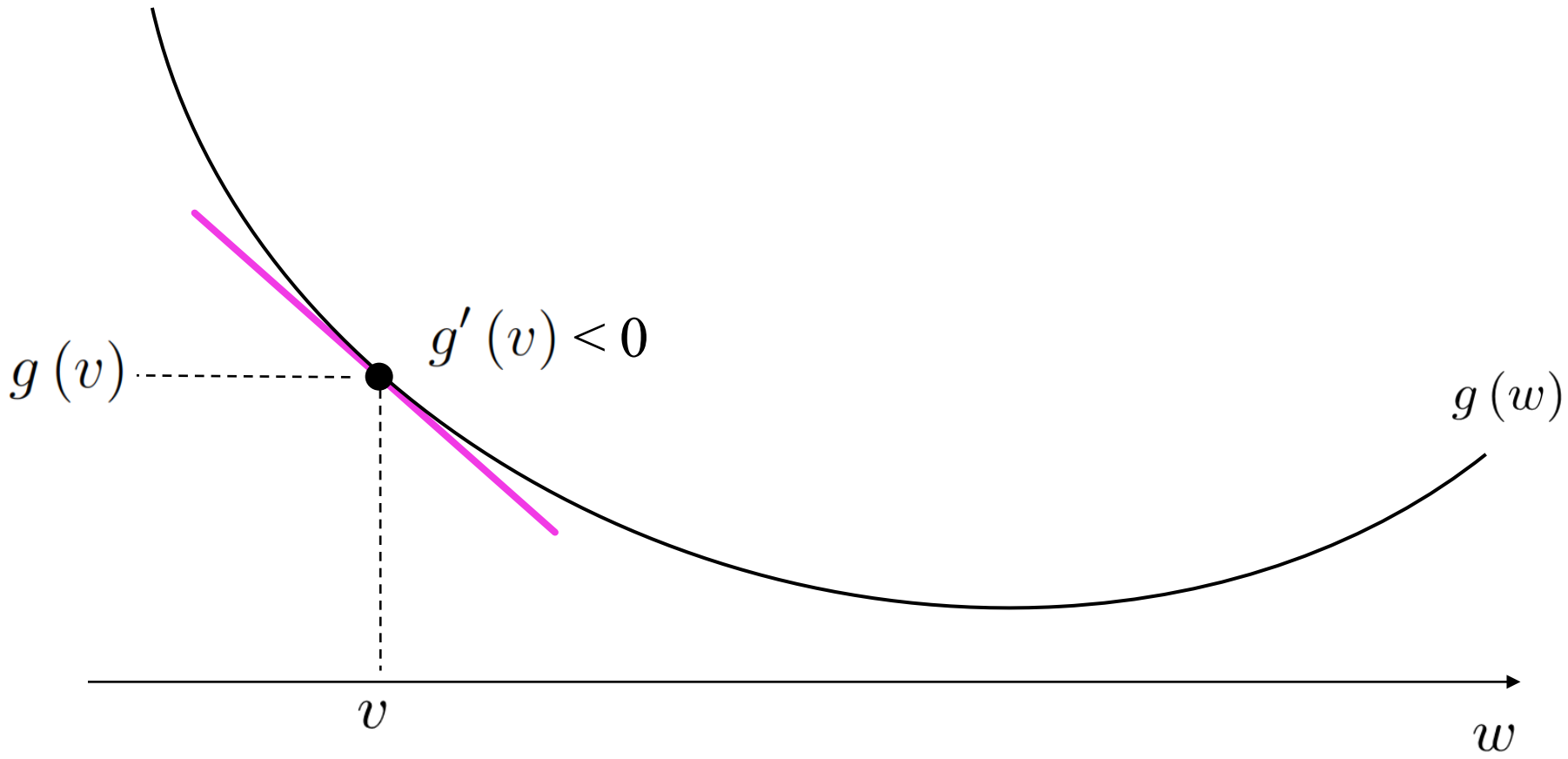
the derivative

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



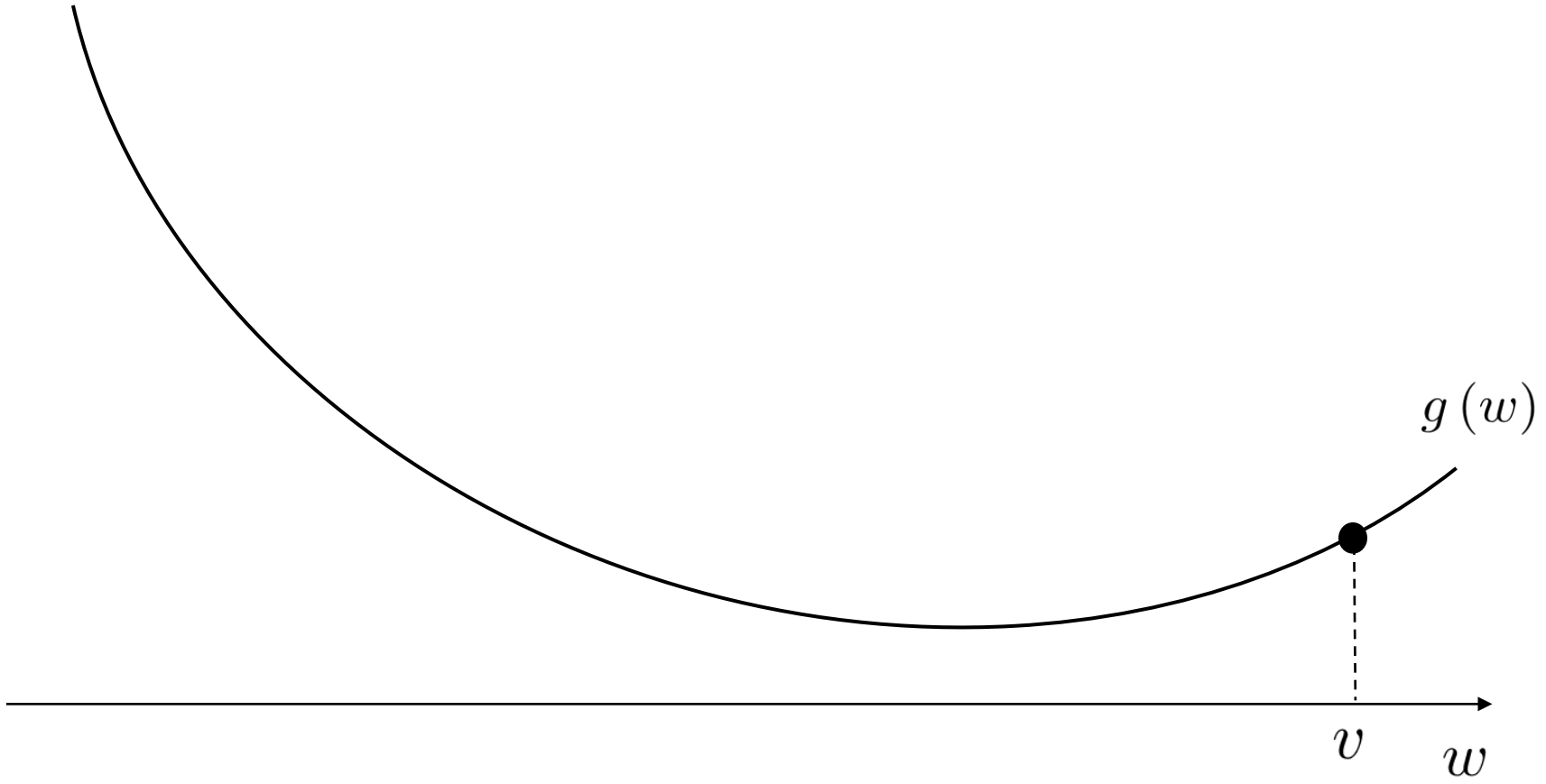
the derivative

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



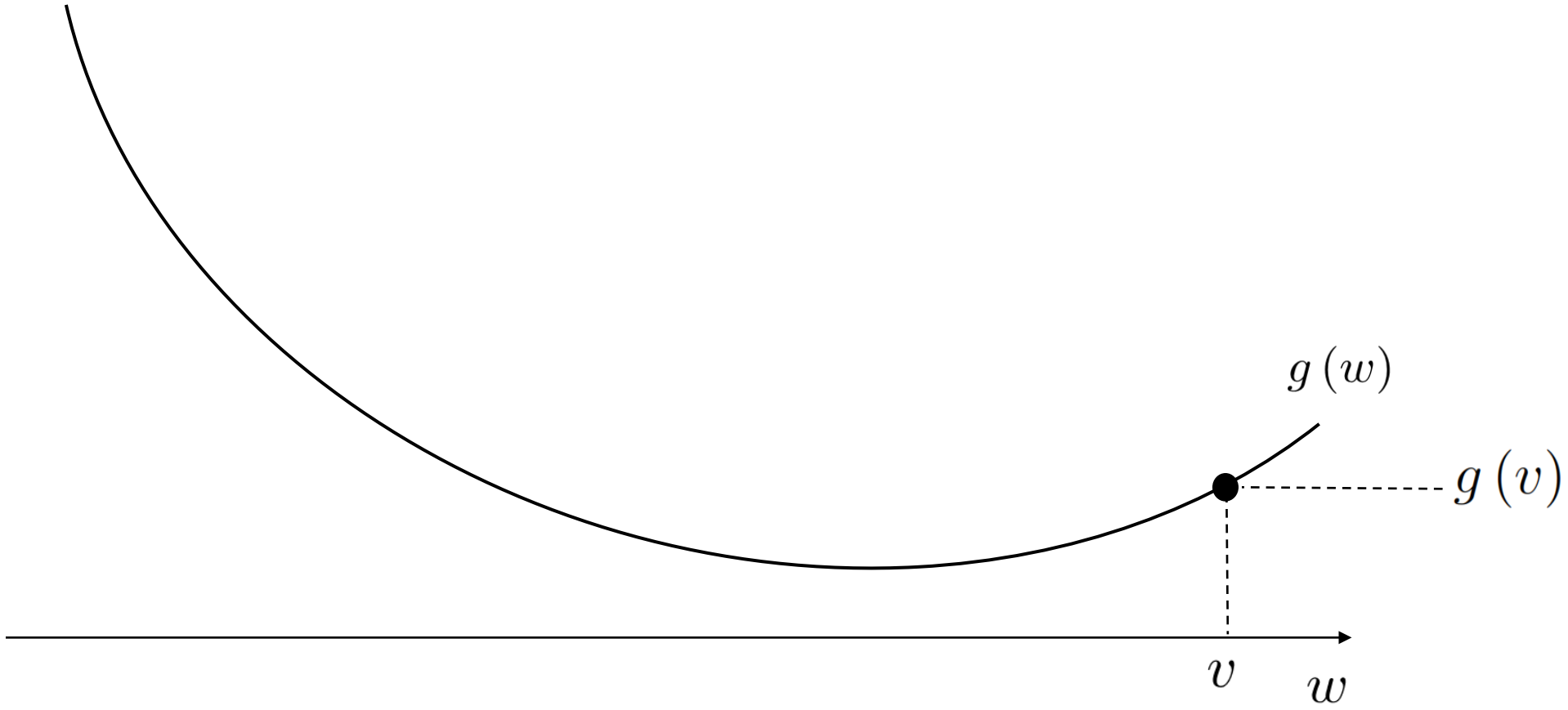
the derivative

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



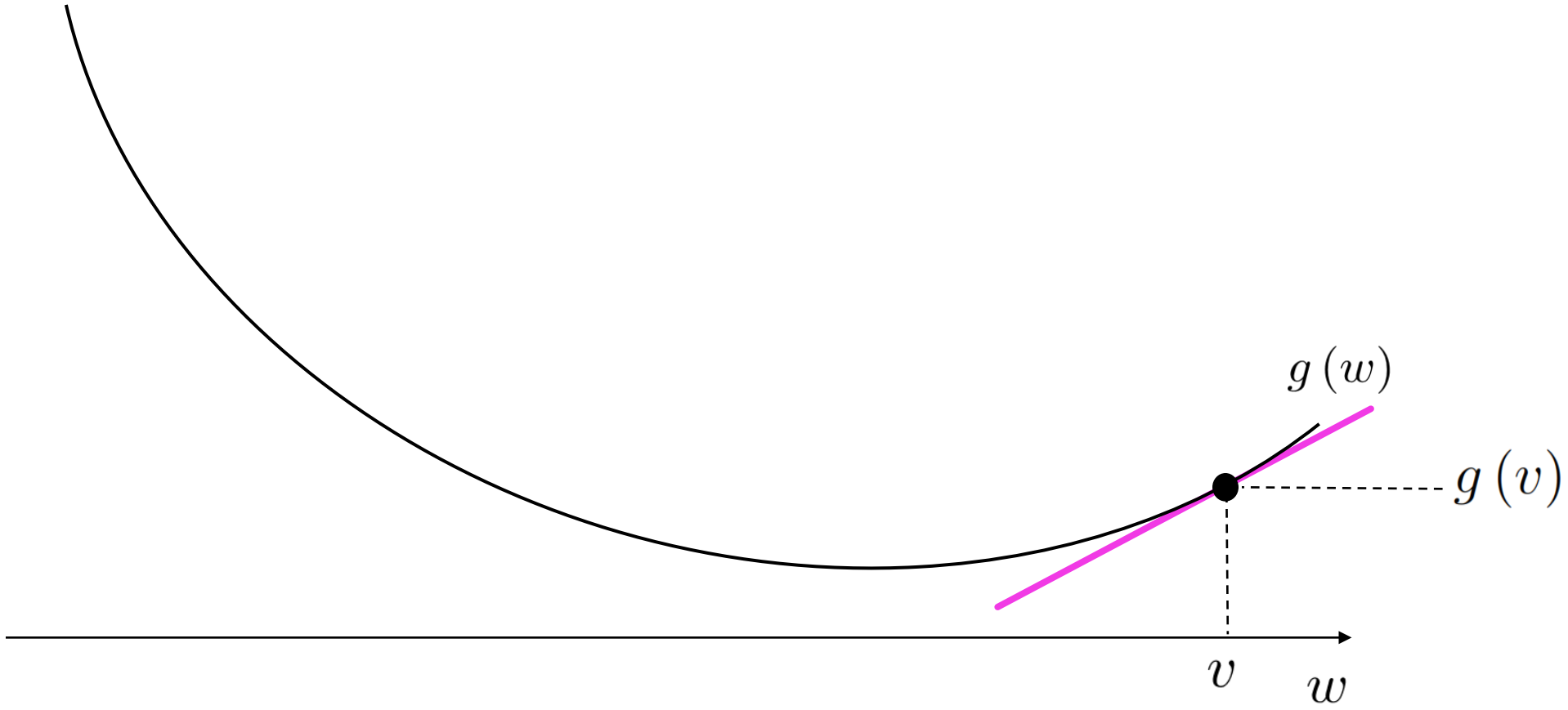
the derivative

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



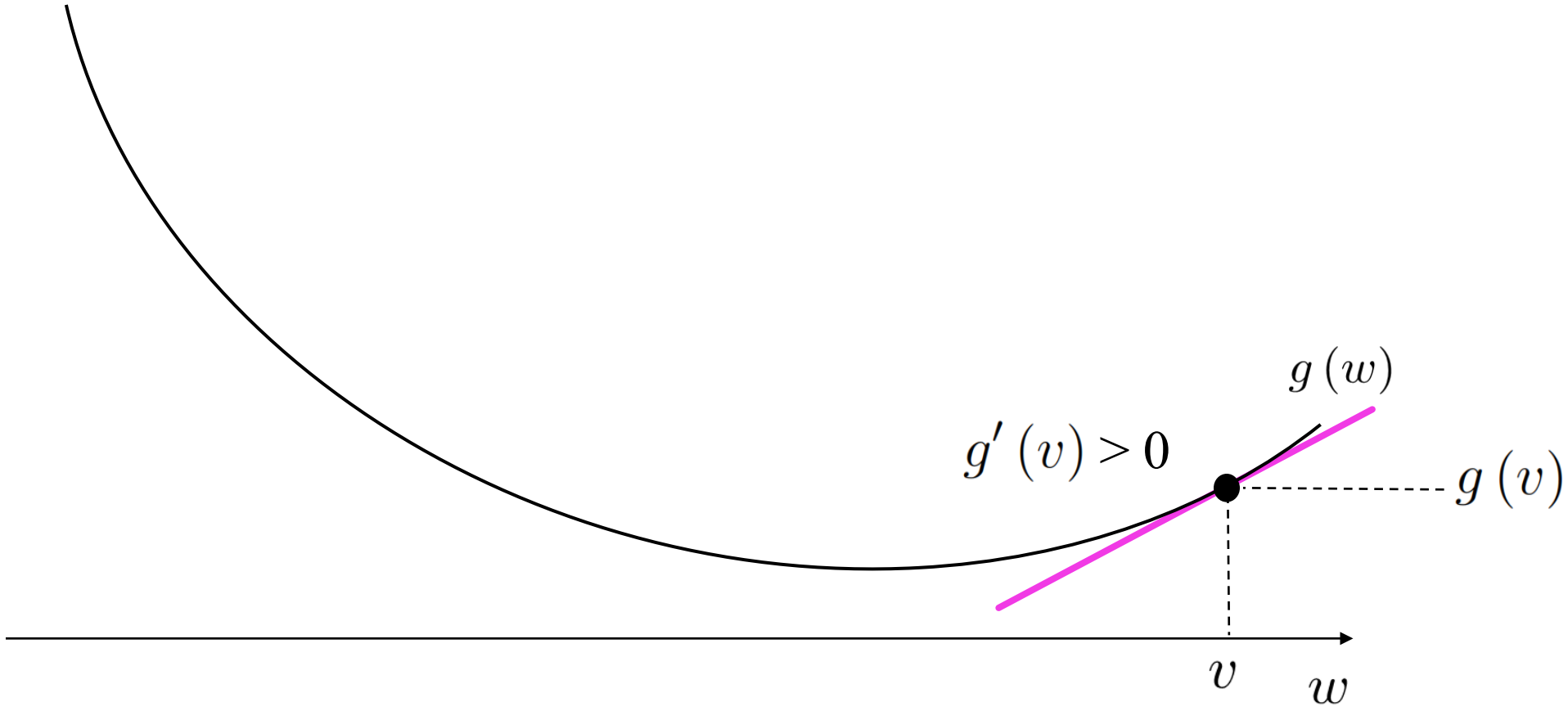
the derivative

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



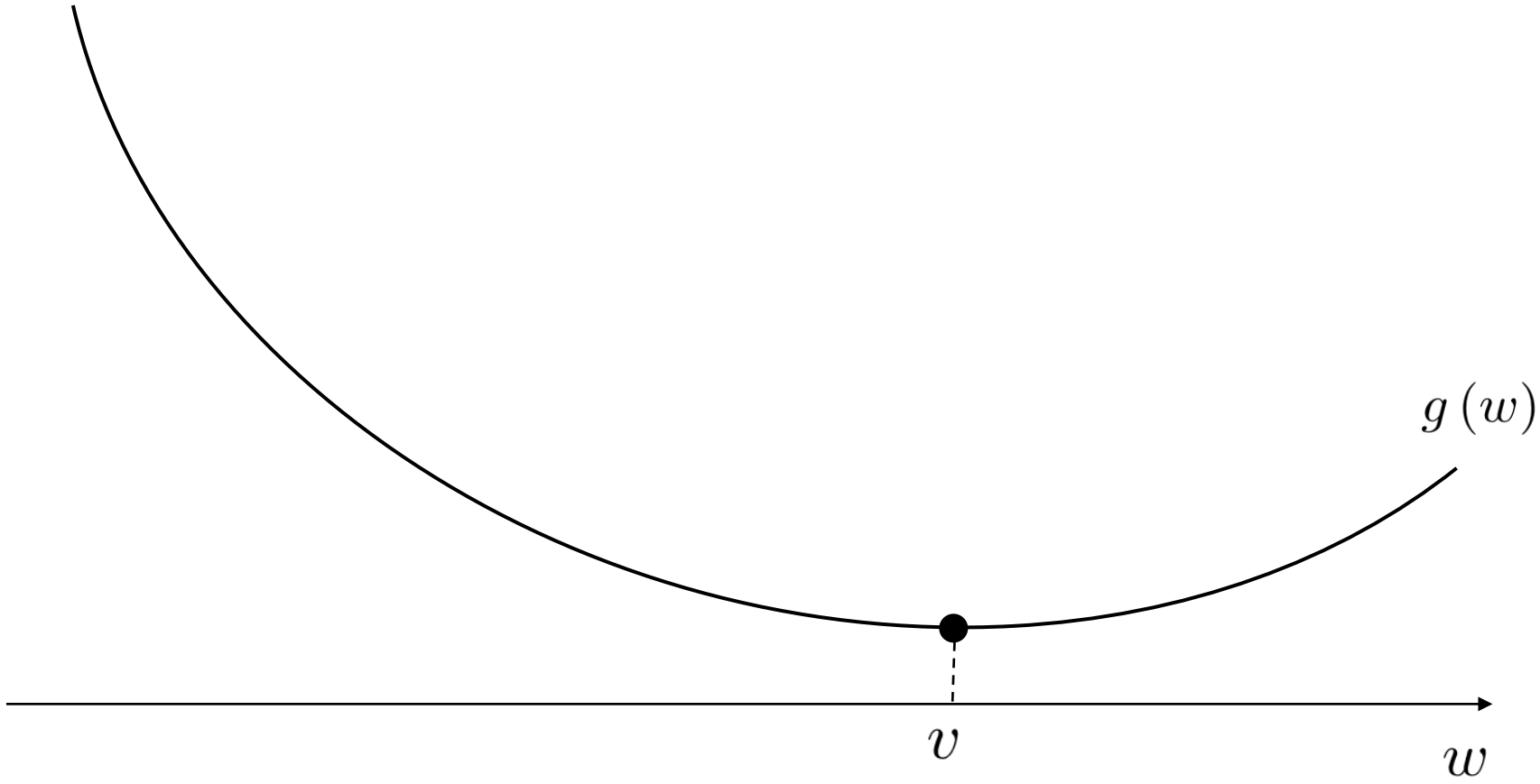
the derivative

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



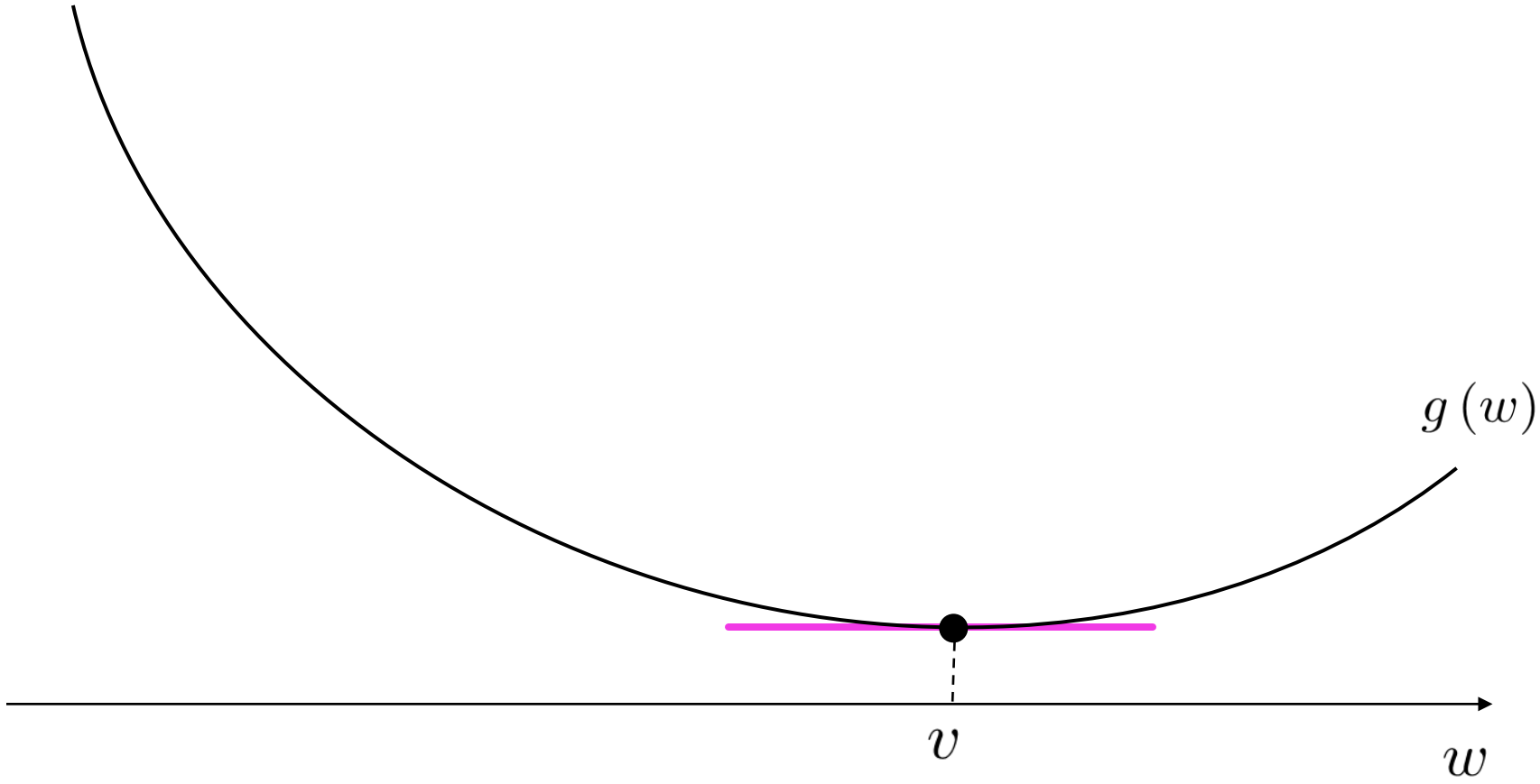
the derivative

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



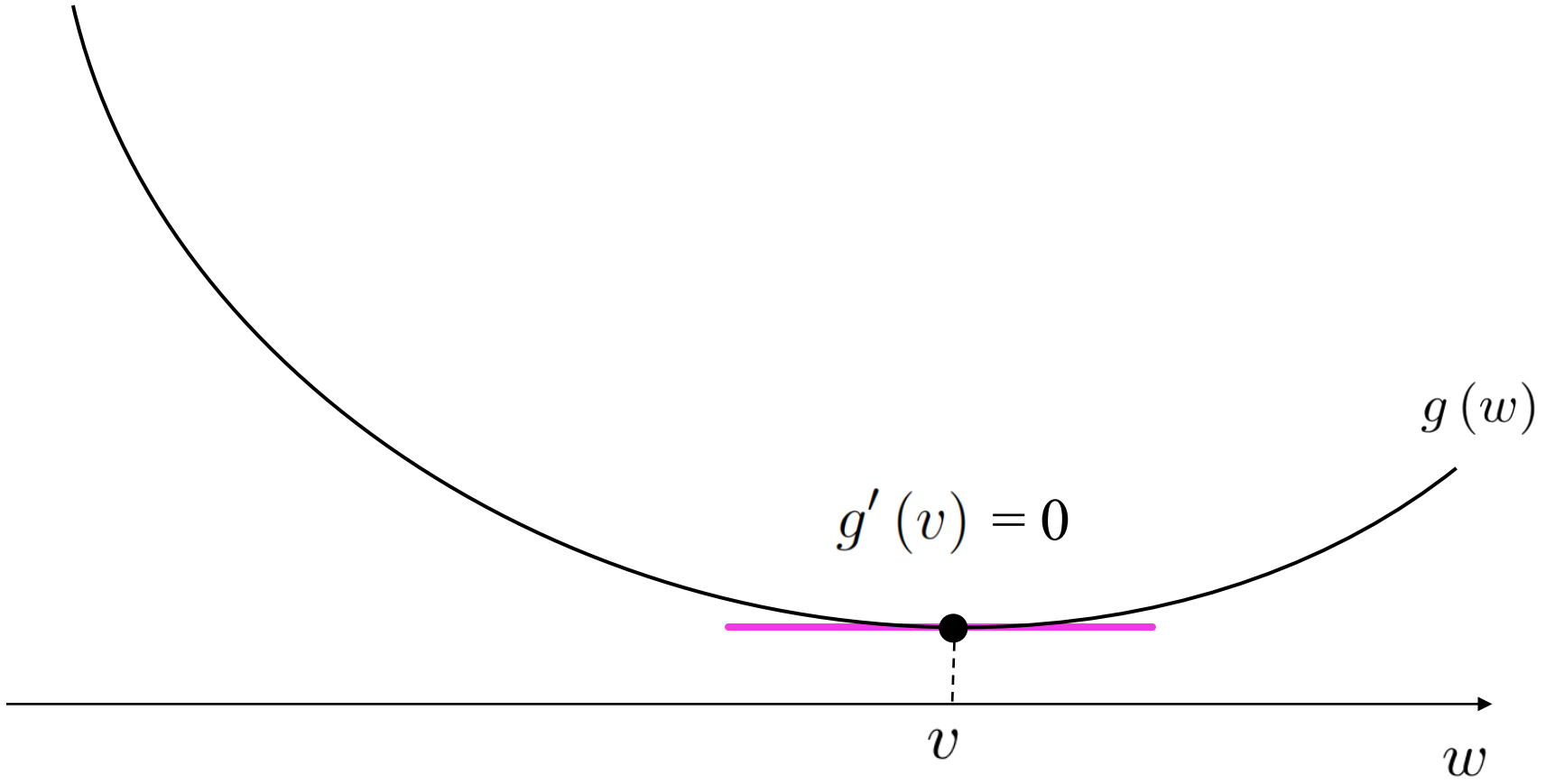
the derivative

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



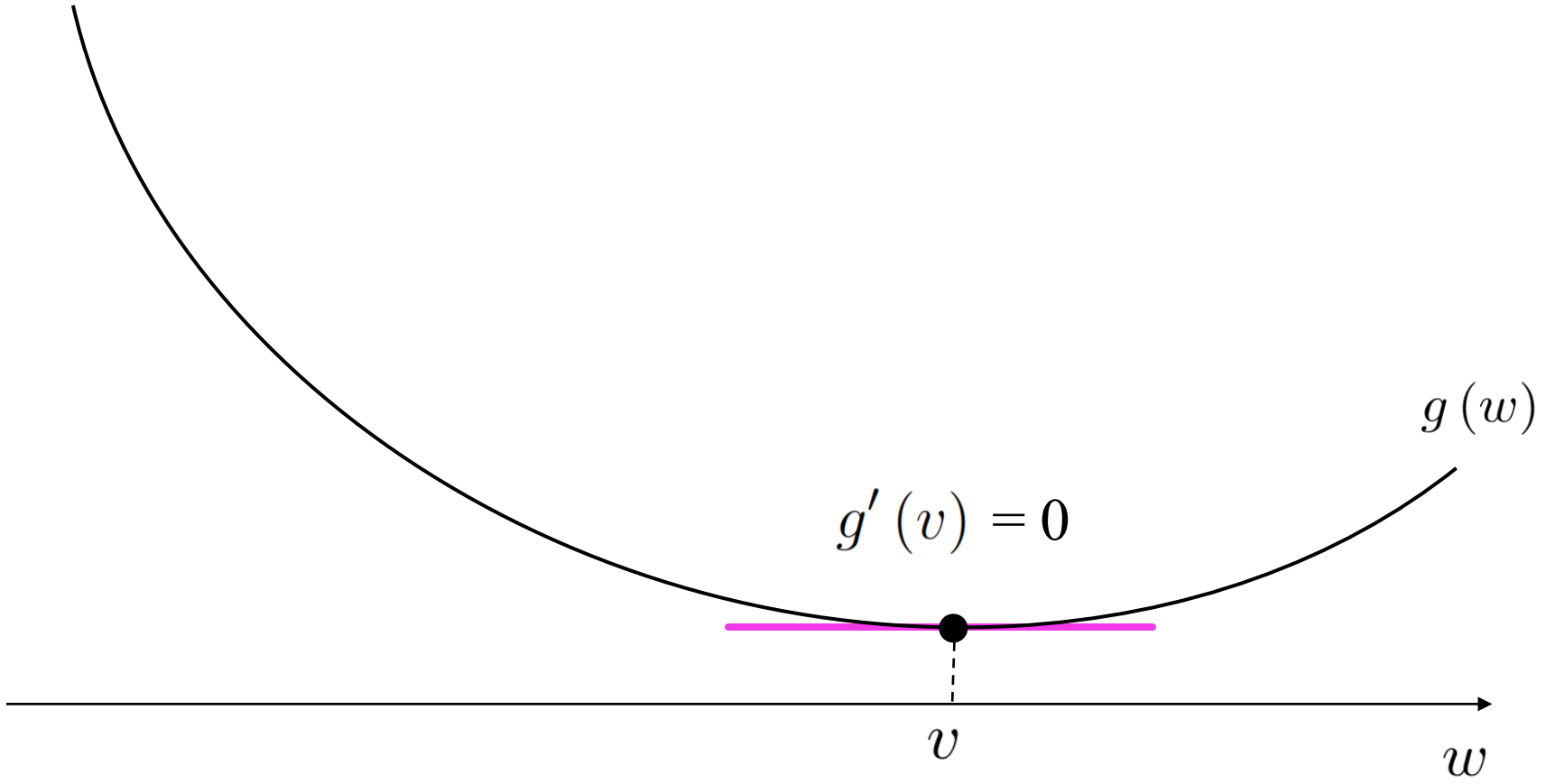
the derivative

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



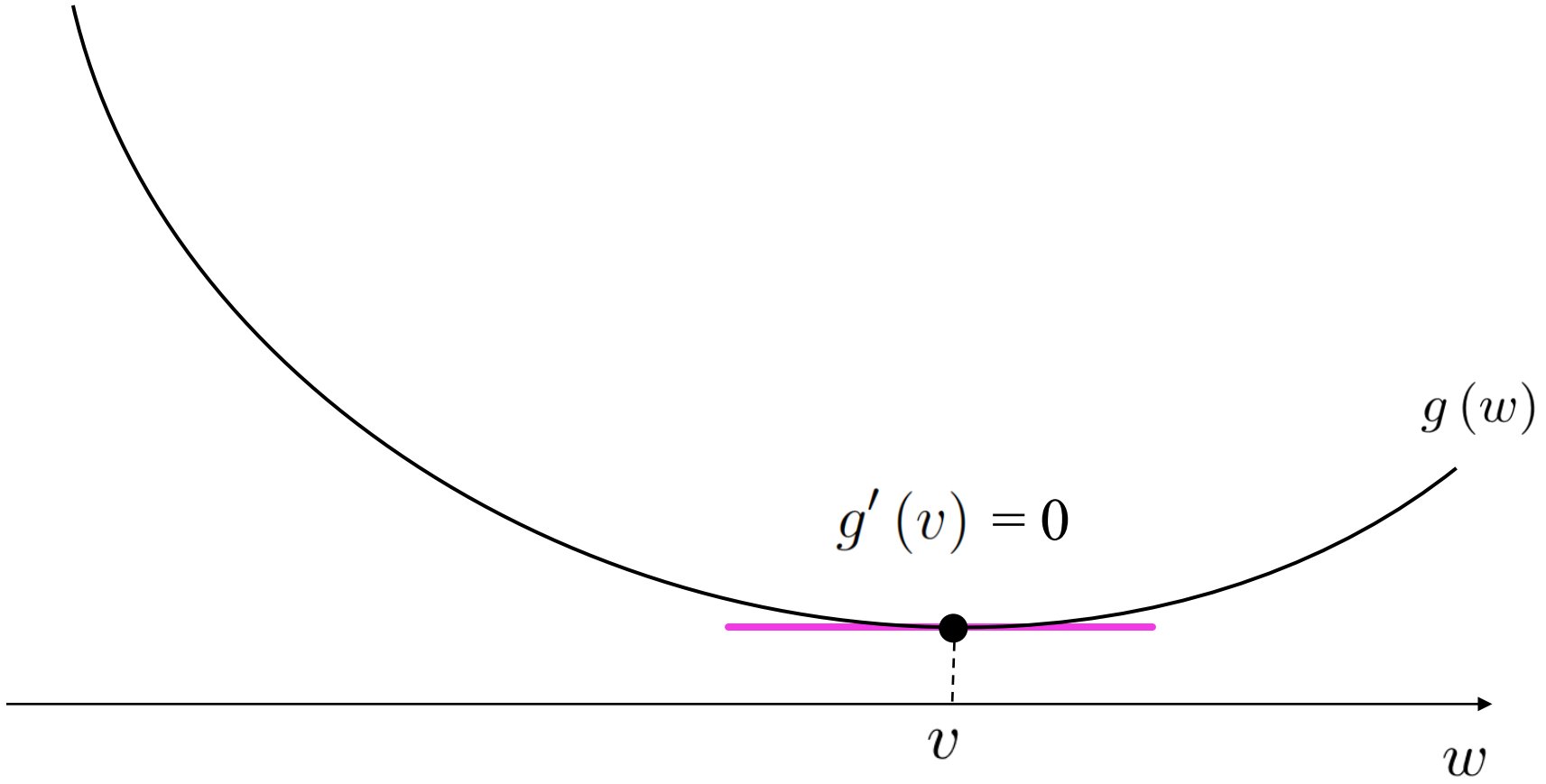
the derivative

- 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



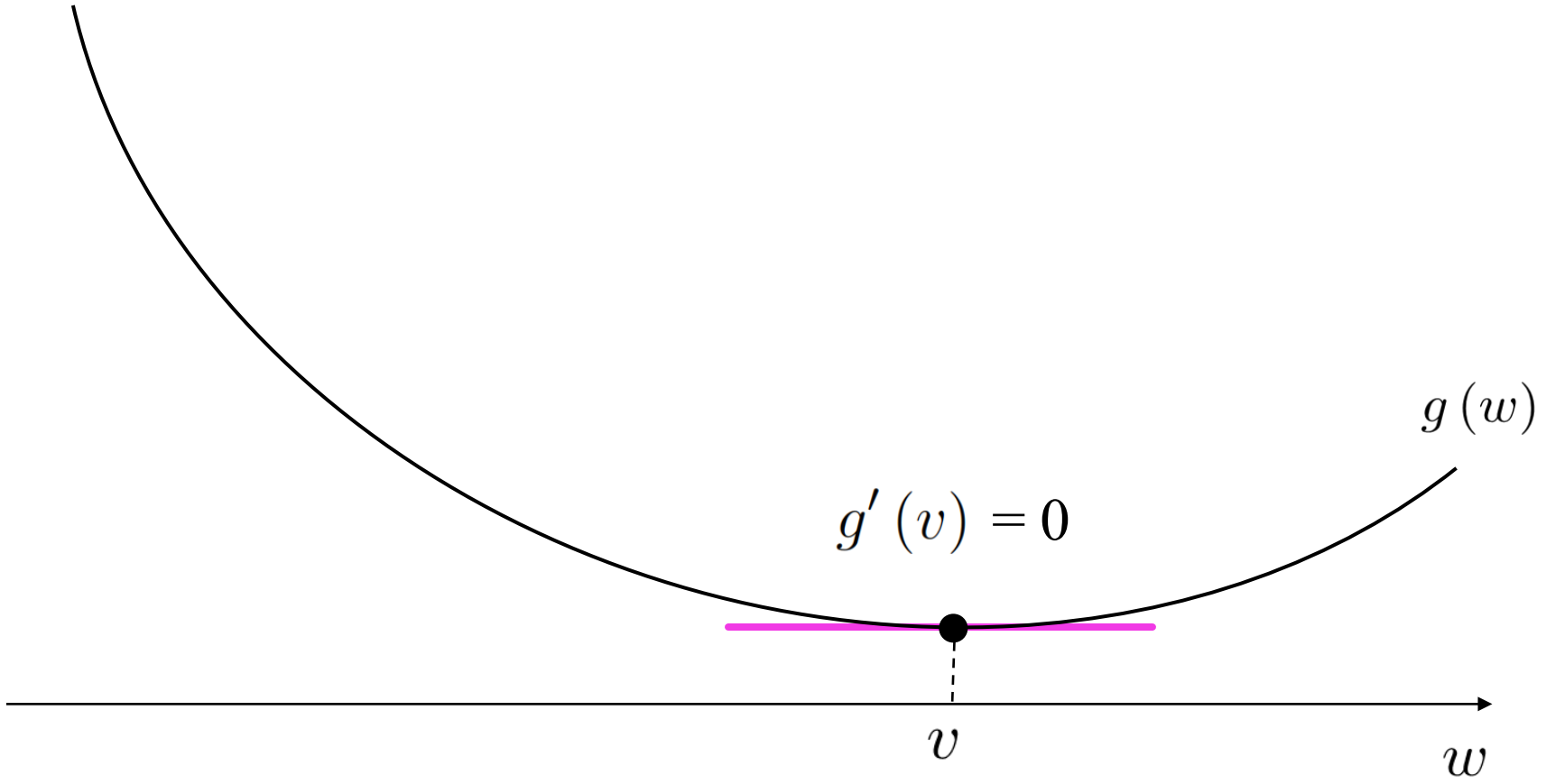
the derivative

- 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



Los Angeles Times

Soaring student loan debt poses risk to nation's future economic growth

By Jim Puzzanghera · Contact Reporter

SEPTEMBER 5, 2015, 10:00 AM | WASHINGTON

The New York Times

A Student Loan System Stacked Against the Borrower

Fair Game

By GRETCHEN MORGENSON OCT. 9, 2015

USA TODAY

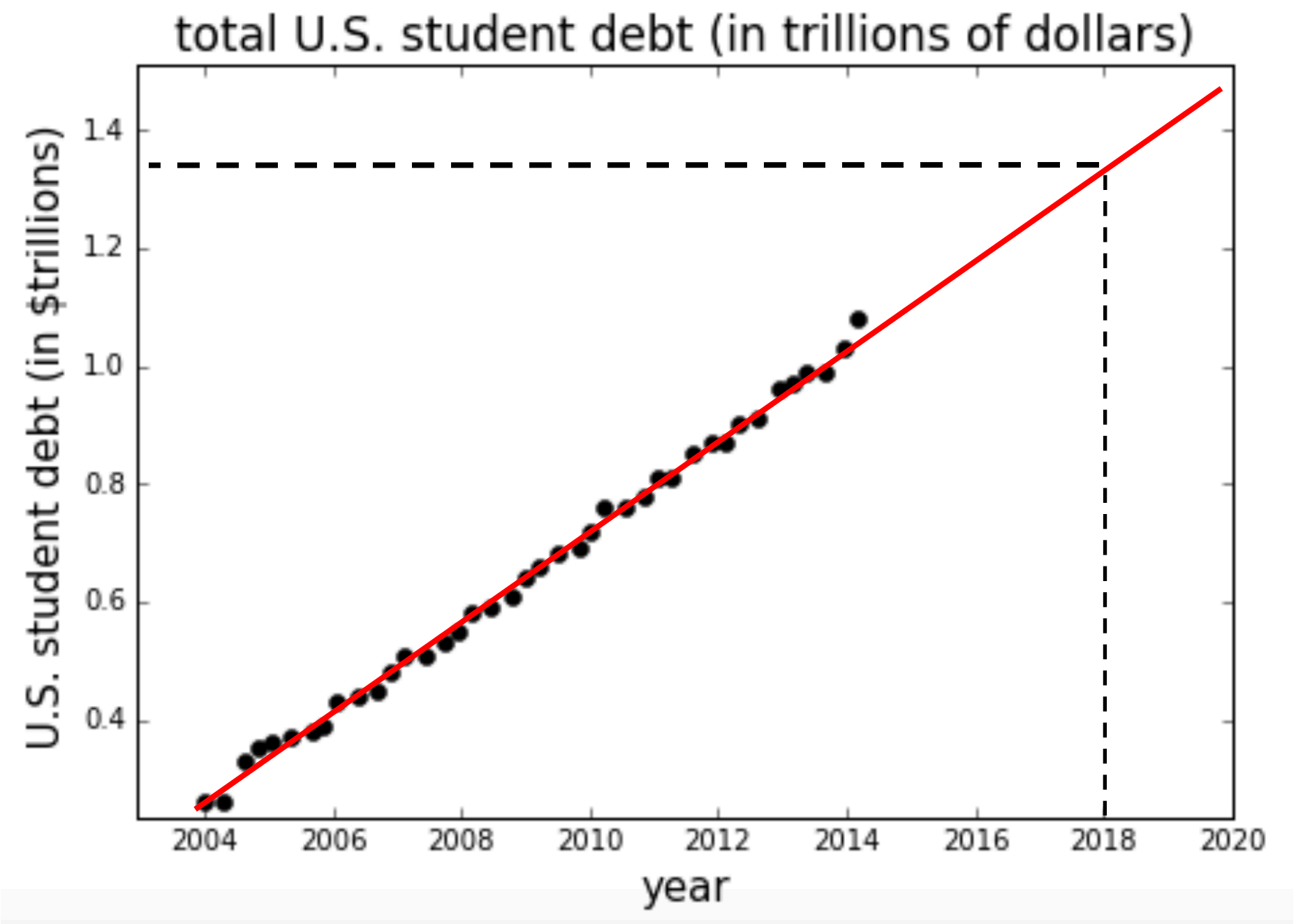
Student loan debt: America's next big crisis

Mitchell D. Weiss, Credit.com *8:48 a.m. EDT August 23, 2015*

Why the Student Loan Crisis Is Even Worse Than People Think

Mark Kantrowitz | Jan. 11, 2016

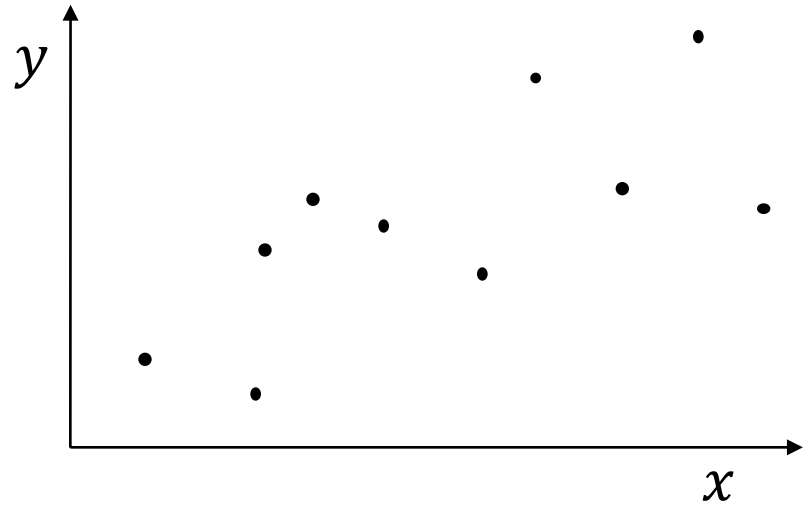
Linear regression



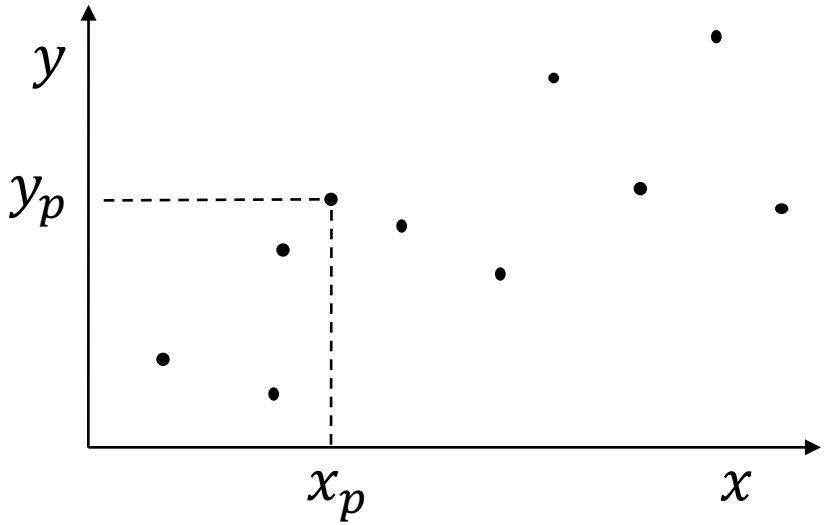
data shown here taken from Donghoon Lee, Wilbert Van der Klaauw, Andrew Haughwout, Meta Brown, and Joelle Scally. Measuring student debt and its performance. *FRB of New York Staff Report*, (668), 2014.

Determining the best fit line

Linear regression



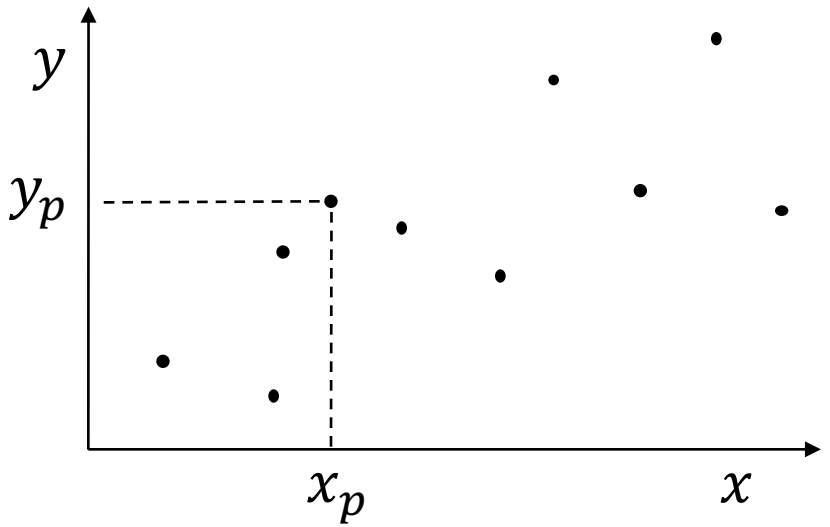
Linear regression



Linear regression

Data: set of P input/output pairs

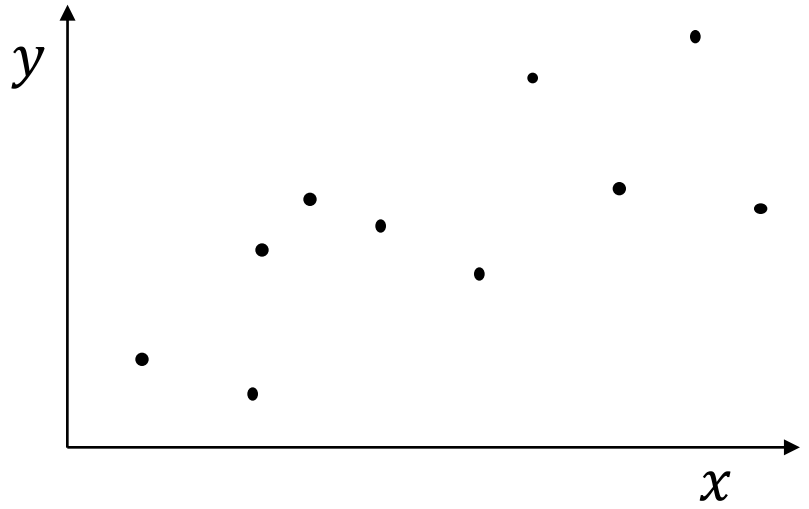
$$(\boldsymbol{x_1}, \boldsymbol{y_1}), (\boldsymbol{x_2}, \boldsymbol{y_2}), \cdots, (\boldsymbol{x_P}, \boldsymbol{y_P})$$



Linear regression

Data: set of P input/output pairs

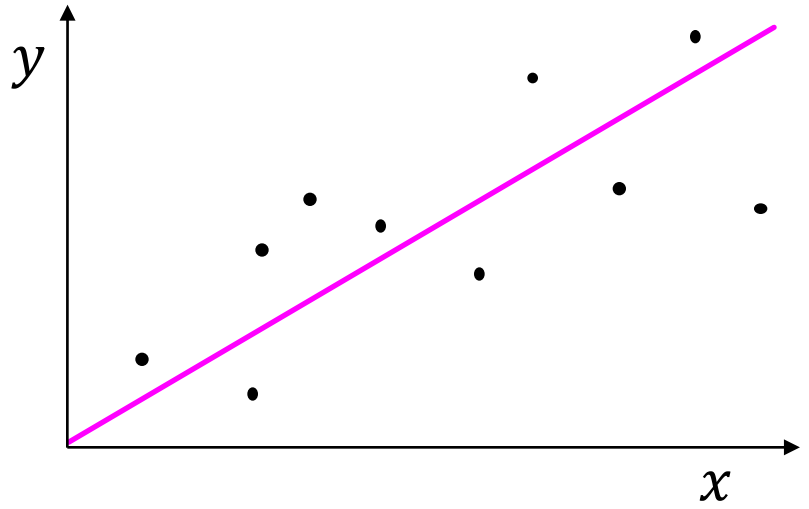
$$(\boldsymbol{x_1}, \boldsymbol{y_1}), (\boldsymbol{x_2}, \boldsymbol{y_2}), \cdots, (\boldsymbol{x_P}, \boldsymbol{y_P})$$



Linear regression

Data: set of P input/output pairs

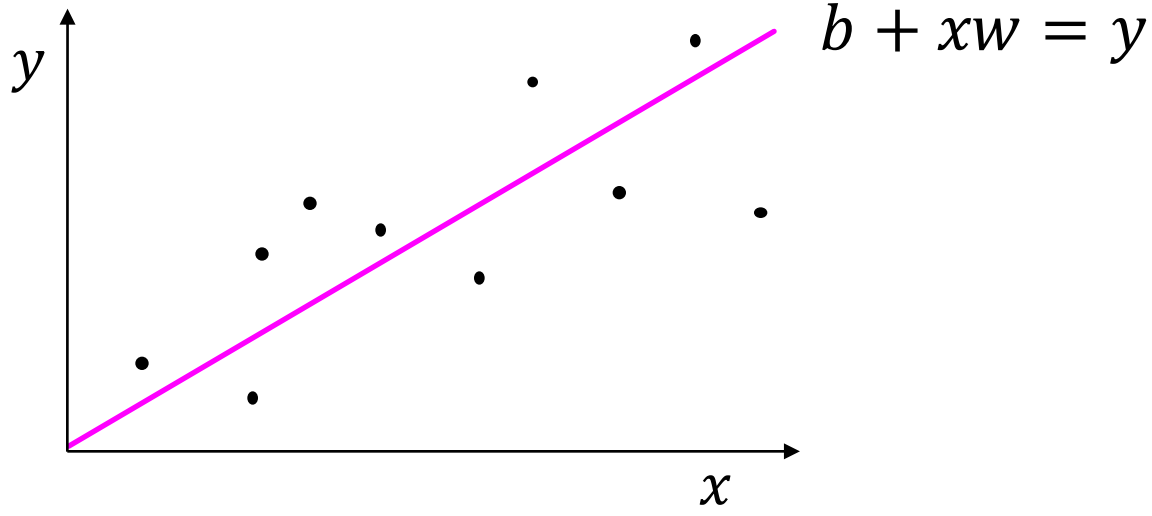
$$(\boldsymbol{x_1}, \boldsymbol{y_1}), (\boldsymbol{x_2}, \boldsymbol{y_2}), \cdots, (\boldsymbol{x_P}, \boldsymbol{y_P})$$



Linear regression

Data: set of P input/output pairs

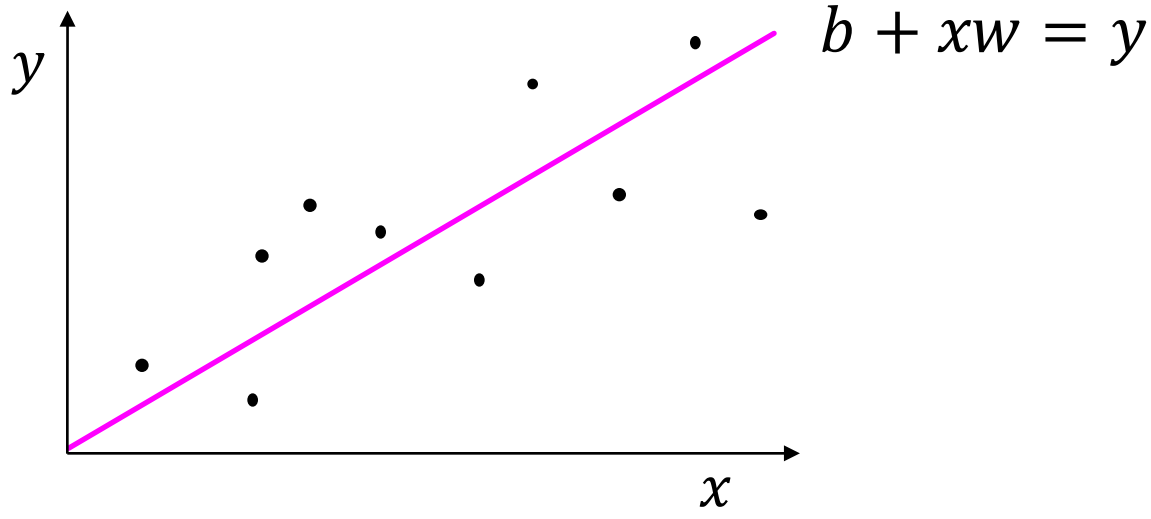
$$(\boldsymbol{x_1}, \boldsymbol{y_1}), (\boldsymbol{x_2}, \boldsymbol{y_2}), \cdots, (\boldsymbol{x_P}, \boldsymbol{y_P})$$



Linear regression

Data: set of P input/output pairs

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



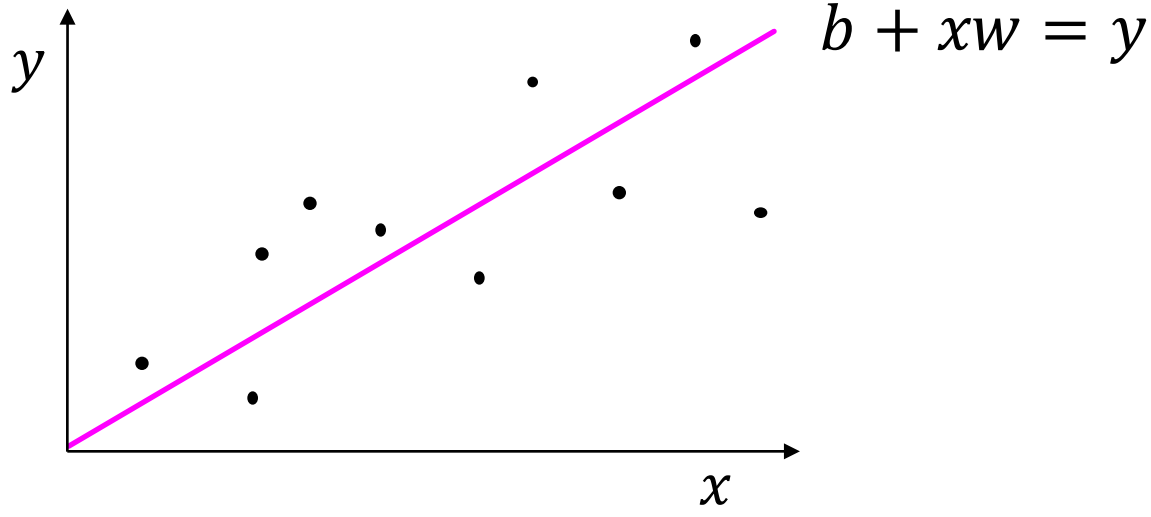
Find intercept b and slope w so line passes through all points as best as possible

$$b + x_p w \approx y_p$$

Linear regression

Data: set of P input/output pairs

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

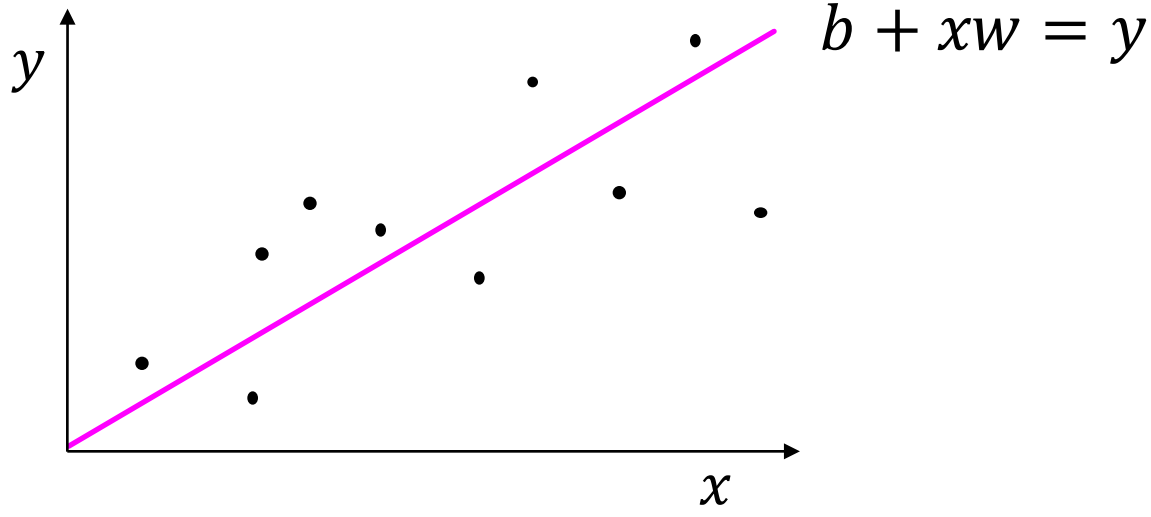


Find intercept b and slope w so line passes through all points as best as possible

$$b + x_p w \approx y_p$$

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

Linear regression

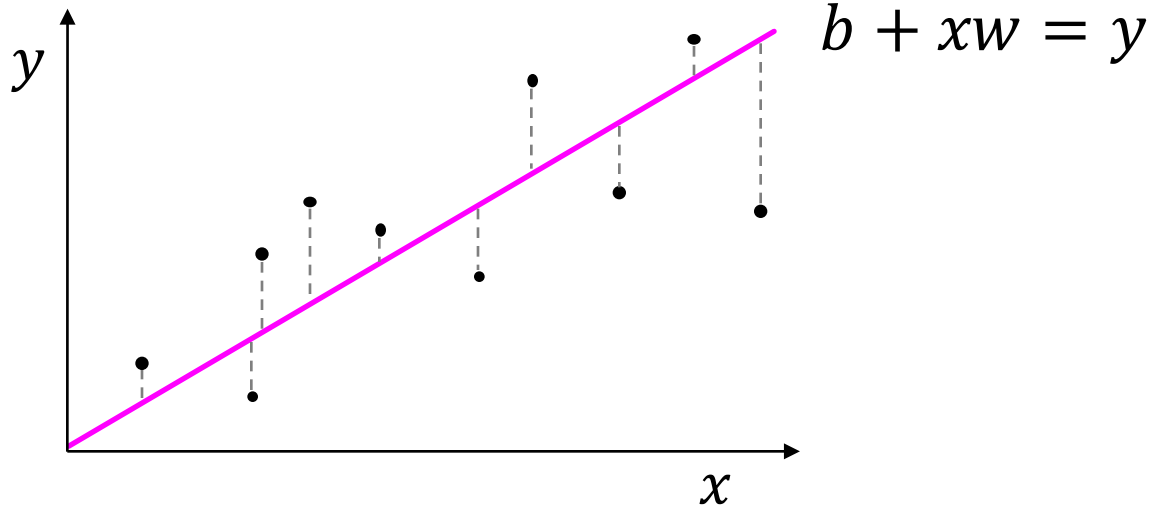


Find intercept b and slope w so line passes through all points as best as possible

$$b + x_p w \approx y_p$$

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

Linear regression

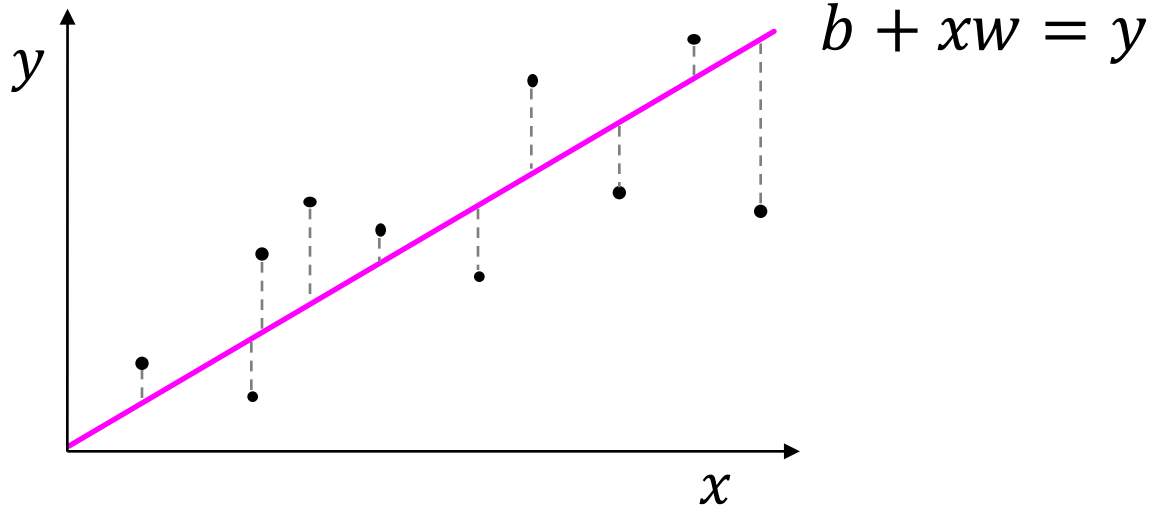


Find intercept b and slope w so line passes through all points as best as possible

$$b + x_p w \approx y_p$$

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

Linear regression

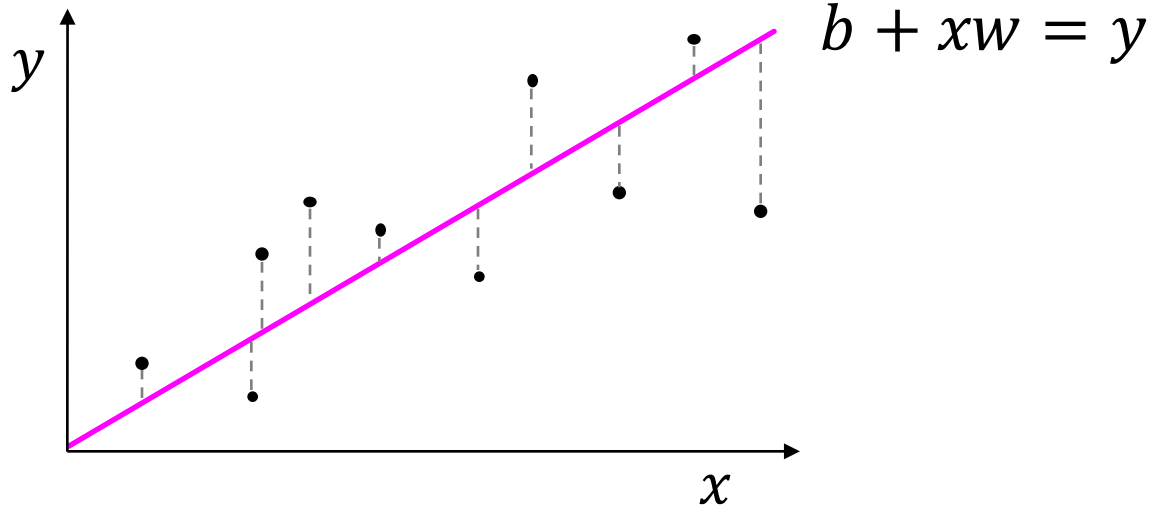


Find intercept b and slope w so line has ***smallest*** squared error with data points

$$b + x_p w \approx y_p$$

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

Linear regression

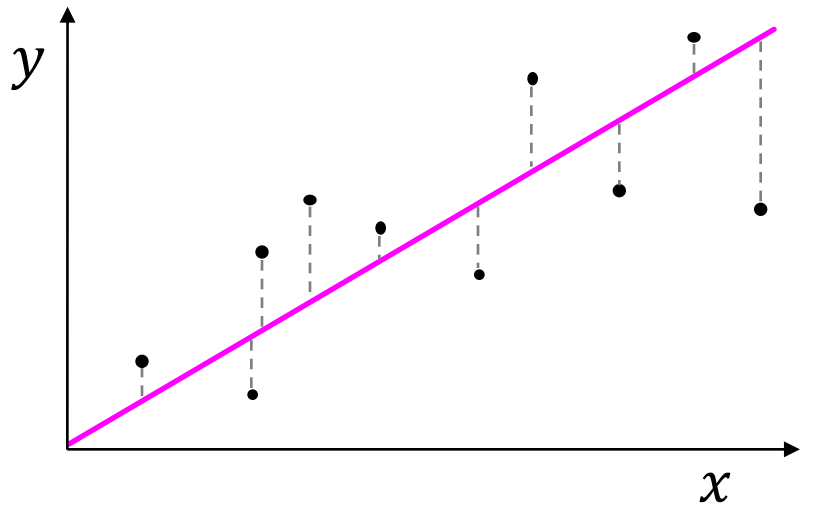


Find intercept b and slope w so line has ***smallest*** squared error with data points

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

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

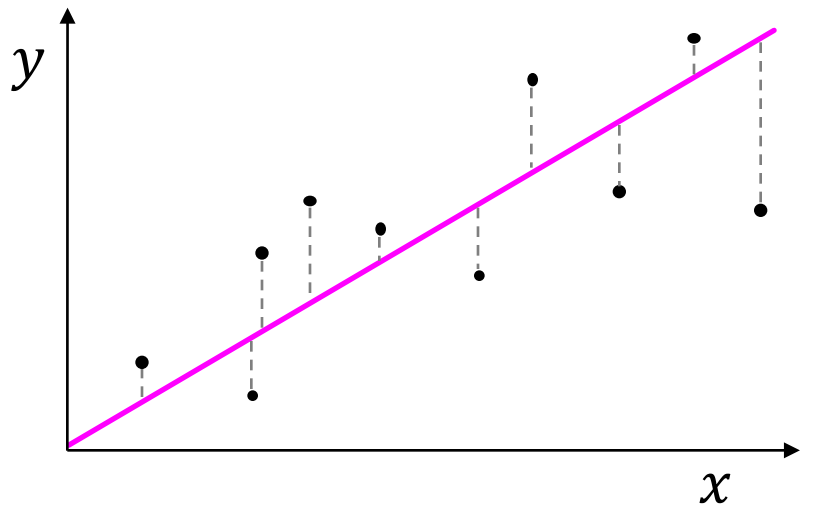
Linear regression



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

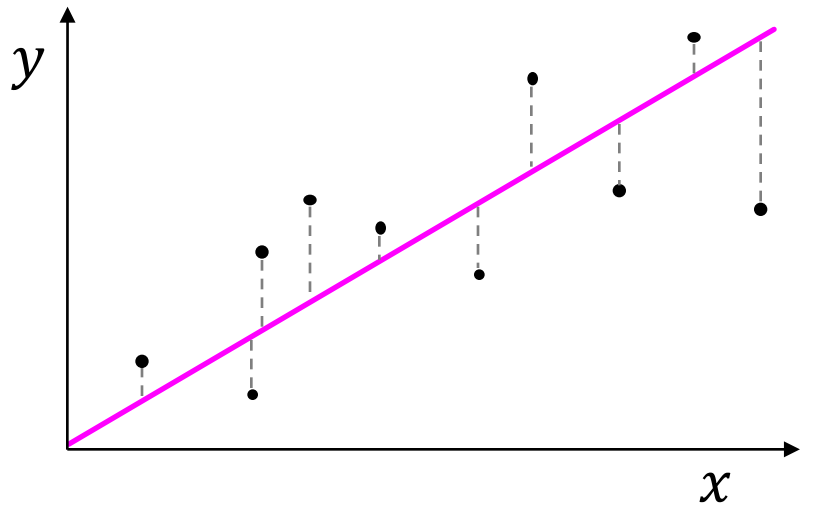


$b + xw = y$

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

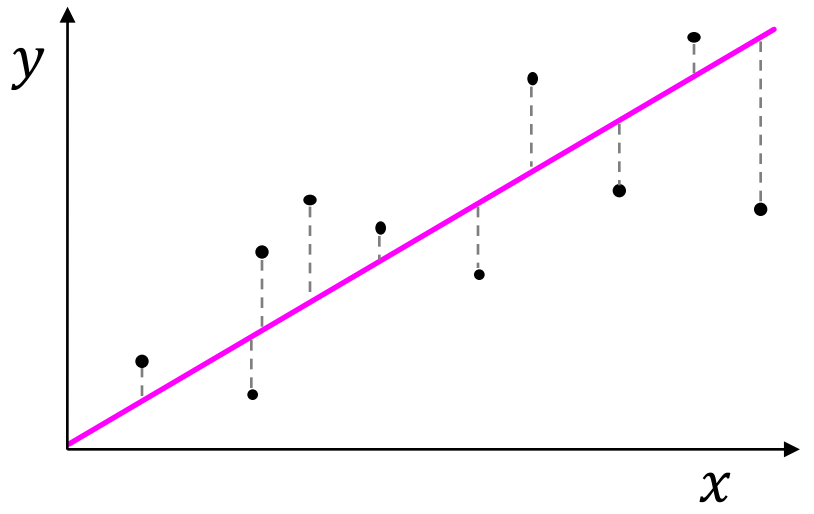


Find intercept b and slope w so line has ***smallest*** squared error with data points

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

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

Linear regression

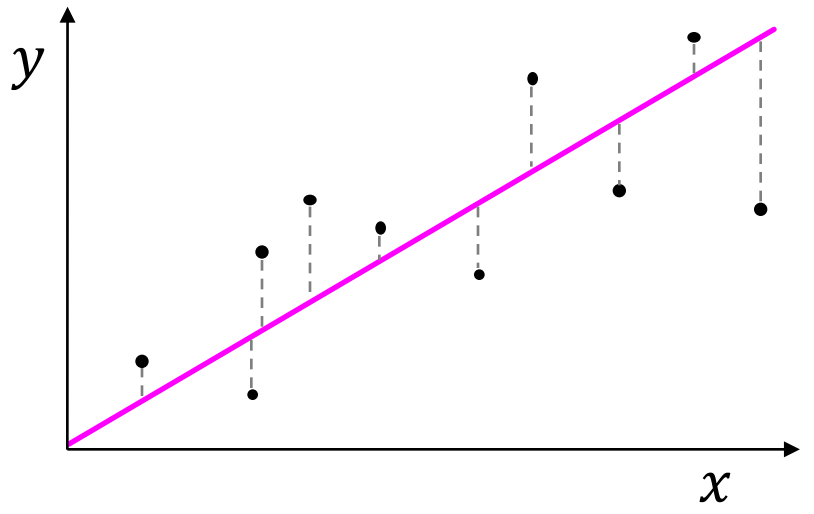


Find intercept b and slope w so line has ***smallest*** squared error with data points

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

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

Linear regression

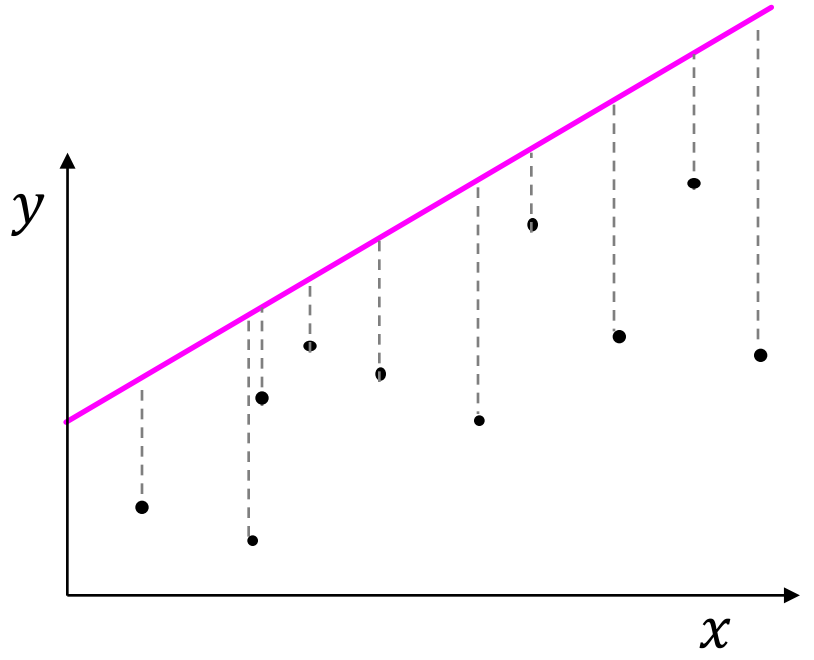


Find intercept b and slope w so line has ***smallest*** ~~squared~~ error with data points

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

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

Linear regression



$b + xw = y$

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

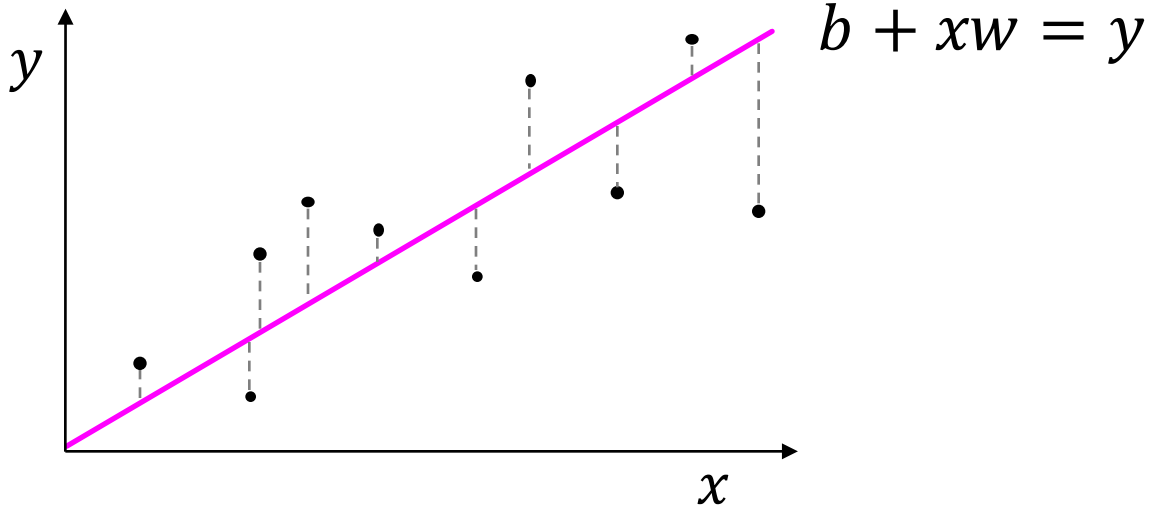
$$g(b,w) = \sum_{p=1}^{P} \left(b + x_p w - y_p\right)$$

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

quiz: problem

The ideal parameters for linear regression minimize squared error between the datapoints and the prospective line because

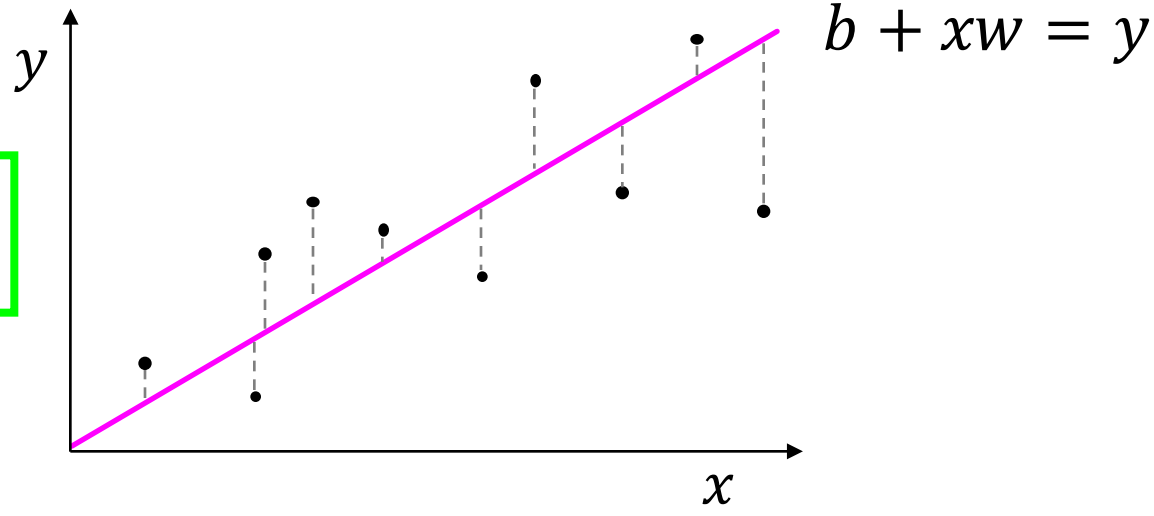
a) squaring the errors allow us to treat positive and negative errors in the same manner
b) squaring the errrors makes them all smaller
c) squaring the errors makes them all larger
d) squaring something always makes it better



quiz: solution

The ideal parameters for linear regression minimize squared error between the datapoints and the prospective line because

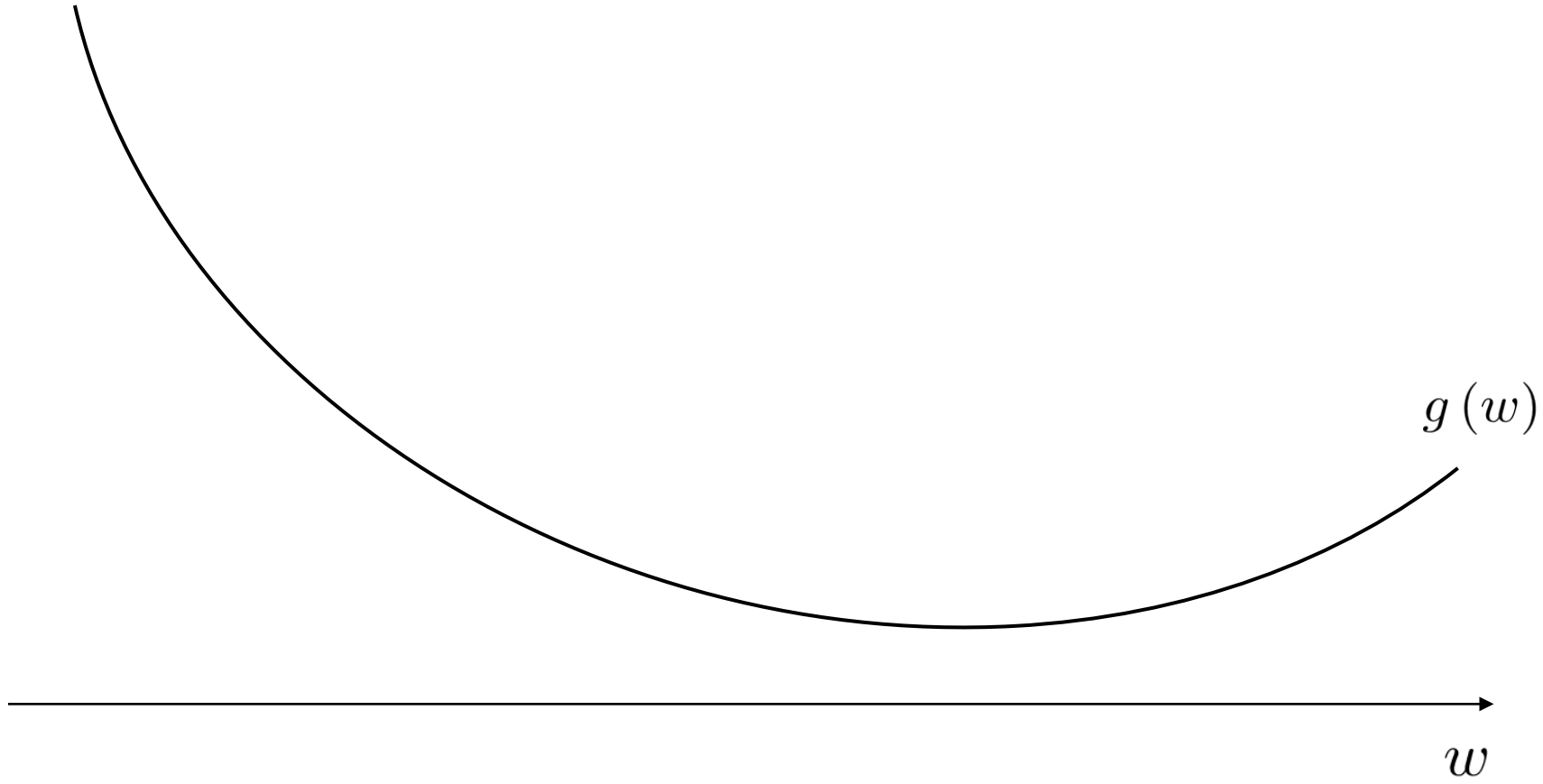
a) squaring the errors allow us to treat positive and negative errors in the same manner
b) squaring the errrors makes them all smaller
c) squaring the errors makes them all larger
d) squaring something always makes it better



minimizing cost functions using basic calculus

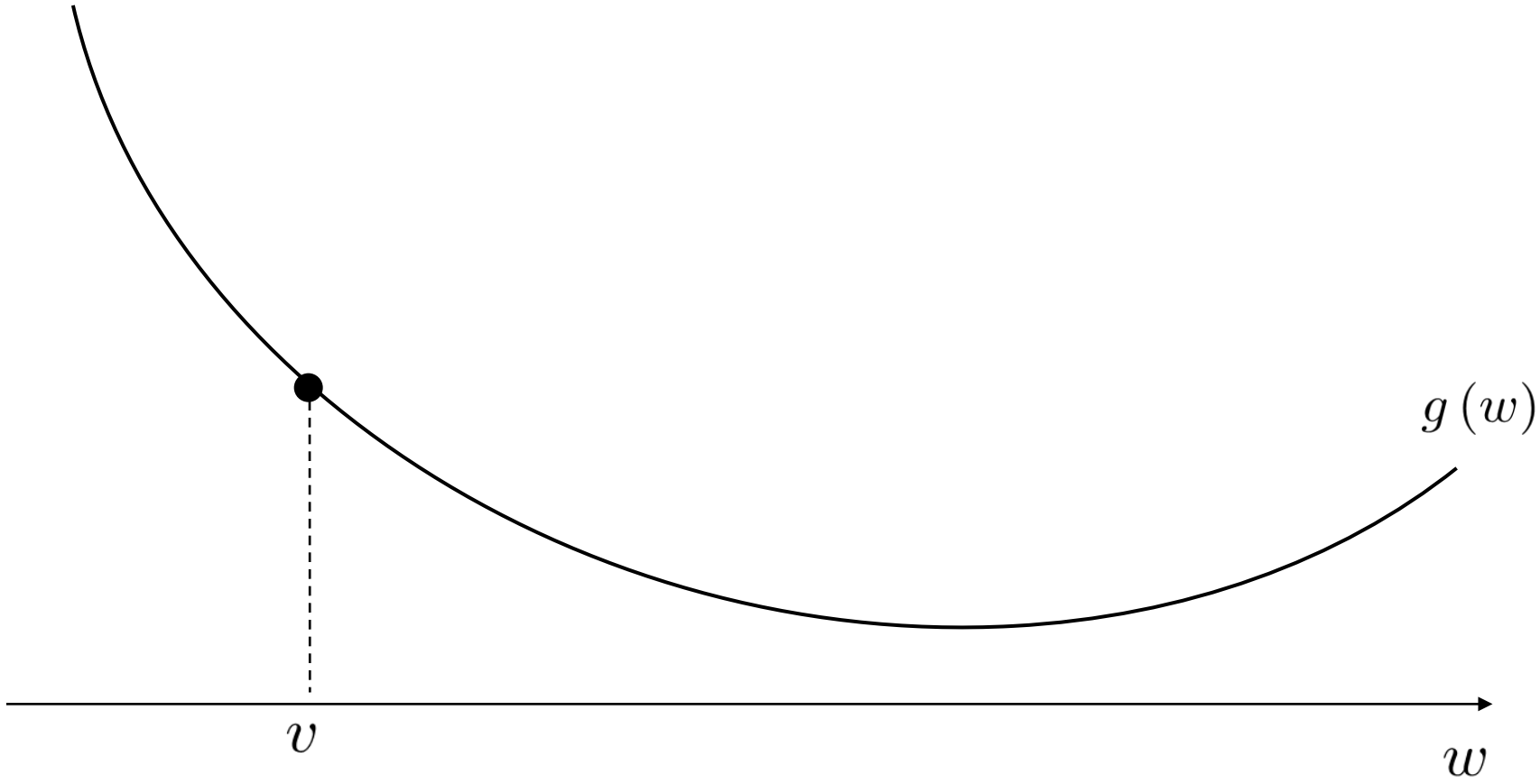
the derivative

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



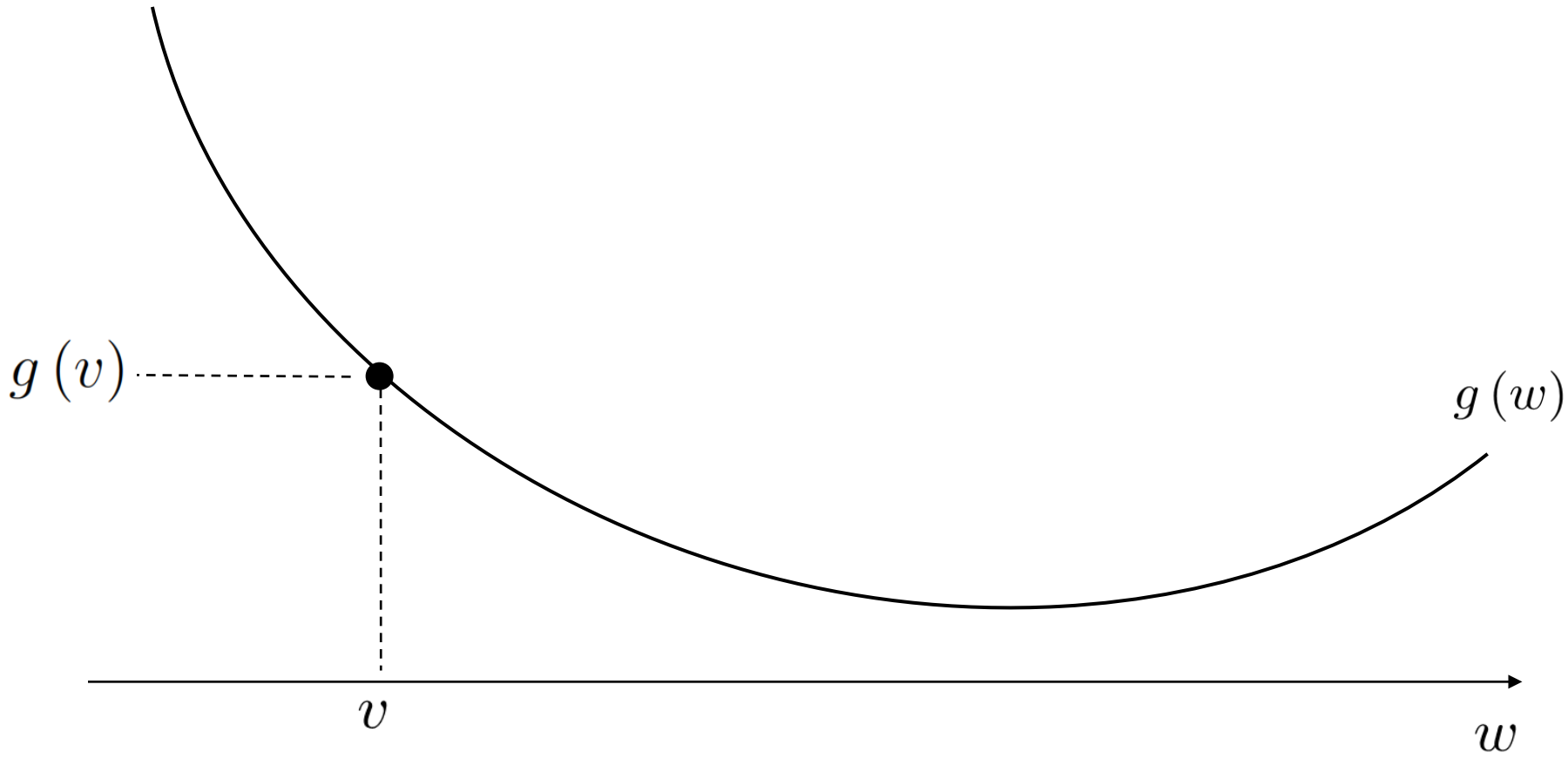
the derivative

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



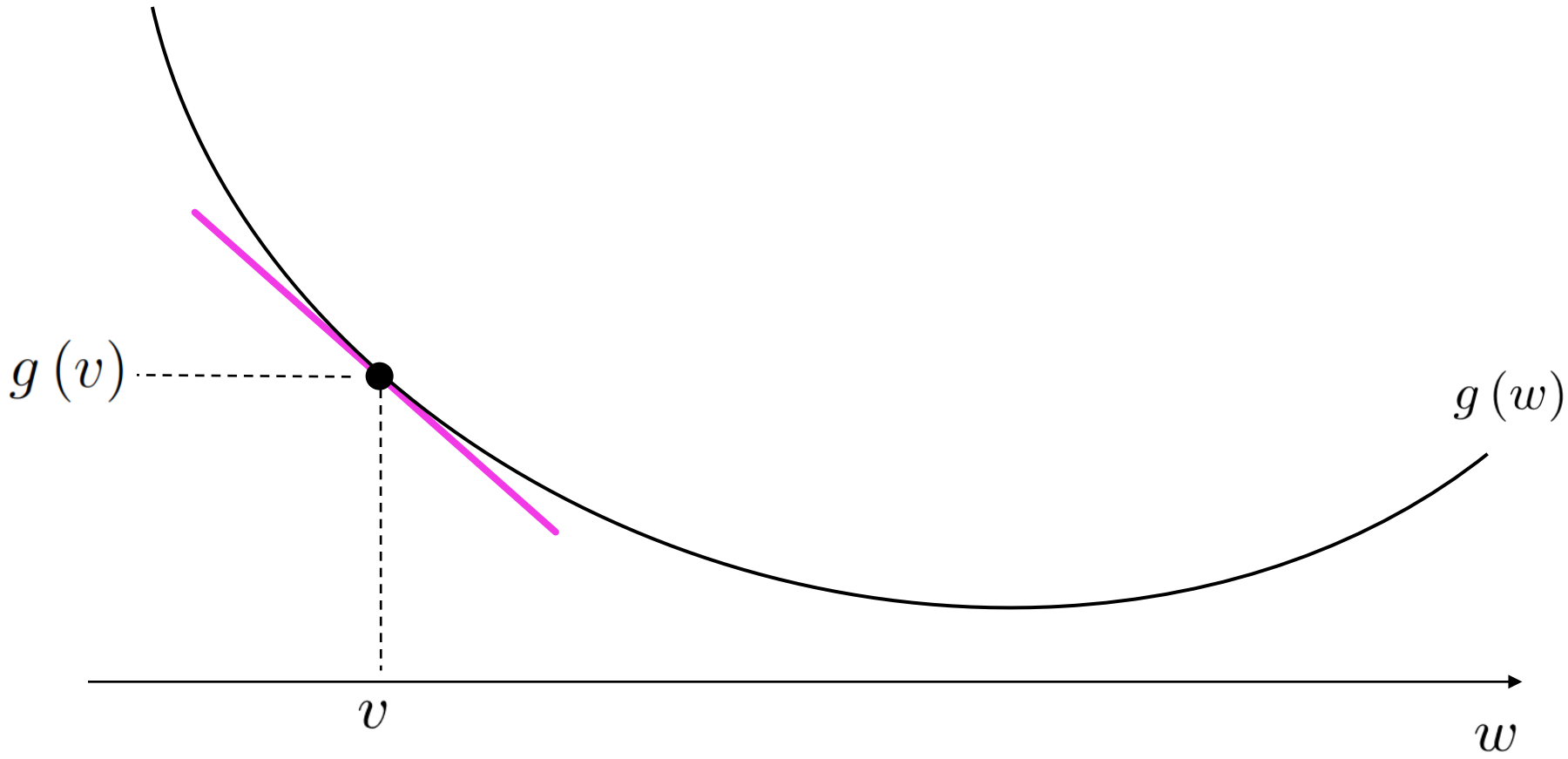
the derivative

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



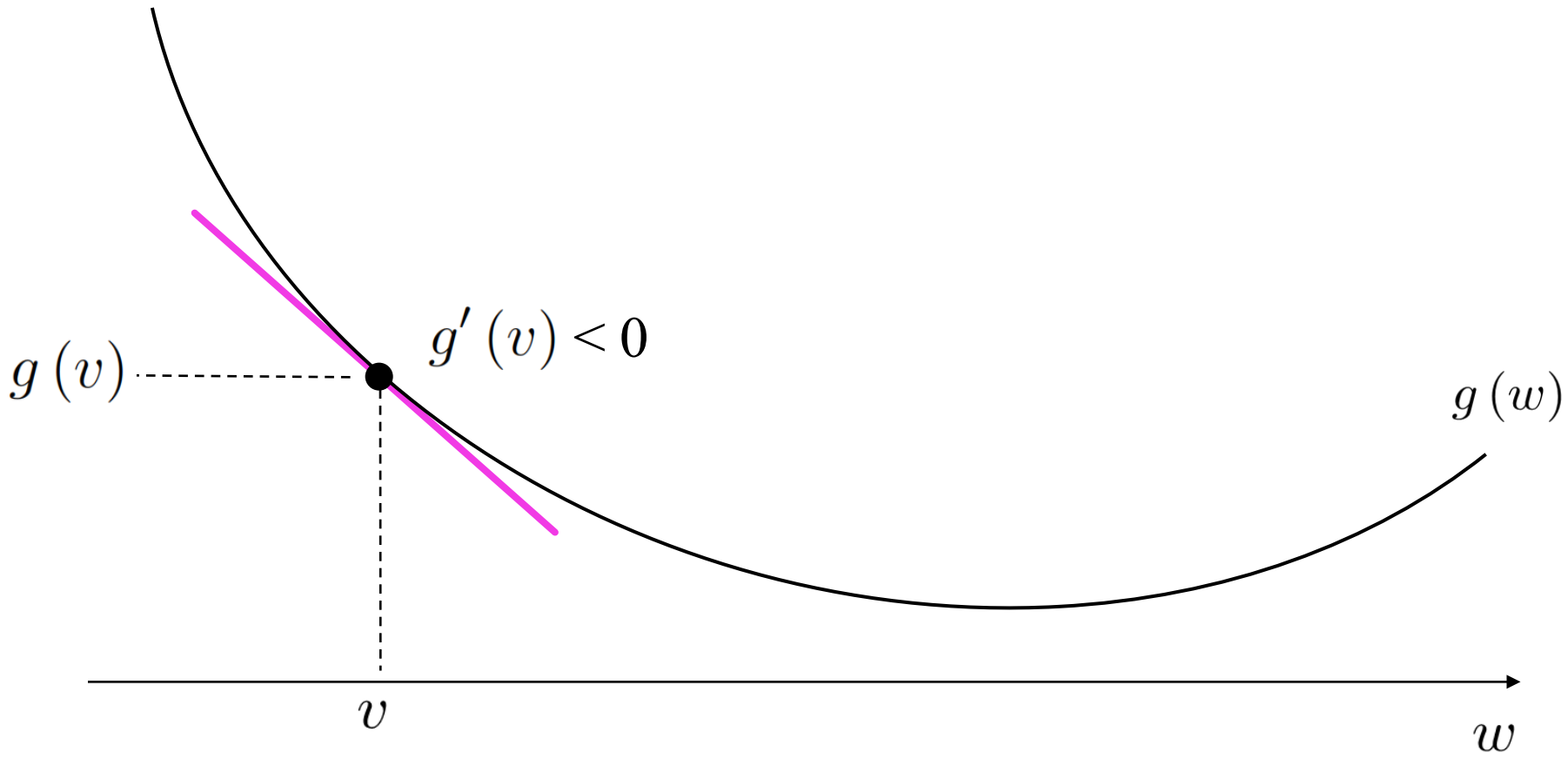
the derivative

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



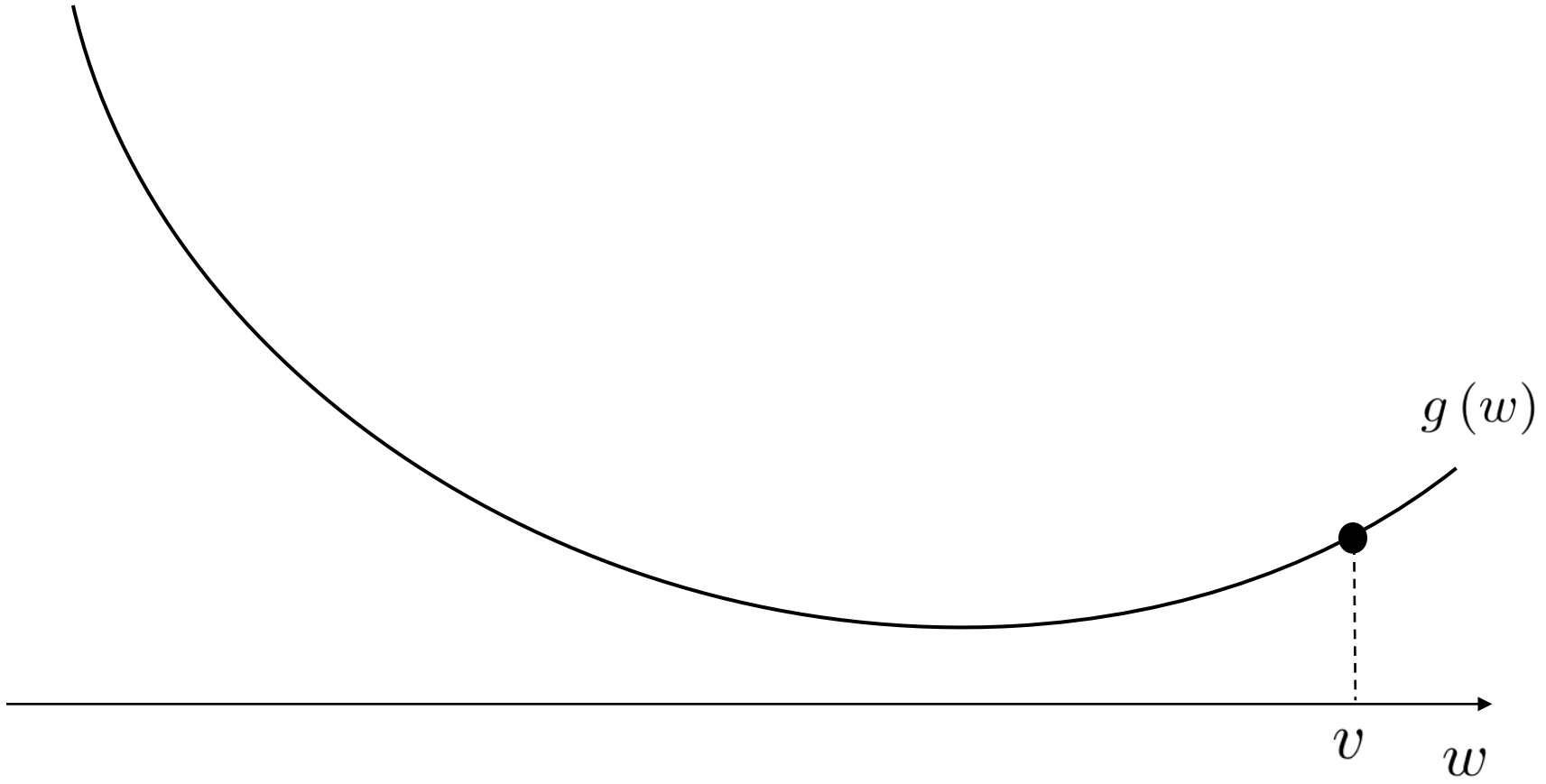
the derivative

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



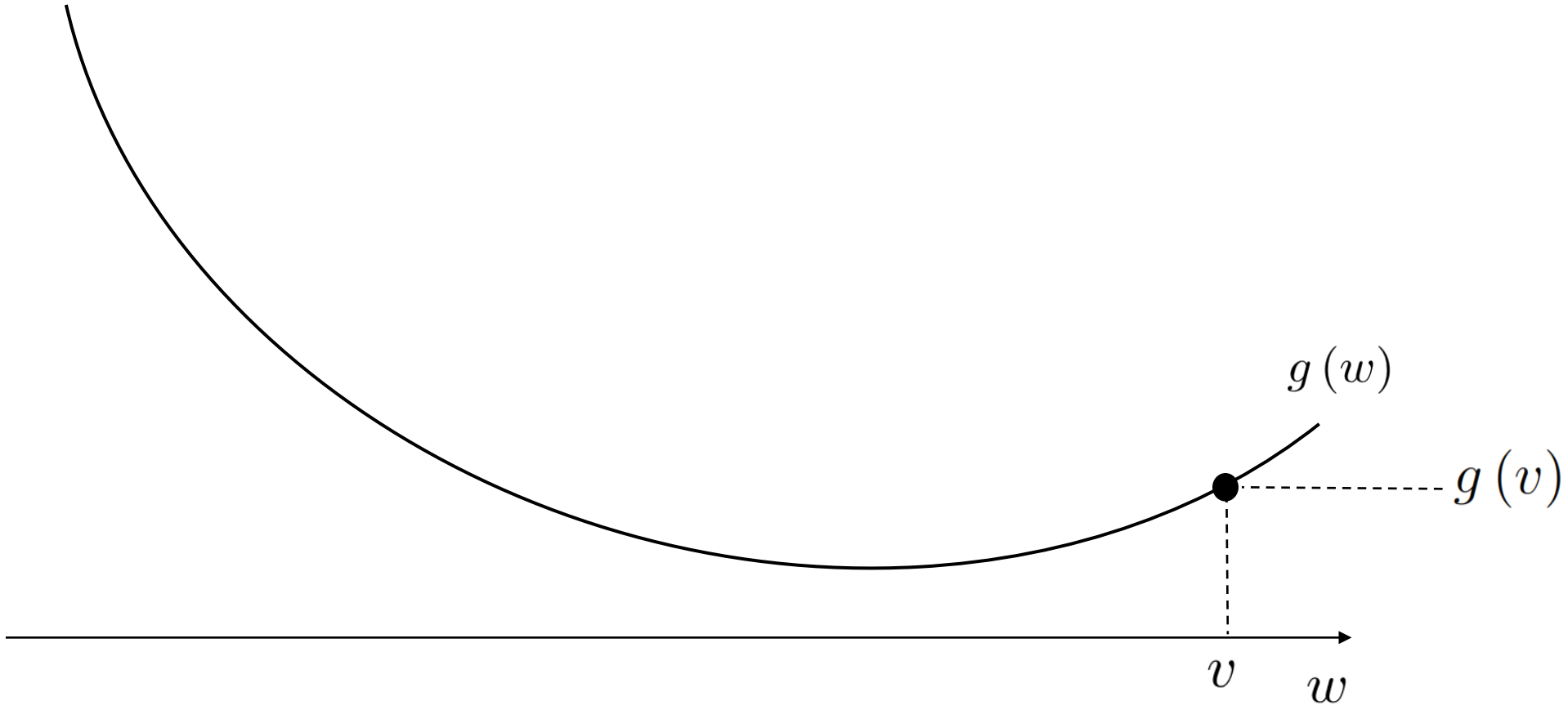
the derivative

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



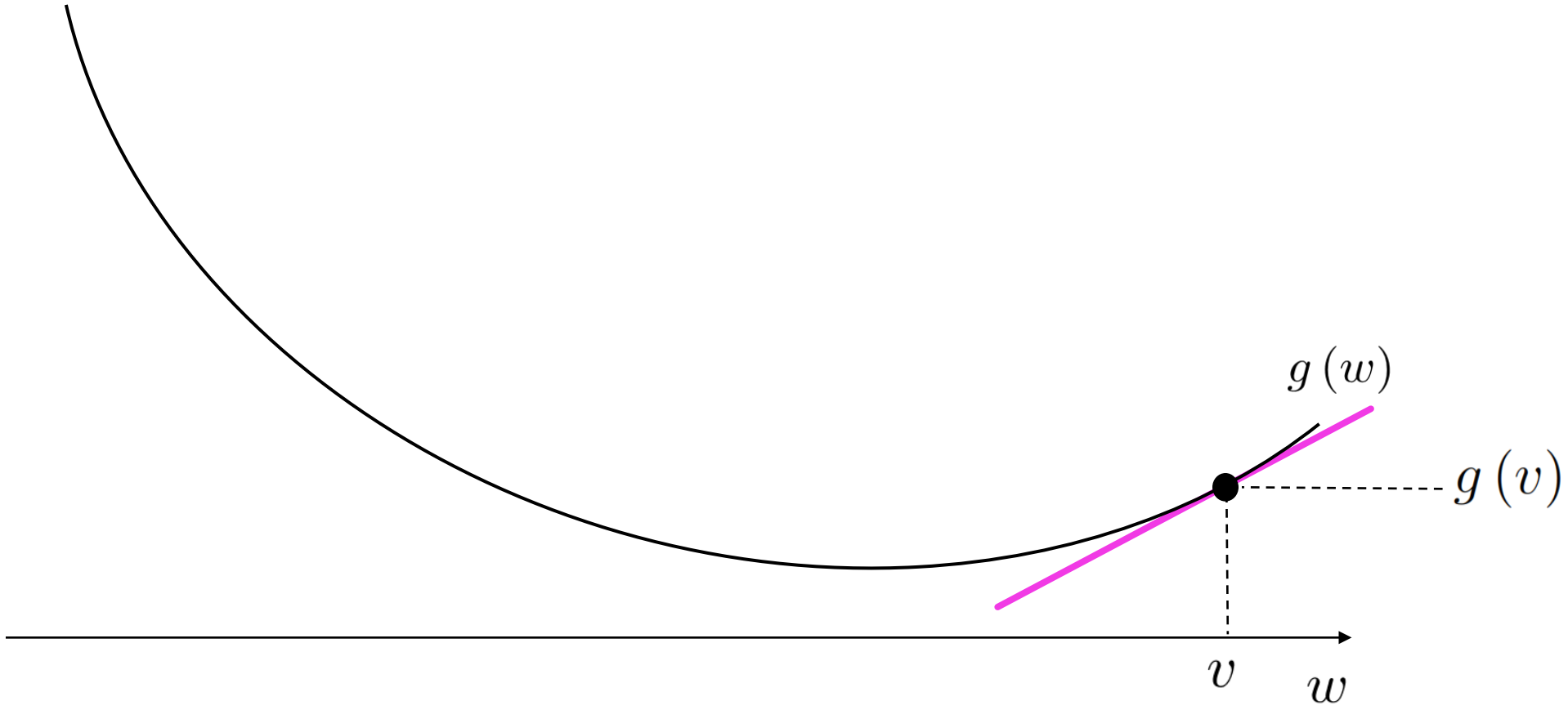
the derivative

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



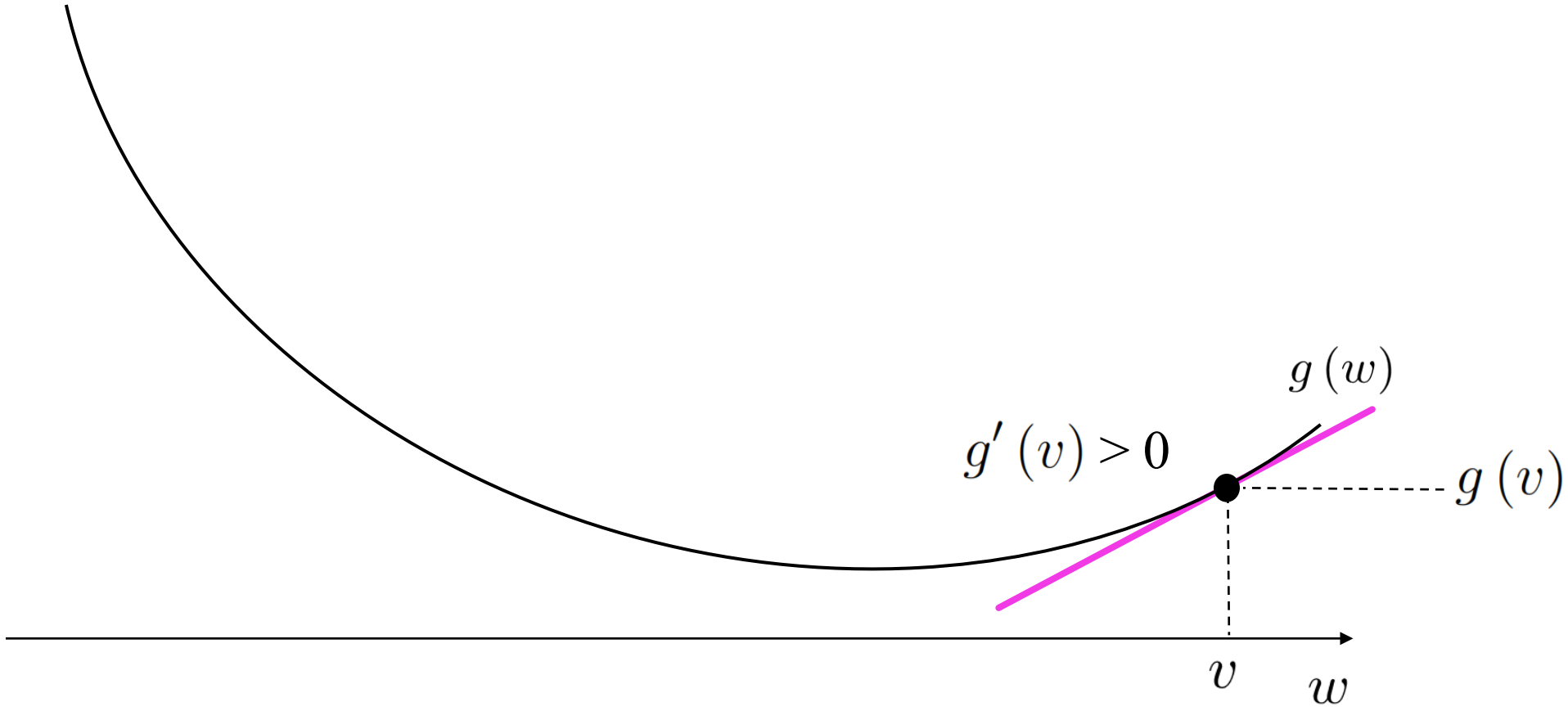
the derivative

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



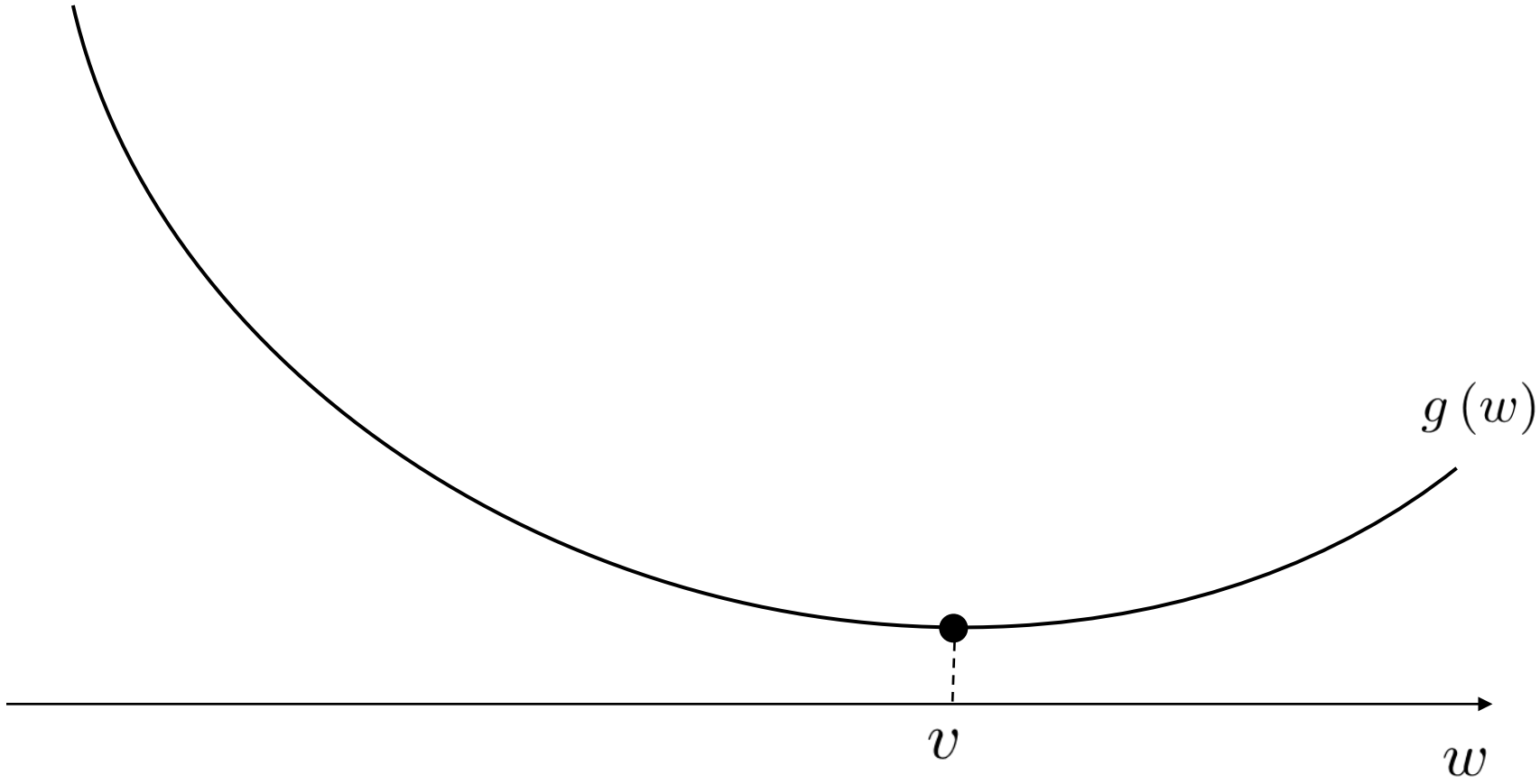
the derivative

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



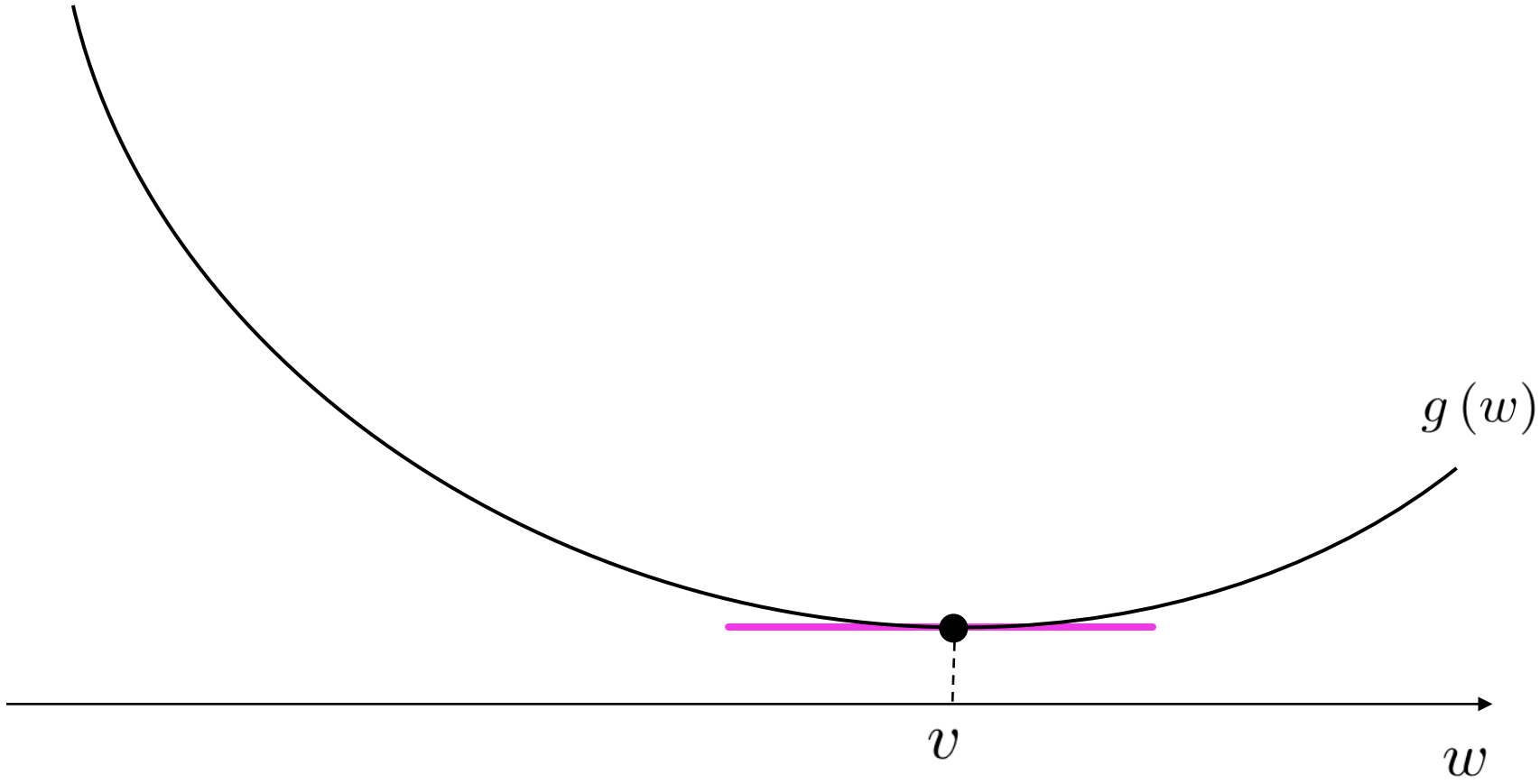
the derivative

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



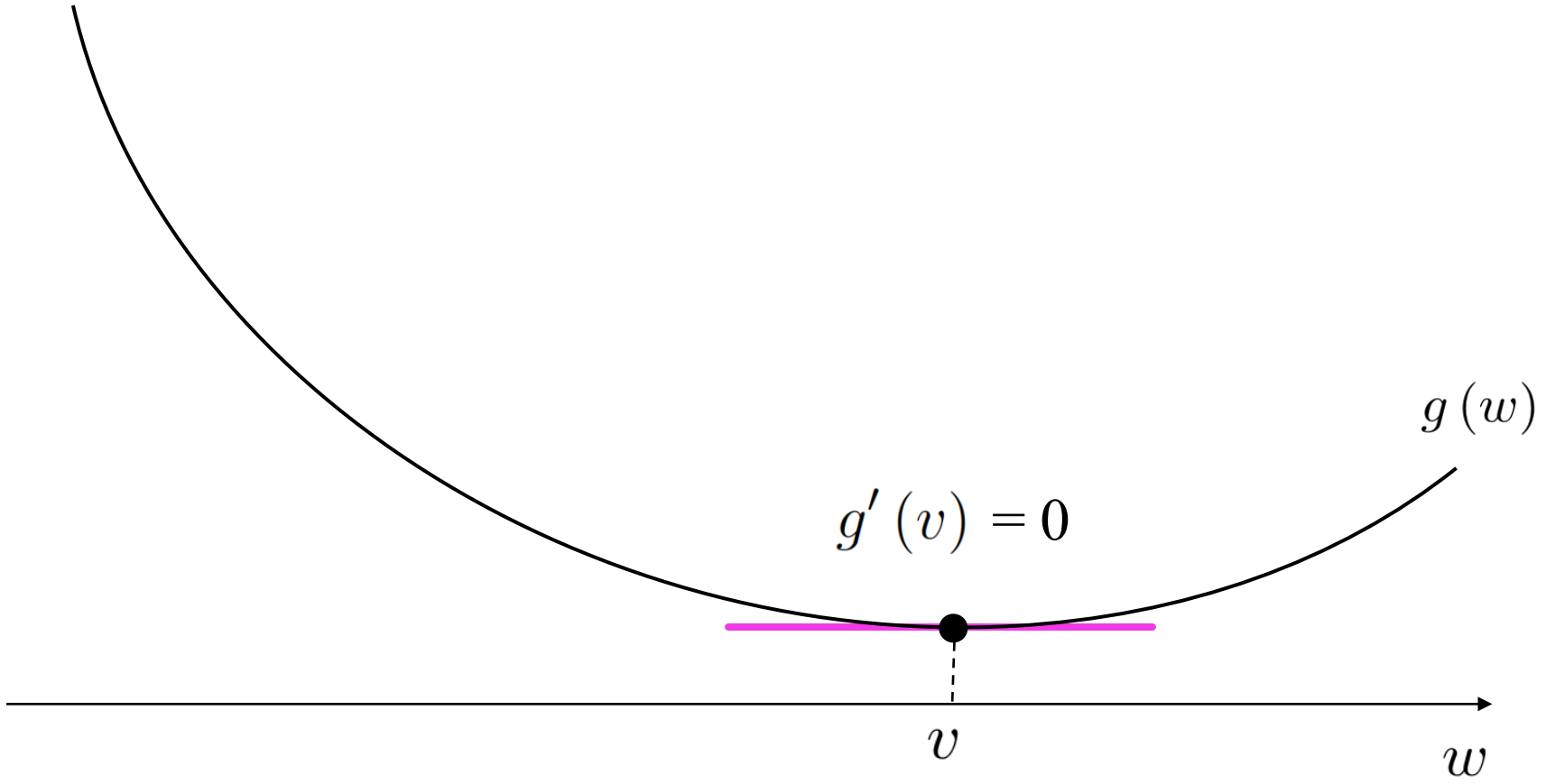
the derivative

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



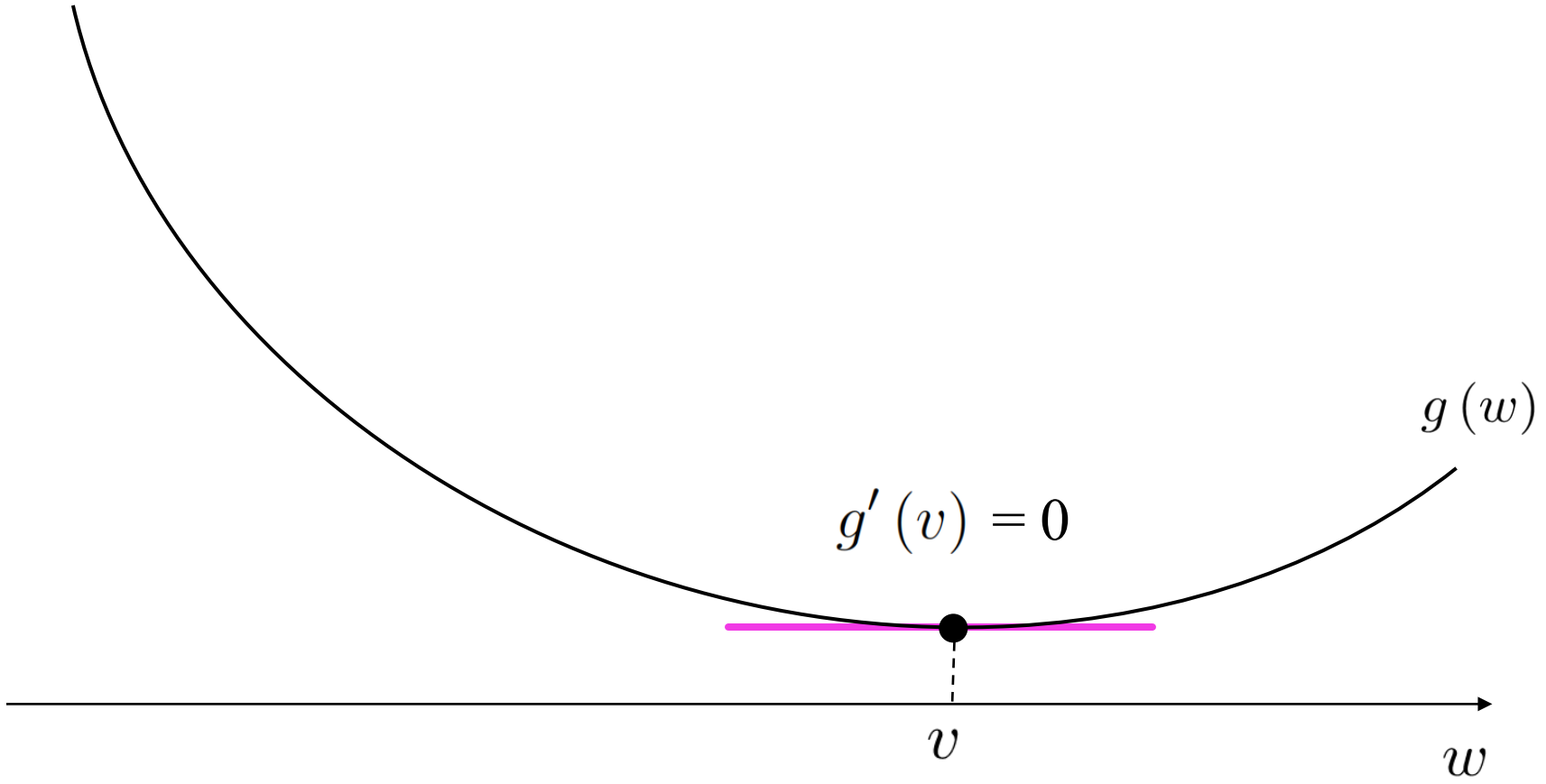
the derivative

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



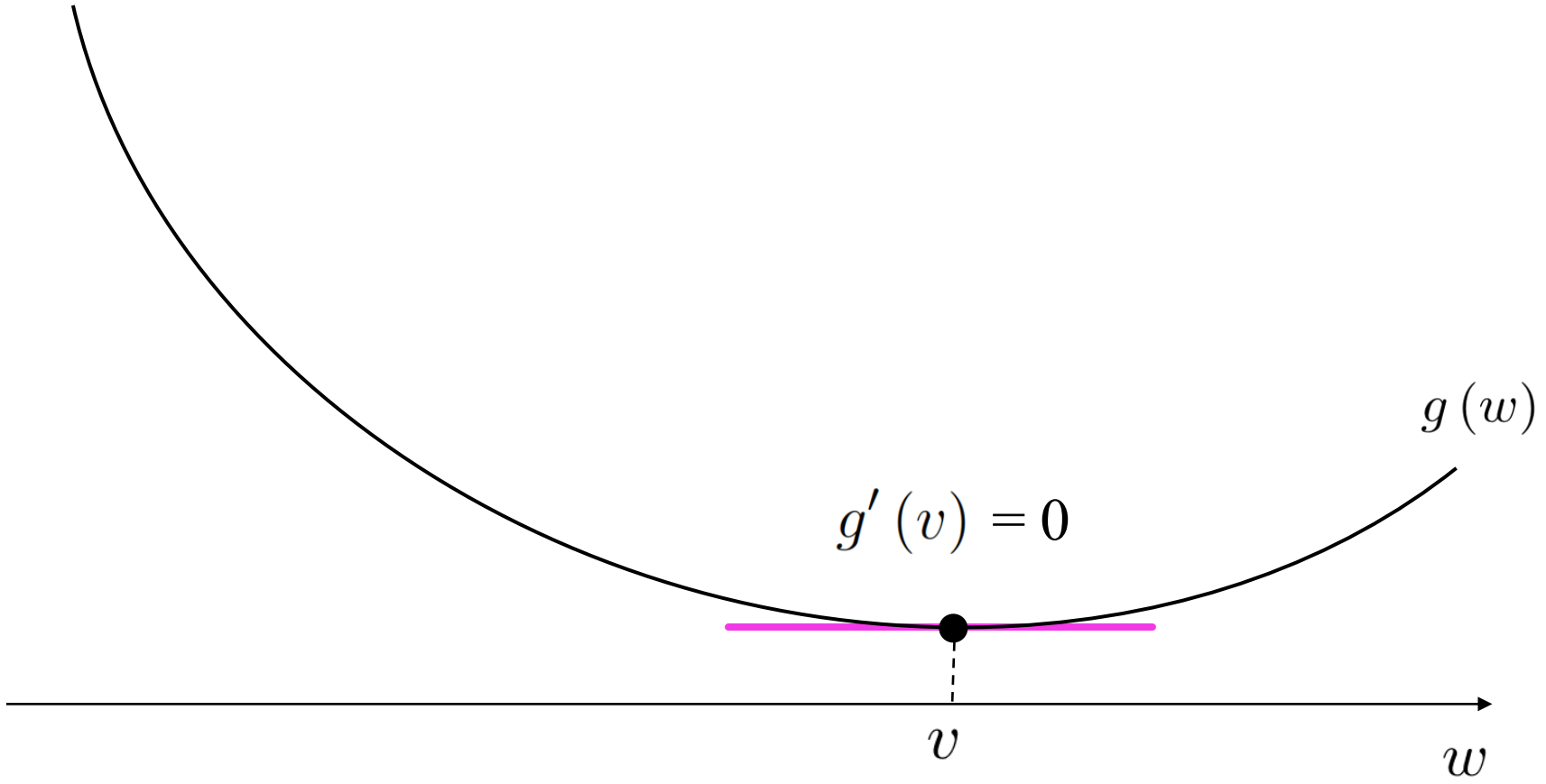
the derivative

- 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



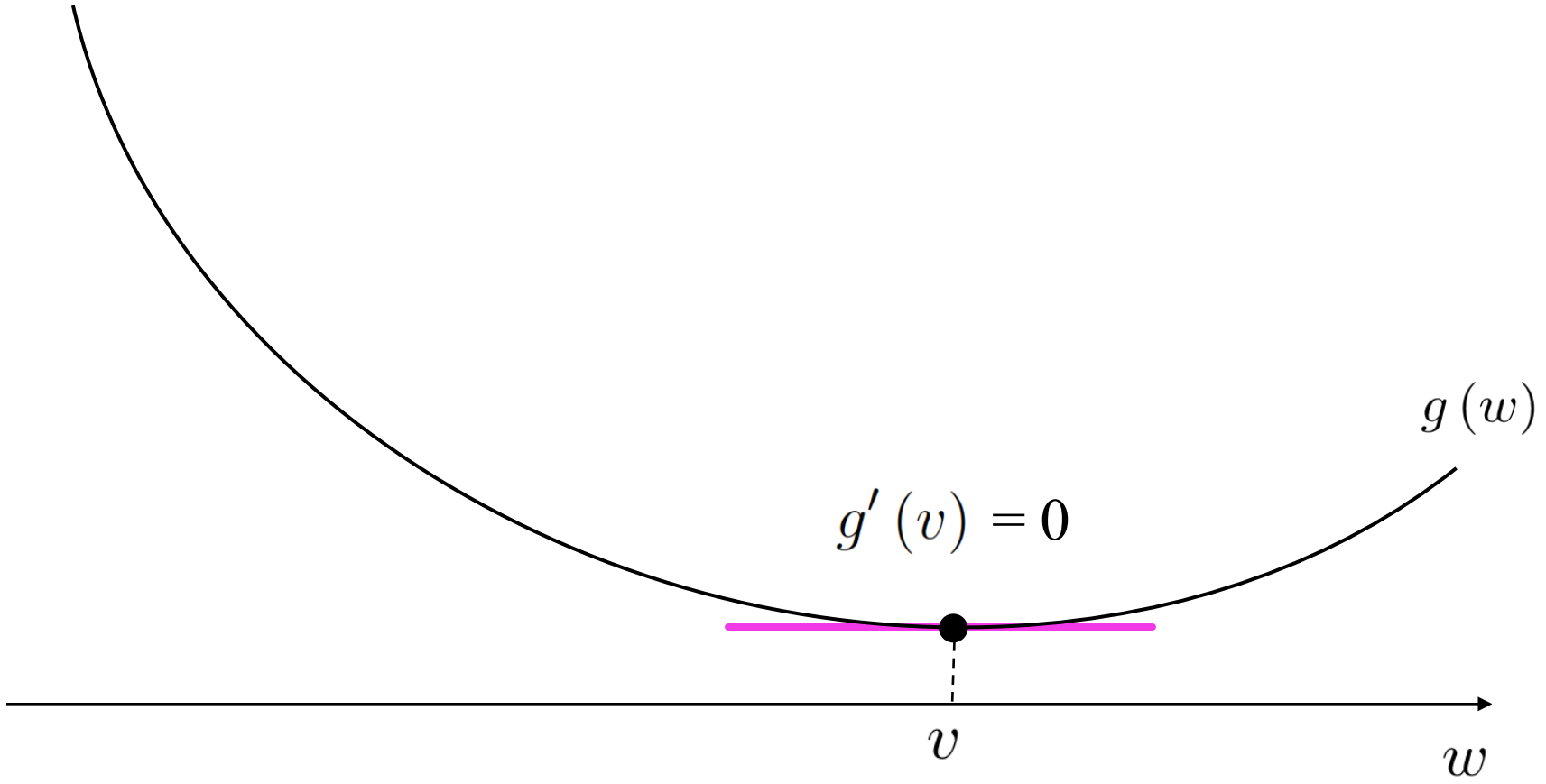
the derivative

- 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

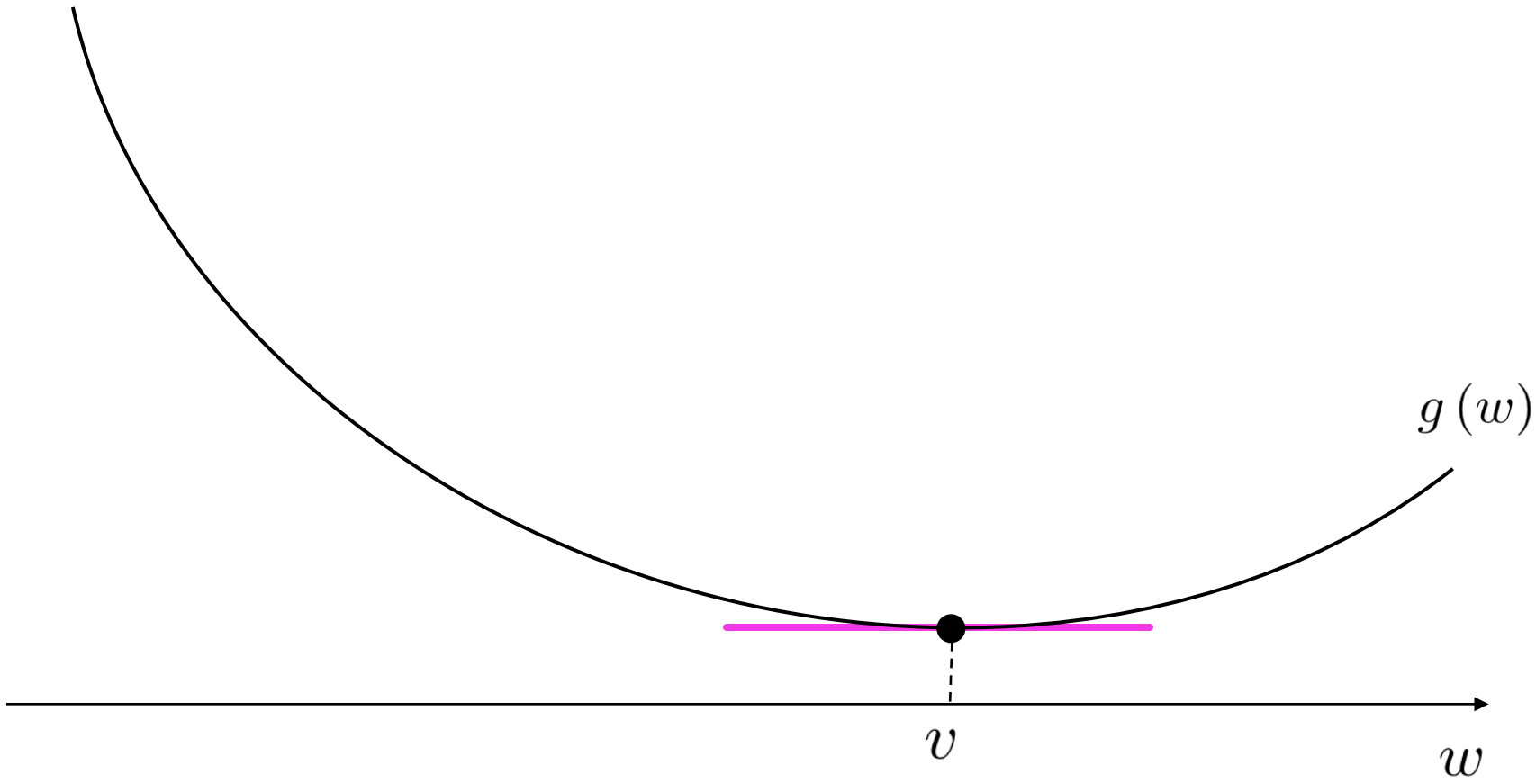


the derivative

- 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

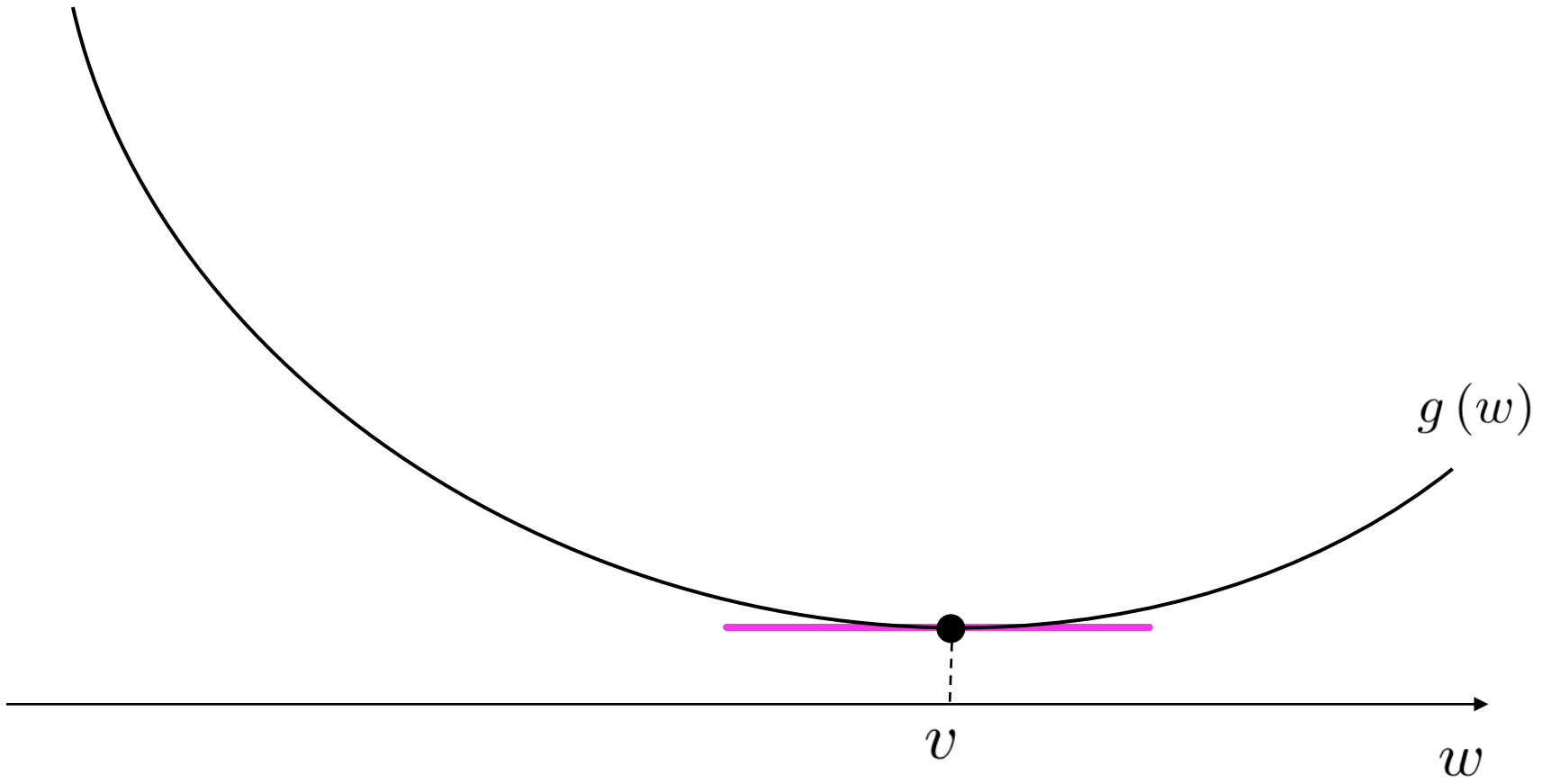


the derivative



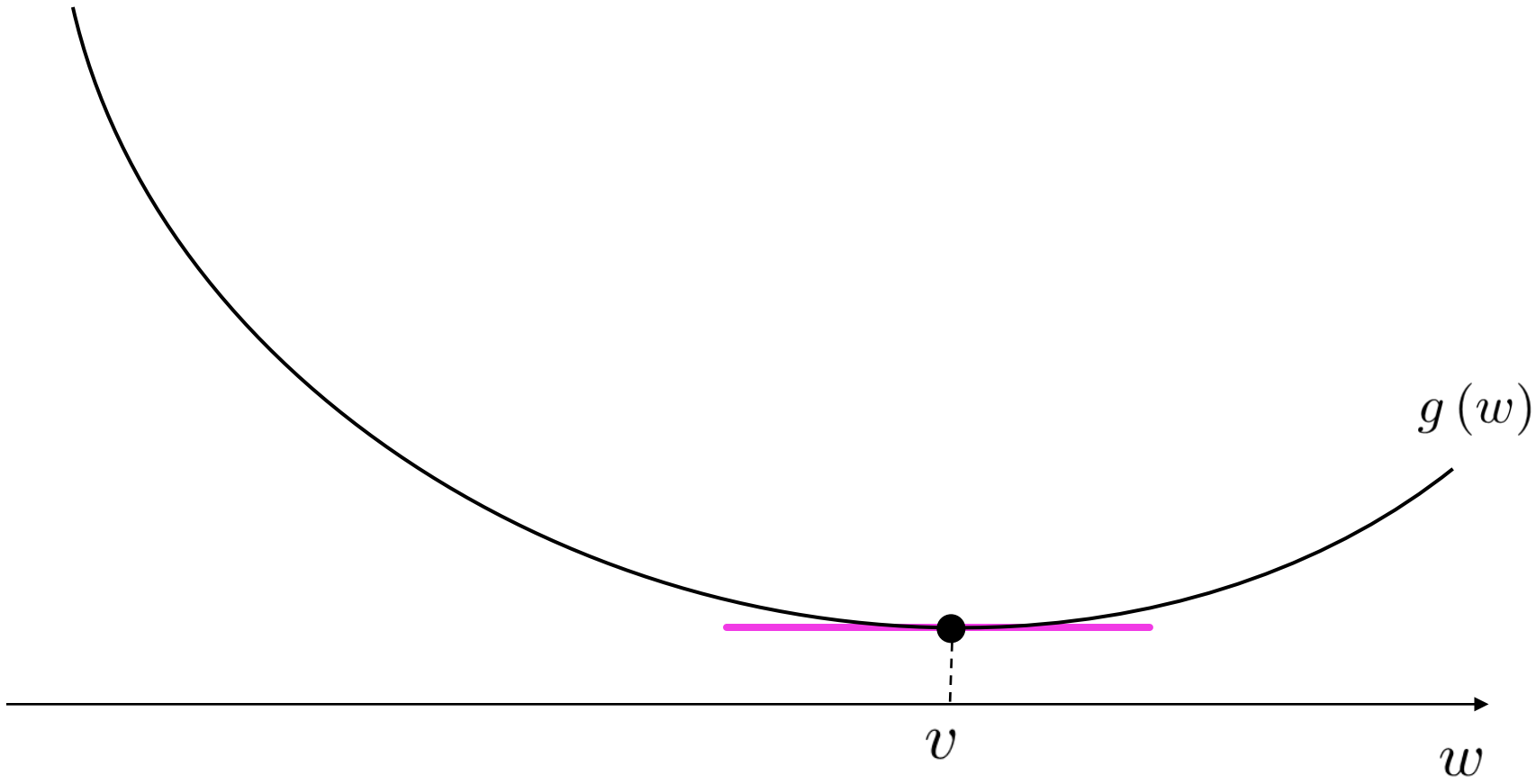
the derivative

- simple examples can be done using pencil and paper calculations



the derivative

- simple examples can be done using pencil and paper calculations
- but machine learning $g\left(w\right)$ costs require algorithmic solutions

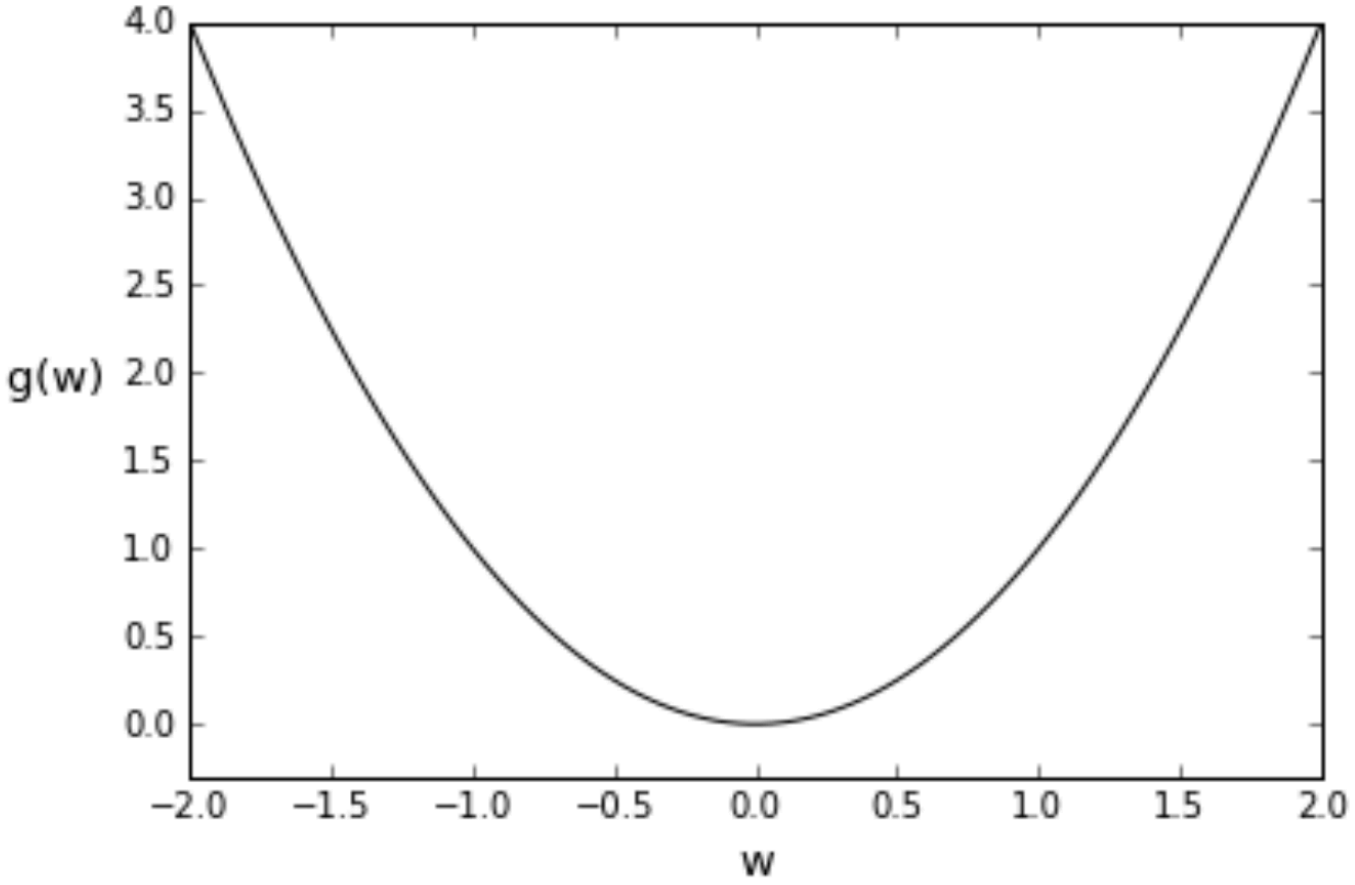


example

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

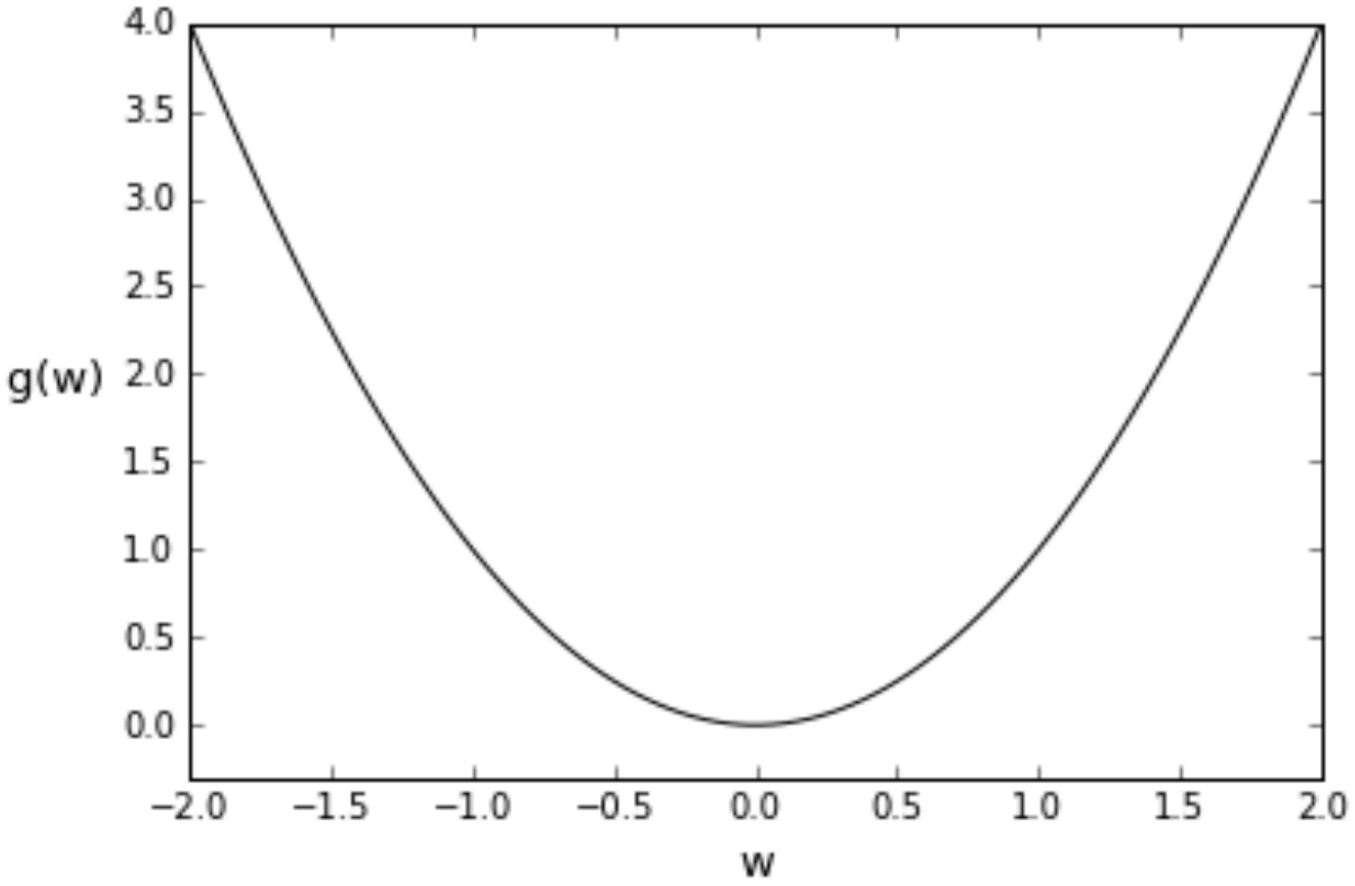
example

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



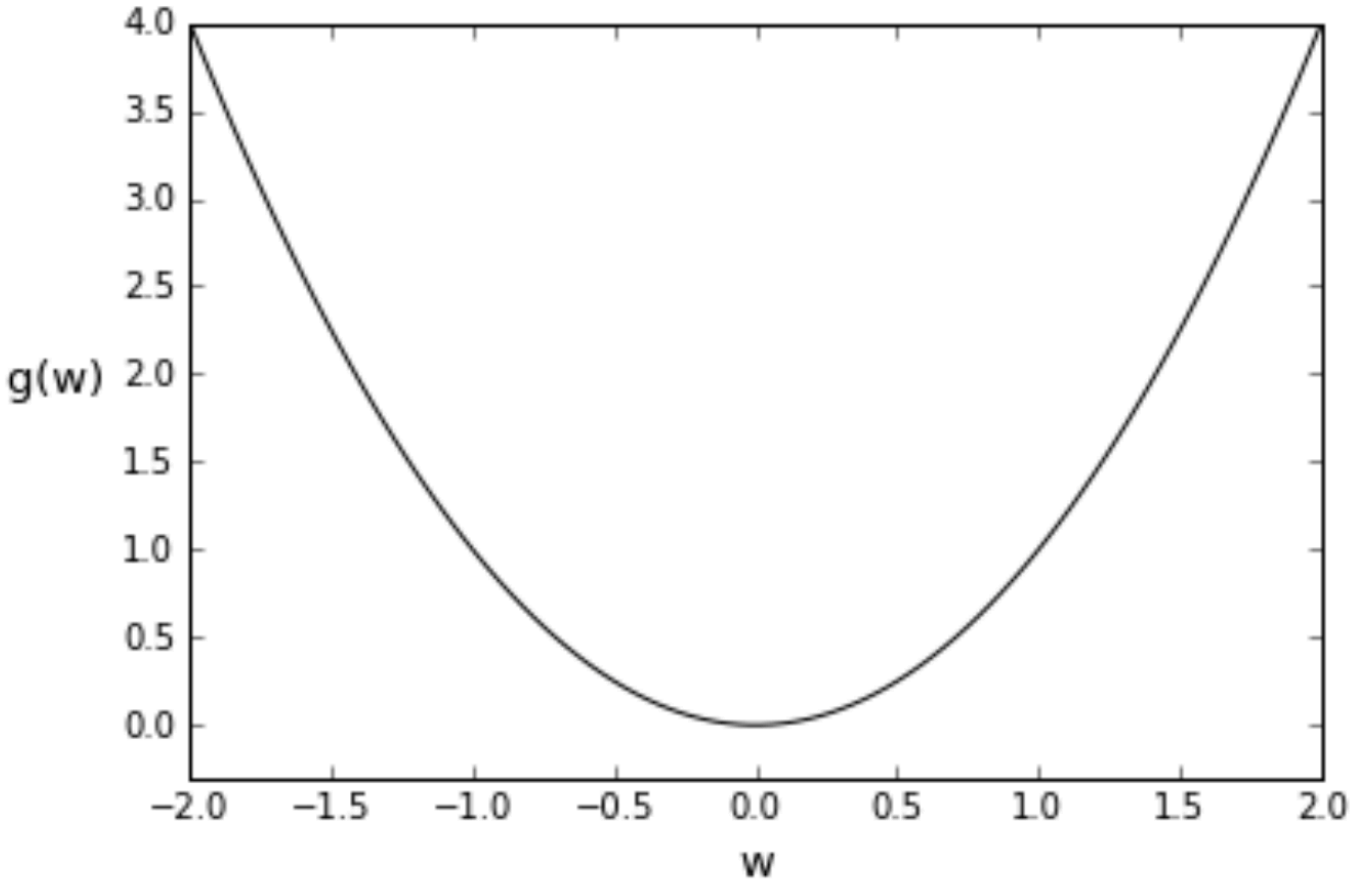
example

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



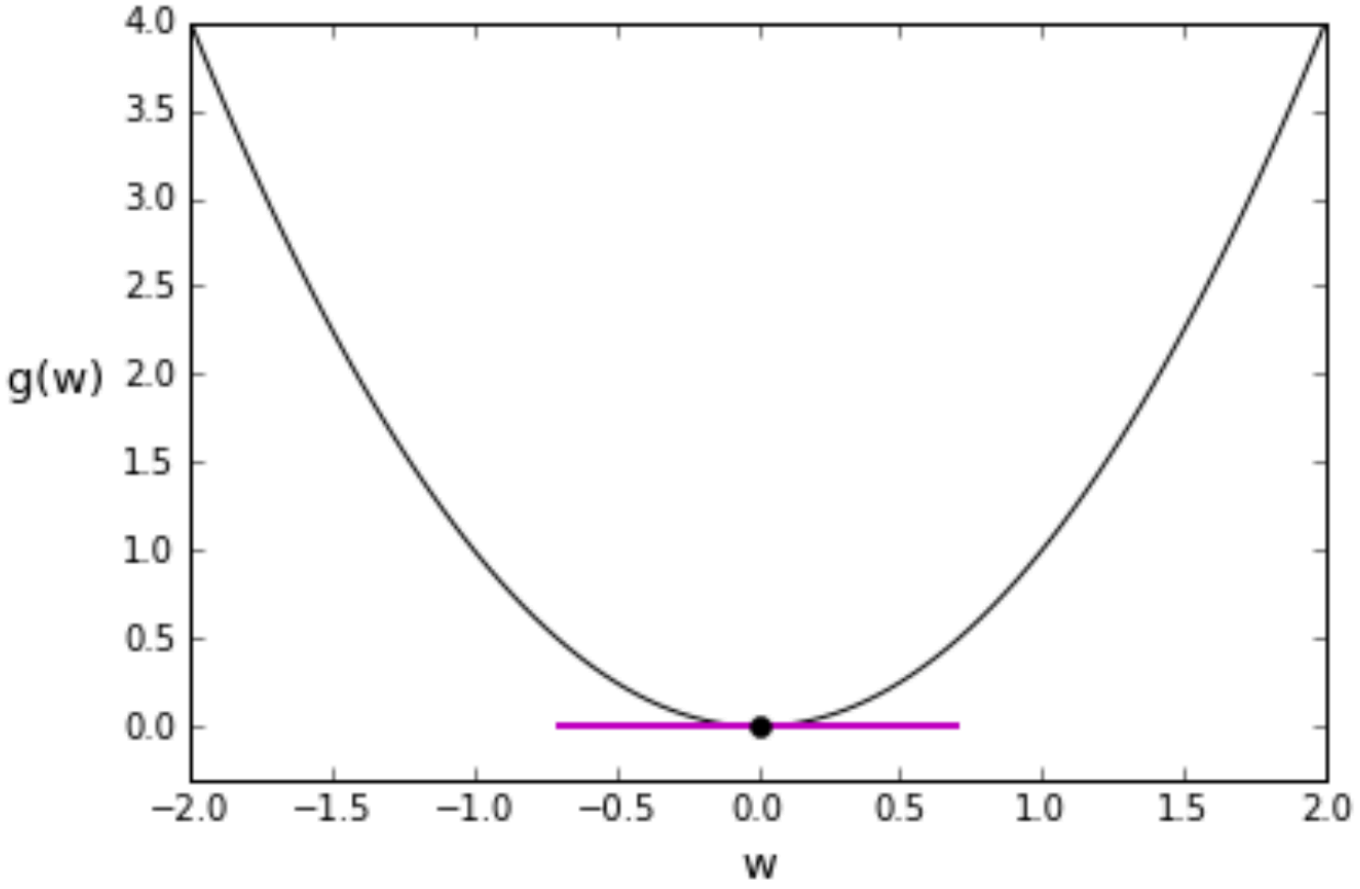
example

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

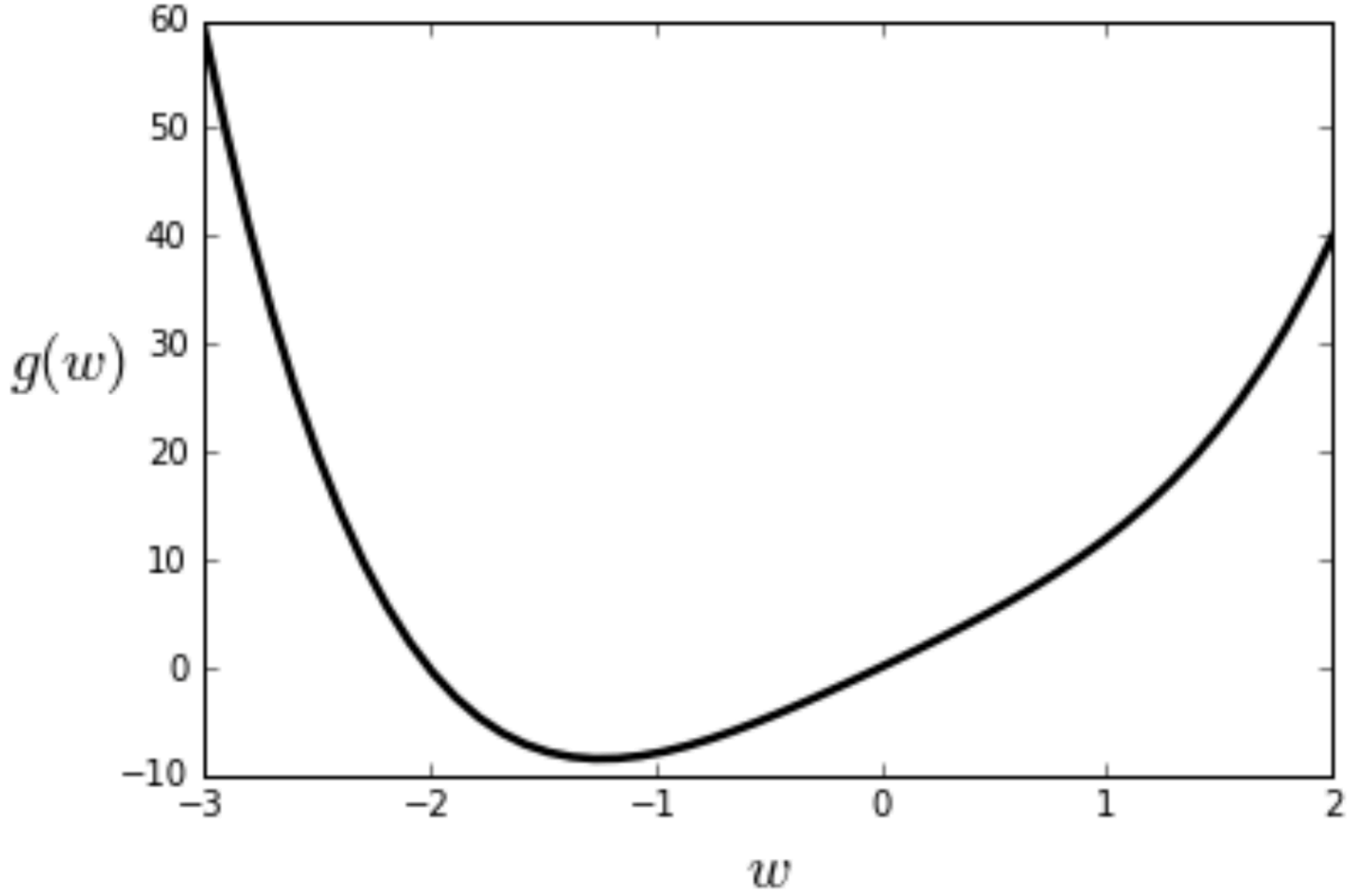


example

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



- OK but how about $g(w) = w^4 + w^2 + 10w$
- Looks simple enough...



WolframAlpha computational knowledge engine

minimum point of w^4 + w^2 + 10w

Examples Random

Input interpretation:

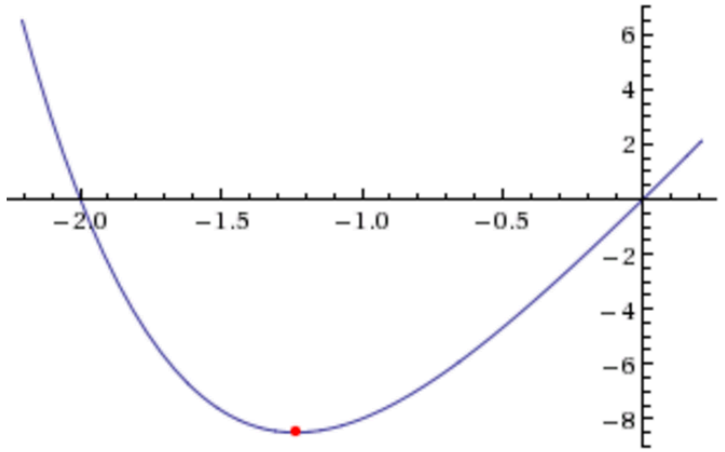
minimize	$w^4 + w^2 + 10\,w$

Global minimum:

Fewer digits More digits

$$\min\{w^4 + w^2 + 10\,w\} \approx -8.498464223154677377534566430142620373354396927609957 \text{ at } w \approx -1.234772825053296980414119894096777410653656407067694$$

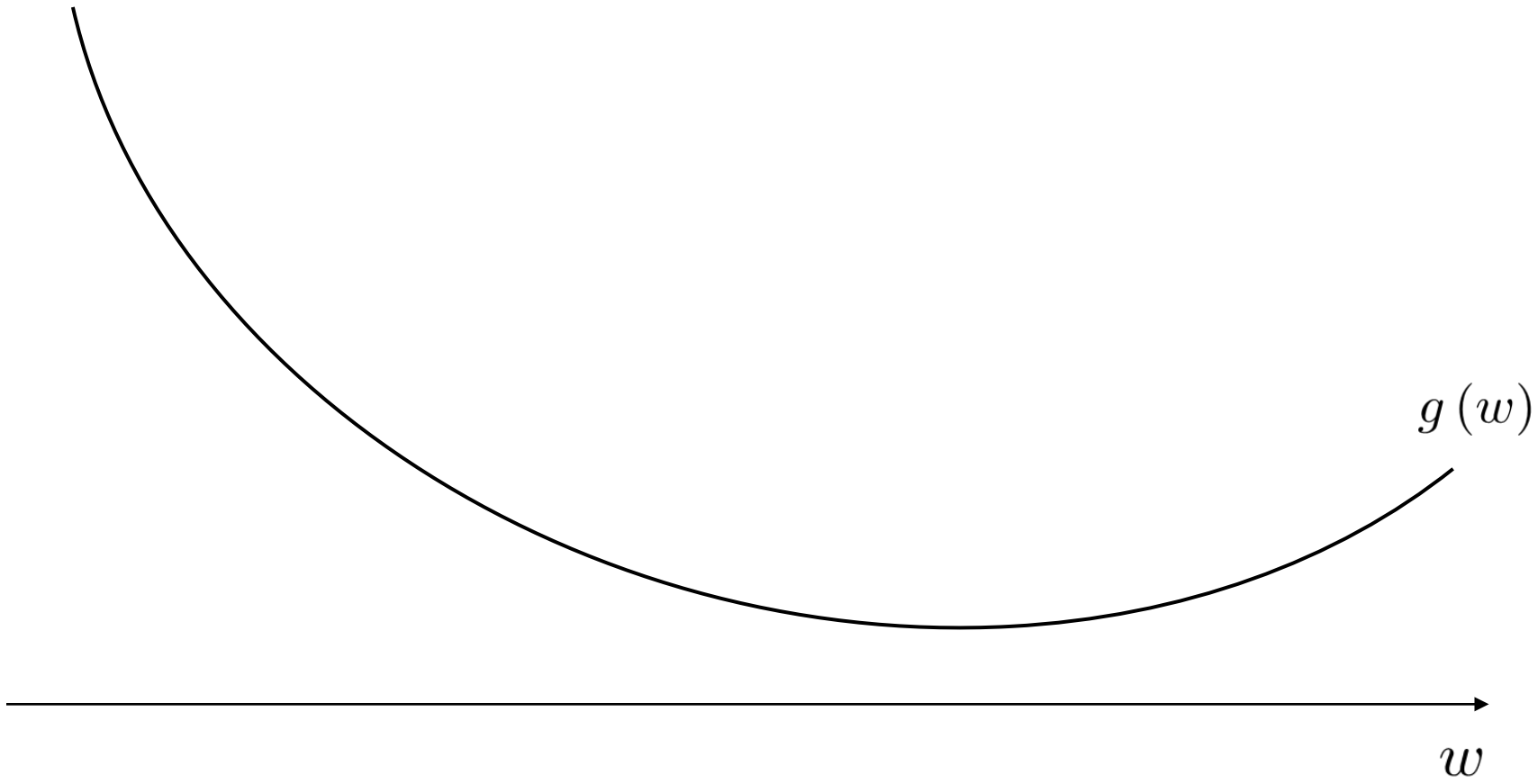
Plot:



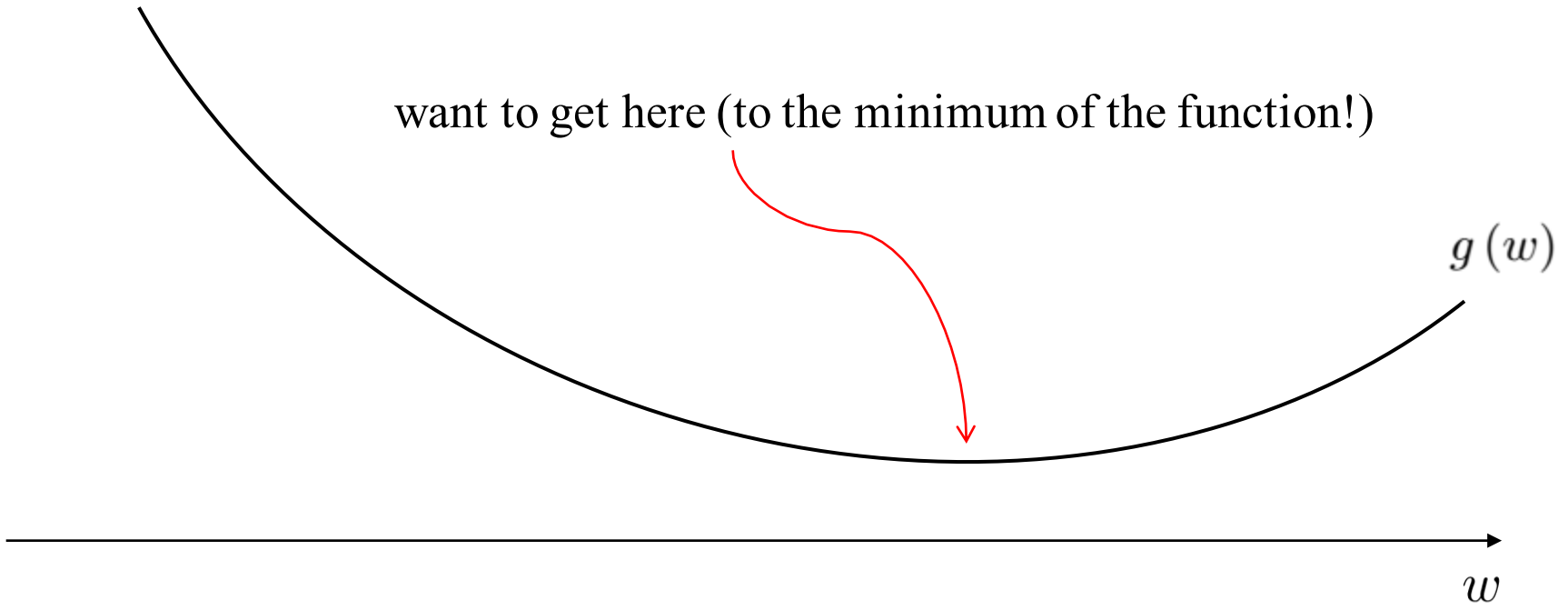
(x from −2.2 to 0.2)

Gradient descent

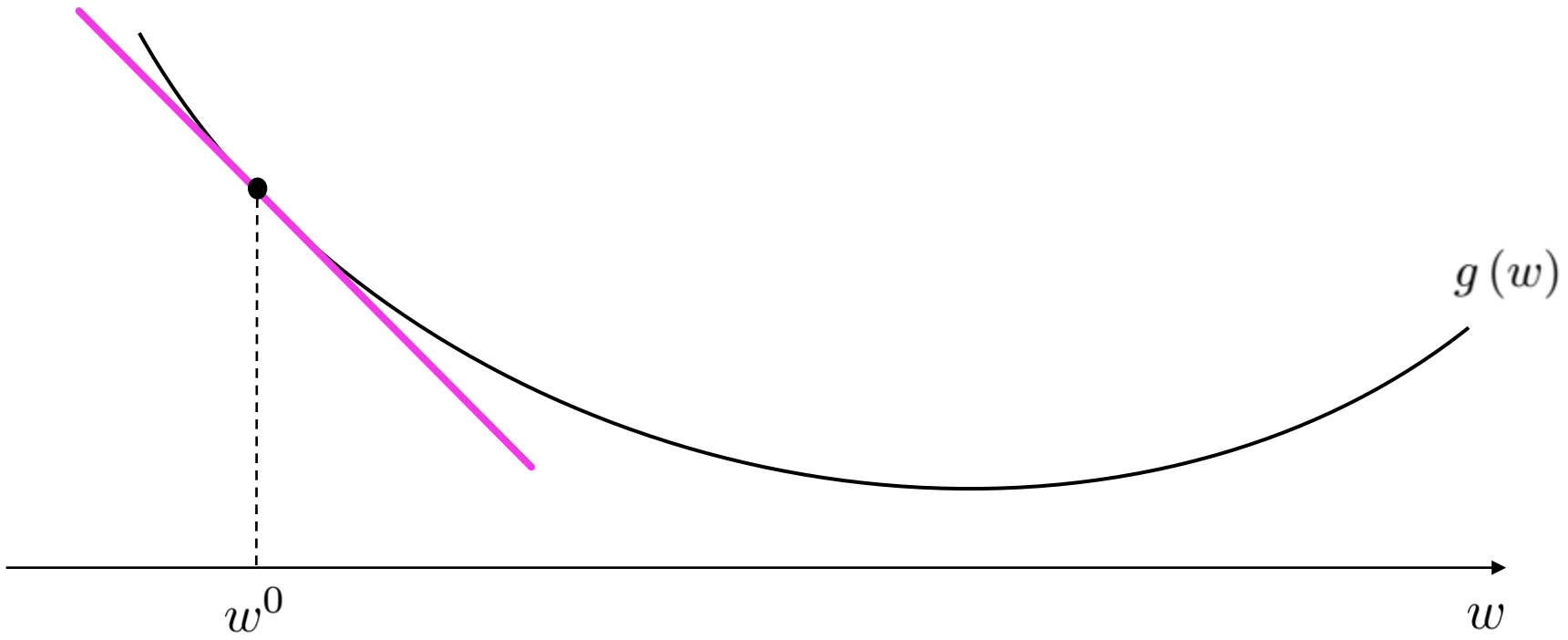
gradient descent



gradient descent

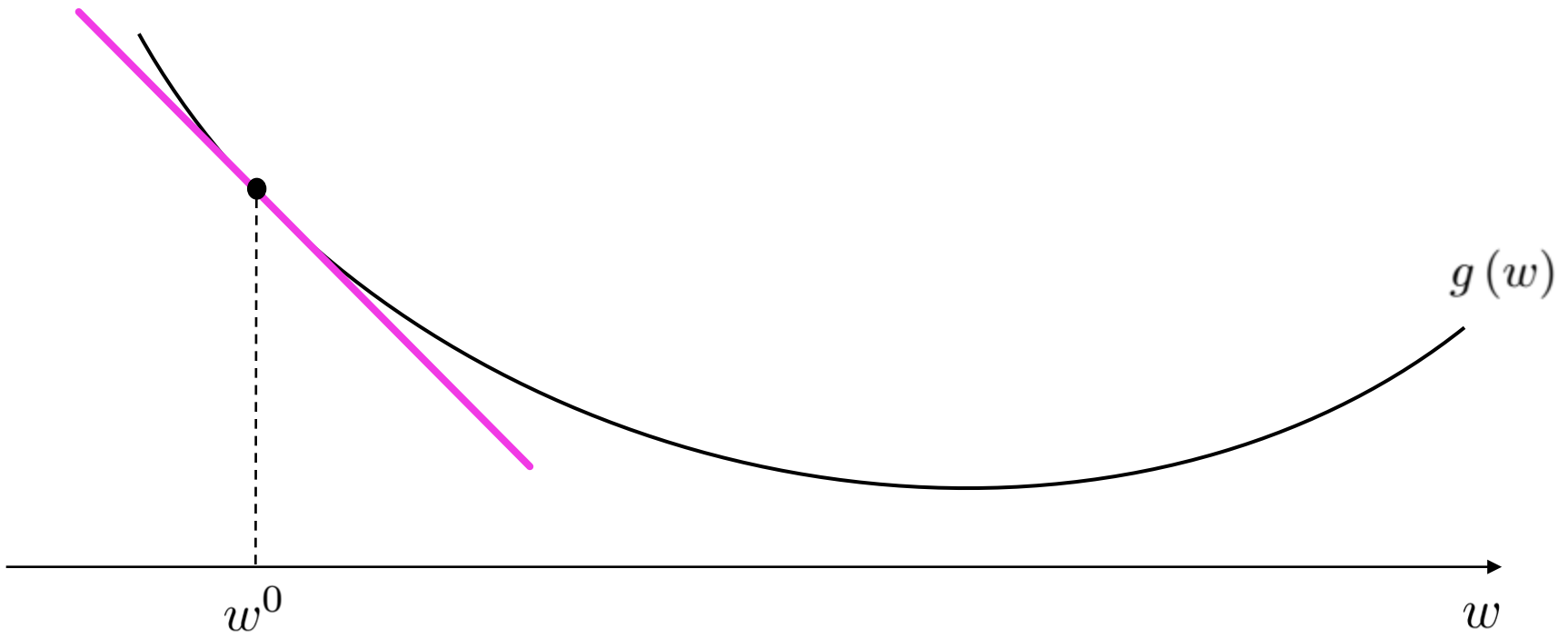


gradient descent



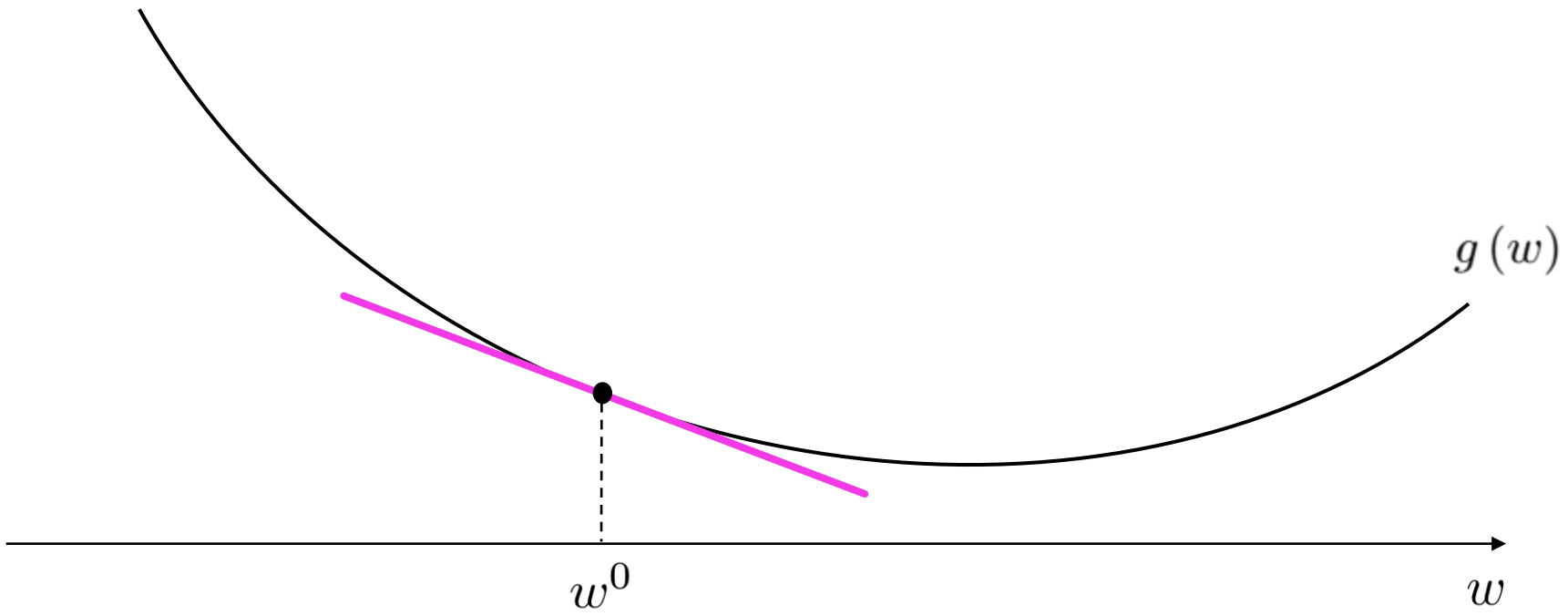
gradient descent

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



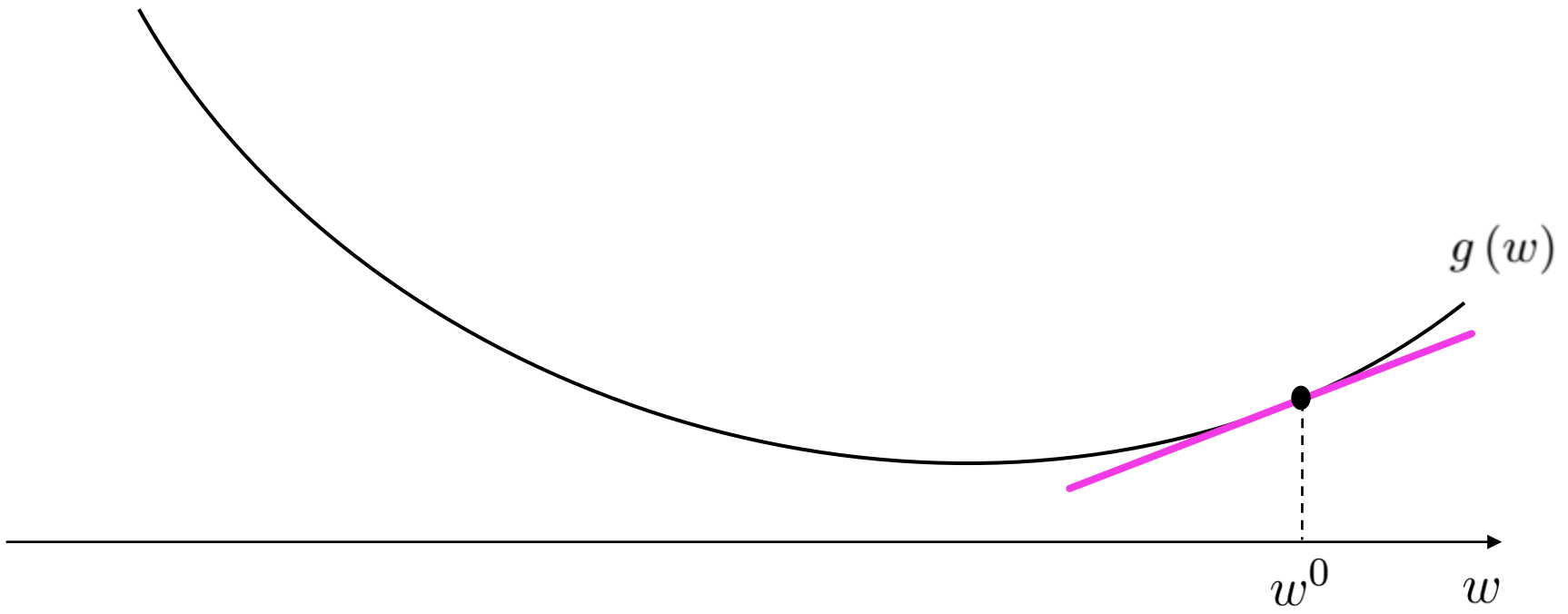
gradient descent

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



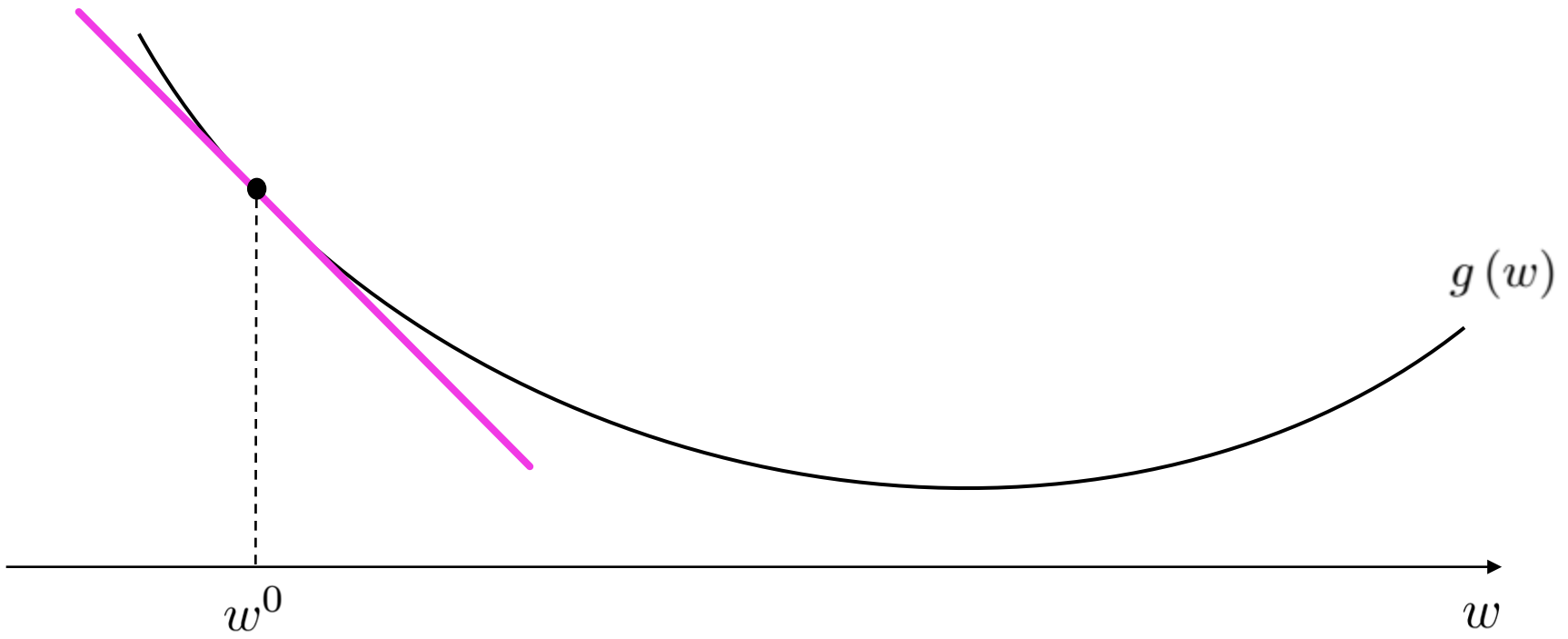
gradient descent

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



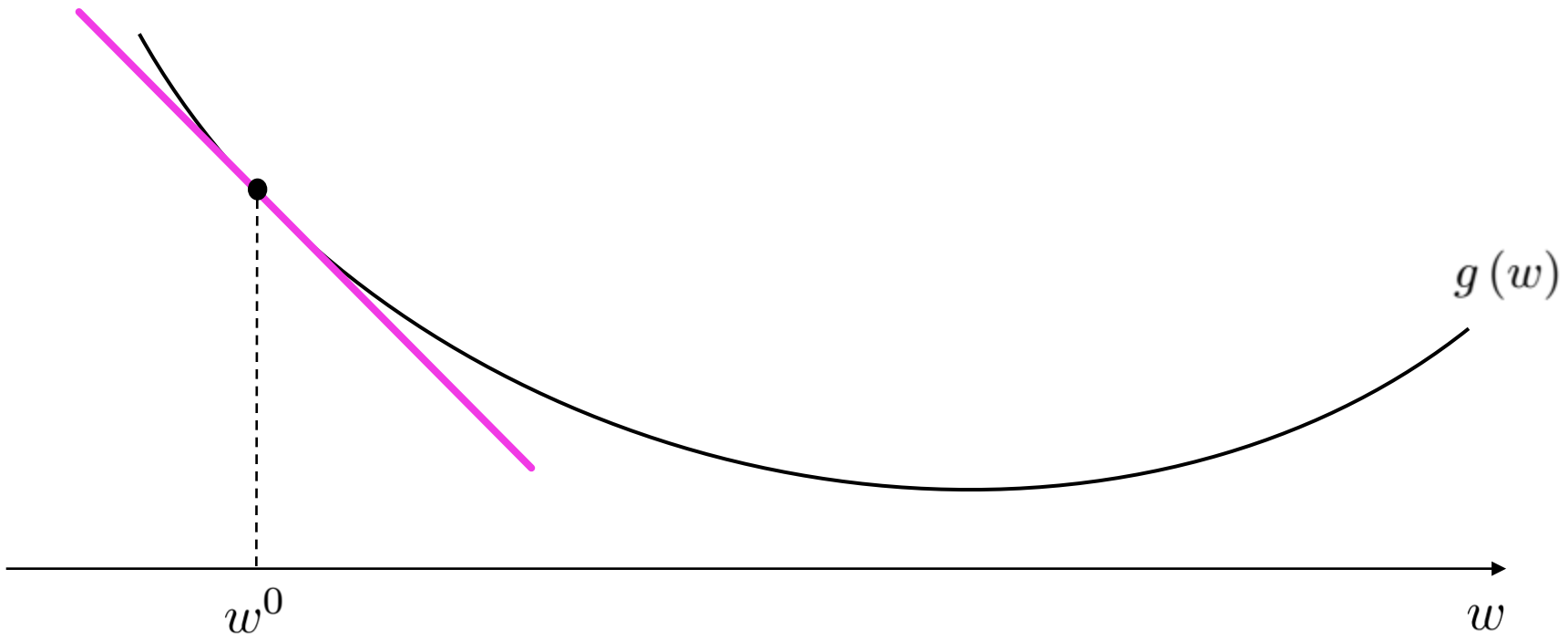
gradient descent

- ✓ 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

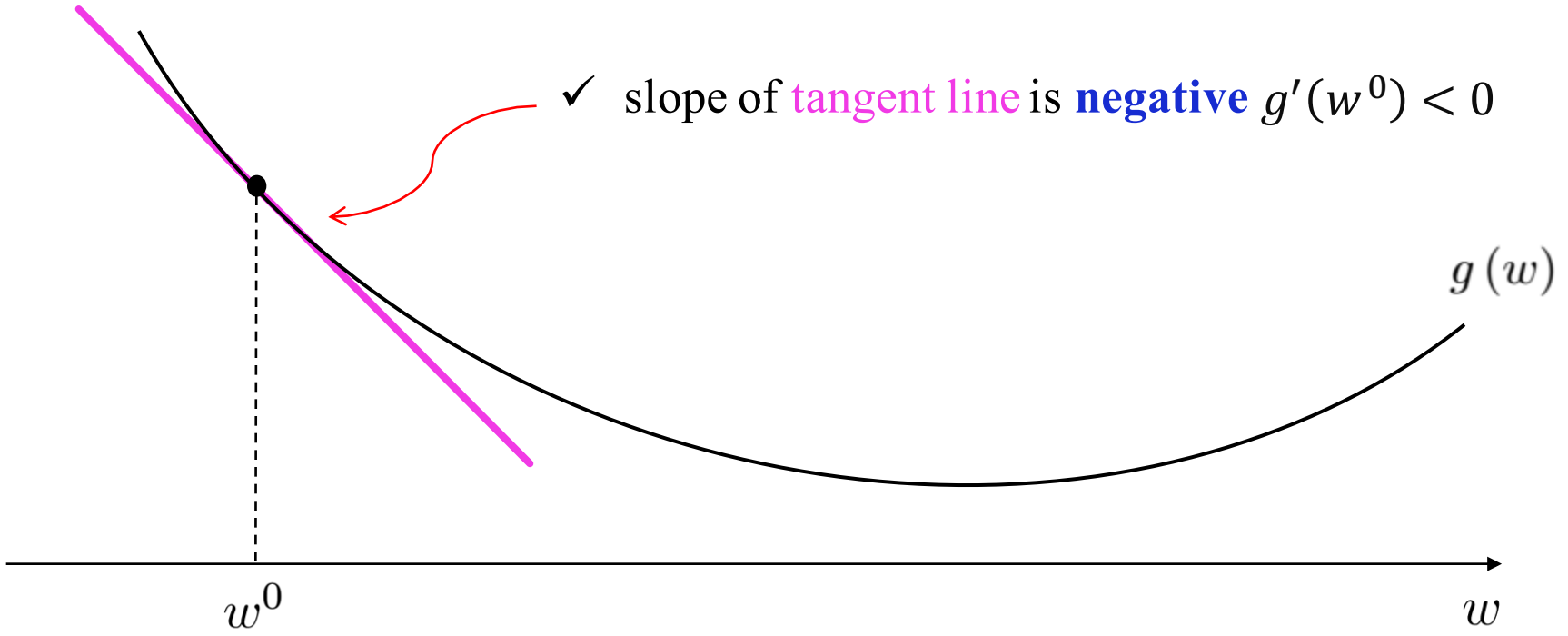


gradient descent

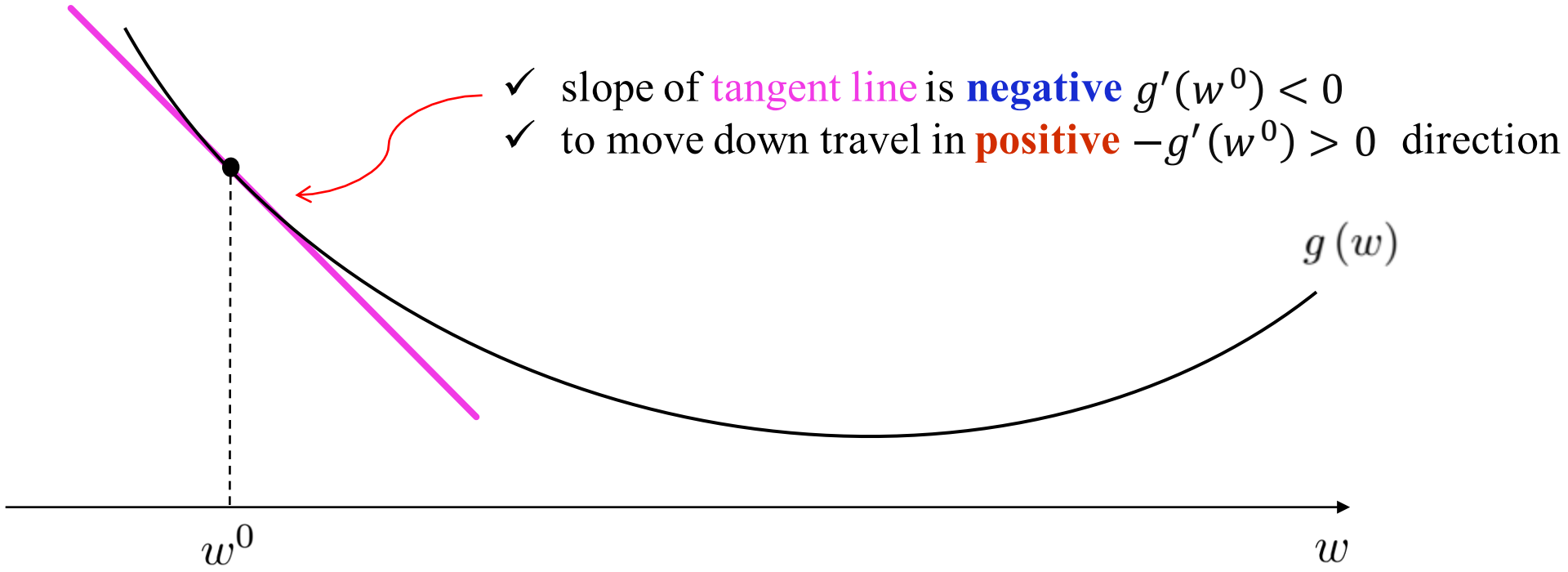
- ✓ the tangent line looks a whole lot like $g(w)$ near w^0
- ✓ traveling downward on line ≈ traveling downward on $g(w)$ near w^0
- ✓ the downward direction on a line is easily computable



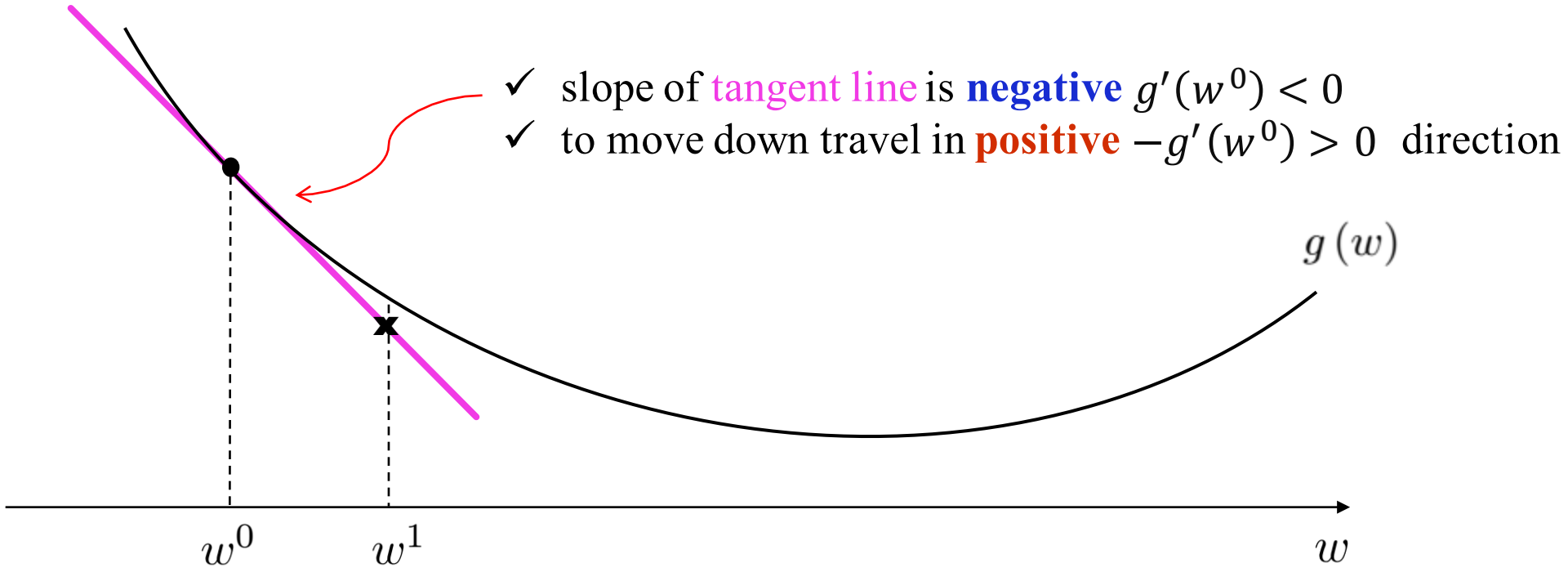
gradient descent



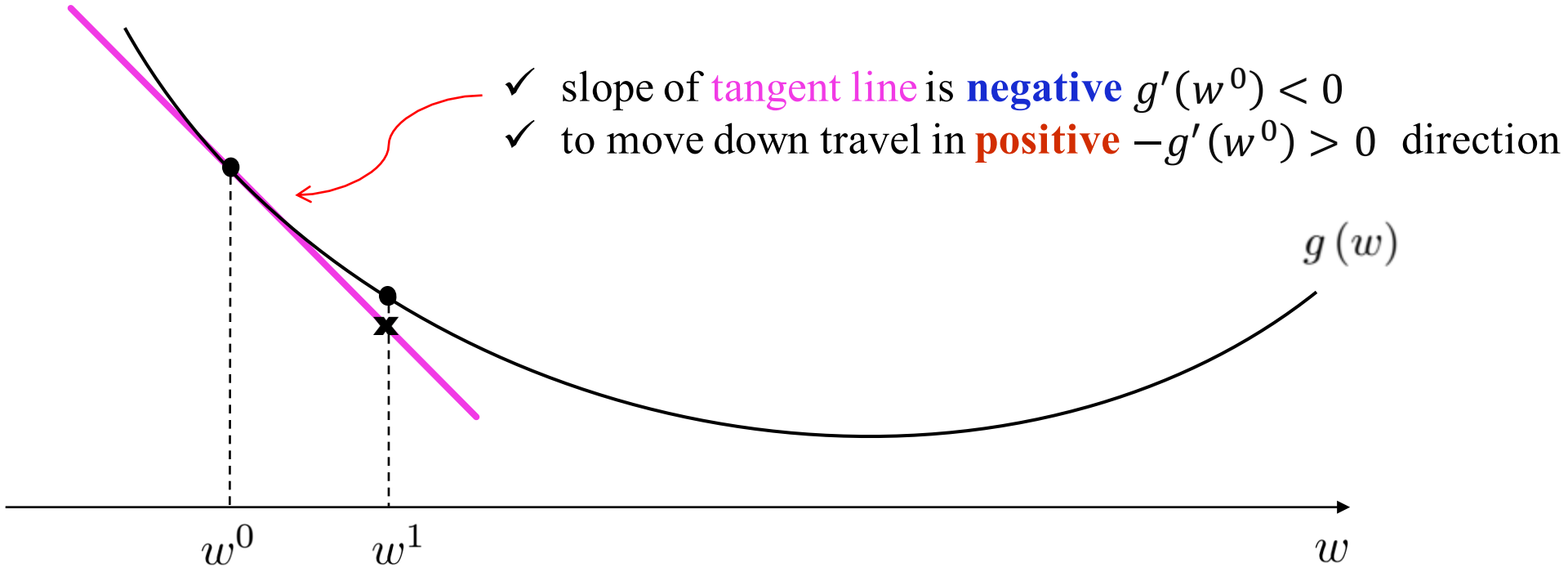
gradient descent



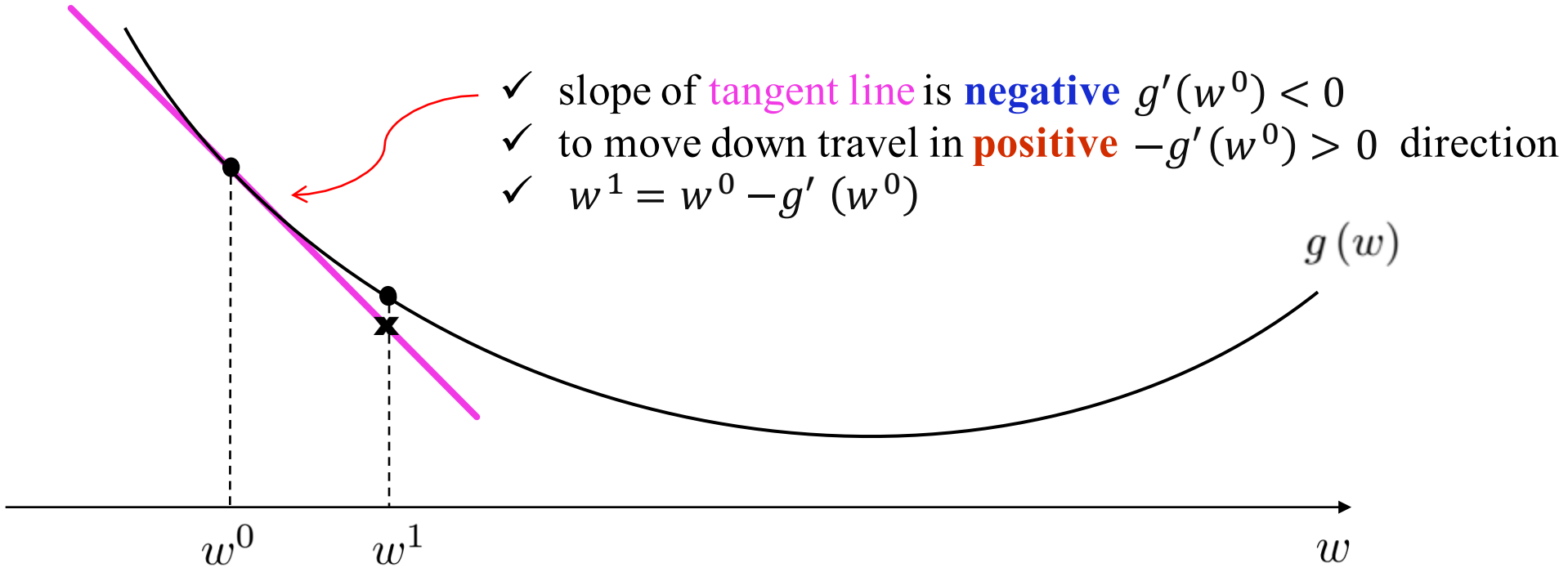
gradient descent



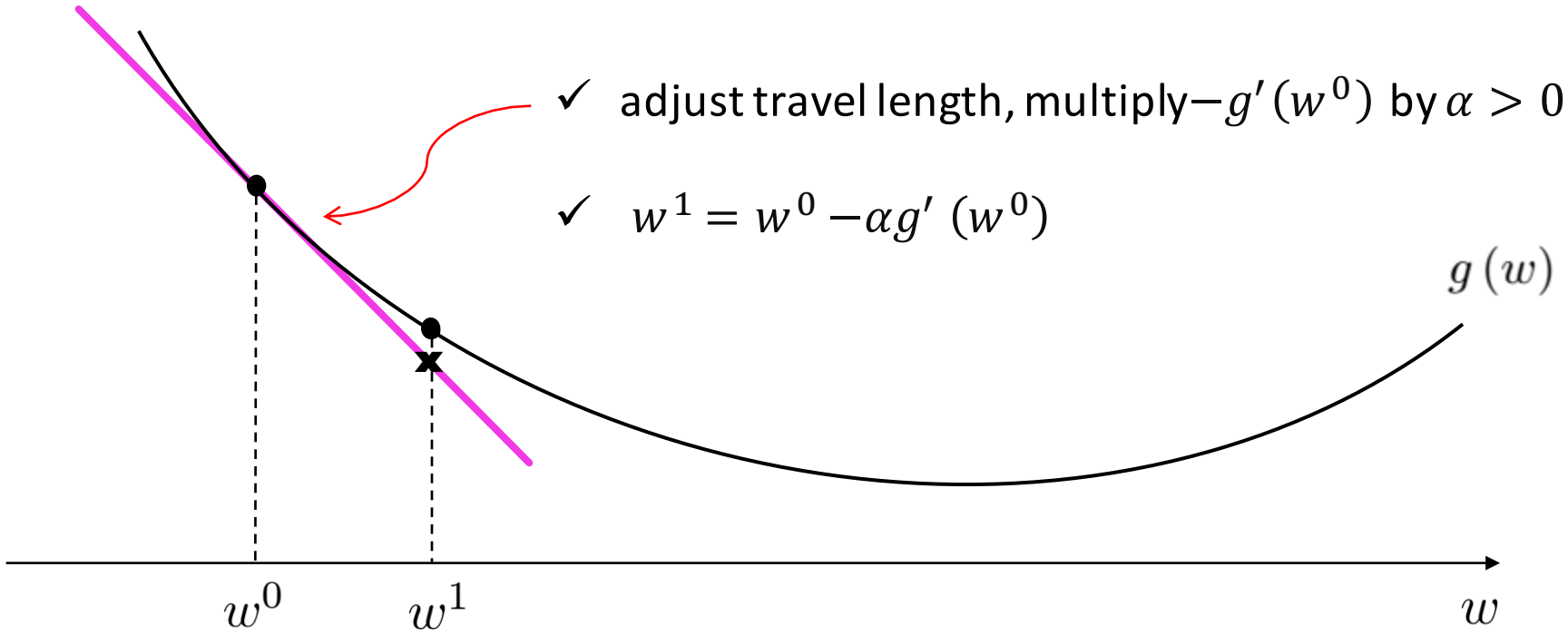
gradient descent



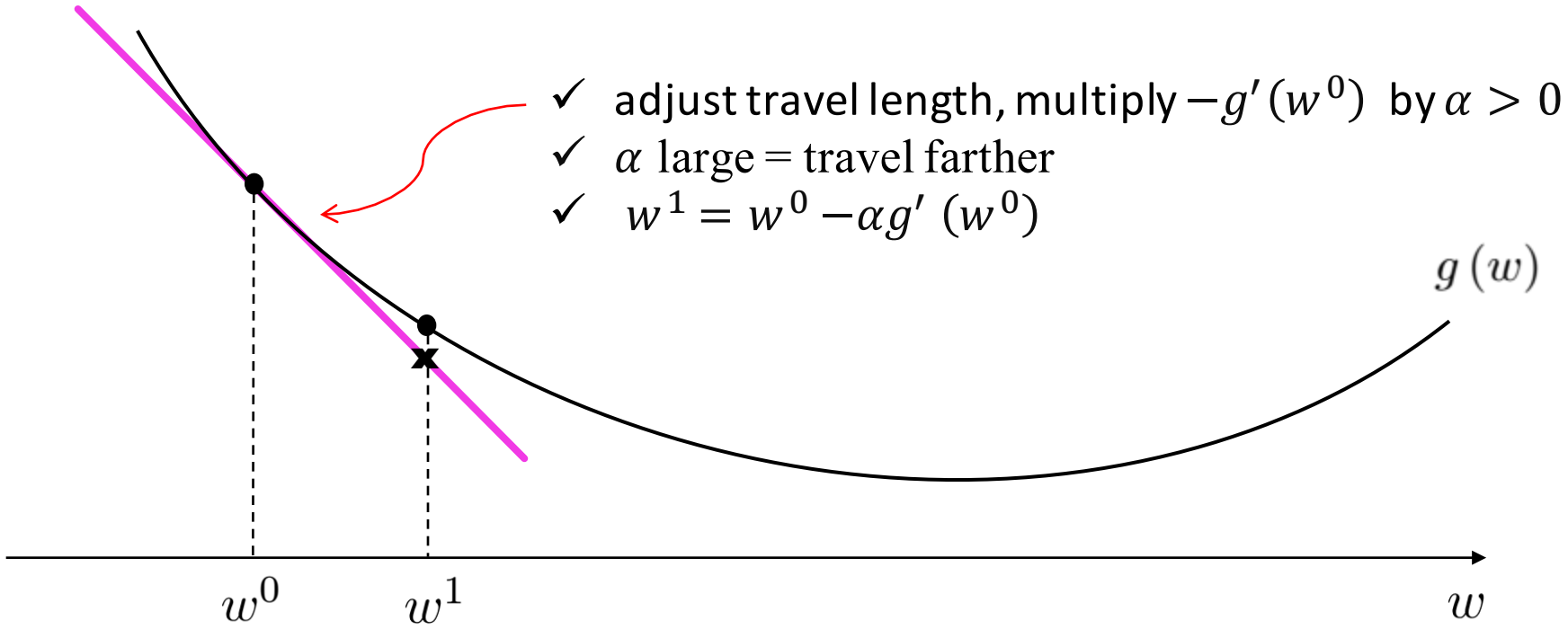
gradient descent



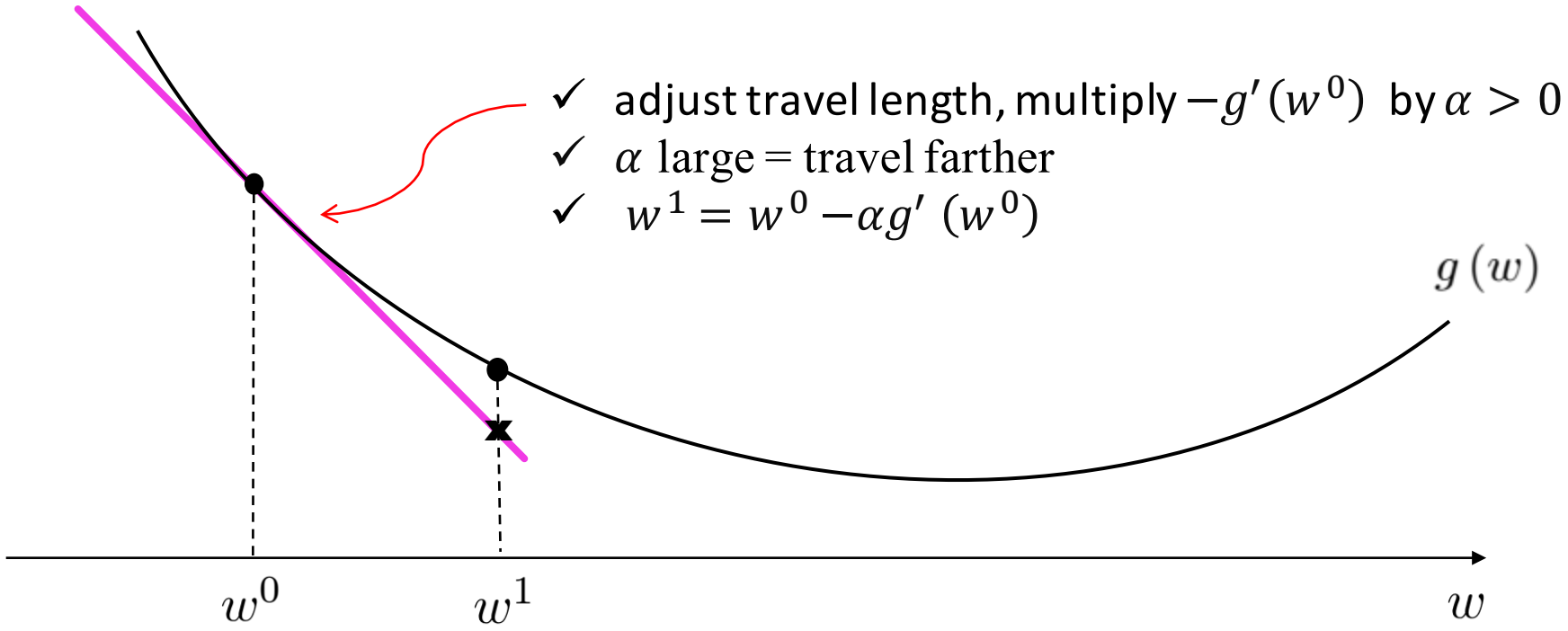
gradient descent



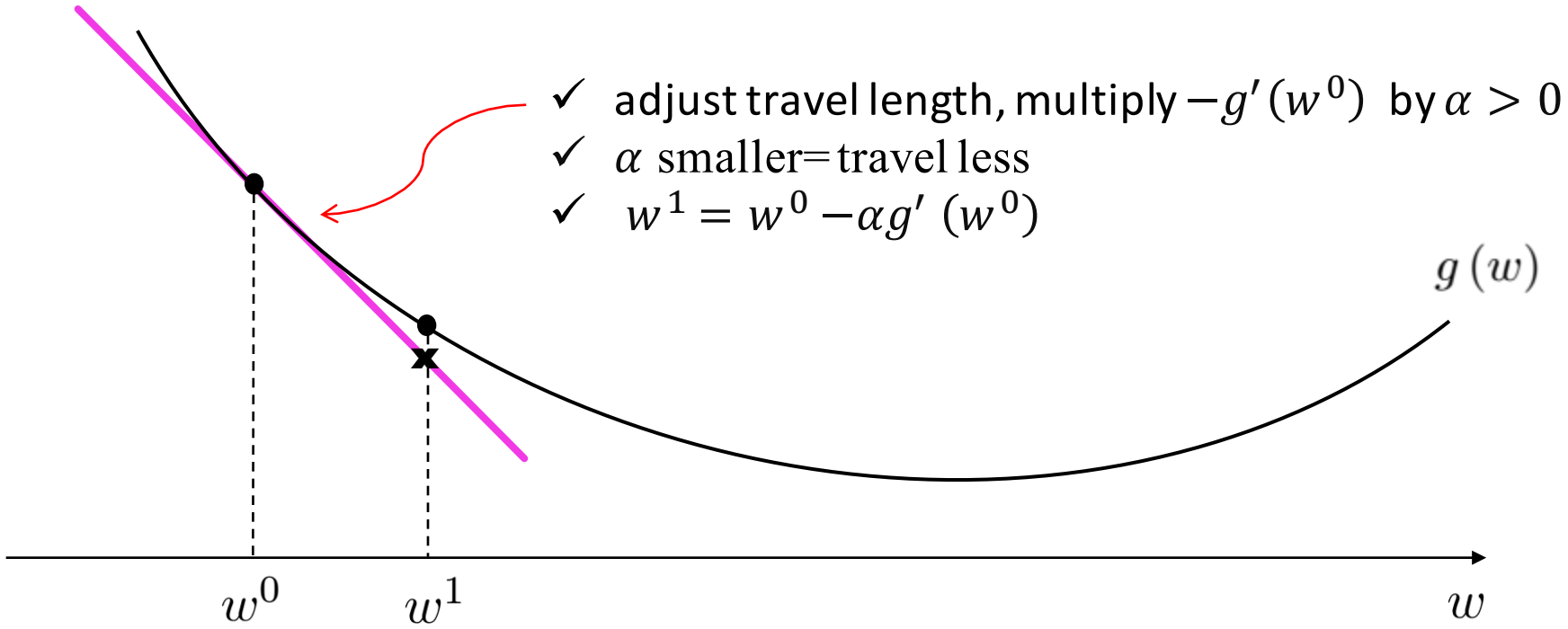
gradient descent



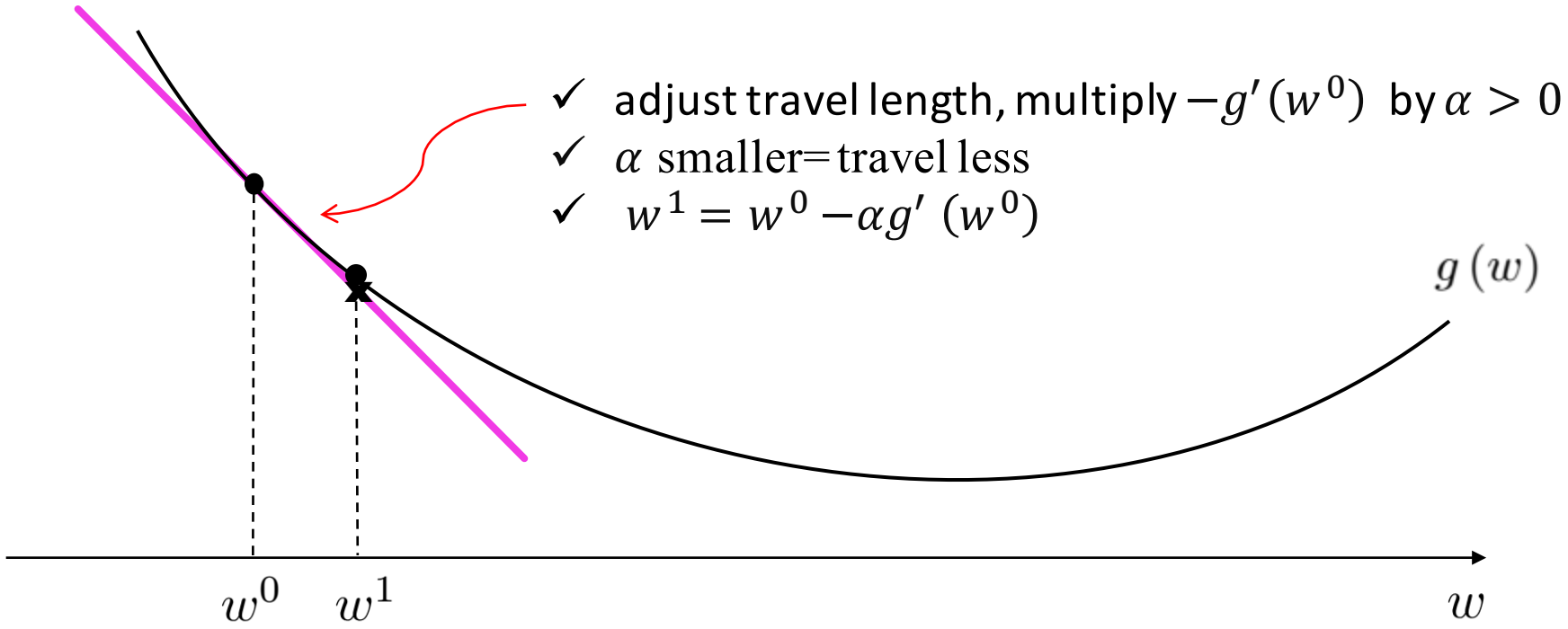
gradient descent



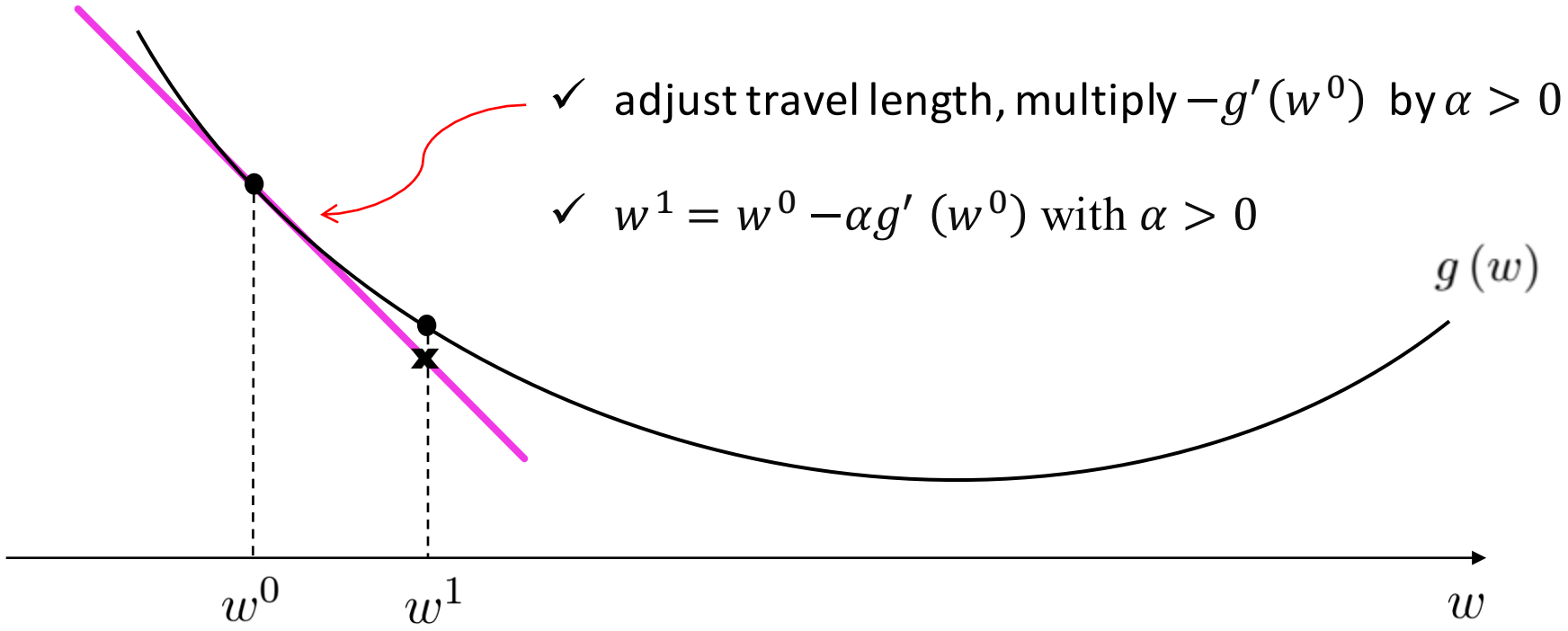
gradient descent



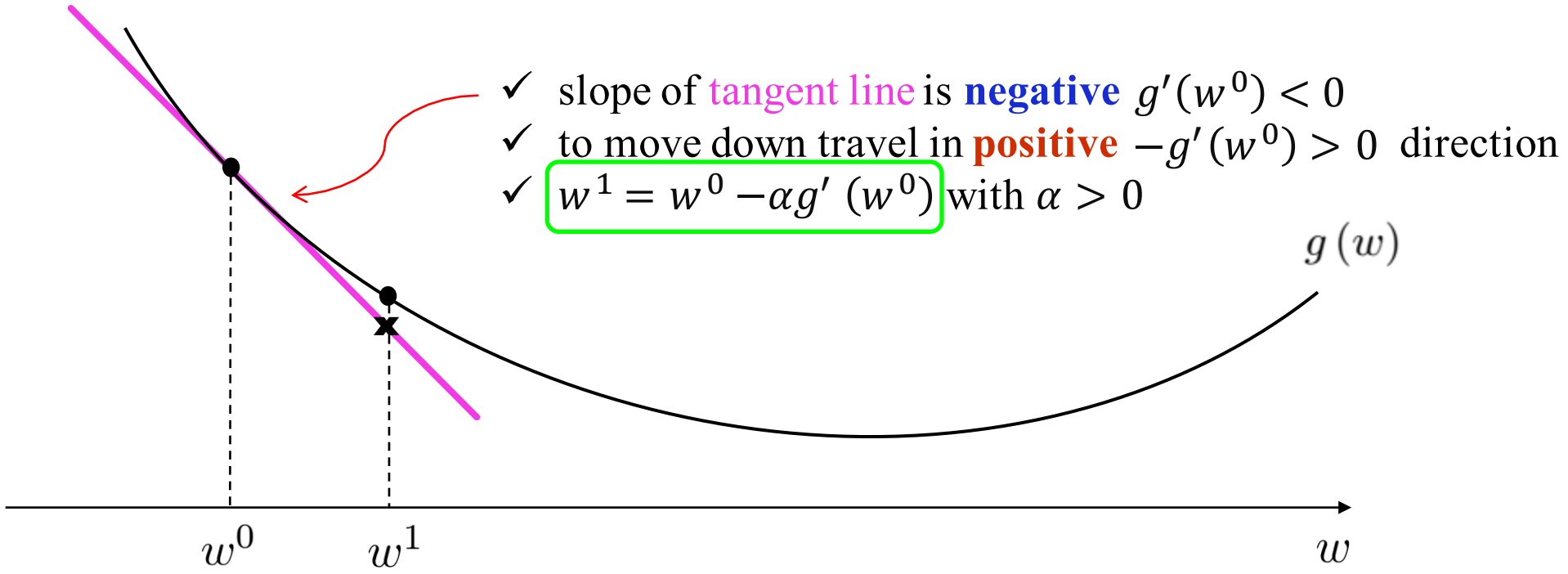
gradient descent



gradient descent

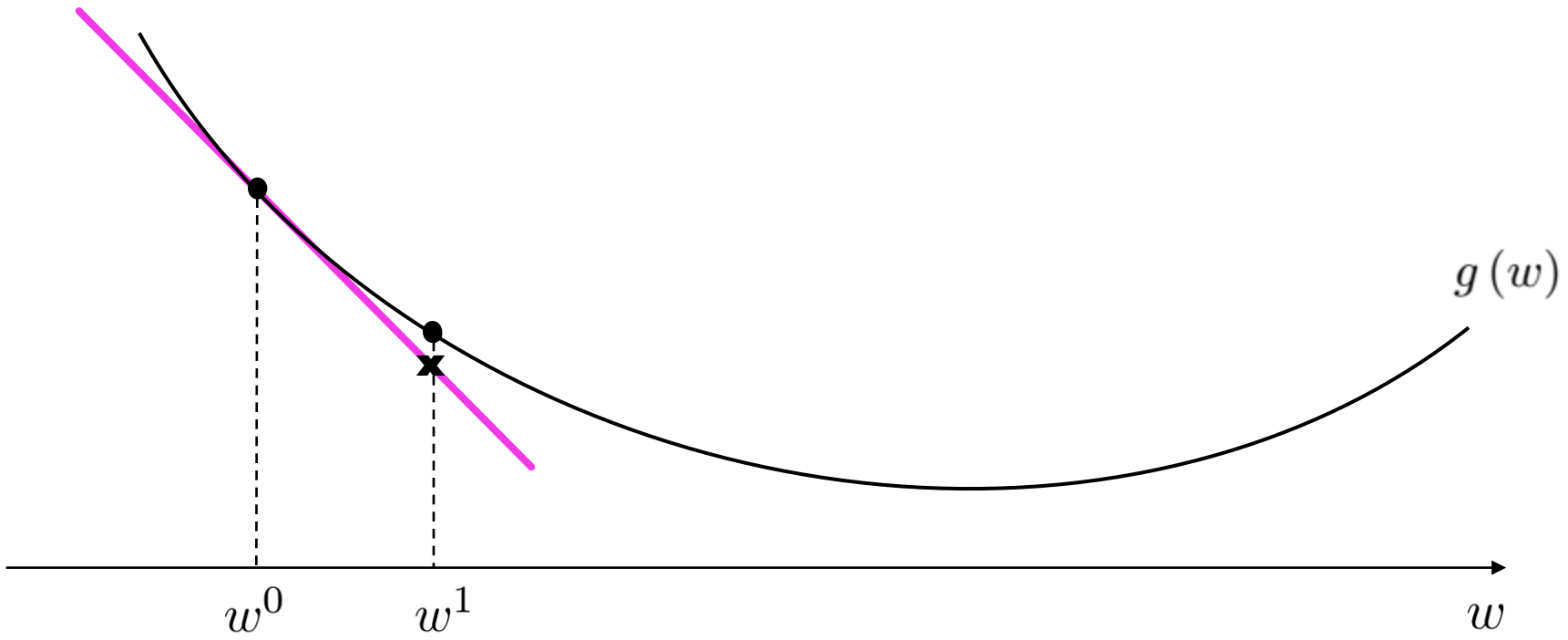


gradient descent



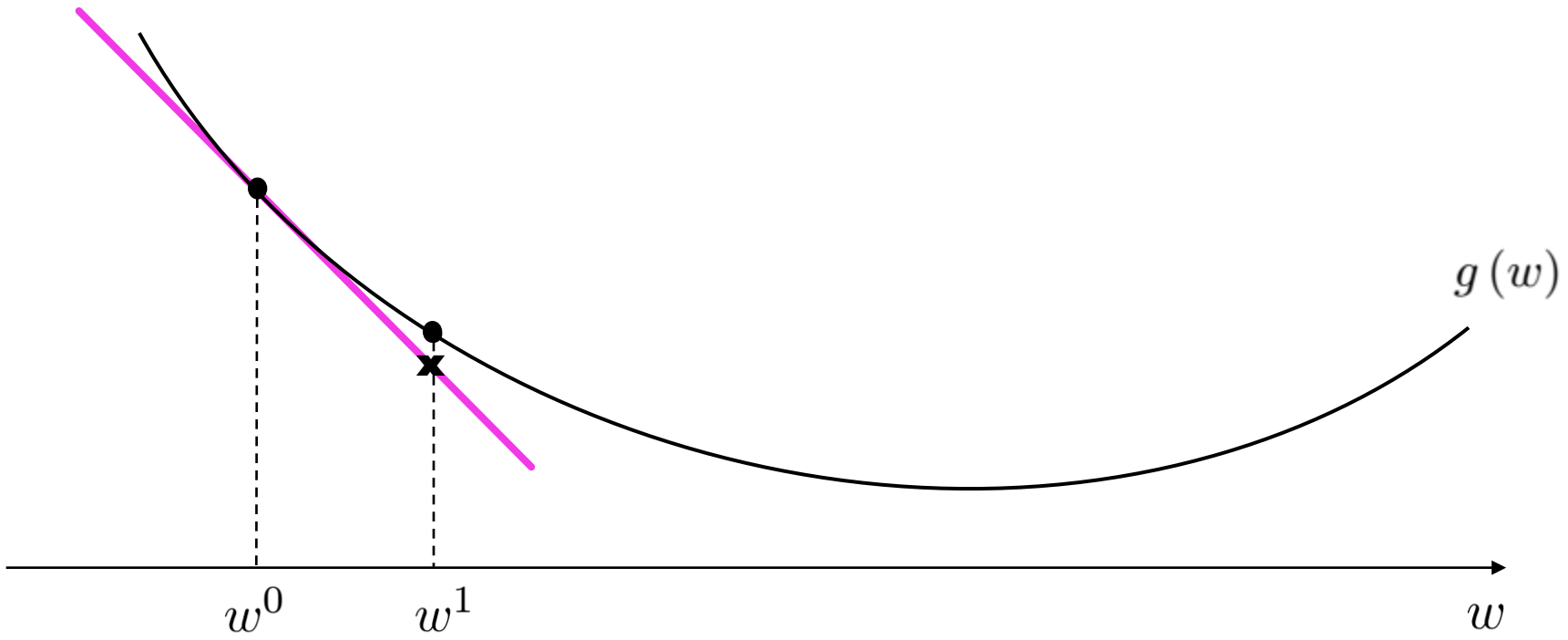
gradient descent

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



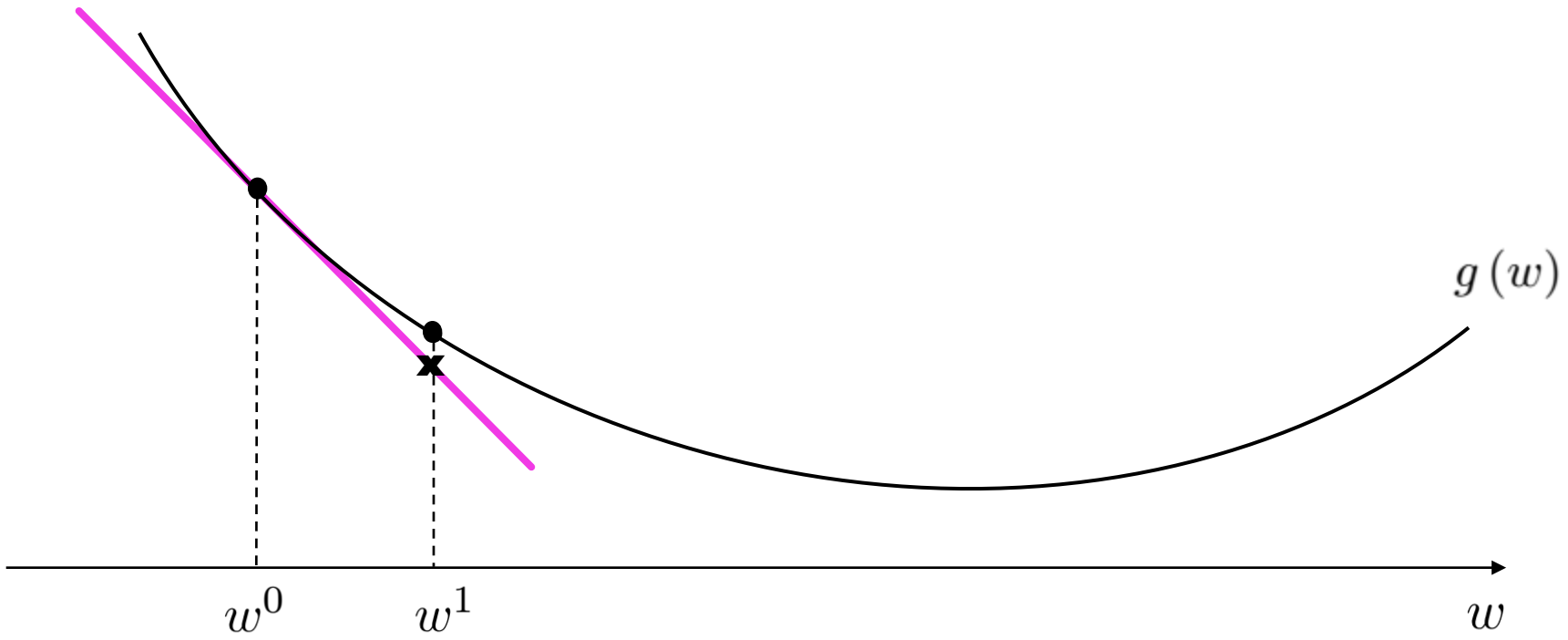
gradient descent

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



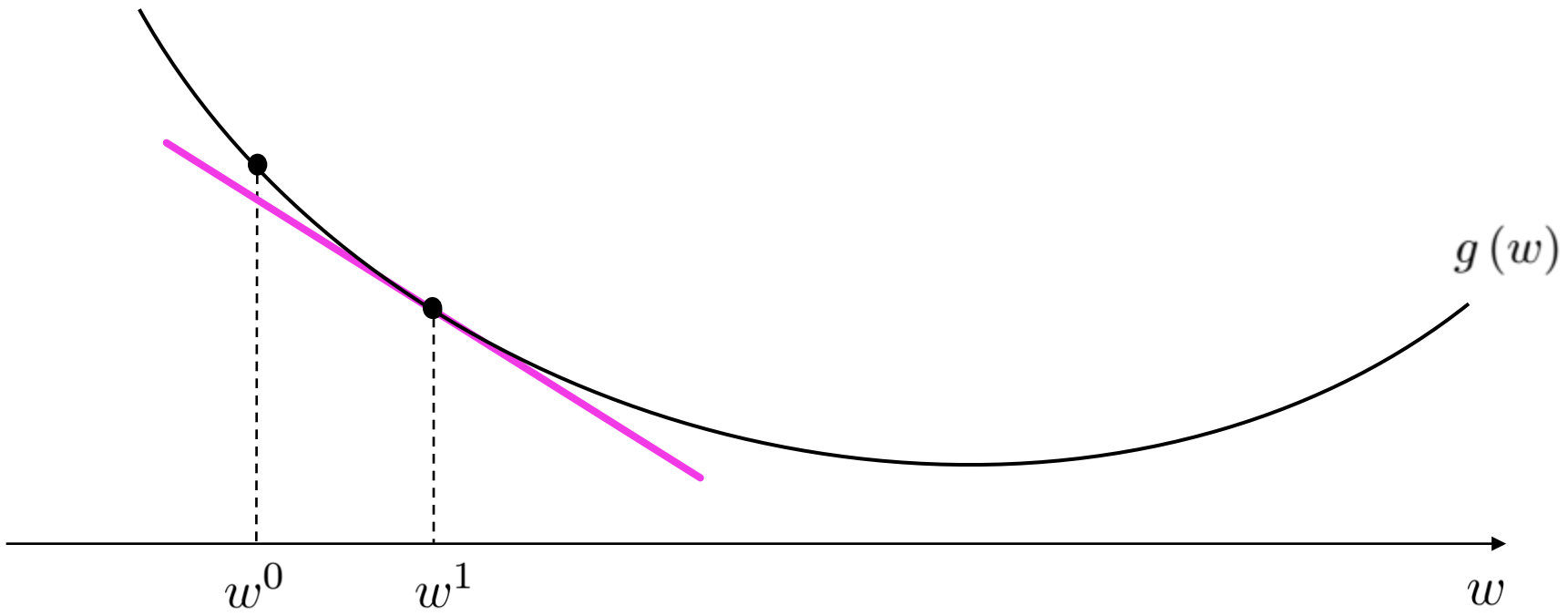
gradient descent

- ✓ 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 $\alpha > 0$



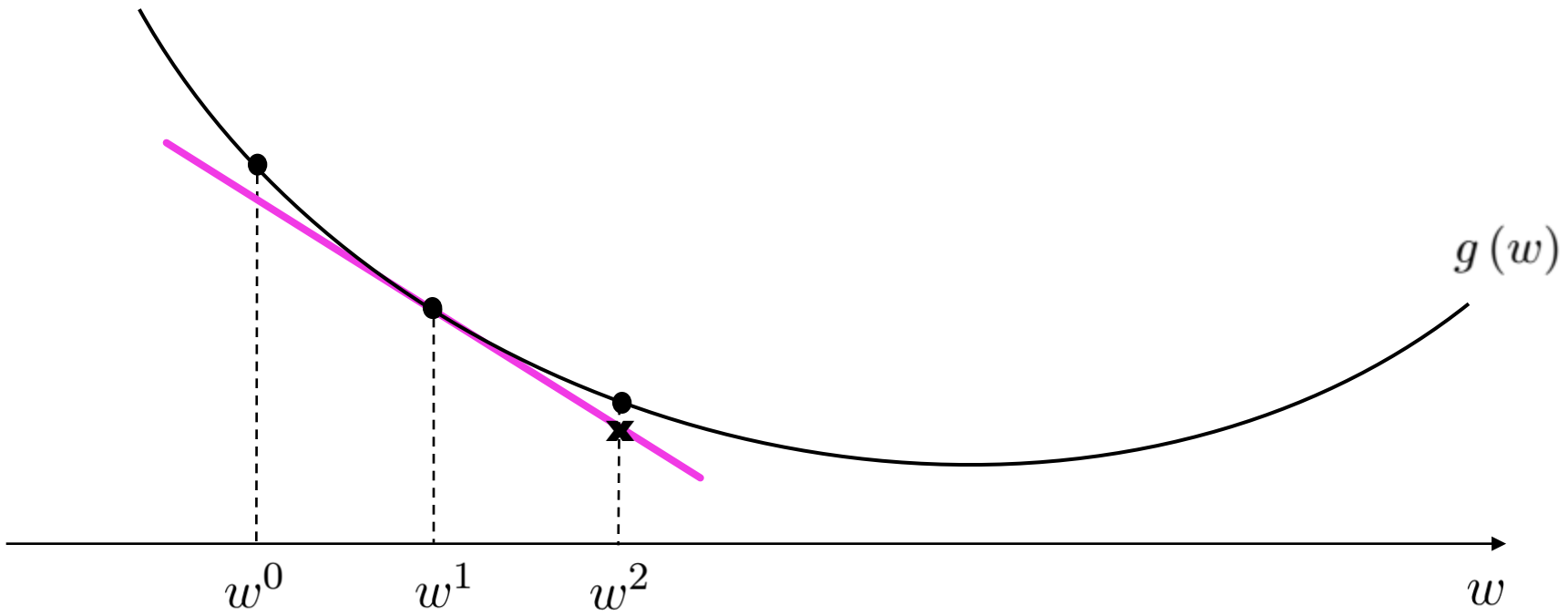
gradient descent

- ✓ 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 $\alpha > 0$



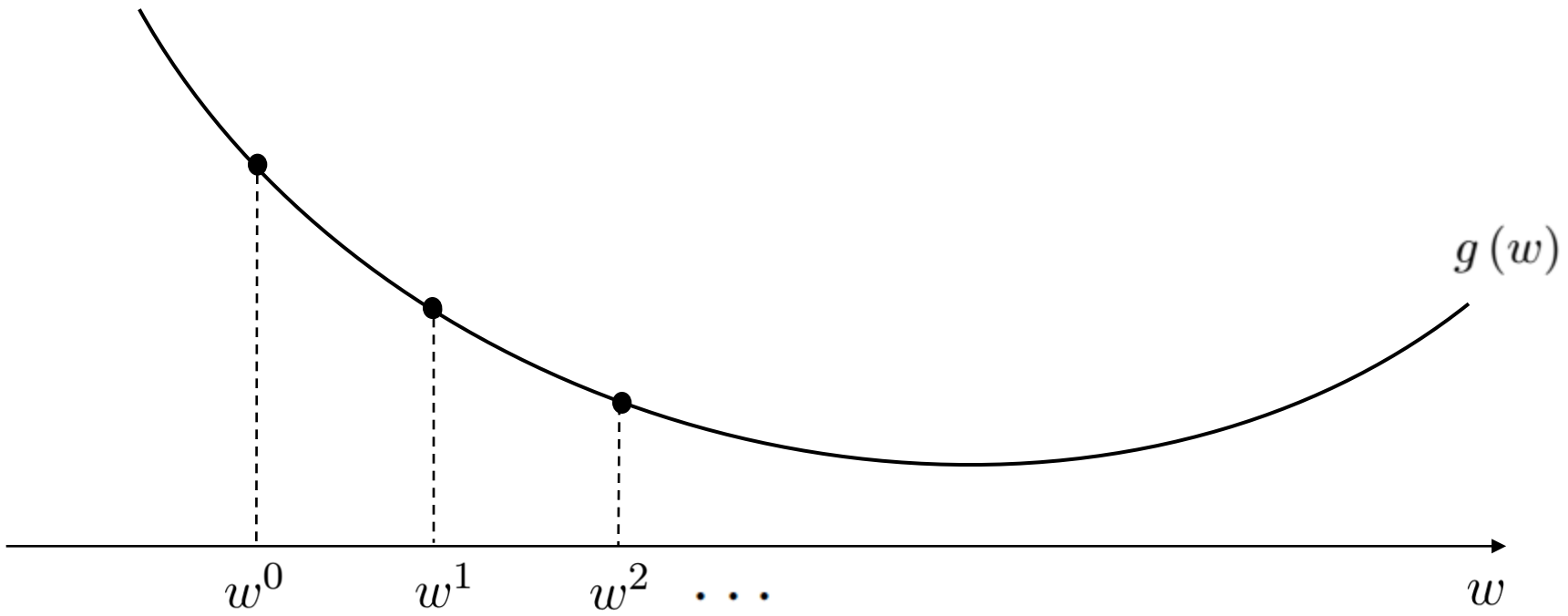
gradient descent

- ✓ 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 $\alpha > 0$



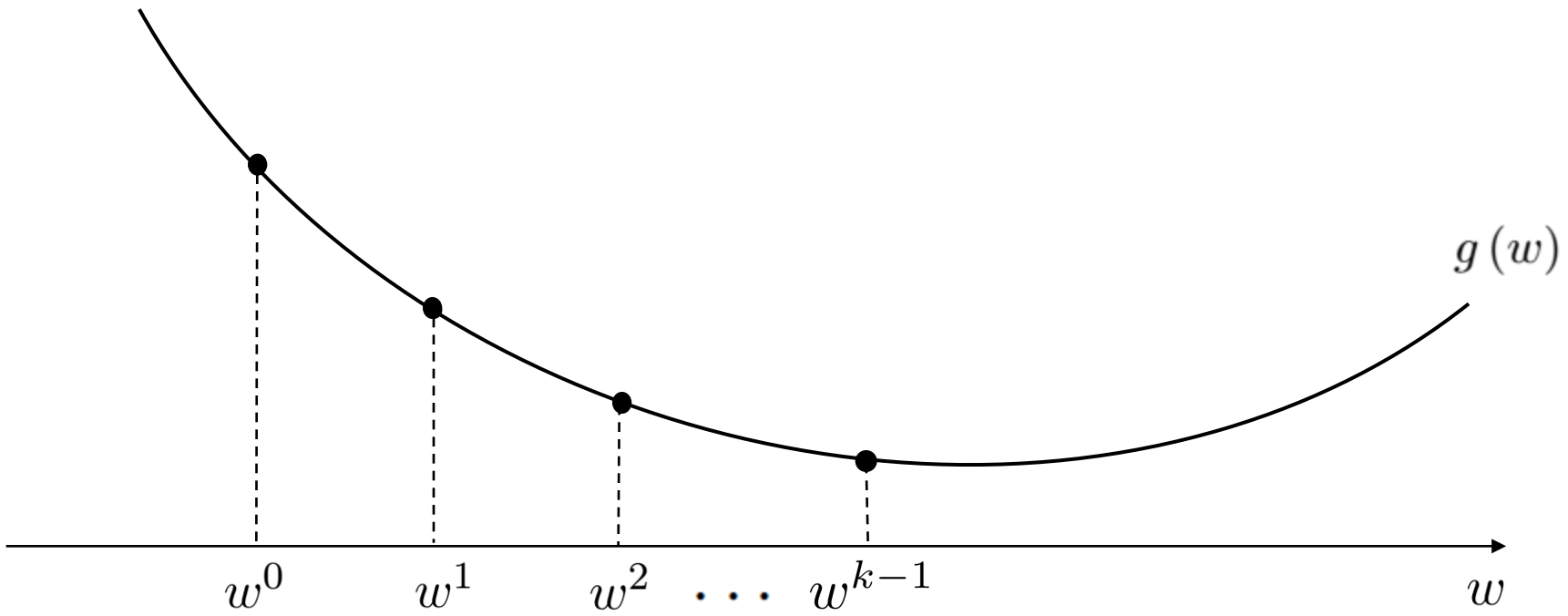
gradient descent

- ✓ 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 $\alpha > 0$



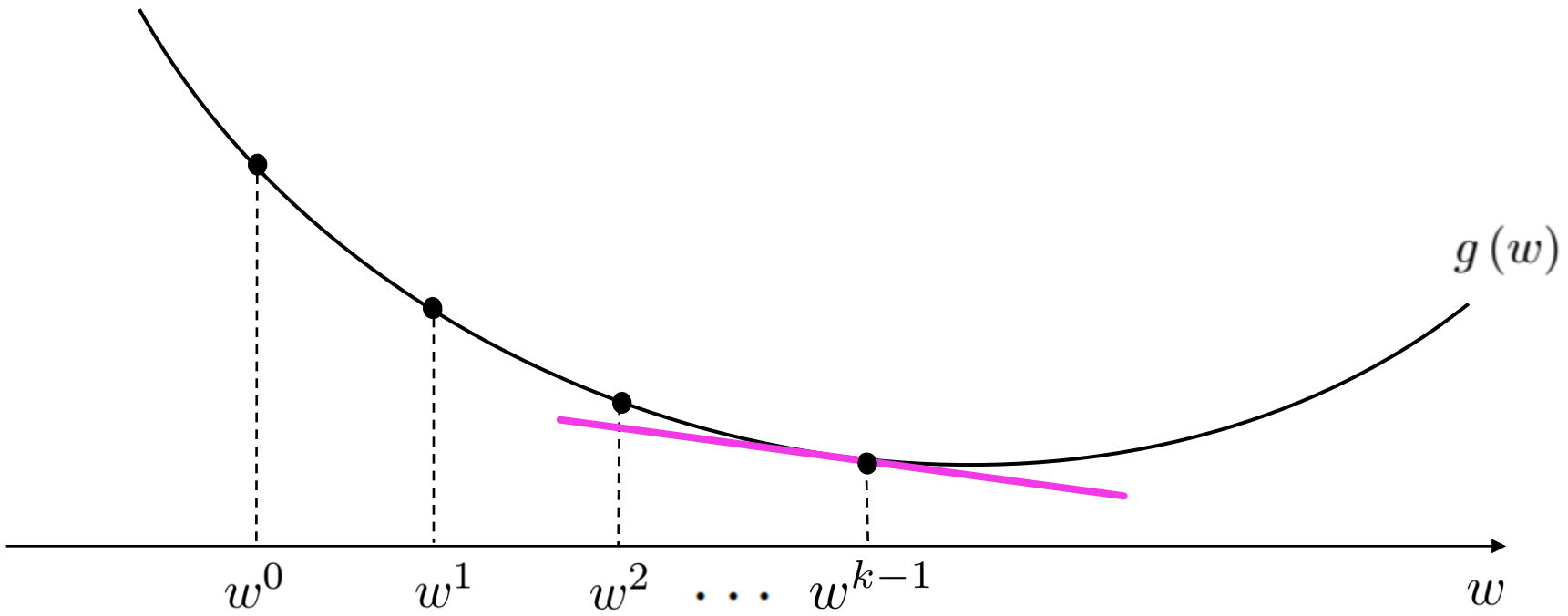
gradient descent

- ✓ 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 $\alpha > 0$



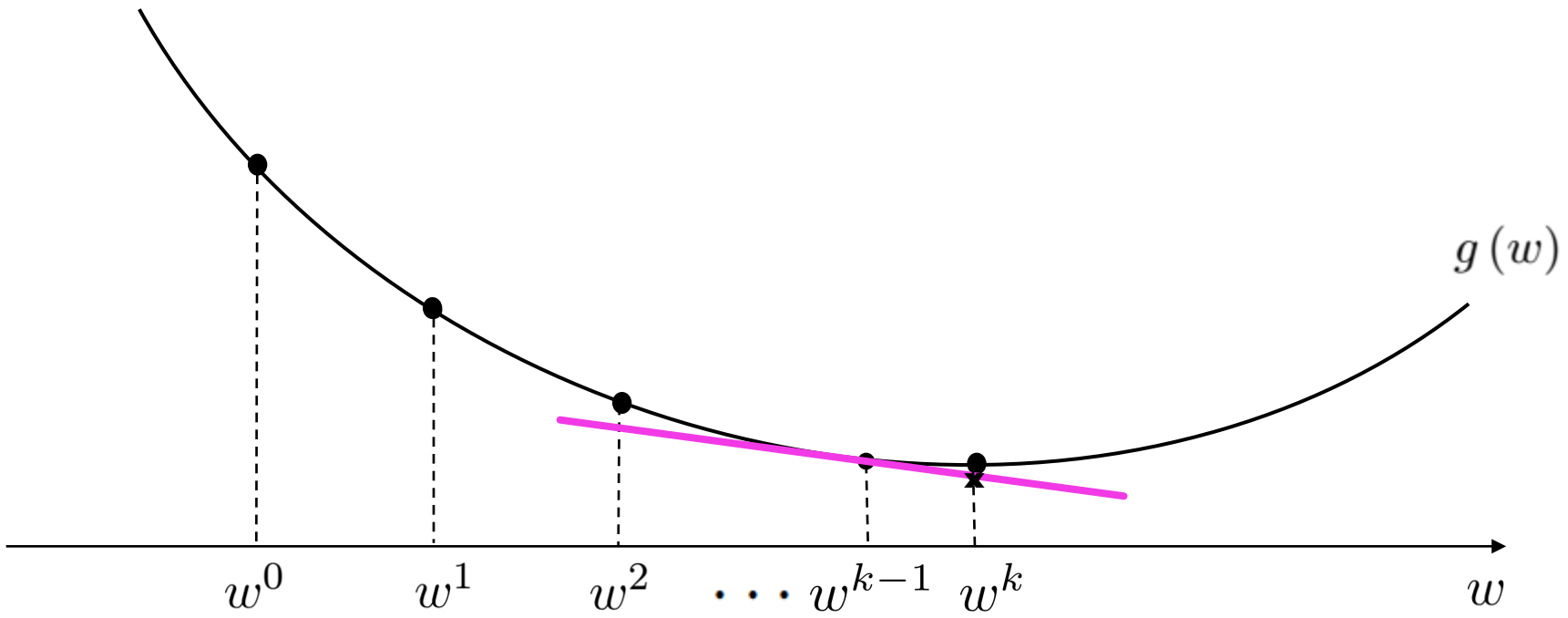
gradient descent

- ✓ 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 $w^k = w^{k-1} - \alpha g'\left(w^{k-1}\right)$ with $\alpha > 0$



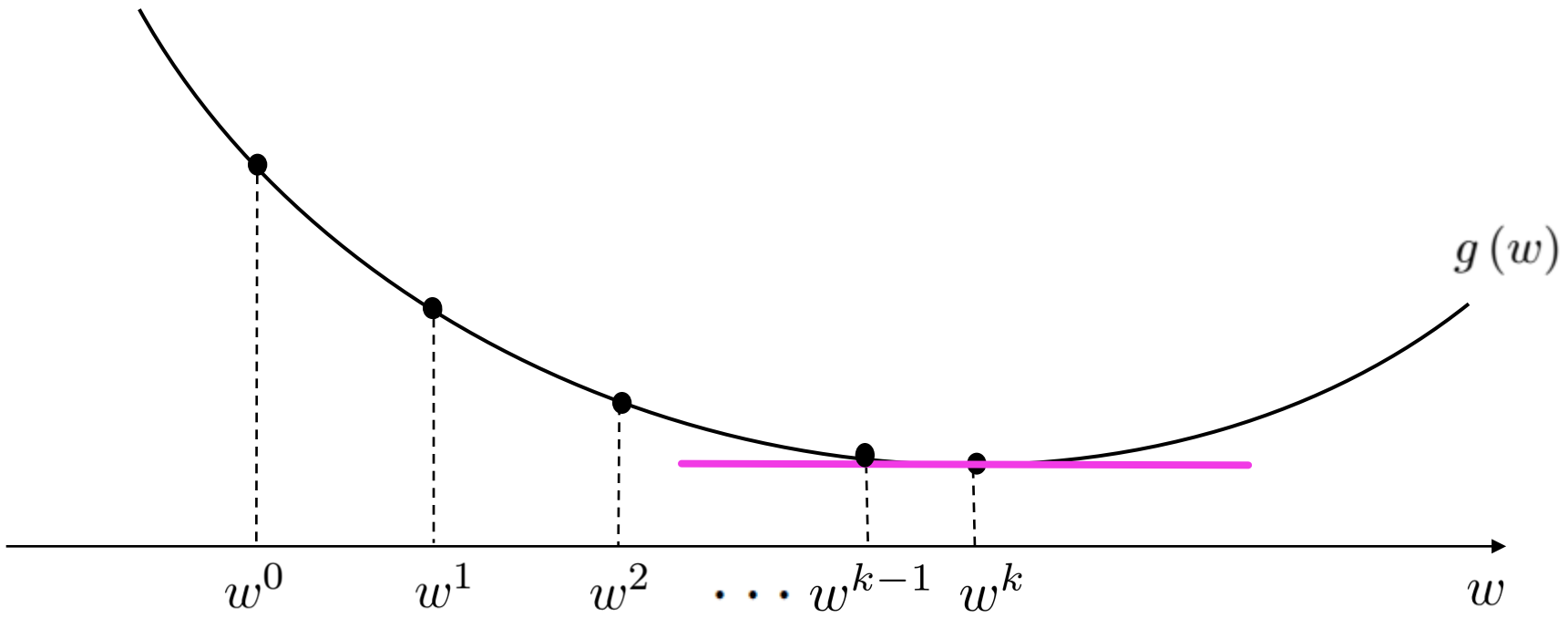
gradient descent

- ✓ 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 $\alpha > 0$



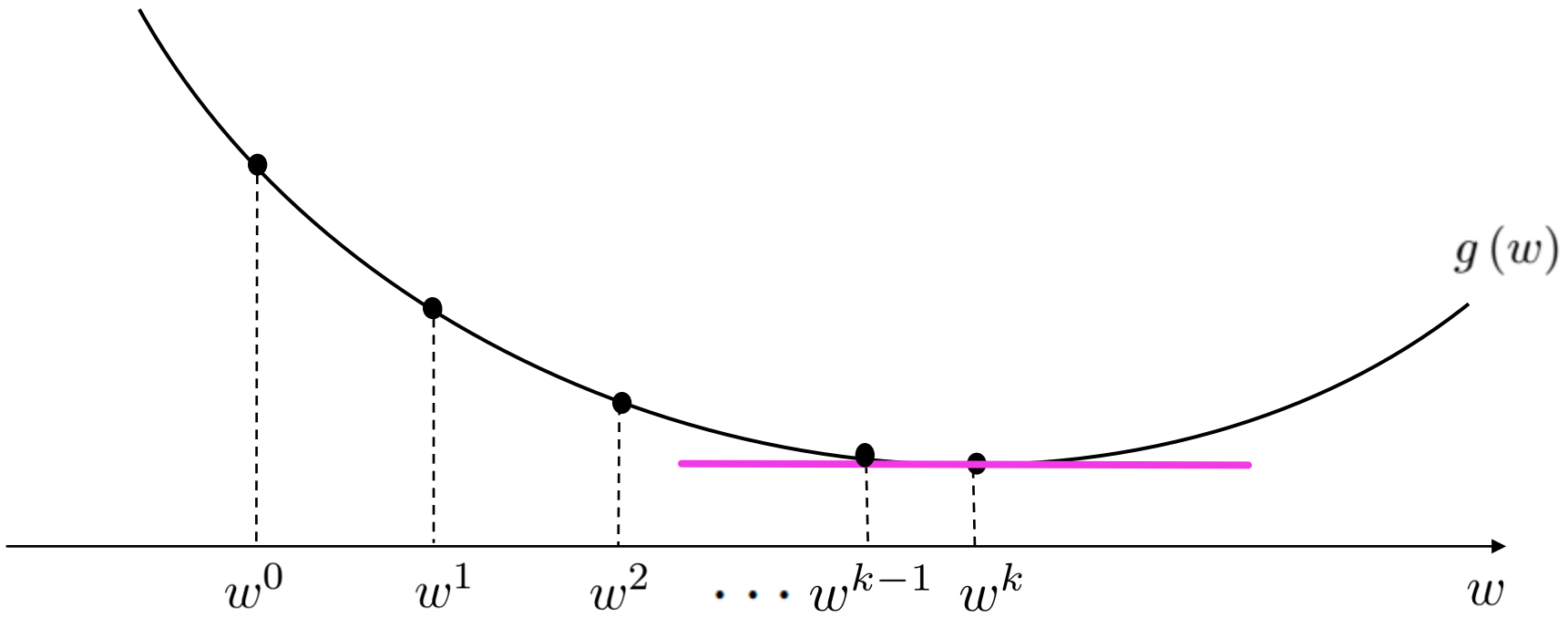
gradient descent

- ✓ 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 $\alpha > 0$



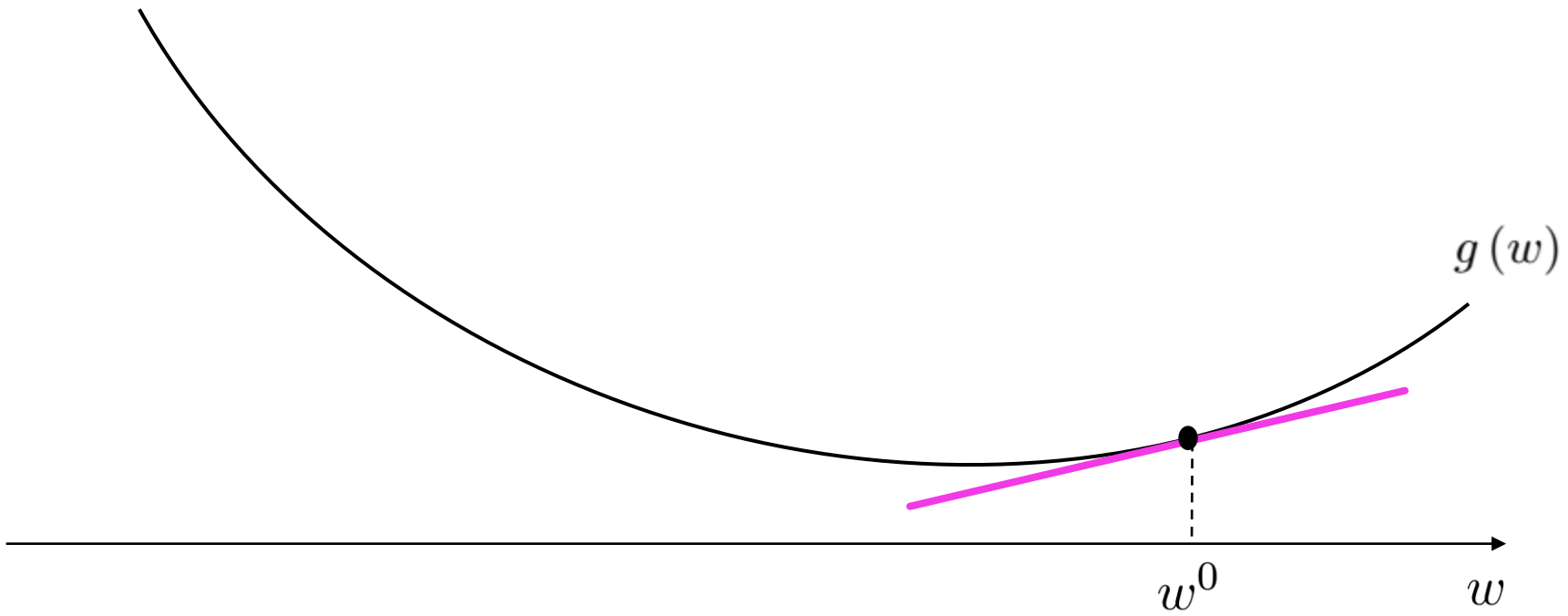
gradient descent

- ✓ 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

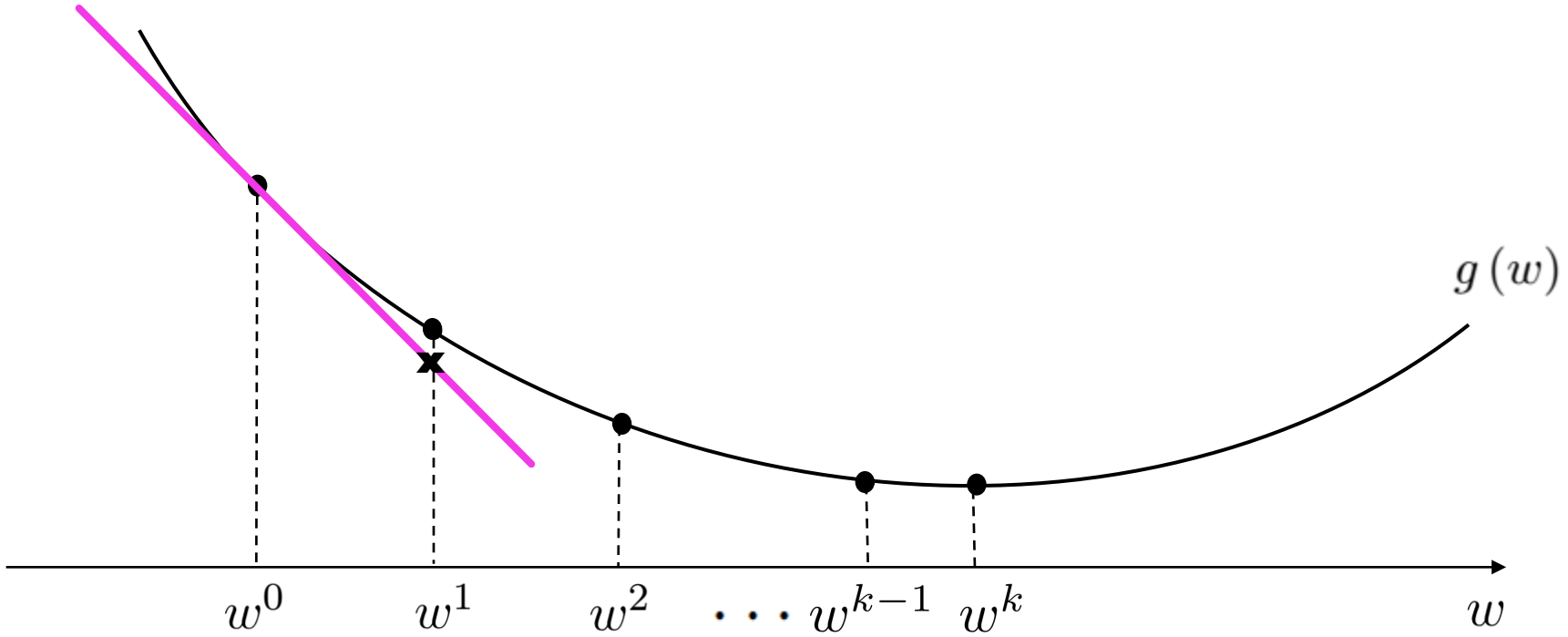
- ✓ same process starting at a different initial w^0 produces the same update rule
- ✓ $w^k = w^{k-1} - \alpha g'\left(w^{k-1}\right)$ with $\alpha > 0$



gradient descent

Full gradient descent algorithm:

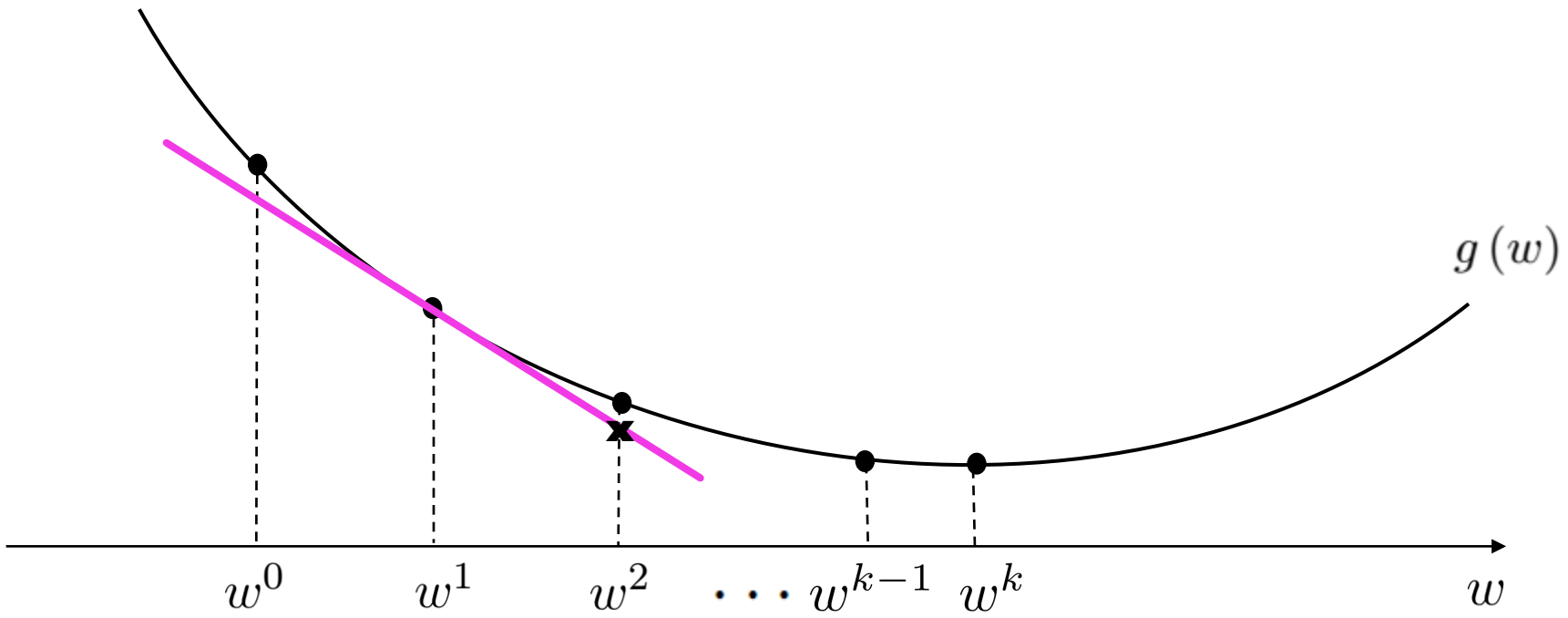
1. start at some w^0, choose value for $\alpha > 0$
2. move down with $w^j = w^{j-1} - \alpha g'\left(w^{j-1}\right)$ with $\alpha > 0$
3. repeat 2. until convergence



gradient descent

Full gradient descent algorithm:

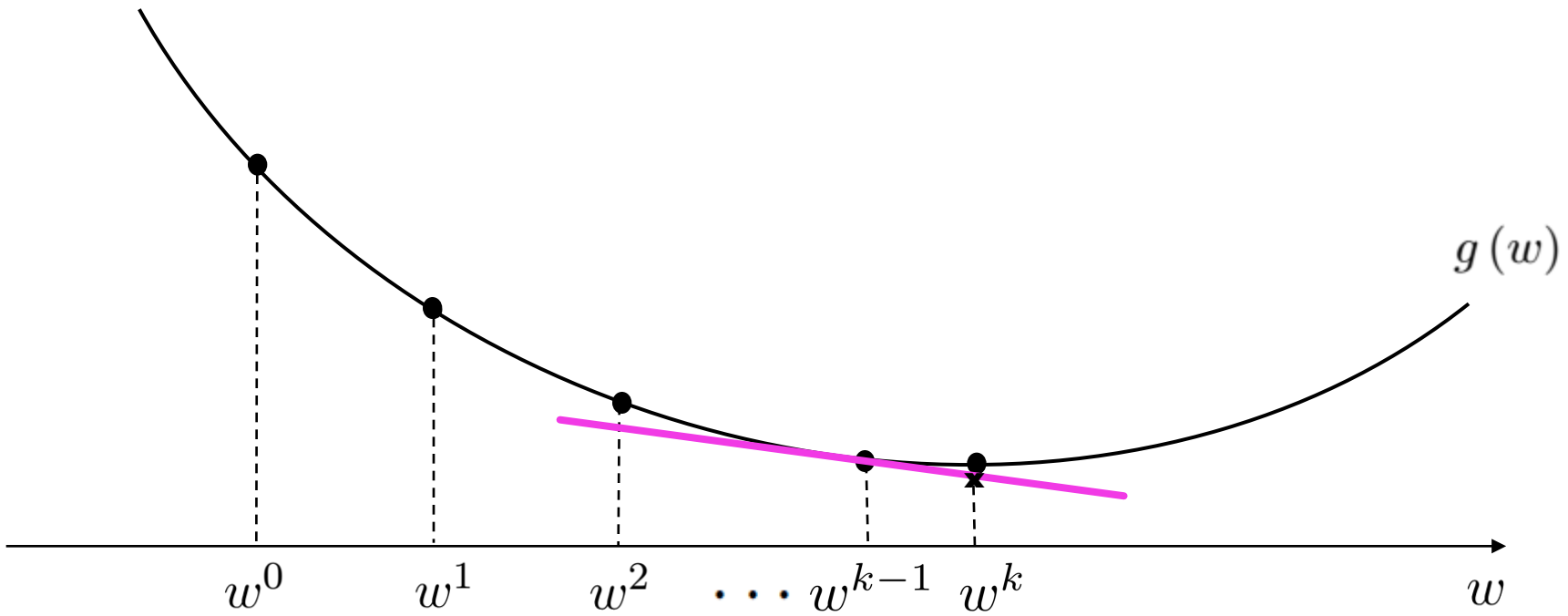
1. start at some w^0, choose value for $\alpha > 0$
2. move down with $w^j = w^{j-1} - \alpha g'\left(w^{j-1}\right)$ with $\alpha > 0$
3. repeat 2. until convergence



gradient descent

Full gradient descent algorithm:

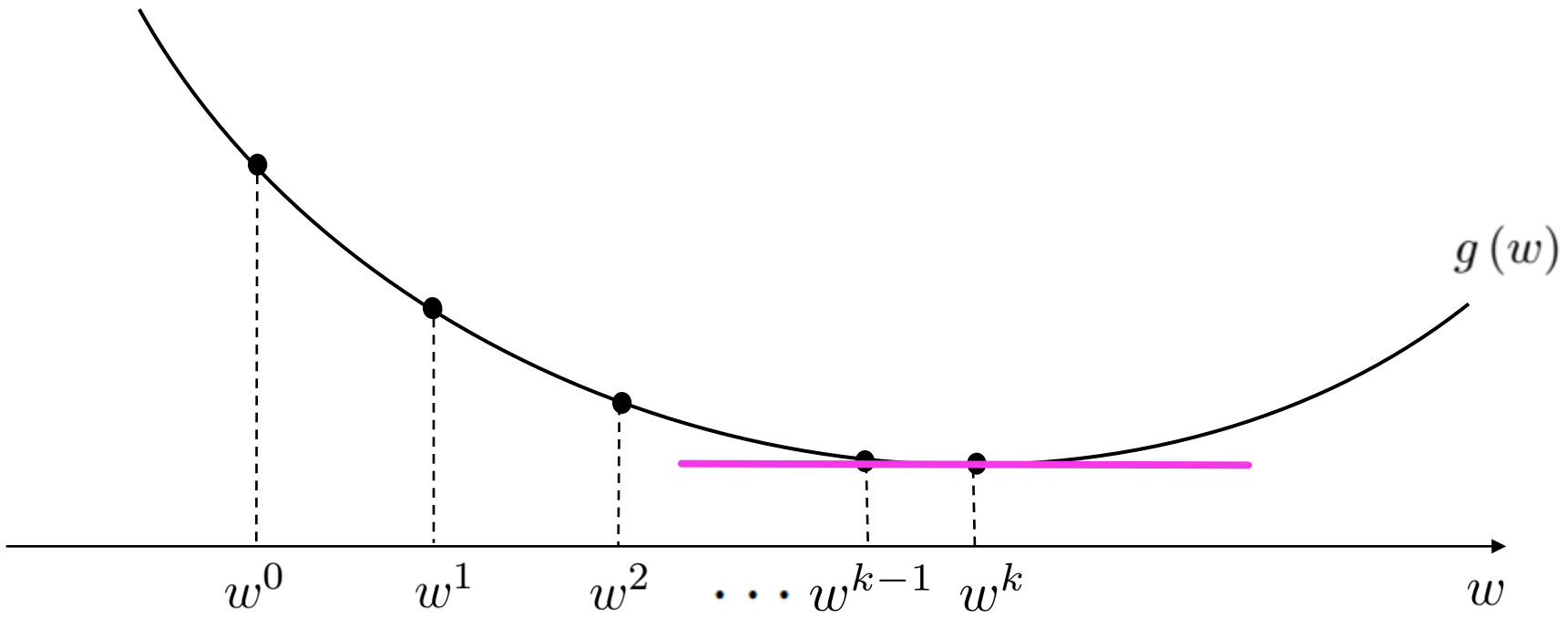
1. start at some w^0, choose value for $\alpha > 0$
2. move down with $w^j = w^{j-1} - \alpha g'\left(w^{j-1}\right)$ with $\alpha > 0$
3. repeat 2. until convergence



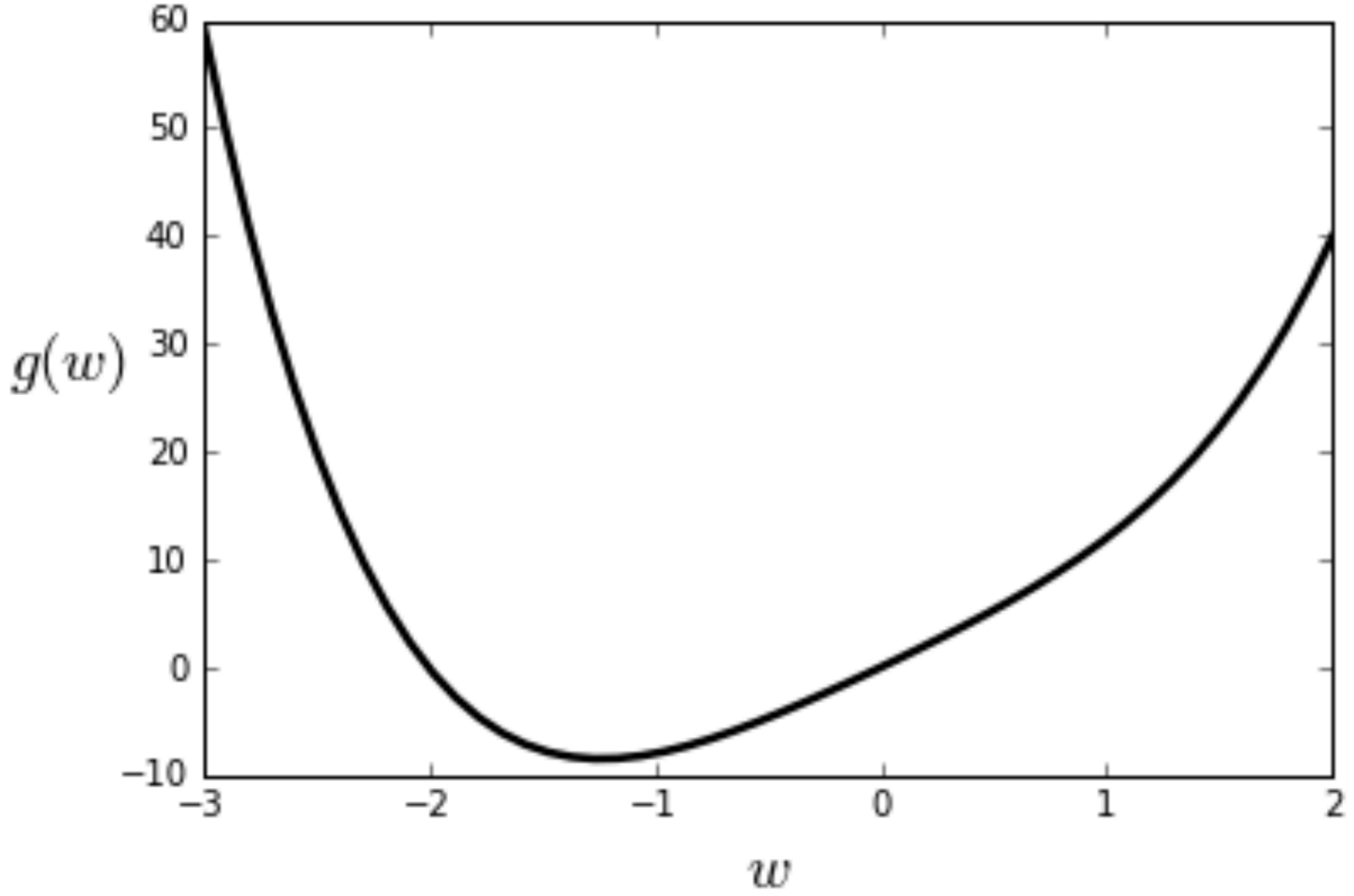
gradient descent

Full gradient descent algorithm:

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



- OK but how about $g(w) = w^4 + w^2 + 10w$
- Looks simple enough...



WolframAlpha computational knowledge engine

minimum point of w^4 + w^2 + 10w

Examples Random

Input interpretation:

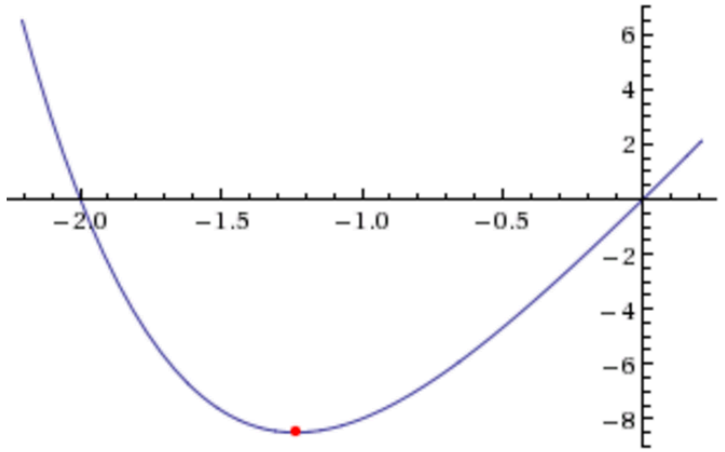
minimize	$w^4 + w^2 + 10\,w$

Global minimum:

Fewer digits More digits

$$\min\{w^4 + w^2 + 10\,w\} \approx -8.498464223154677377534566430142620373354396927609957 \text{ at } w \approx -1.234772825053296980414119894096777410653656407067694$$

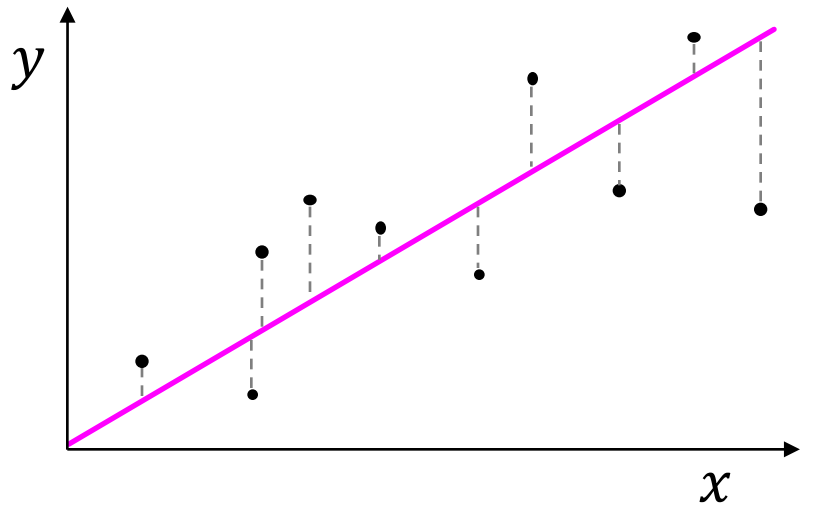
Plot:



(x from −2.2 to 0.2)

Linear regression

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



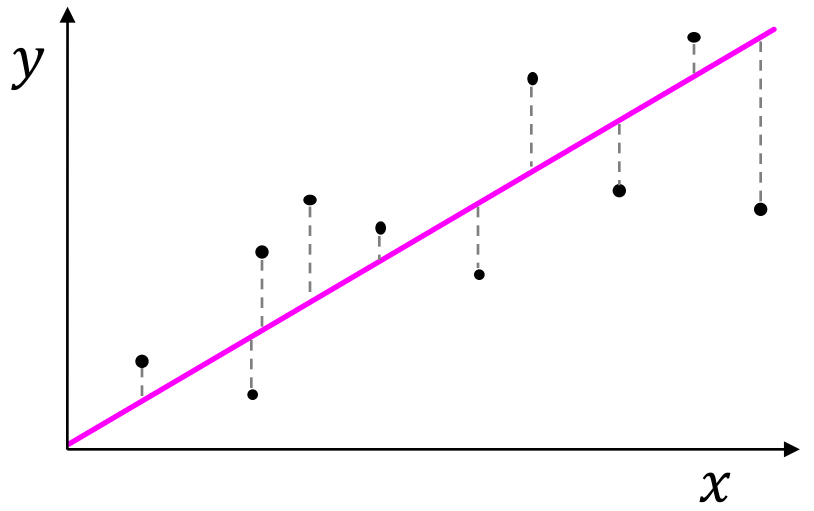
$$b + xw = y$$

Linear regression

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

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

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



$$b + xw = y$$

Demo # 3 convex grad descent

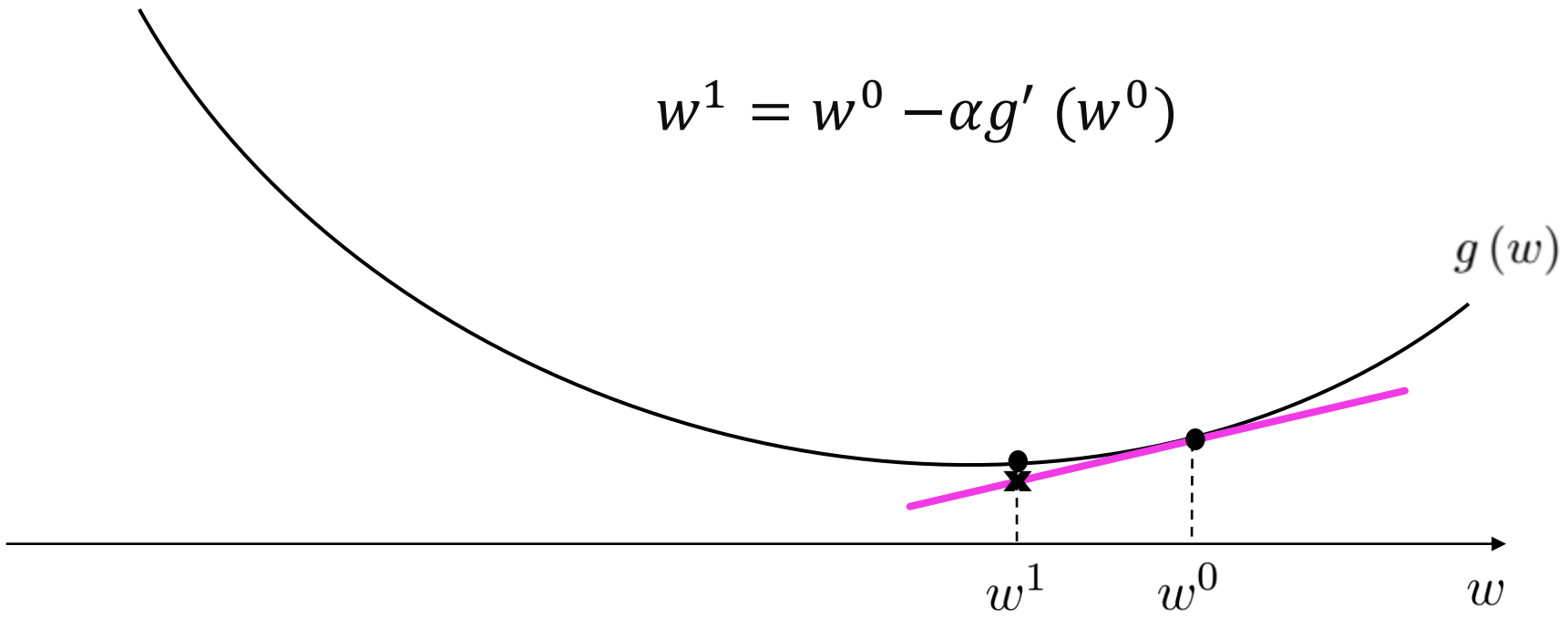
- to the notebook!

Demo # 4 linear regression

- to the notebook!

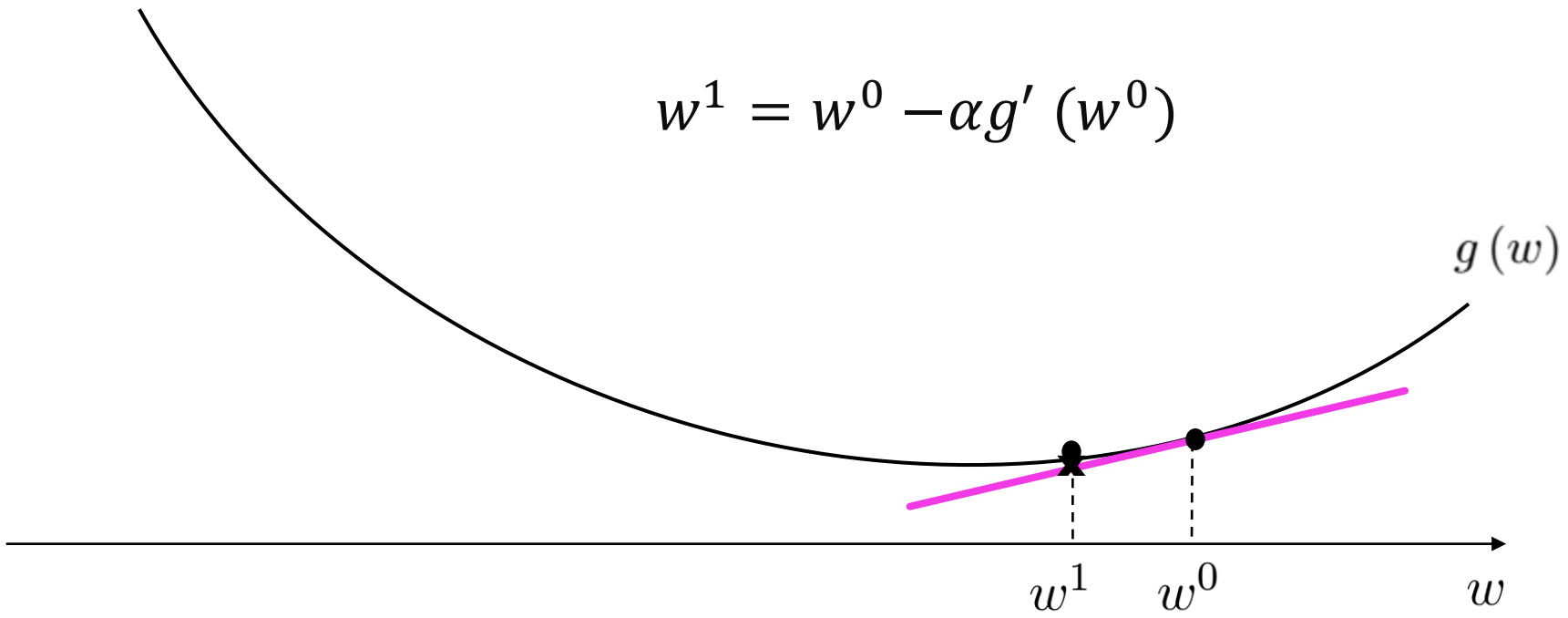
quiz: question

$$w^1 = w^0 - \alpha g'(w^0)$$



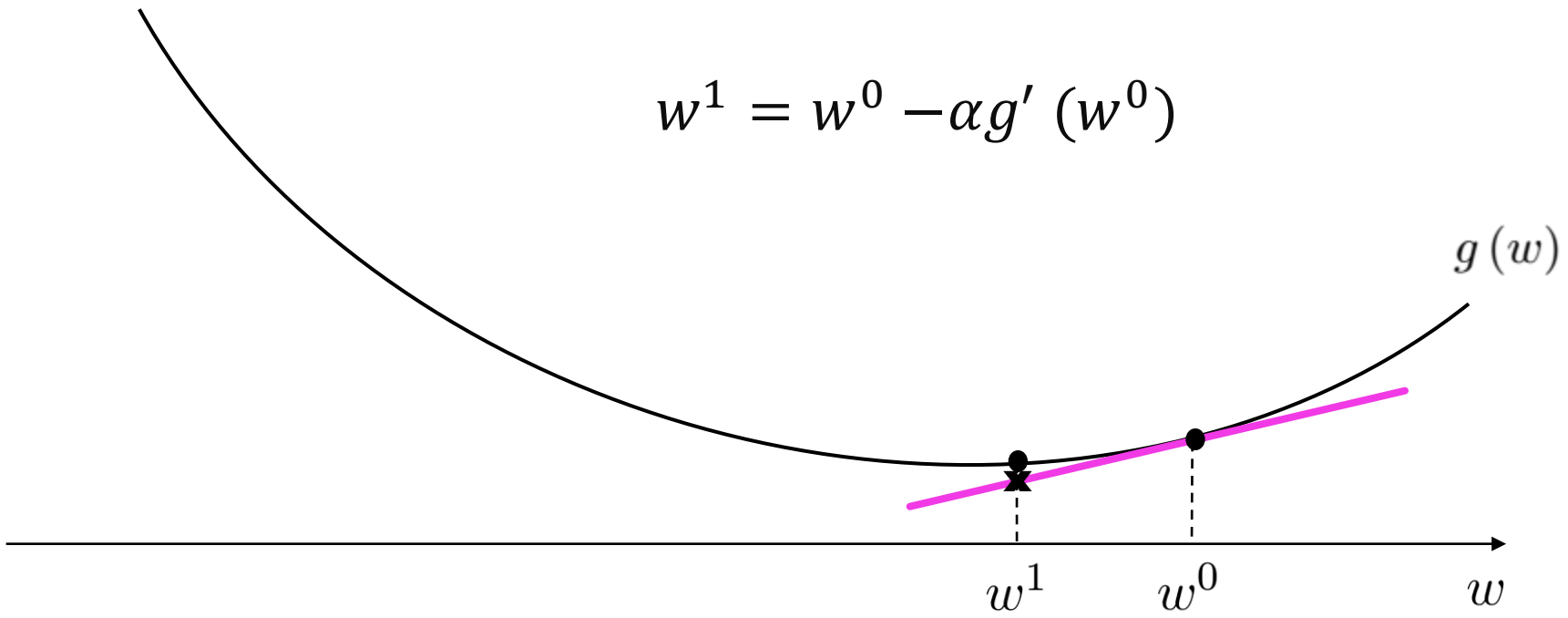
quiz: question

$$w^1 = w^0 - \alpha g'(w^0)$$



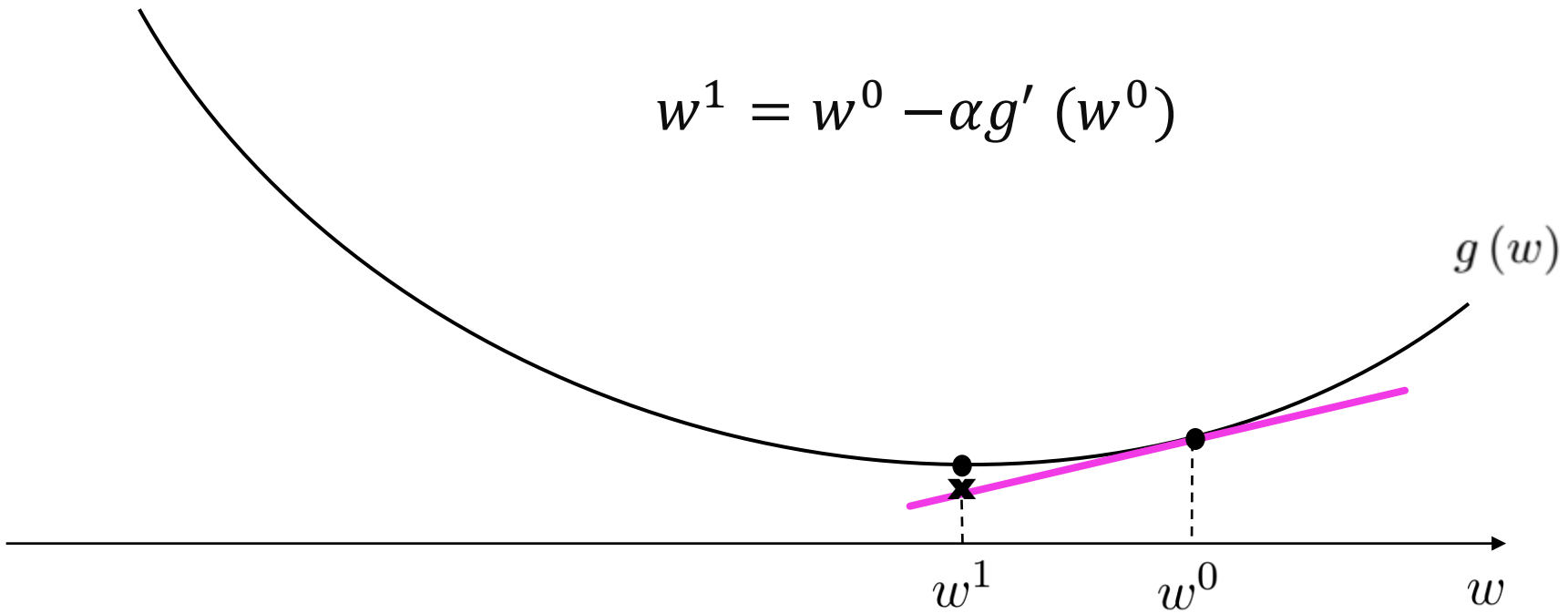
quiz: question

$$w^1 = w^0 - \alpha g'(w^0)$$



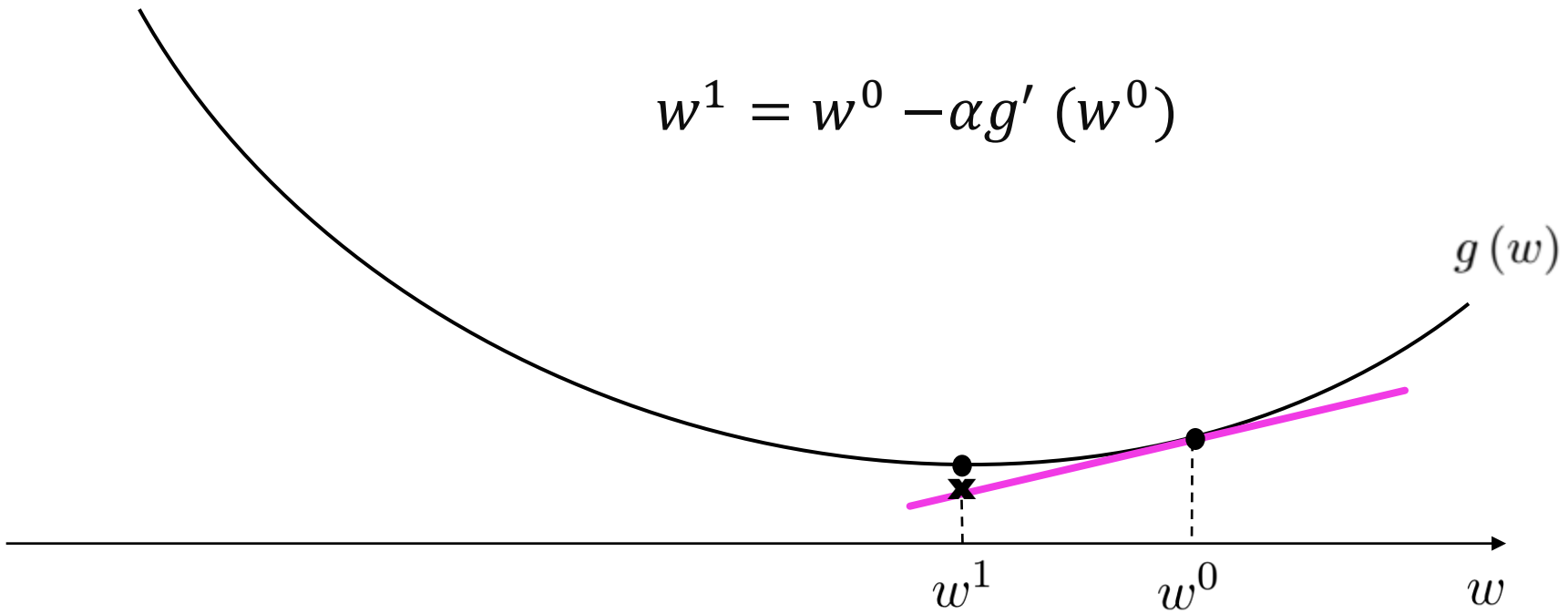
quiz: question

$$w^1 = w^0 - \alpha g'(w^0)$$



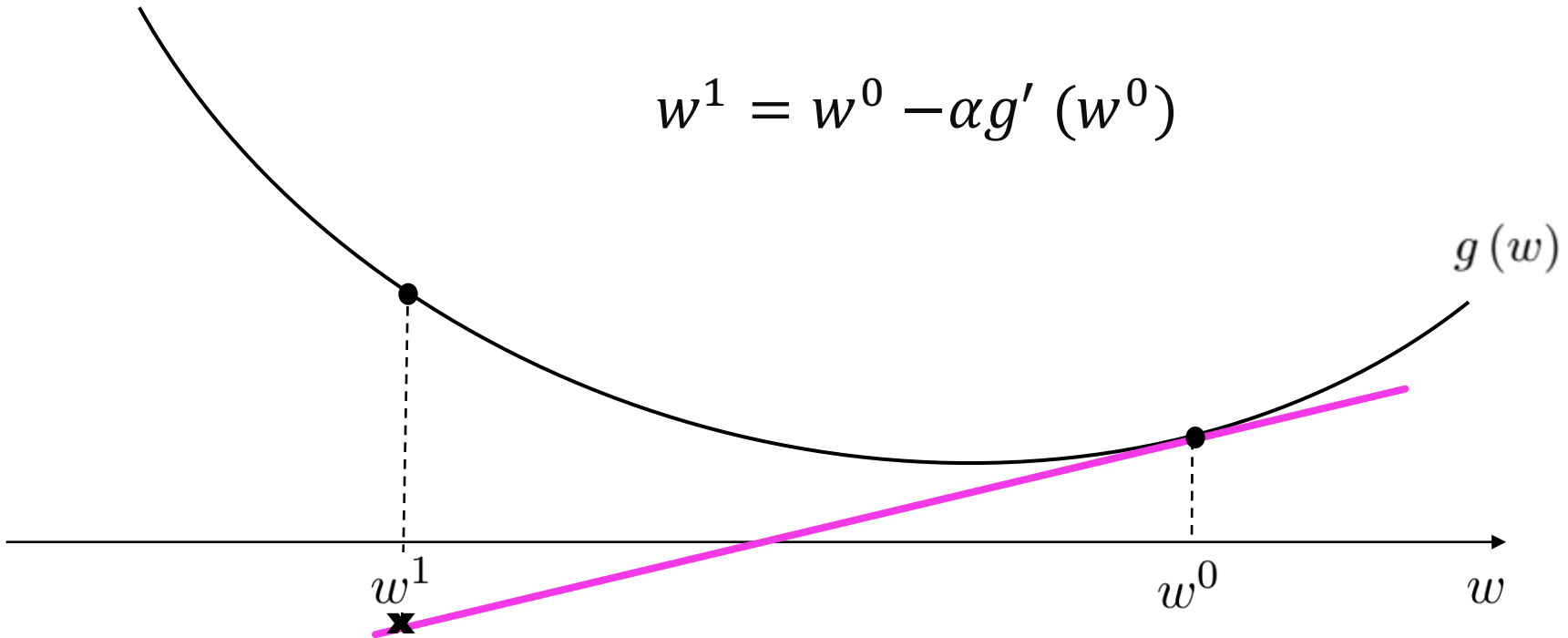
quiz: solution

$$w^1 = w^0 - \alpha g'(w^0)$$



quiz: solution

$$w^1 = w^0 - \alpha g'(w^0)$$



Gradient descent and classification

Classification ⇒ distinguish between different classes of data

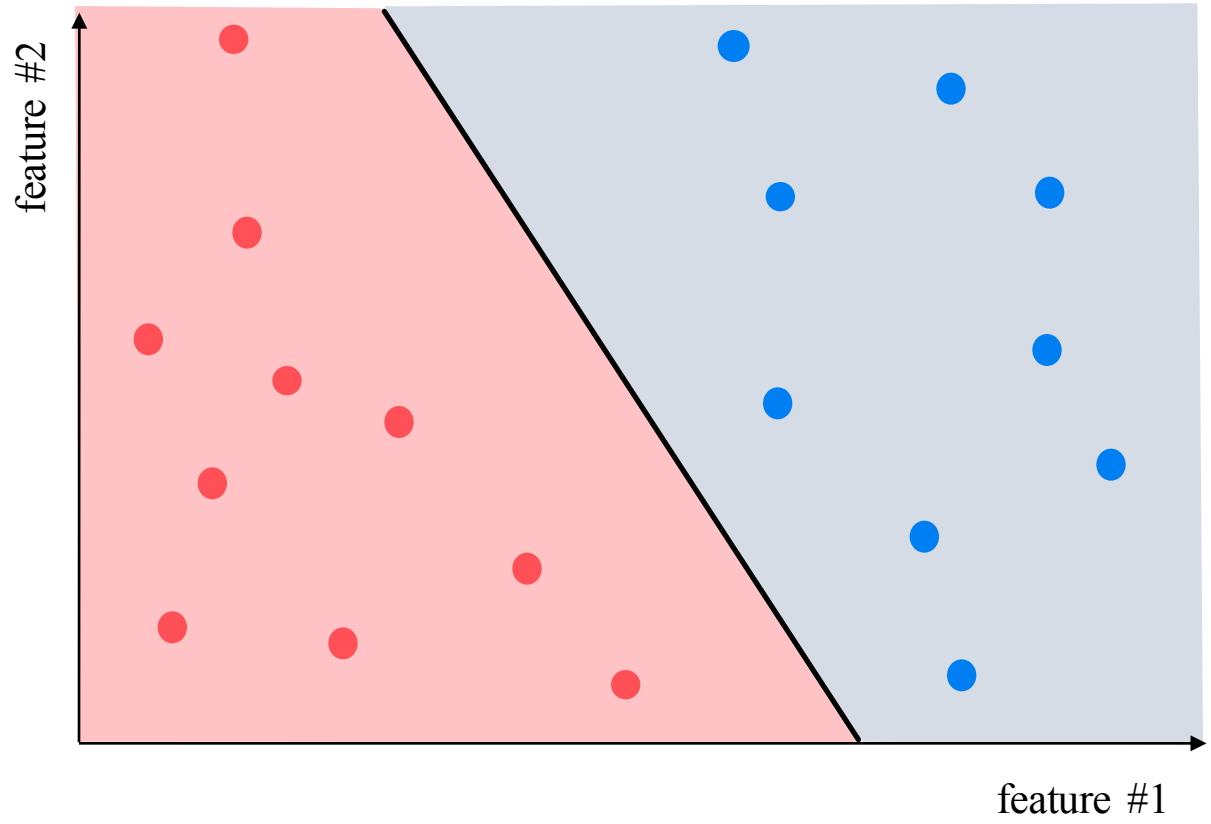


Image taken from [29]

Classification ⇒ distinguish between different classes of data

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

Classification ⇒ distinguish between different classes of data

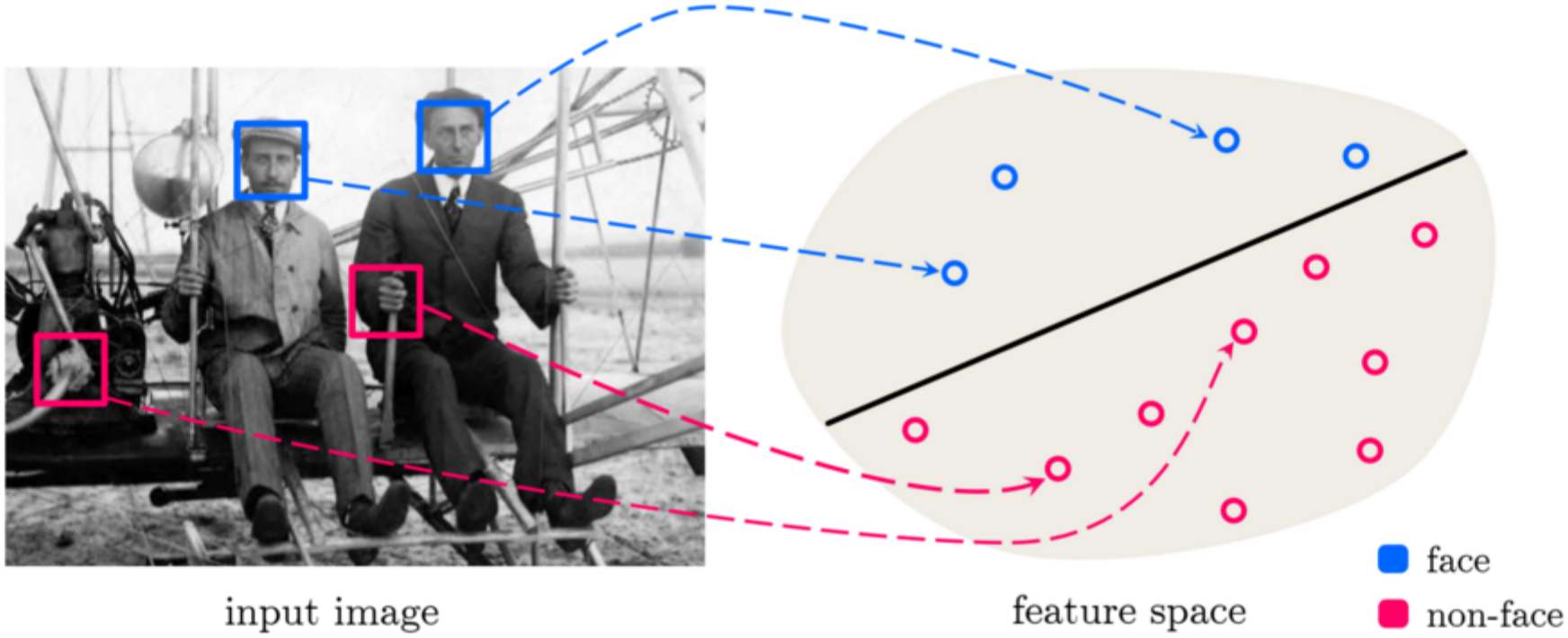


Image taken from [29]

Classification ⇒ distinguish between different classes of data

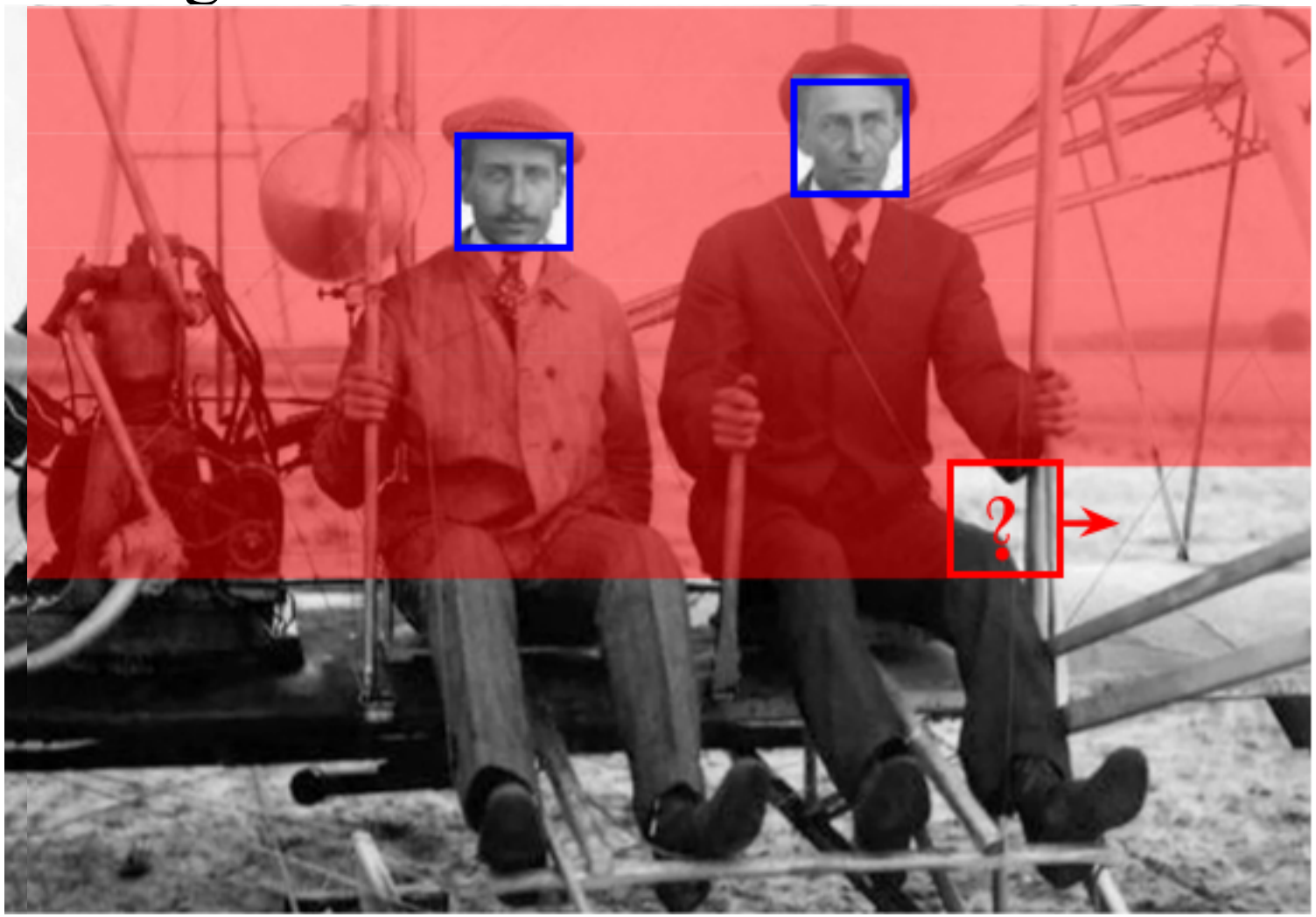


Image taken from [29]

Classification ⇒ distinguish between different classes of data

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

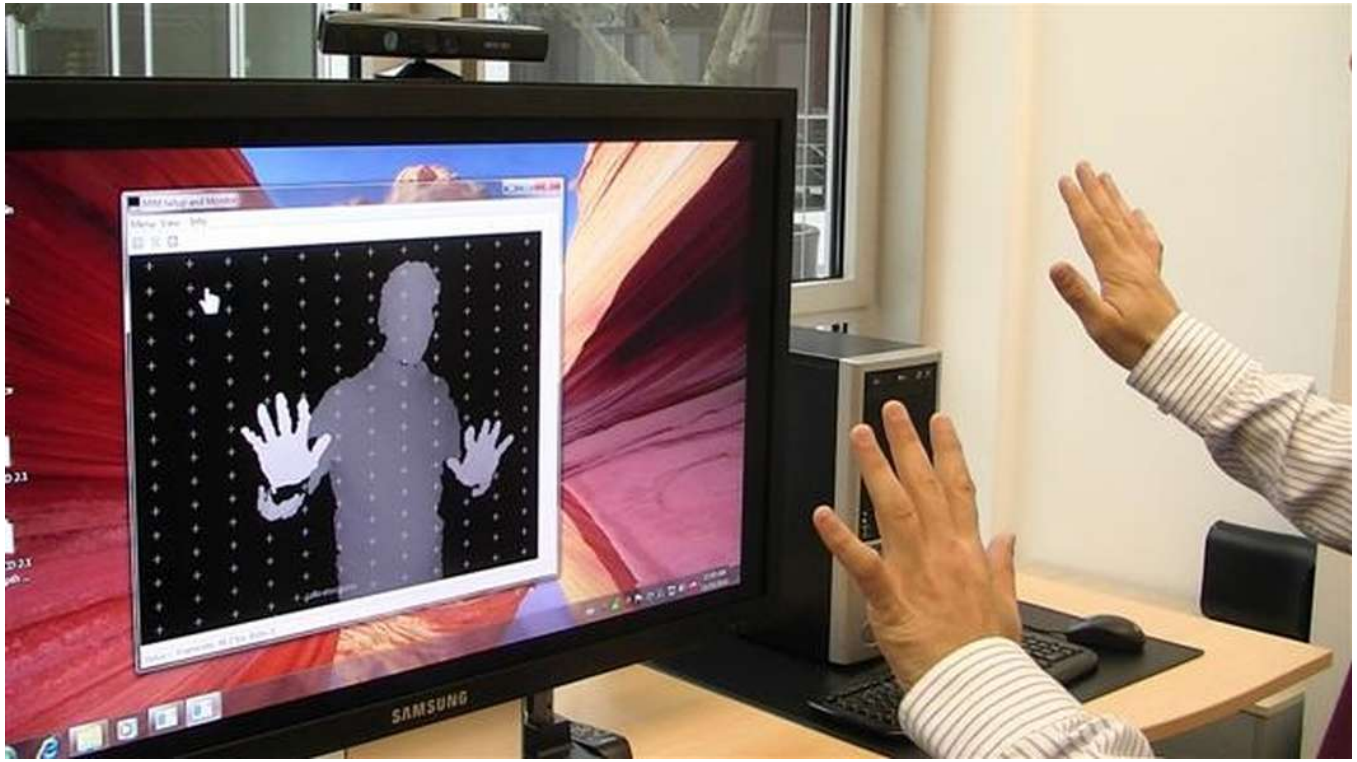


Image taken from http://www.gizmag.com/kinect-used-to-control-windows-7/16997/

Classification ⇒ distinguish between different classes of data

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

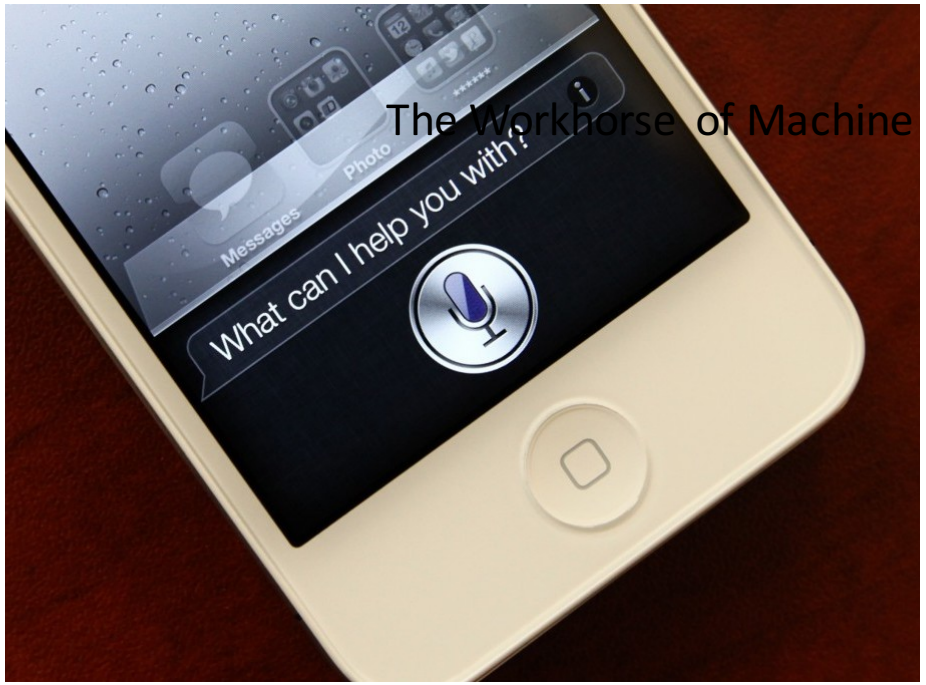
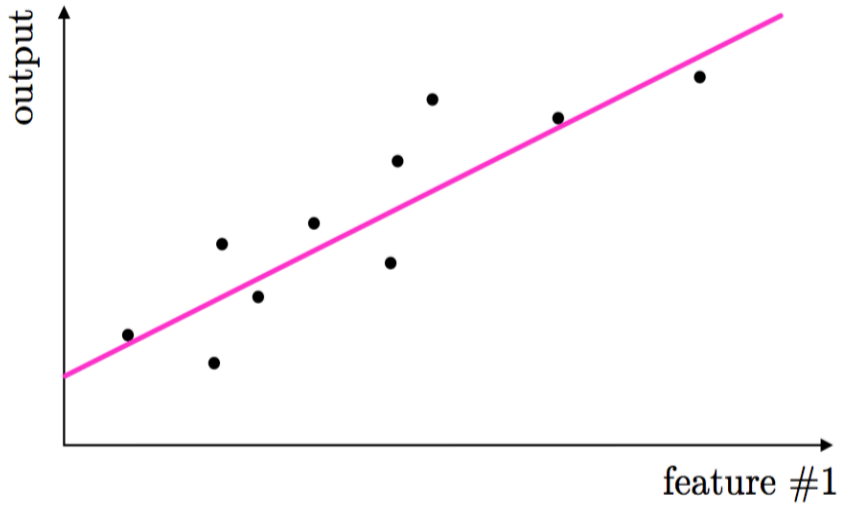


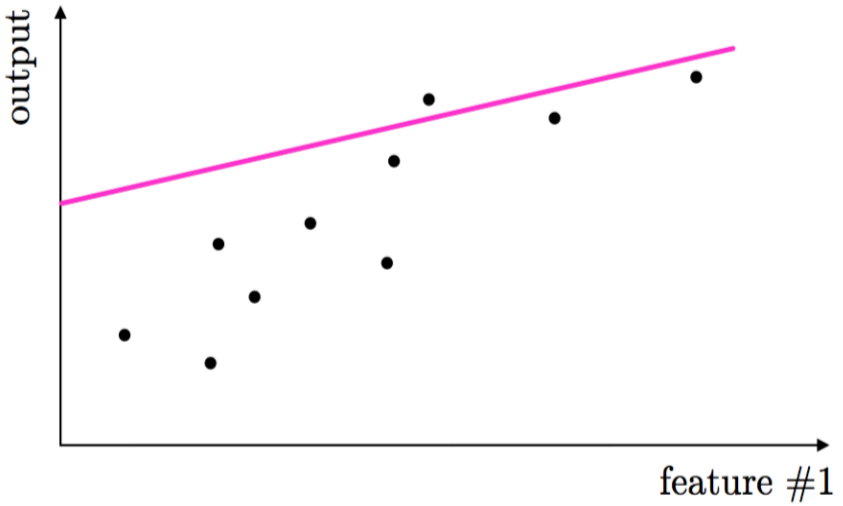
Image taken from (left) http://tiny.cc/n25z8x and (right) www.amazon.com

Parameter tuning for optimal learning

parameter tuning: regression

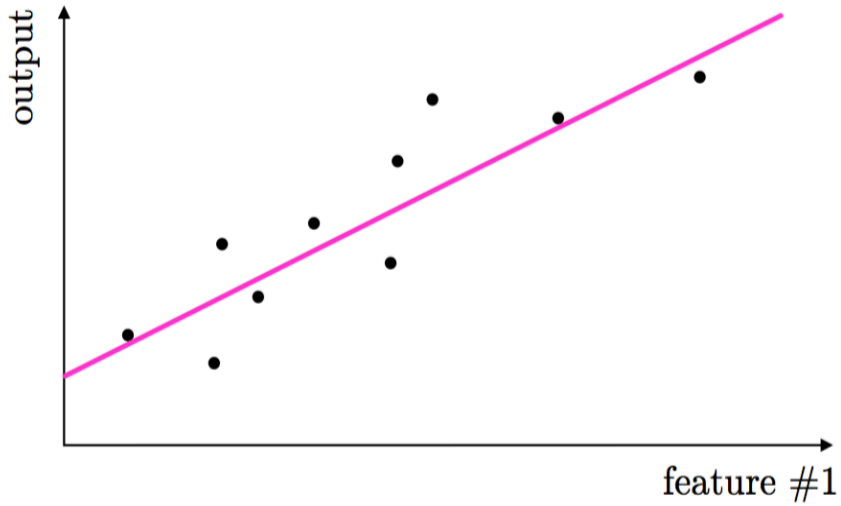
- parameters must be tuned properly to ensure optimal learning

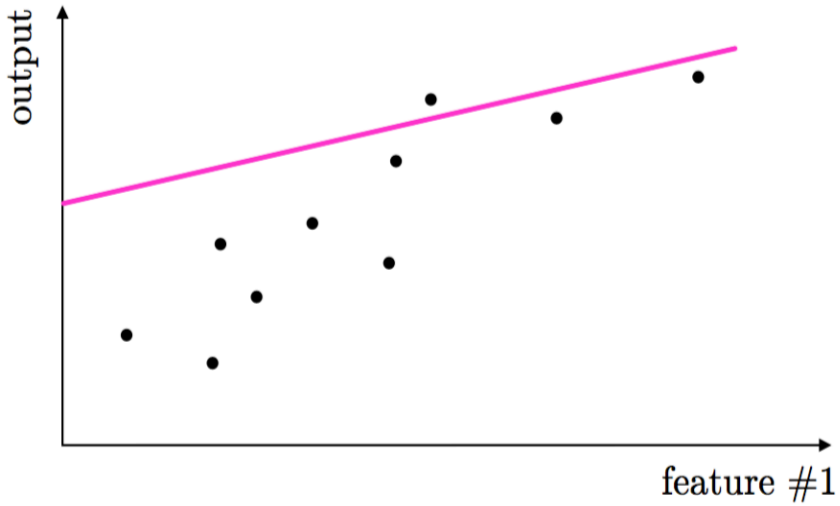




parameter tuning: regression

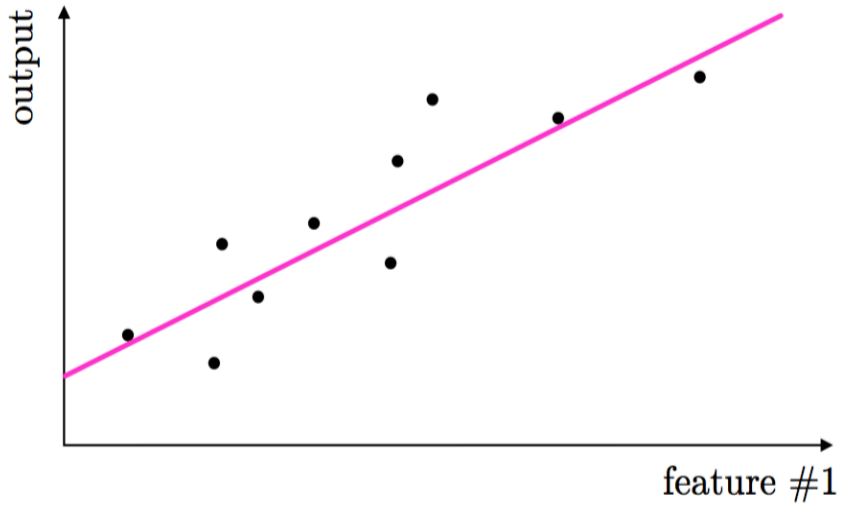
- parameters must be tuned properly to ensure optimal learning
- a 'cost function' measures how well a model ***fits*** data given a set of parameters

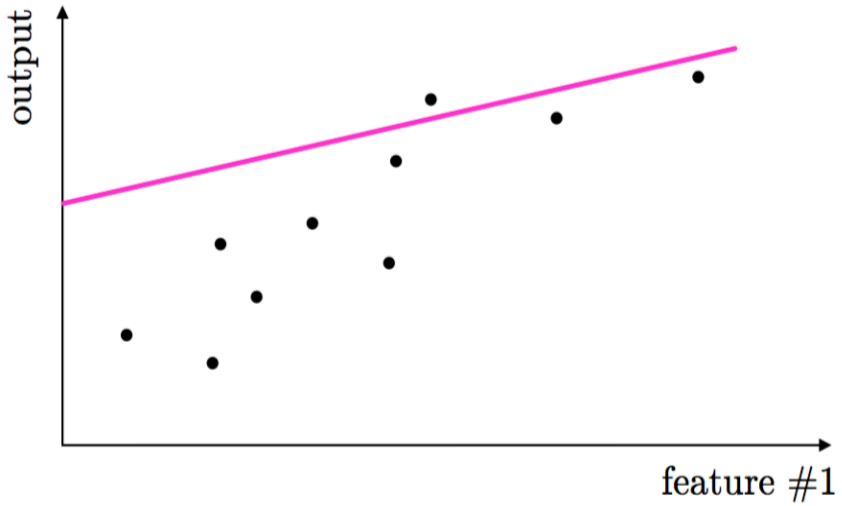




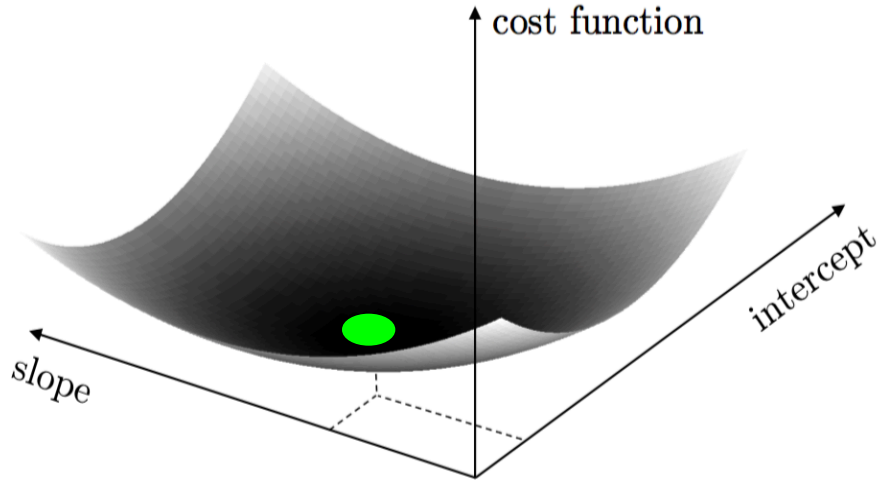
parameter tuning: regression

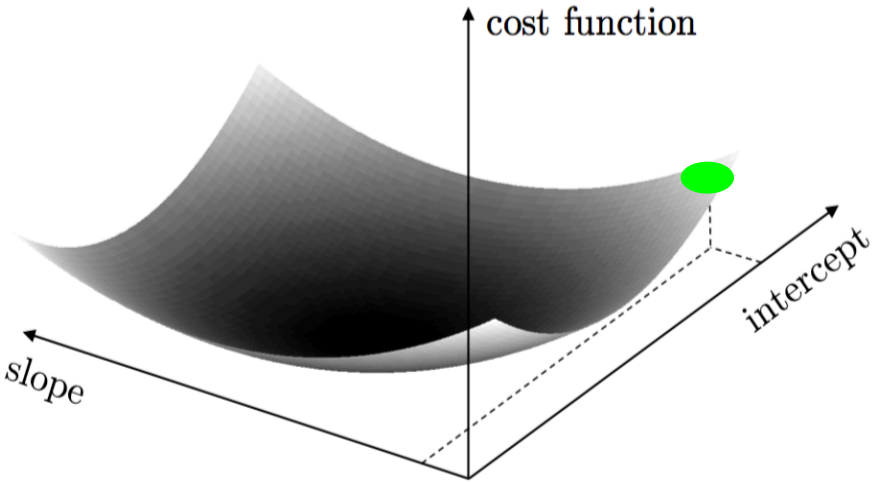
- parameters must be tuned properly to ensure optimal learning
- a 'cost function' measures how well a model ***fits*** data given a set of parameters
- better parameters provide a better fit, and ***lower*** value of cost function

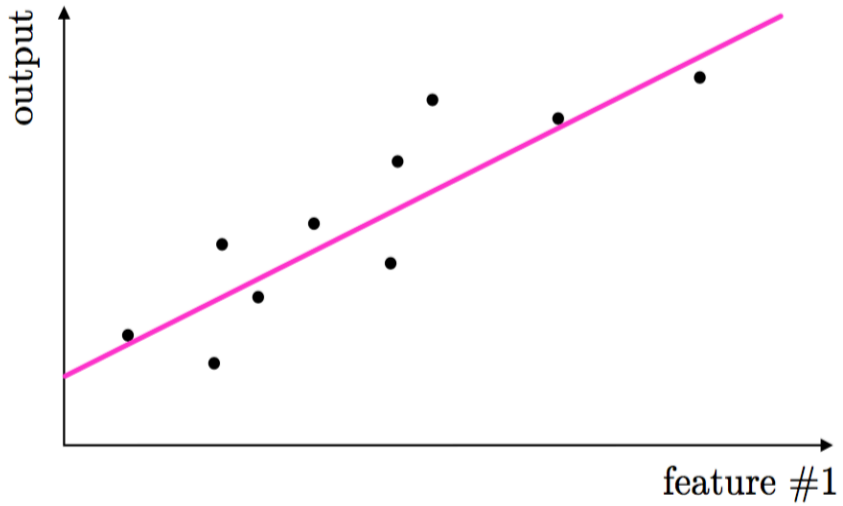


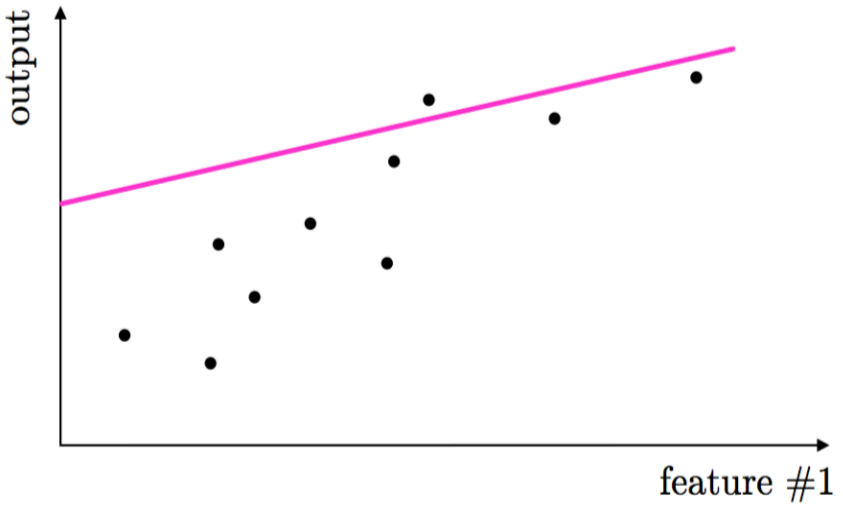


parameter tuning: regression



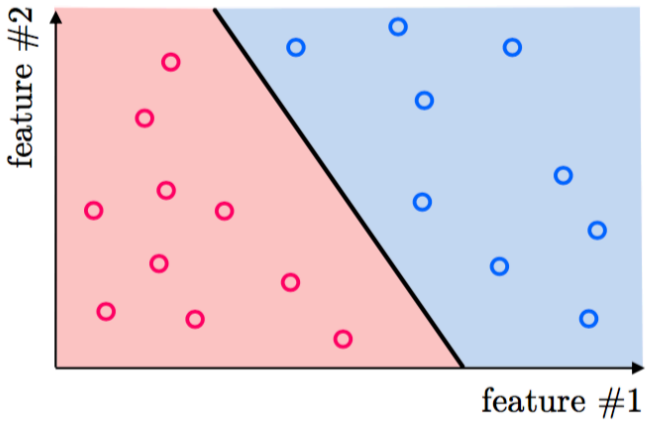


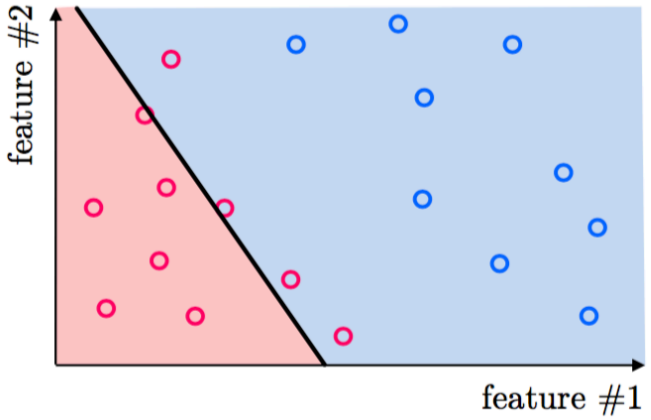




parameter tuning: classification

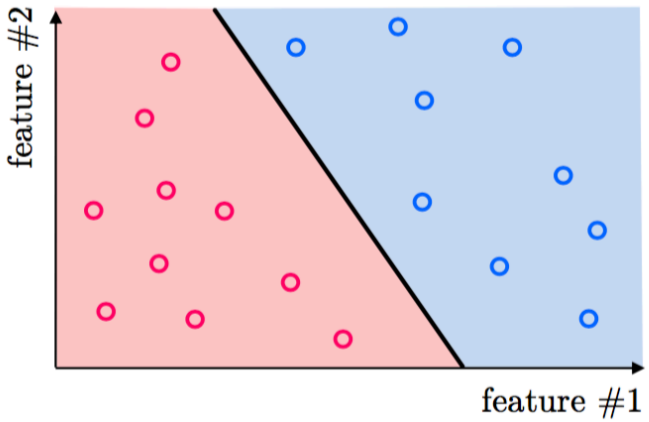
- precisely the same situation with classification

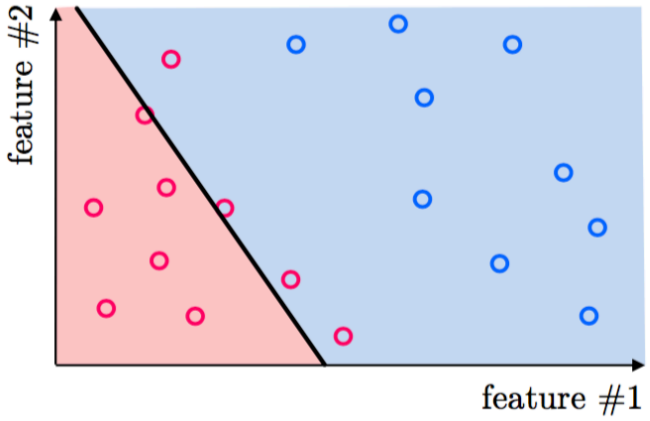




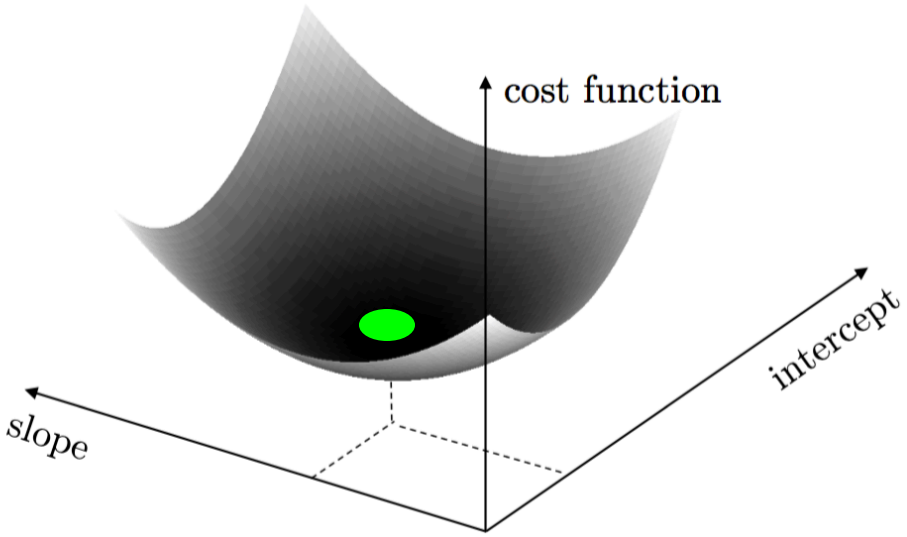
parameter tuning: classification

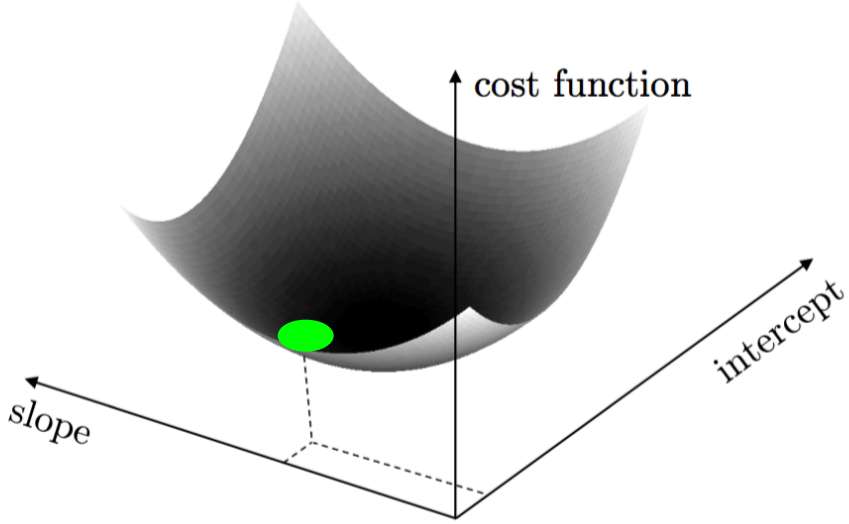
- precisely the same situation with classification
- a 'cost function' measures how well a model ***splits*** data given a set of parameters
- better parameters provide a better fit, and ***lower*** value of cost function

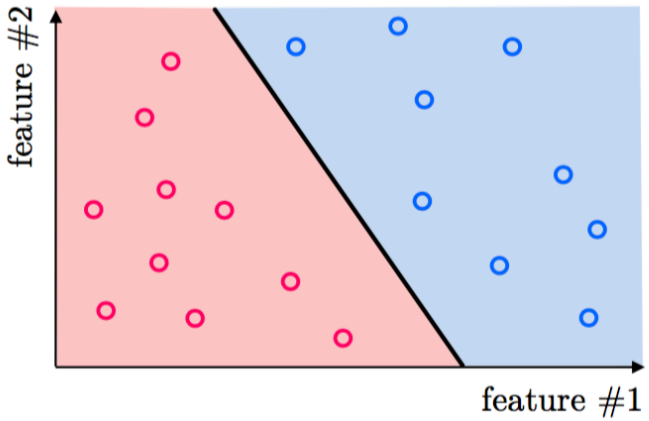


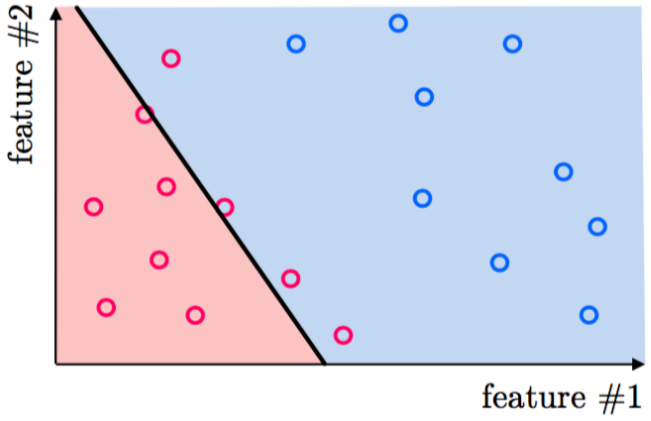


parameter tuning



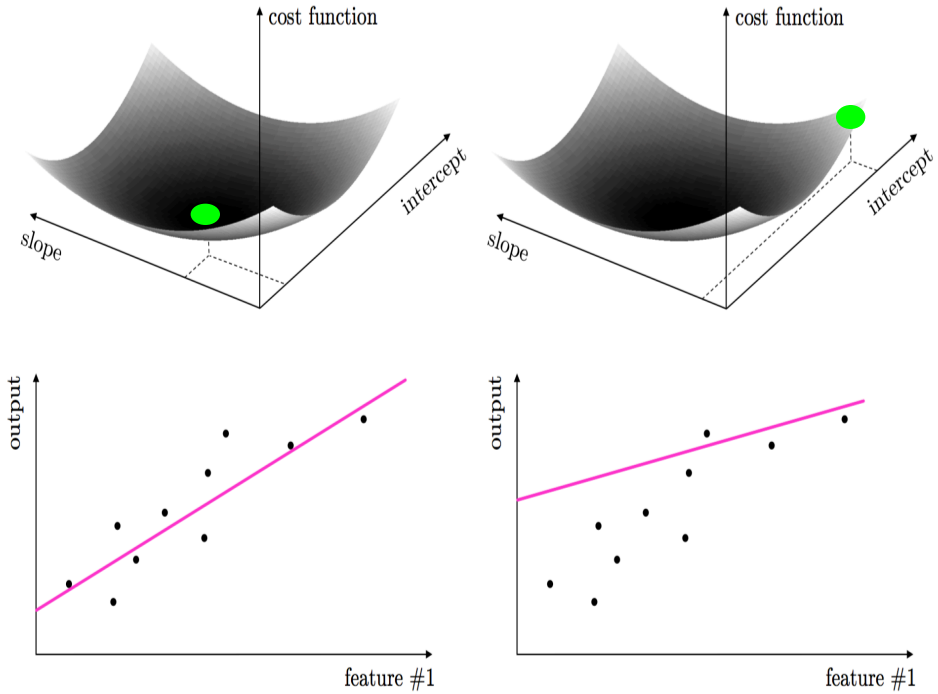


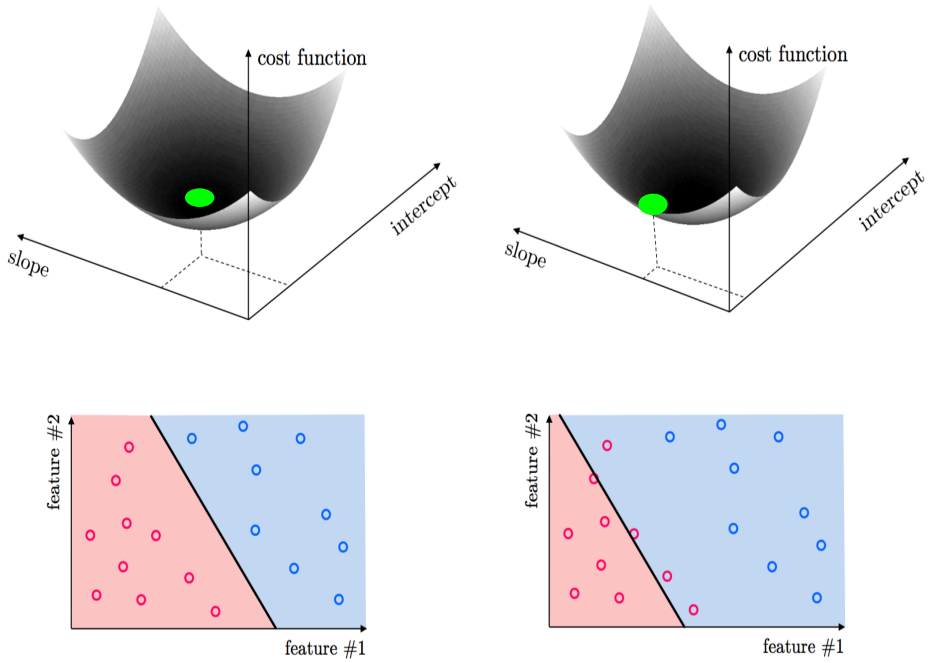




parameter tuning: in general

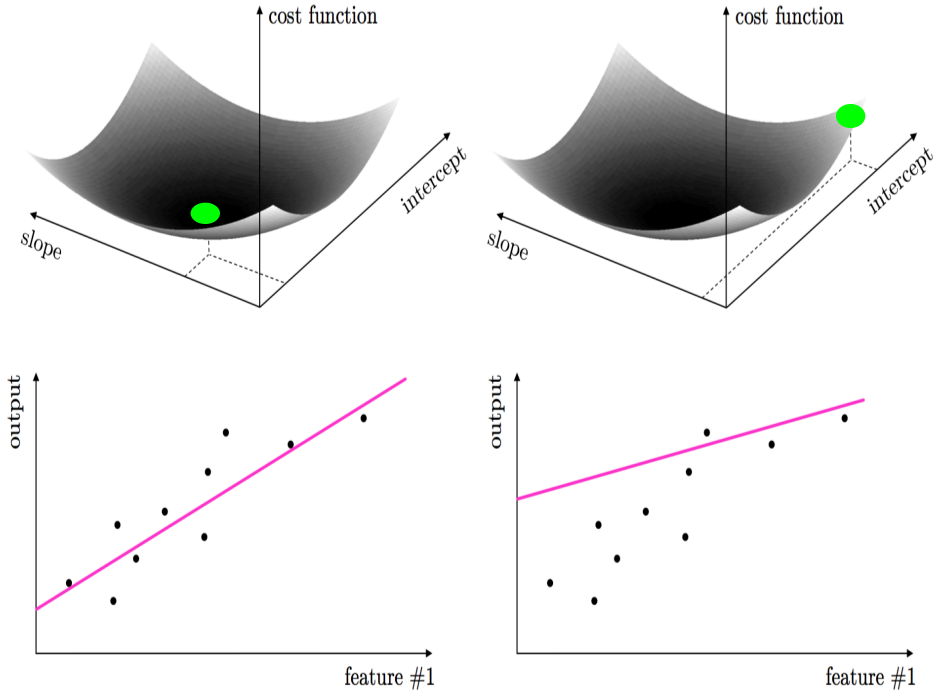
- regardless if regression/classification, tuning parameters is the same issue

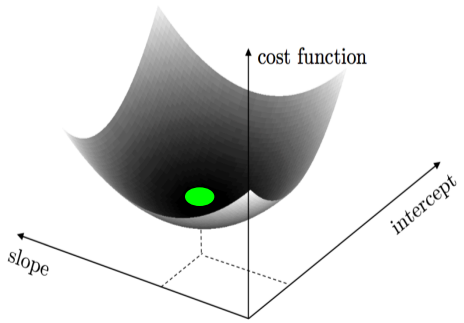


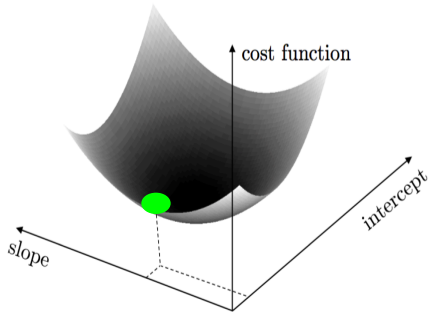


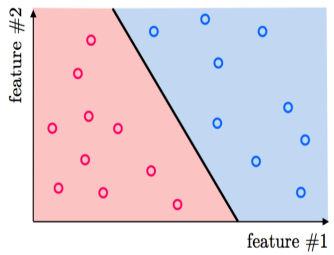
parameter tuning: in general

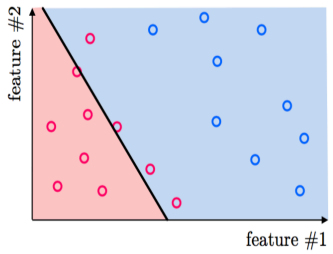
- regardless if regression/classification, tuning parameters is the same issue
- how to find the set of parameters that ***minimizes*** a differentiable cost function





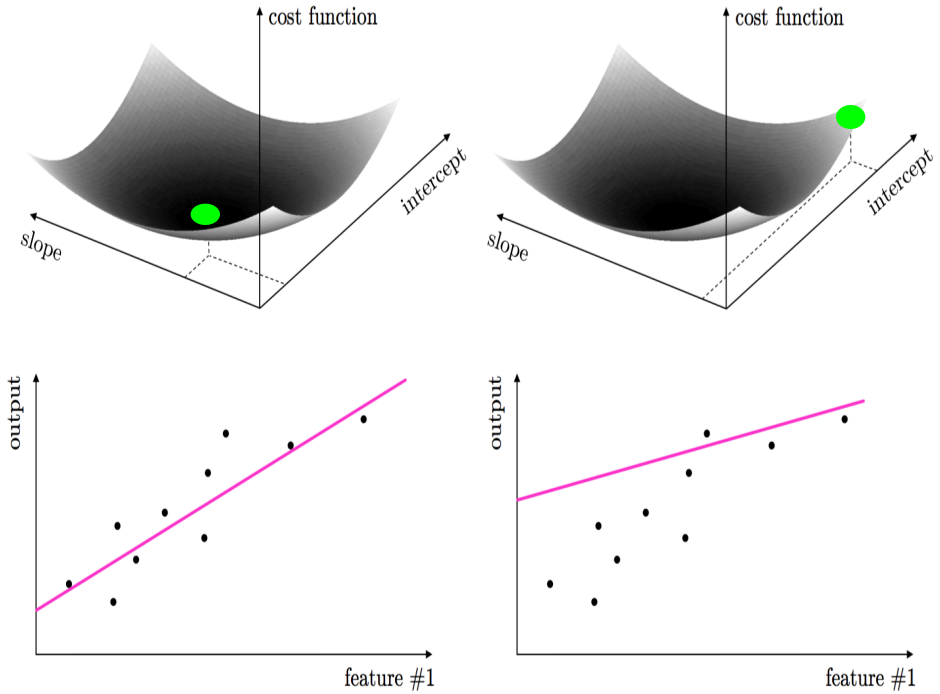


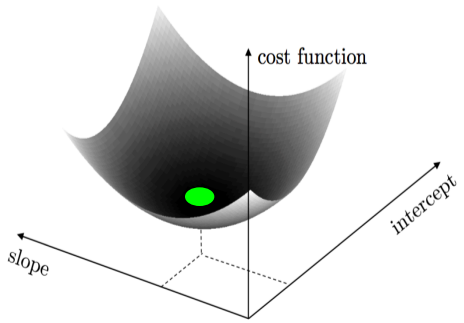


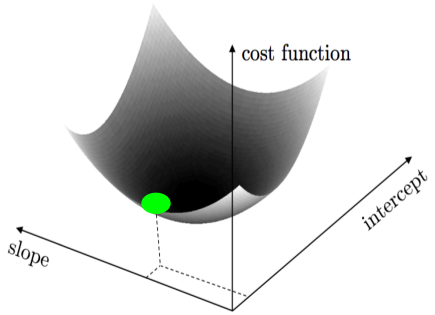


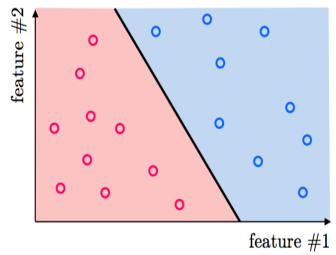
parameter tuning: in general

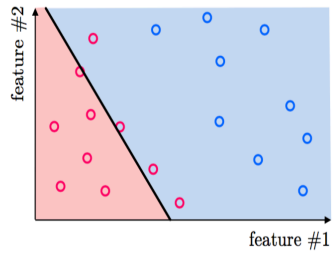
- **problem**: typically cannot visualize cost function, number of parameters > 2





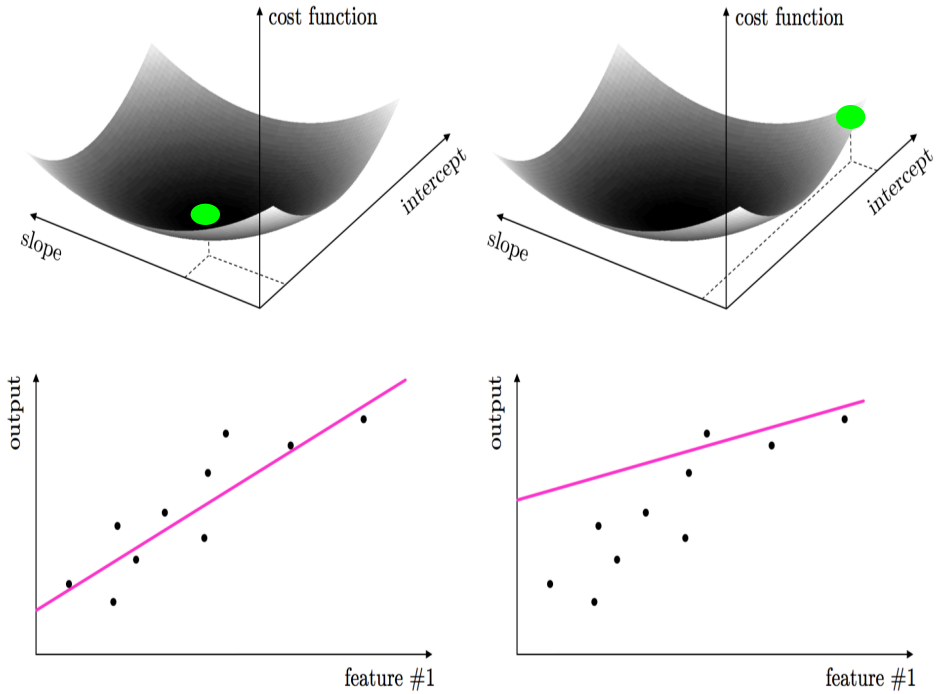


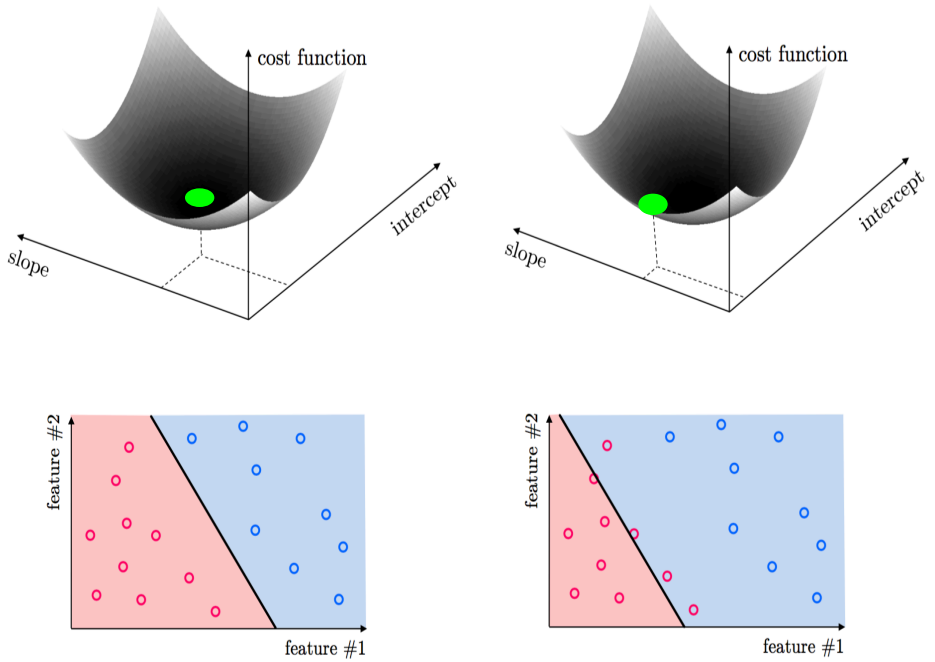




parameter tuning: in general

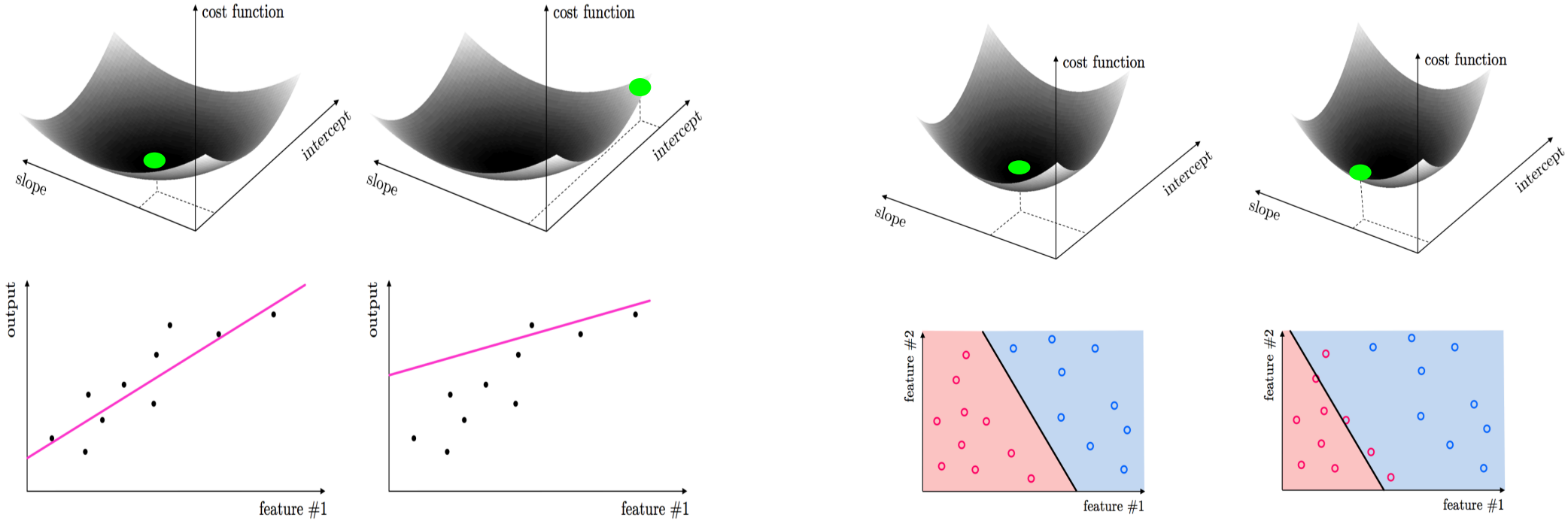
- **problem**: typically cannot visualize cost function, number of parameters > 2
- **solution**: host of algorithms based on simple calculus





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

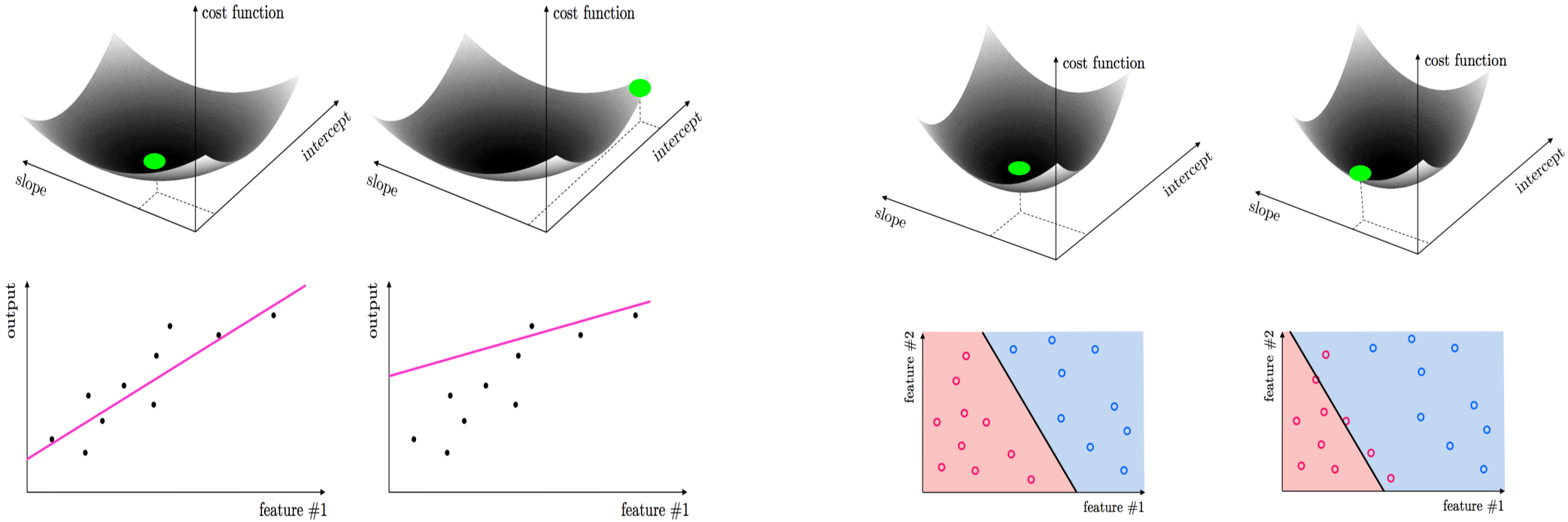


Demo # 6,7,etc, – classification

- to the notebook!

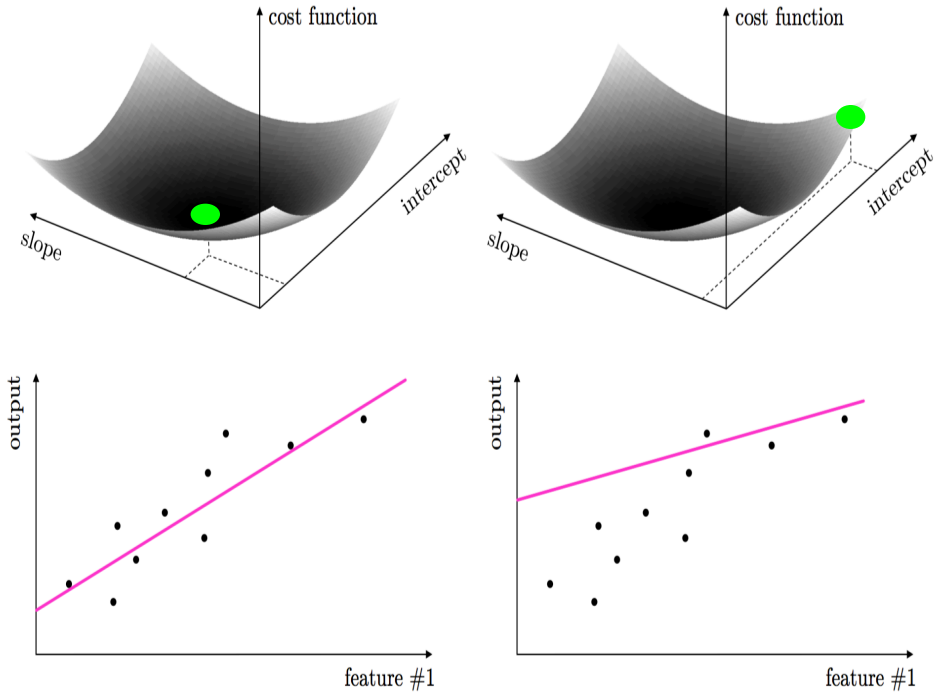
Conclusions

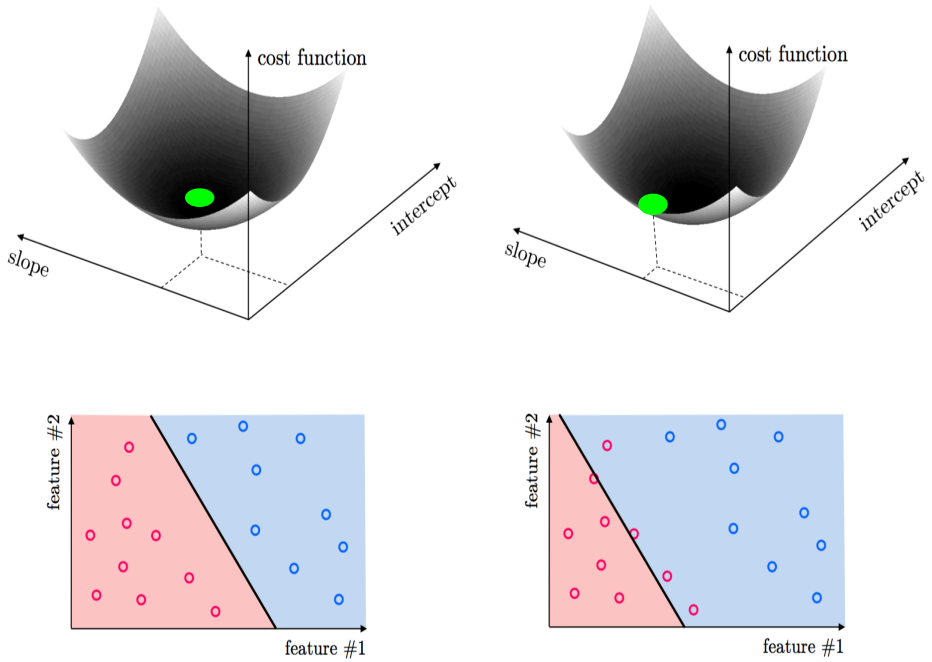
- Machine learning models have parameters to tune for optimal learning



Conclusions

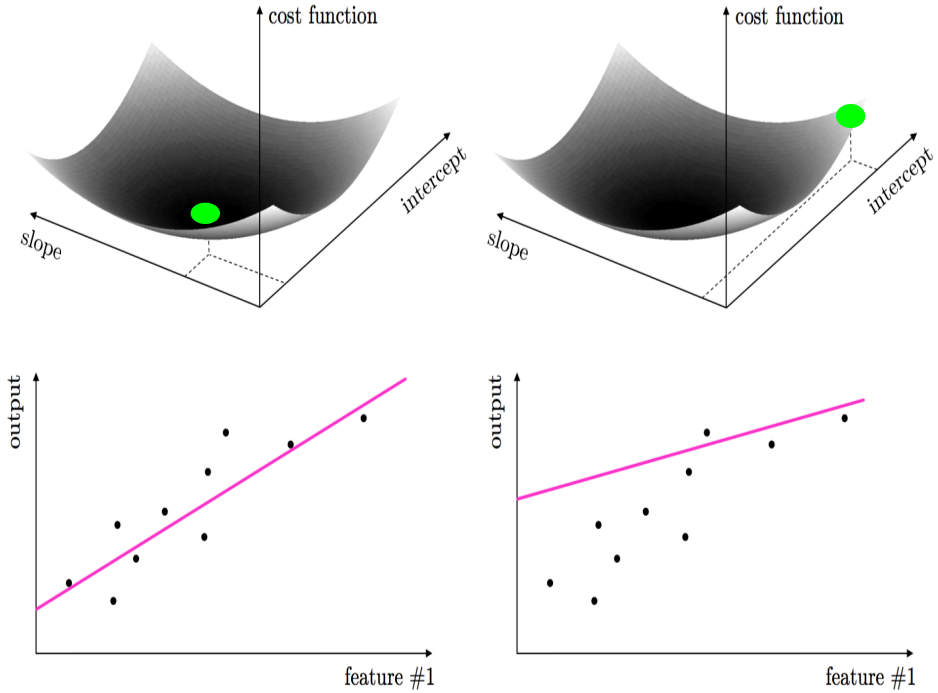
- All machine learning models have parameters to tune for optimal learning
- To tune parameters a cost function is formed which must be ***minimized***

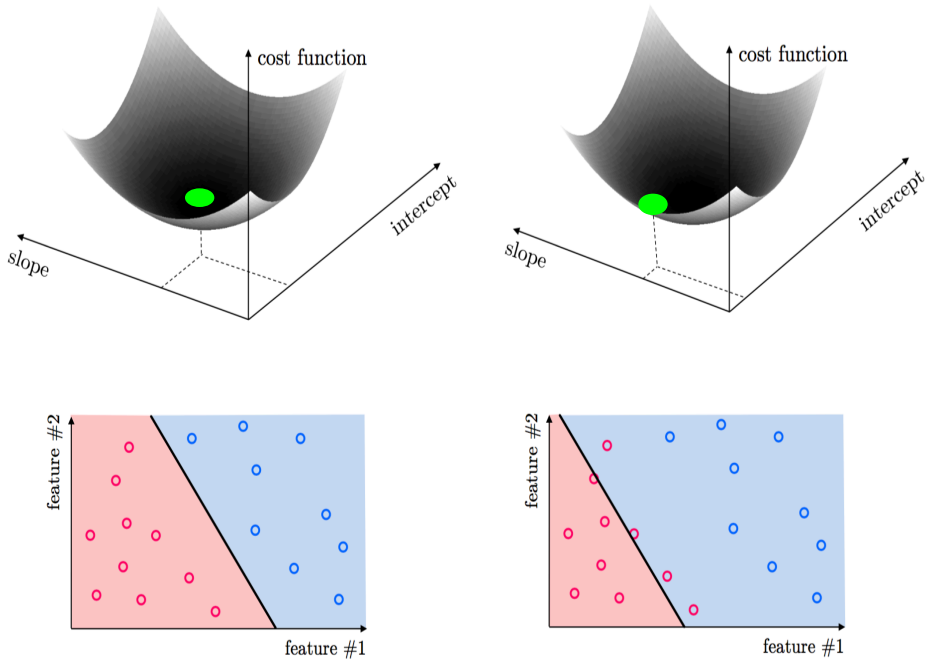




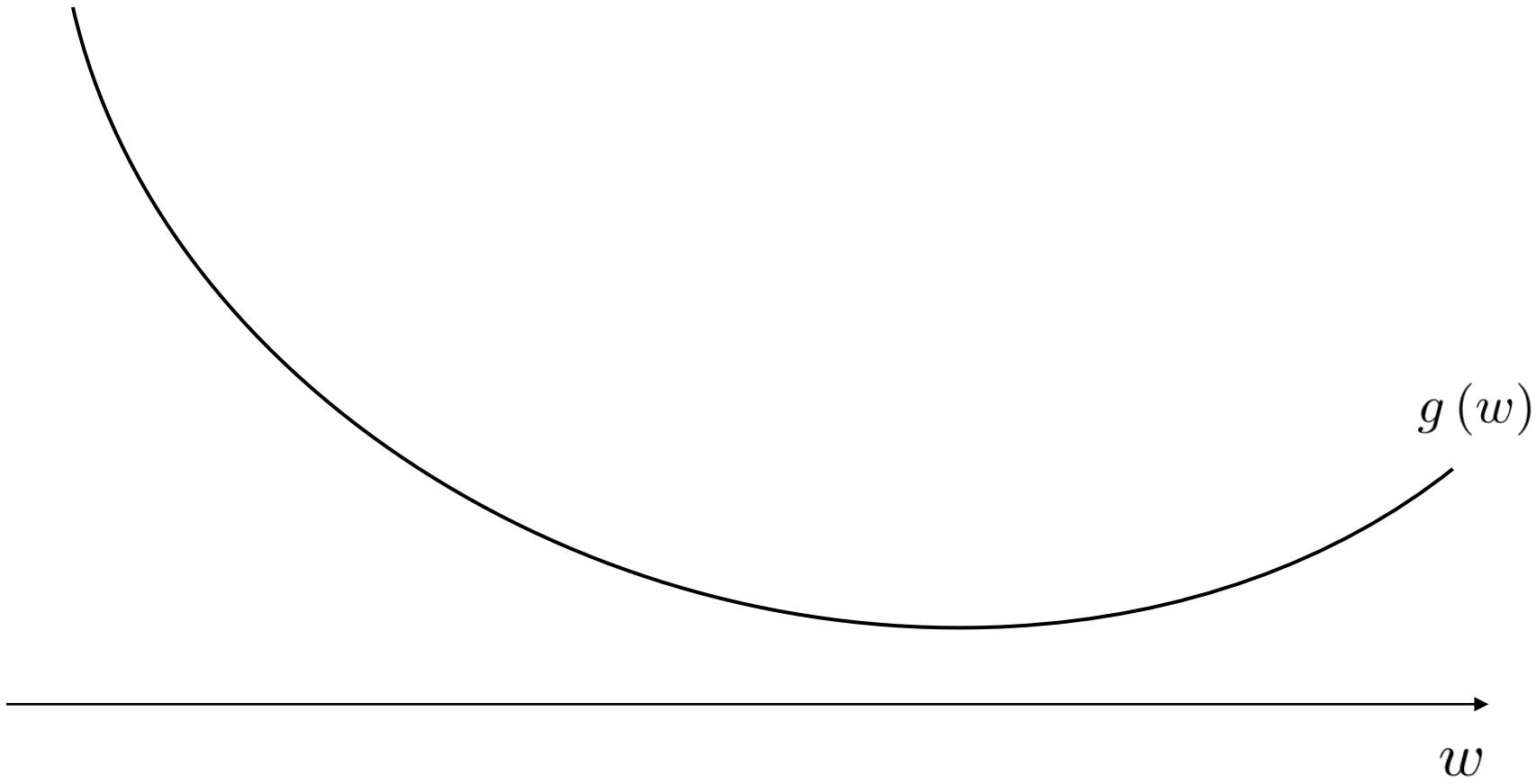
Conclusions

- All machine learning models have parameters to tune for optimal learning
- To tune parameters a cost function is formed which must be ***minimized***
- Gradient descent (and extensions) most popular way of doing this



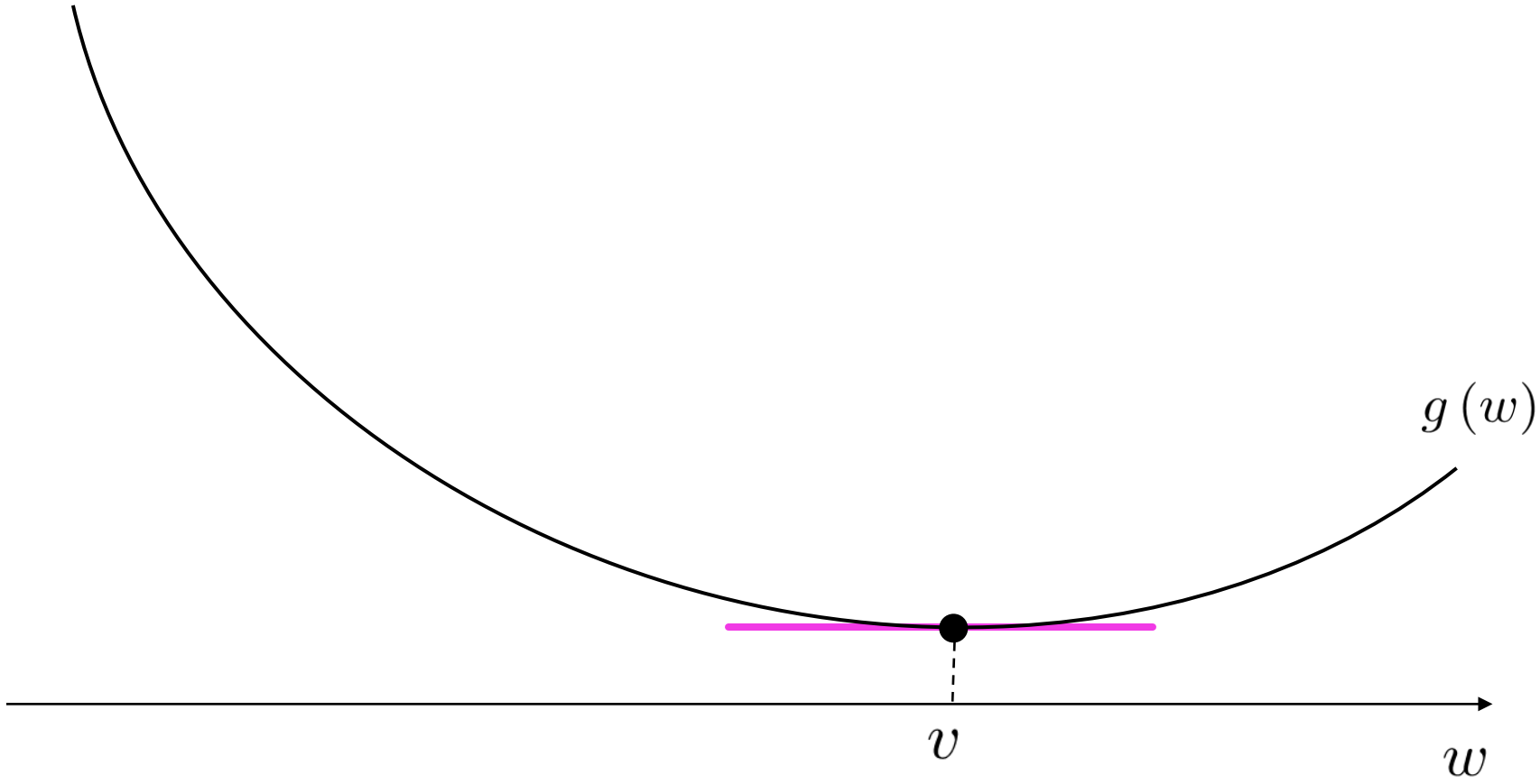


the derivative



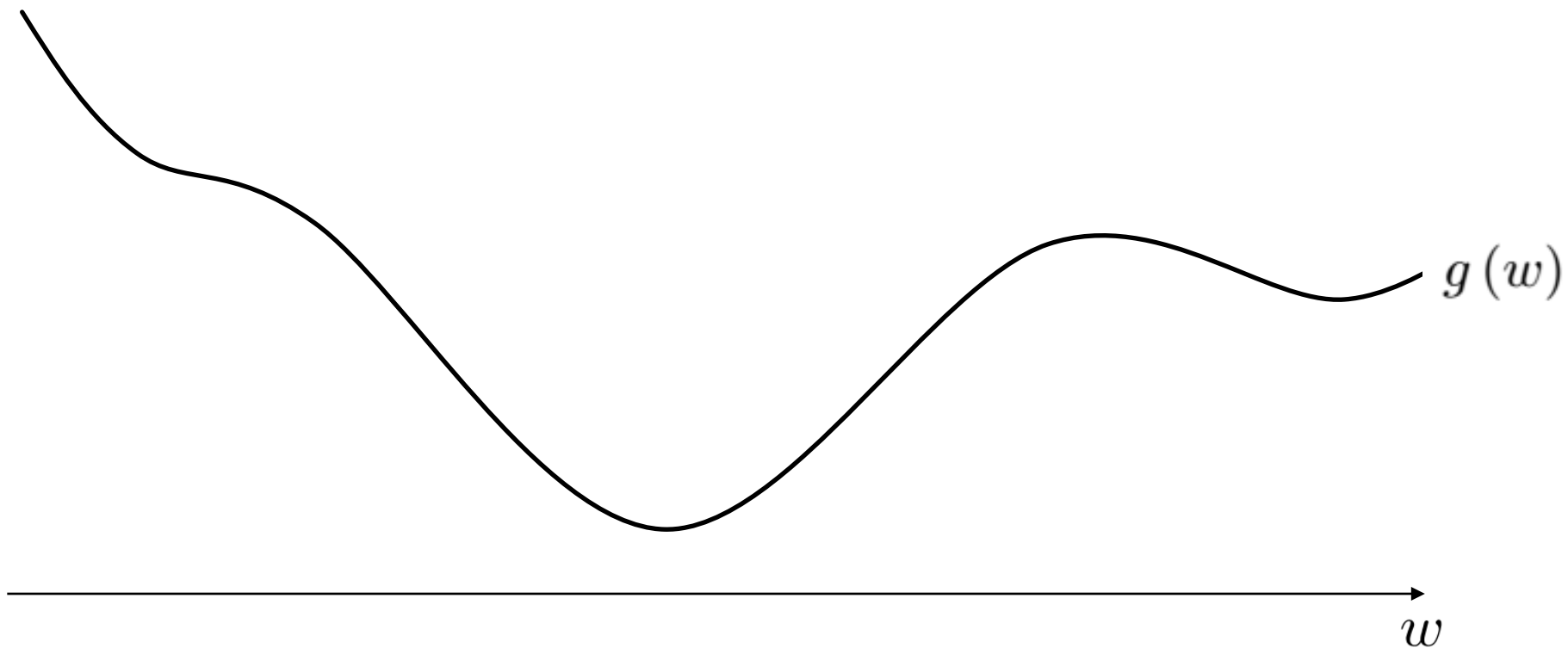
the derivative

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



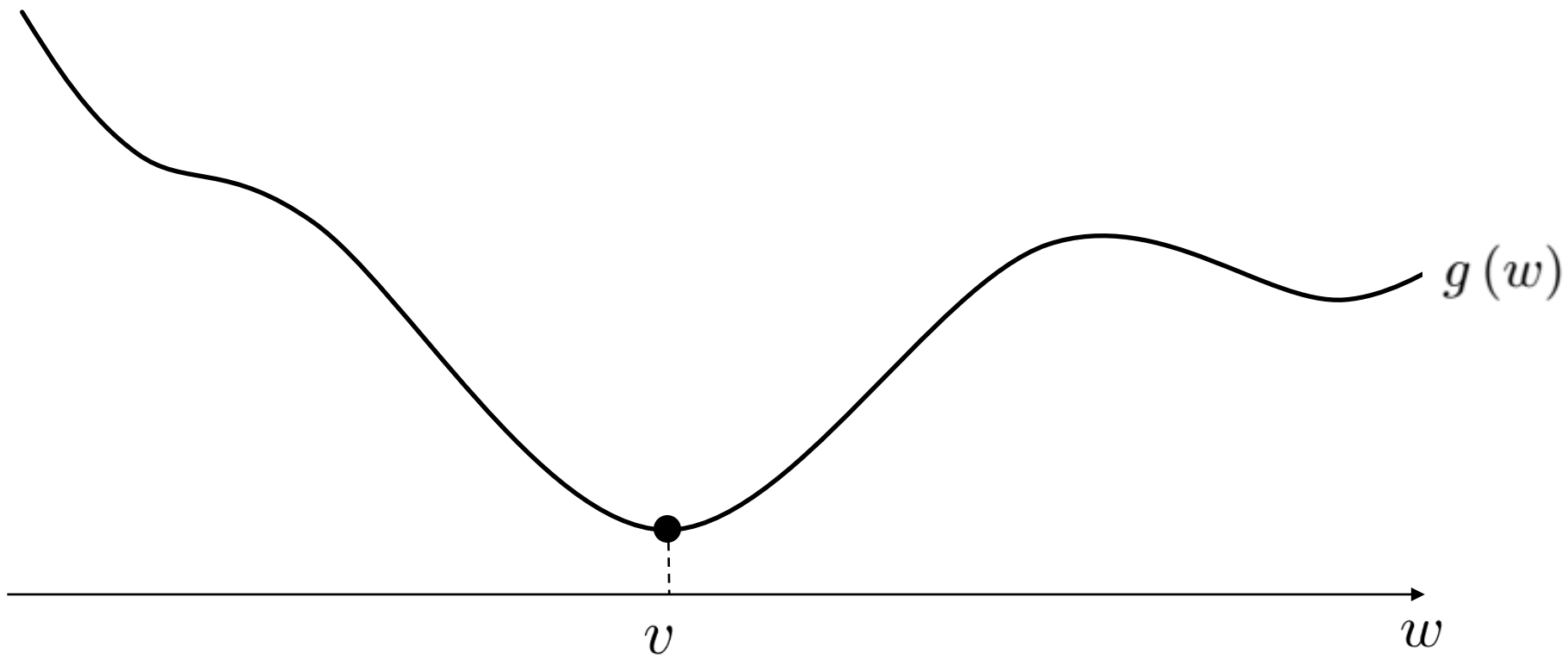
the derivative

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



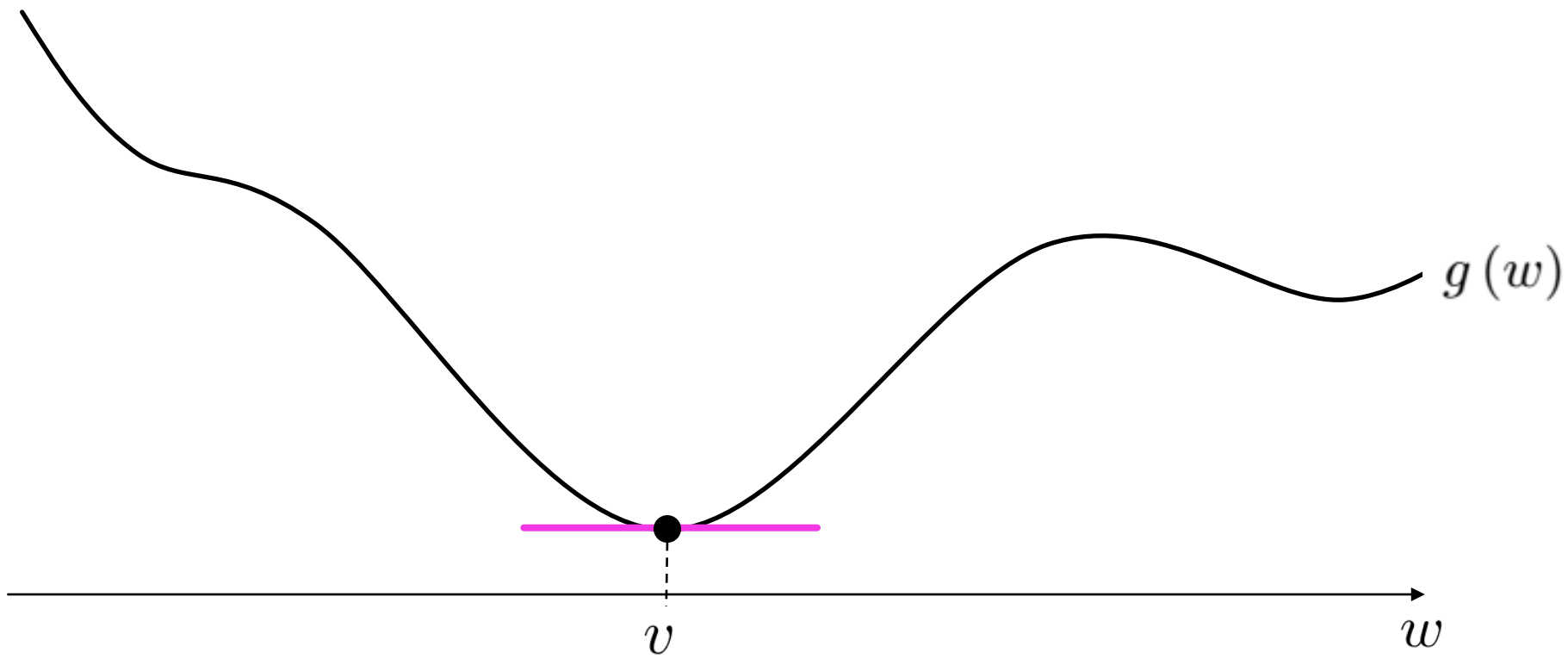
the derivative

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



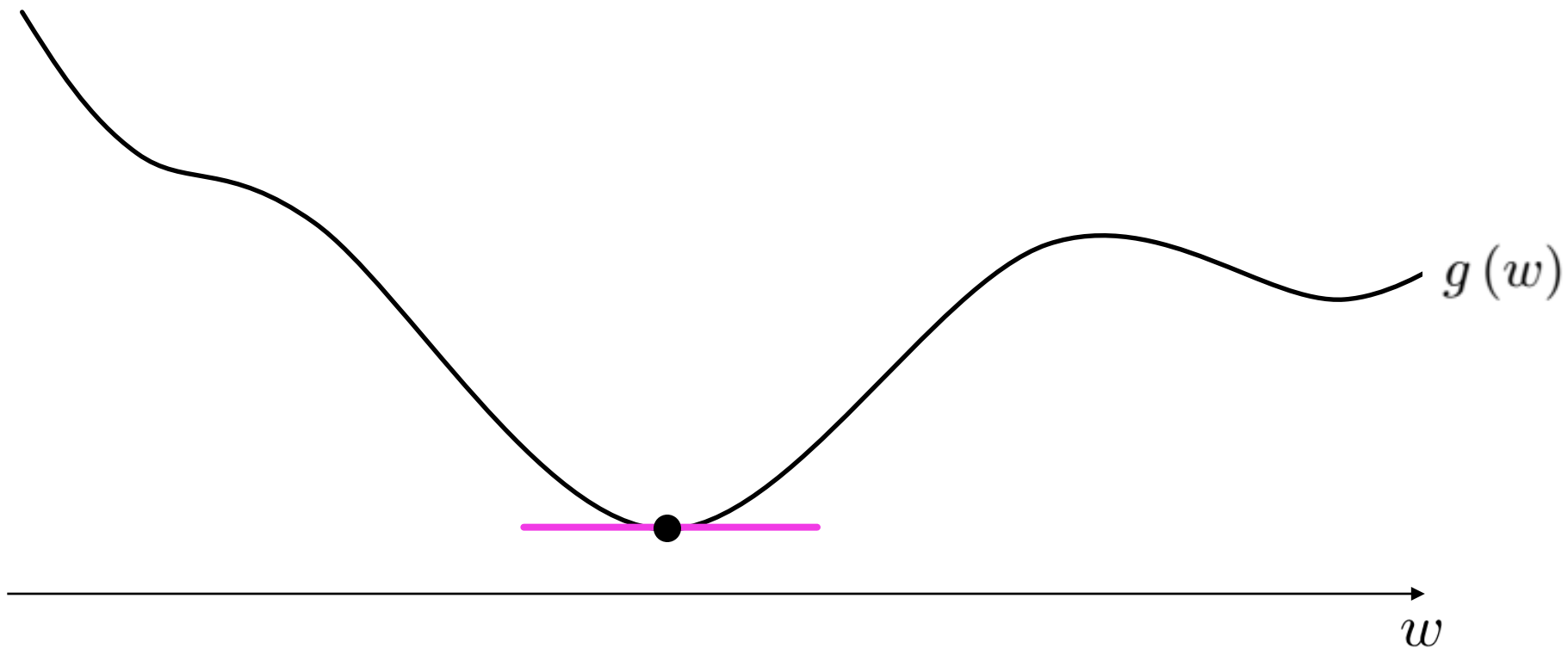
the derivative

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



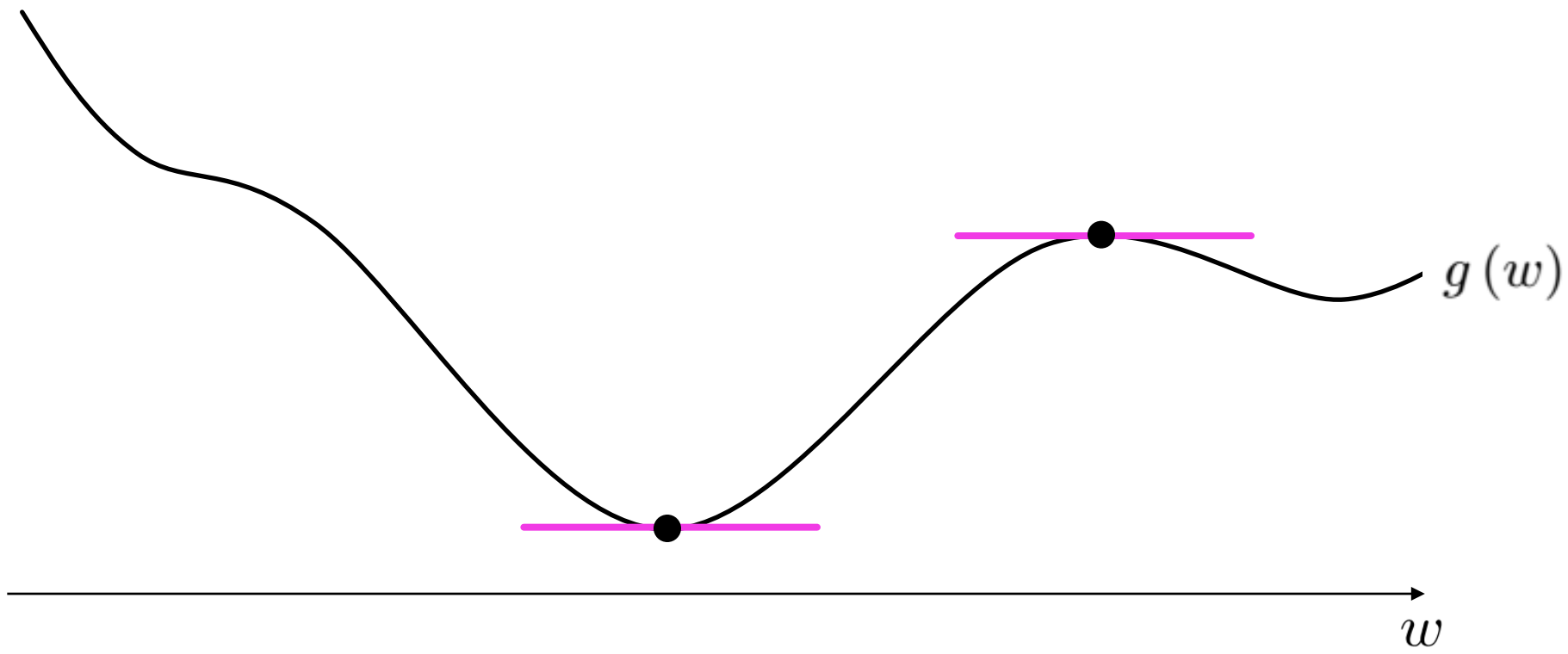
the derivative

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



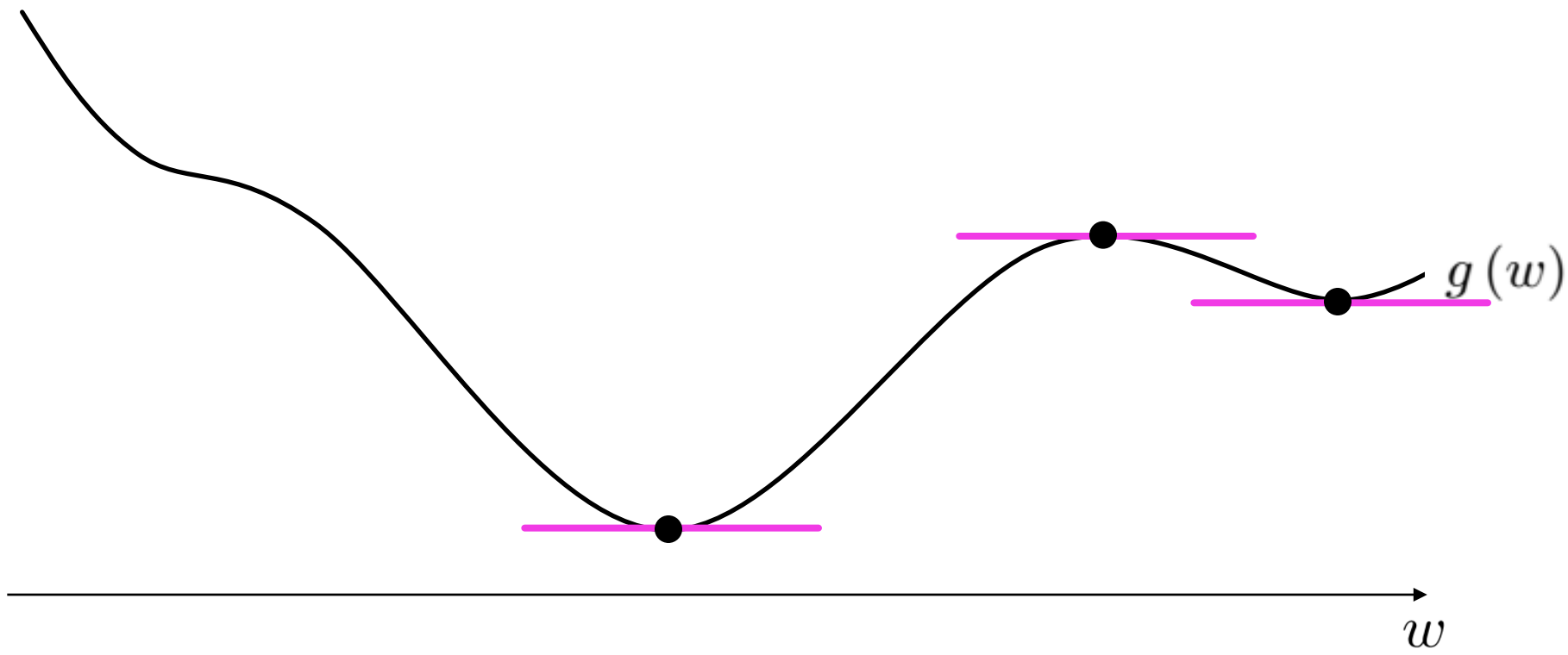
the derivative

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



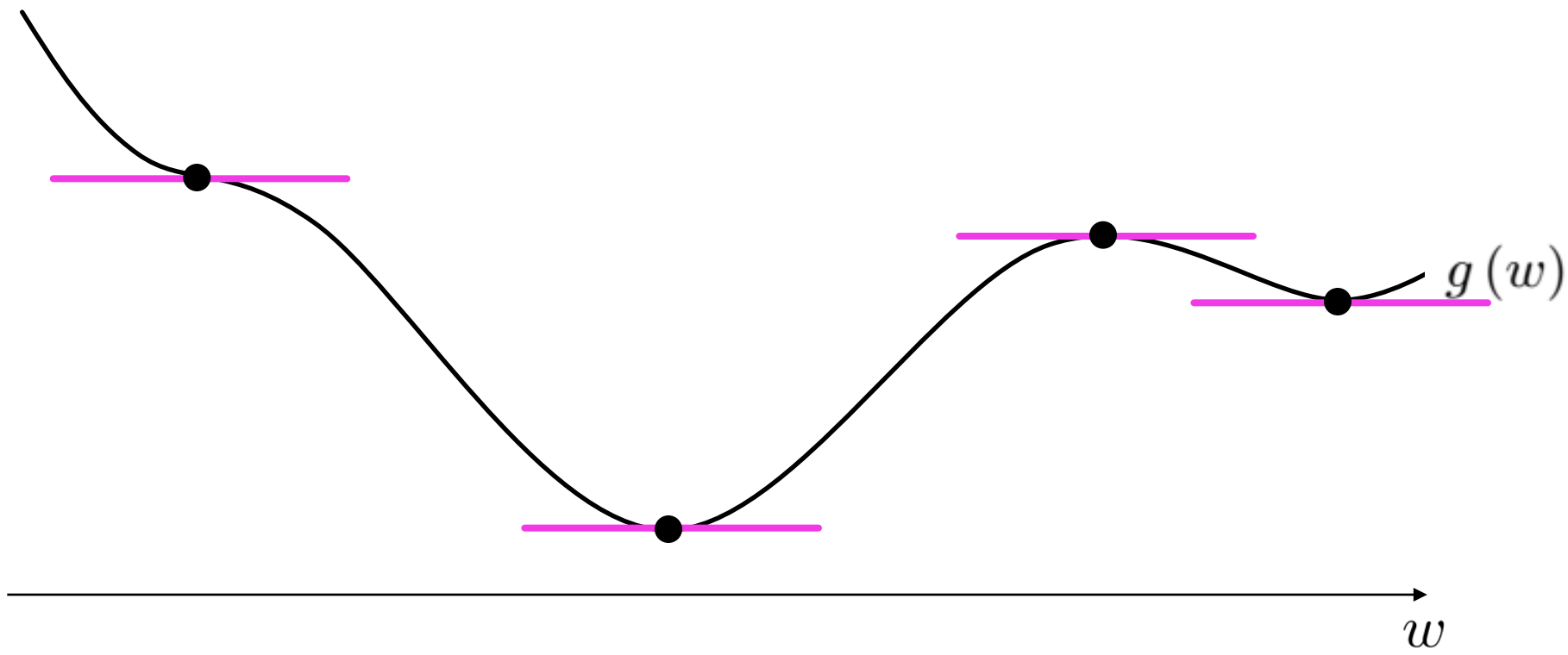
the derivative

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



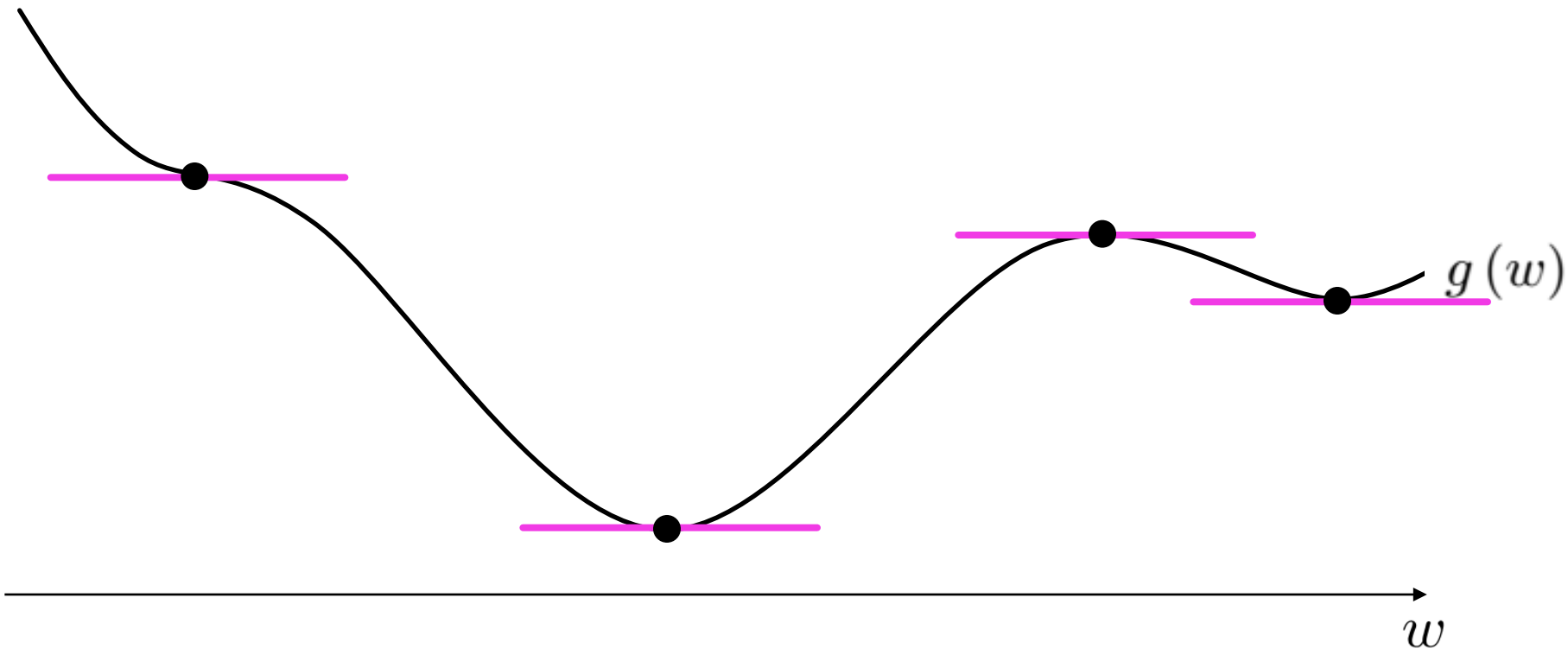
the derivative

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



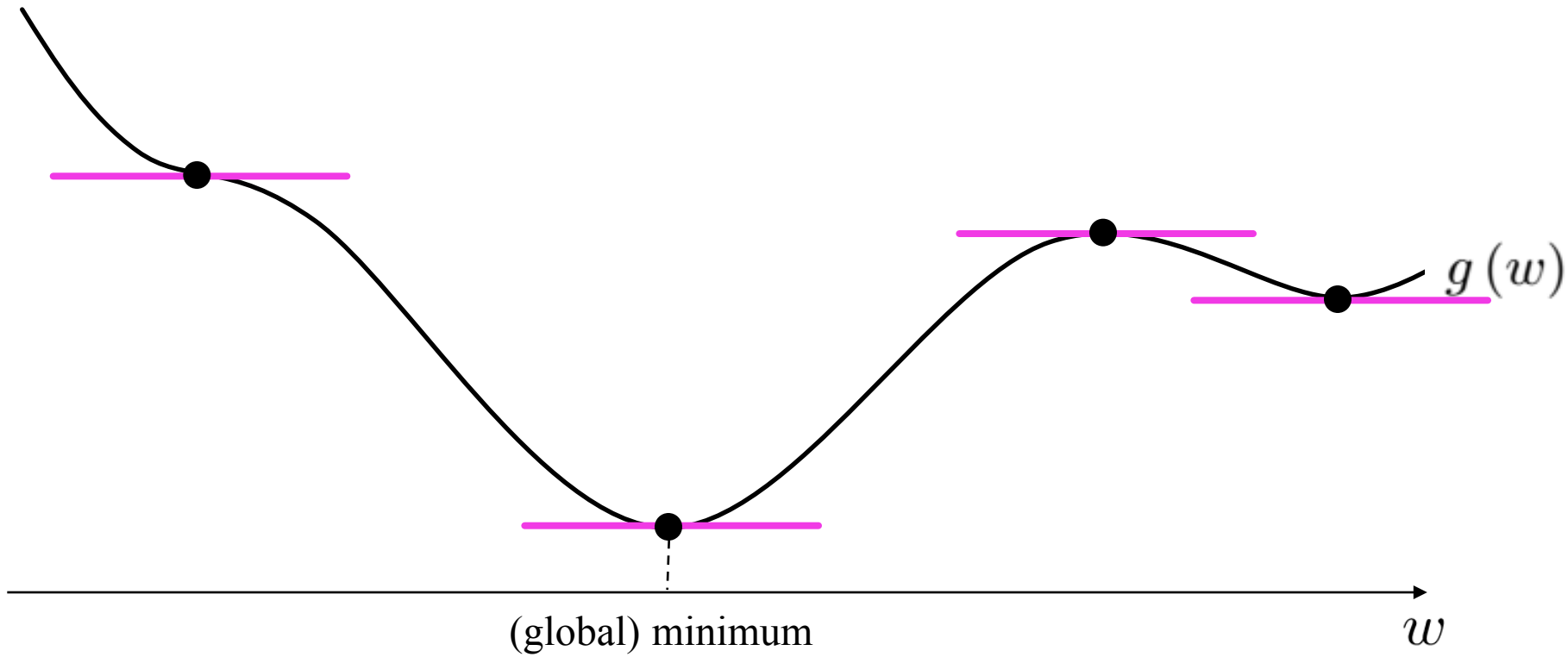
the derivative

- 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***



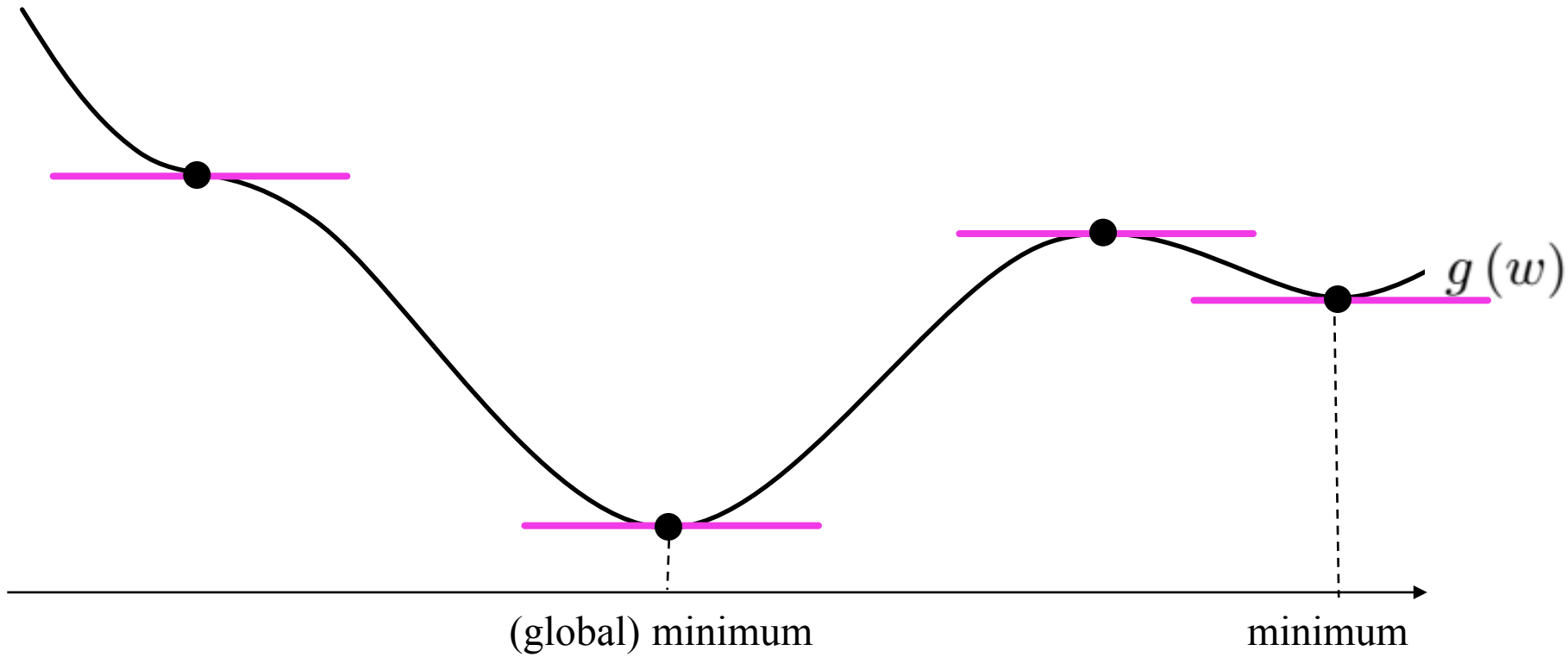
the derivative

- 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***



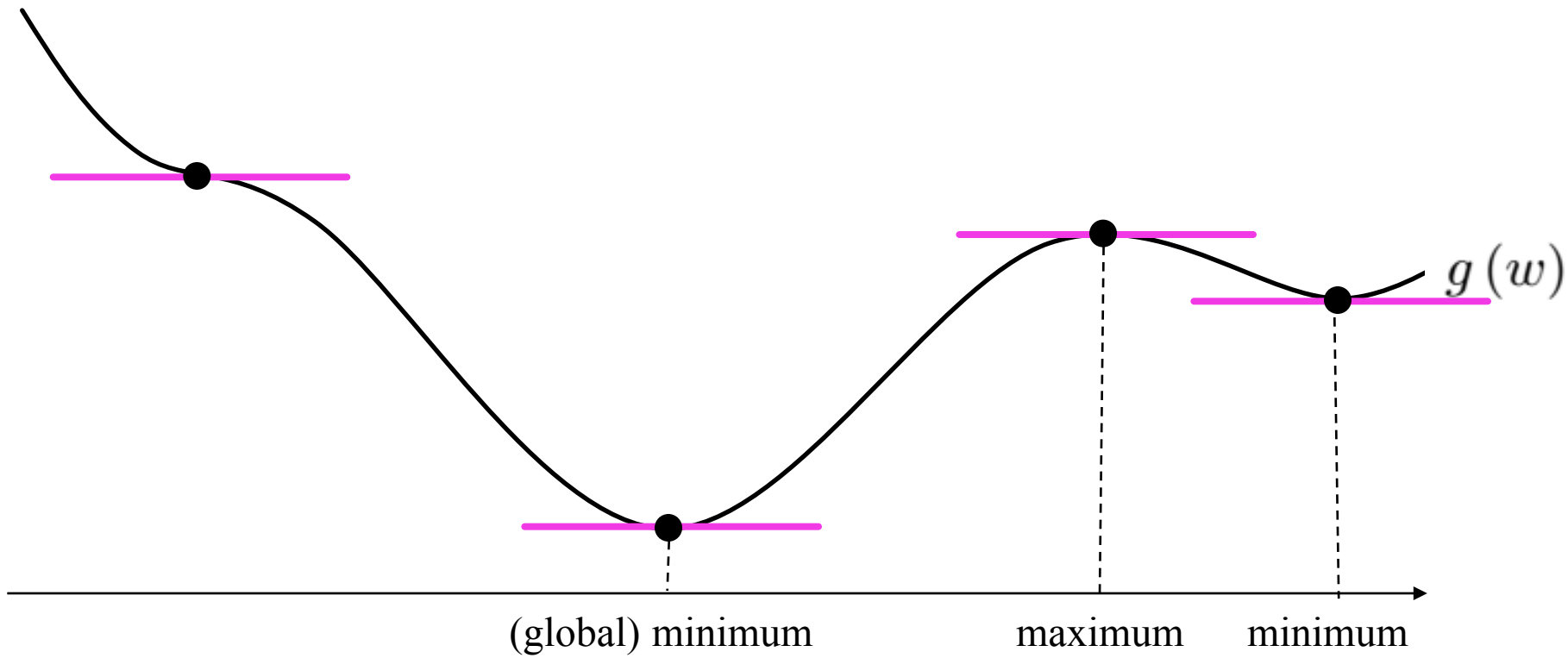
the derivative

- 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***



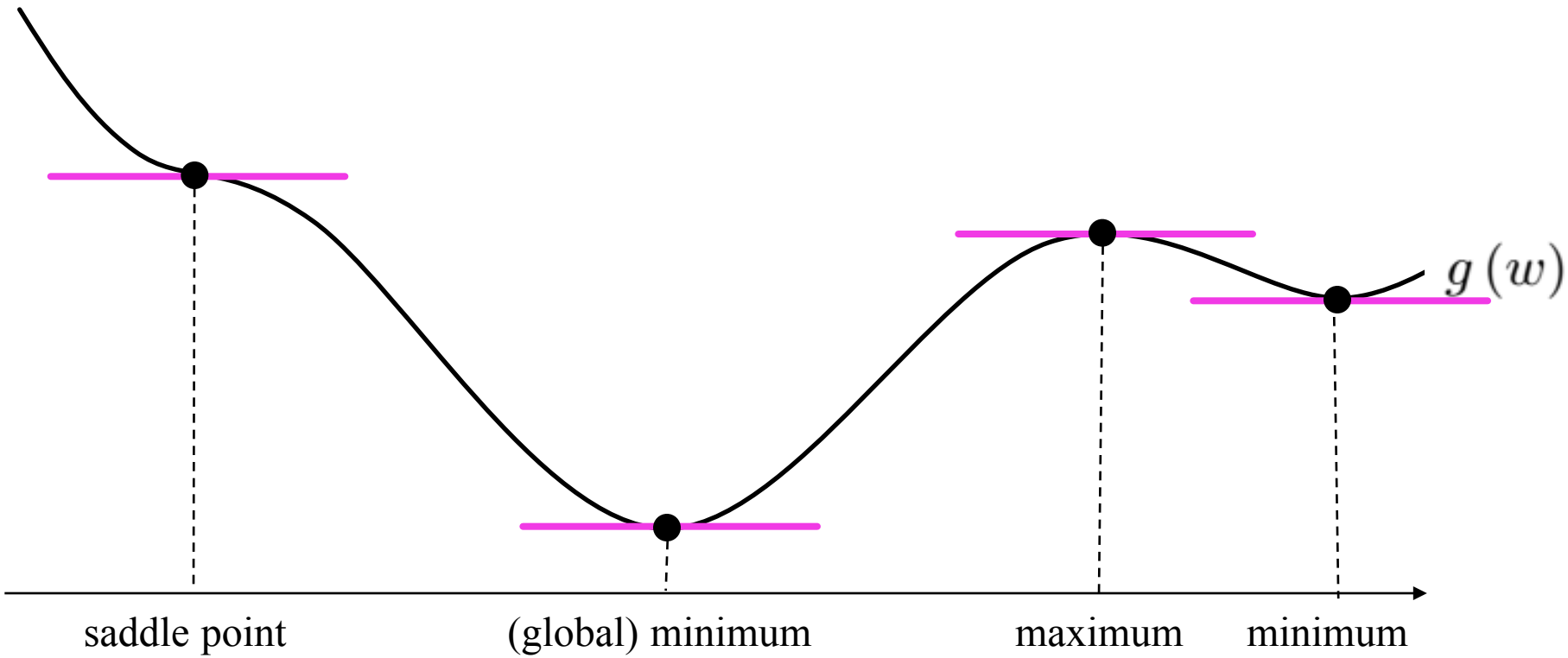
the derivative

- 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***



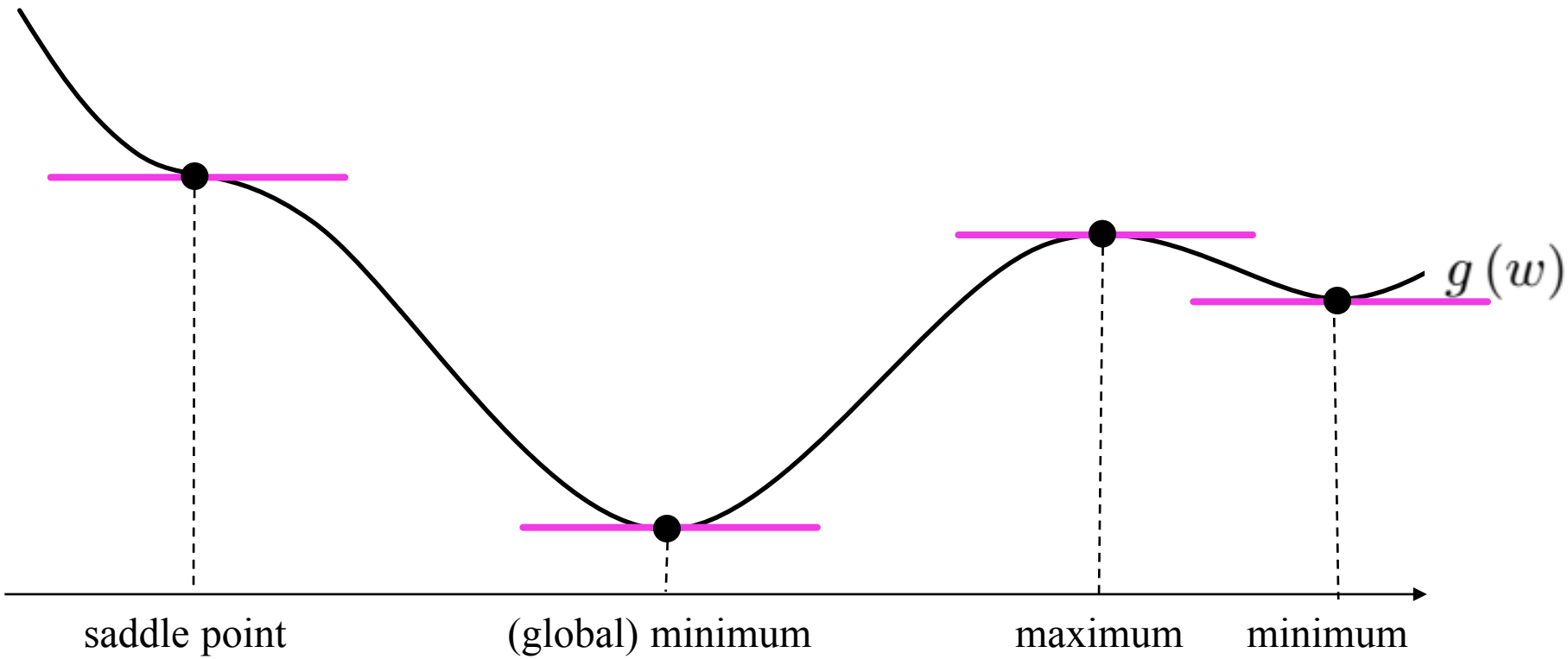
the derivative

- 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***



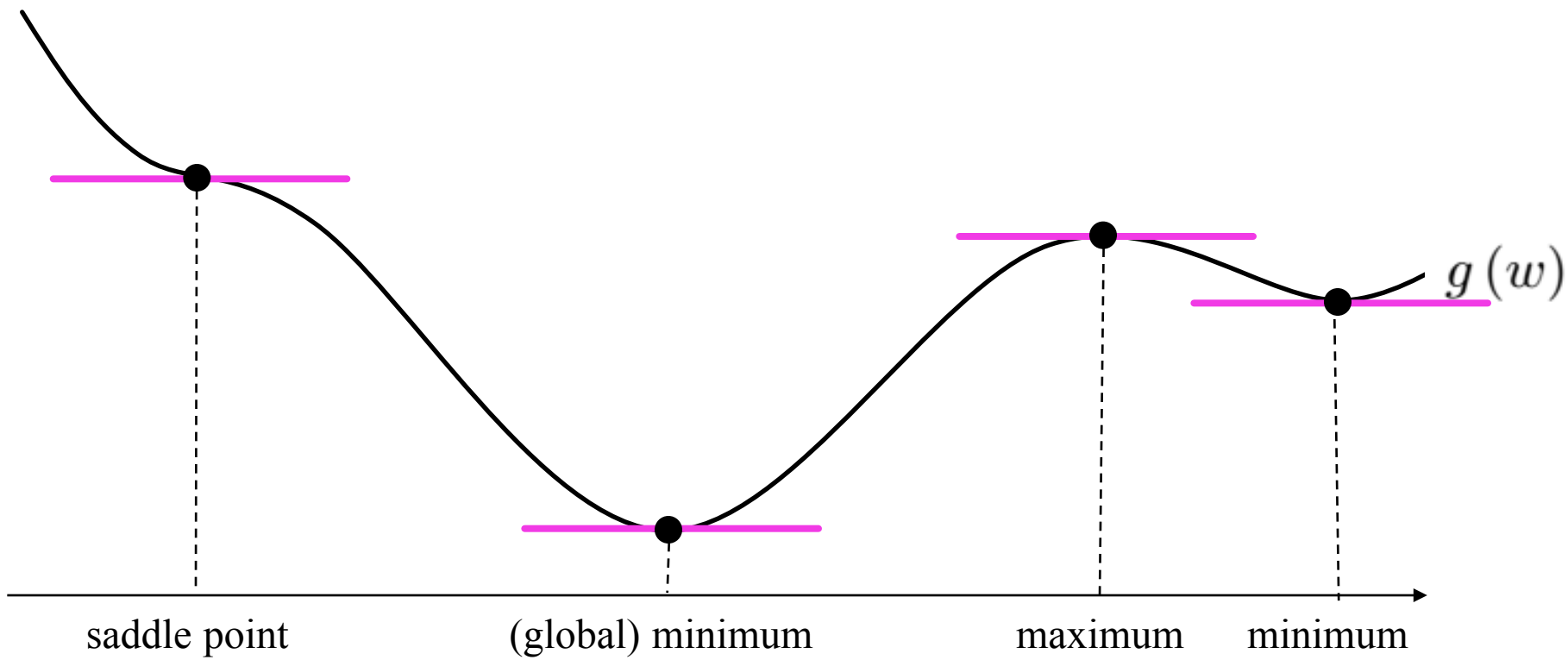
the derivative

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



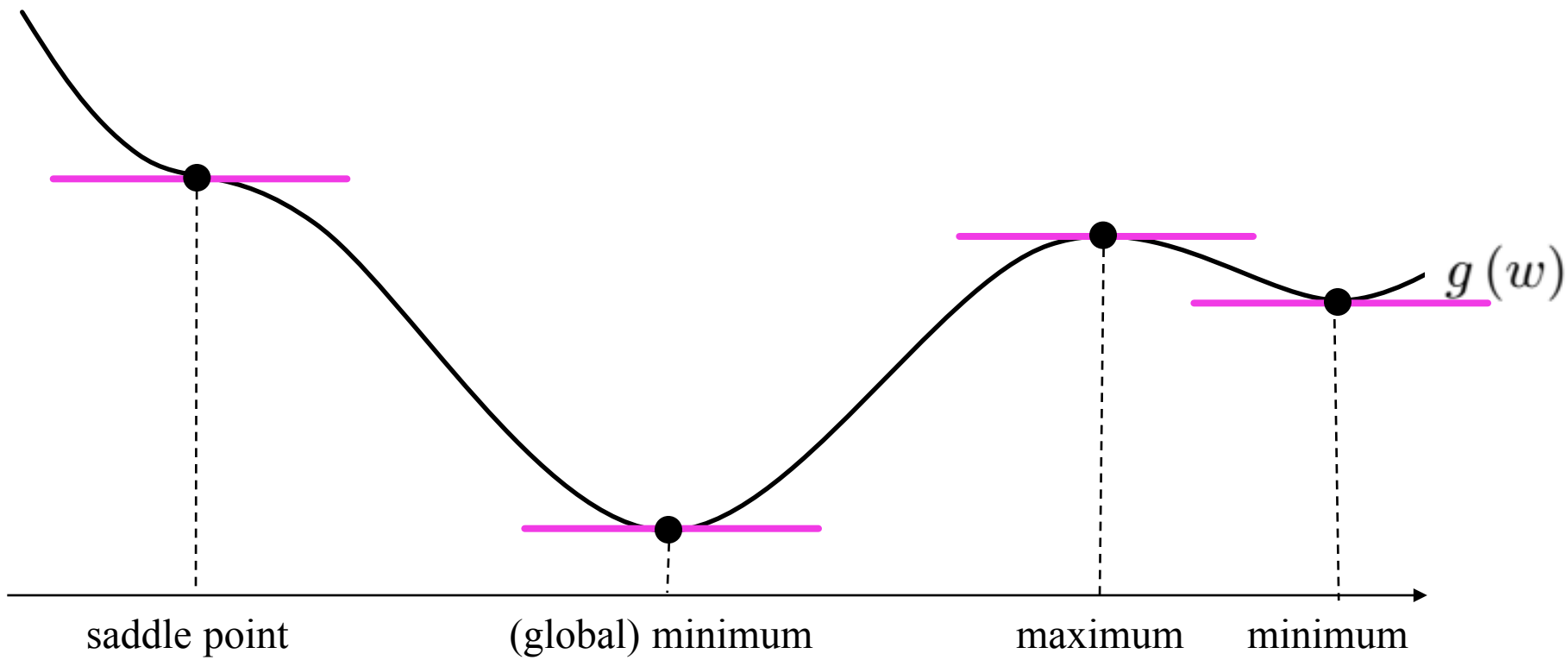
the derivative

- 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!

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.