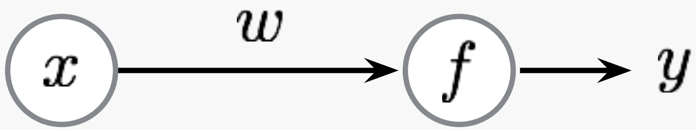
Training perceptrons

Let’s start easy

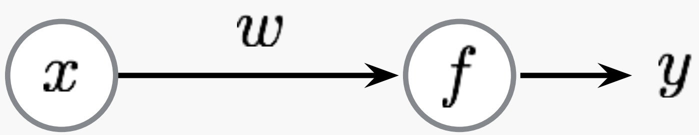
world’s smallest perceptron!



$$y = wx$$

What does this look like?

world's smallest perceptron!



$$y = wx$$

(a.k.a. line equation, linear regression)

Learning a Perceptron

Given a set of samples and a Perceptron

$$\{x_i, y_i\}$$

$$y = f_{\text{PER}}(x; w)$$

Estimate the parameters of the Perceptron

$$w$$

Given training data:

x	y
10	10.1
2	1.9
3.5	3.4
1	1.1

What do you think the weight parameter is?

$$y = wx$$

Given training data:

x	y
10	10.1
2	1.9
3.5	3.4
1	1.1

What do you think the weight parameter is?

$$y = wx$$

not so obvious as the network gets more complicated so we use

…

An Incremental Learning Strategy

(gradient descent)

Given several examples

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$$

and a perceptron

$$\hat{y} = wx$$

An Incremental Learning Strategy

(gradient descent)

Given several examples

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$$

and a perceptron

$$\hat{y} = wx$$

Modify weight w such that $\hat{y}$ gets **'closer'** to y

An Incremental Learning Strategy

(gradient descent)

Given several examples

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$$

and a perceptron

$$\hat{y} = wx$$

Modify weight w such that $\hat{y}$ gets **'closer'** to y

perceptron parameter

perceptron output

true label

An Incremental Learning Strategy

(gradient descent)

Given several examples

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$$

and a perceptron

$$\hat{y} = wx$$

Modify weight w such that $\hat{y}$ gets **'closer'** to y

perceptron parameter

perceptron output

what does this mean?

true label

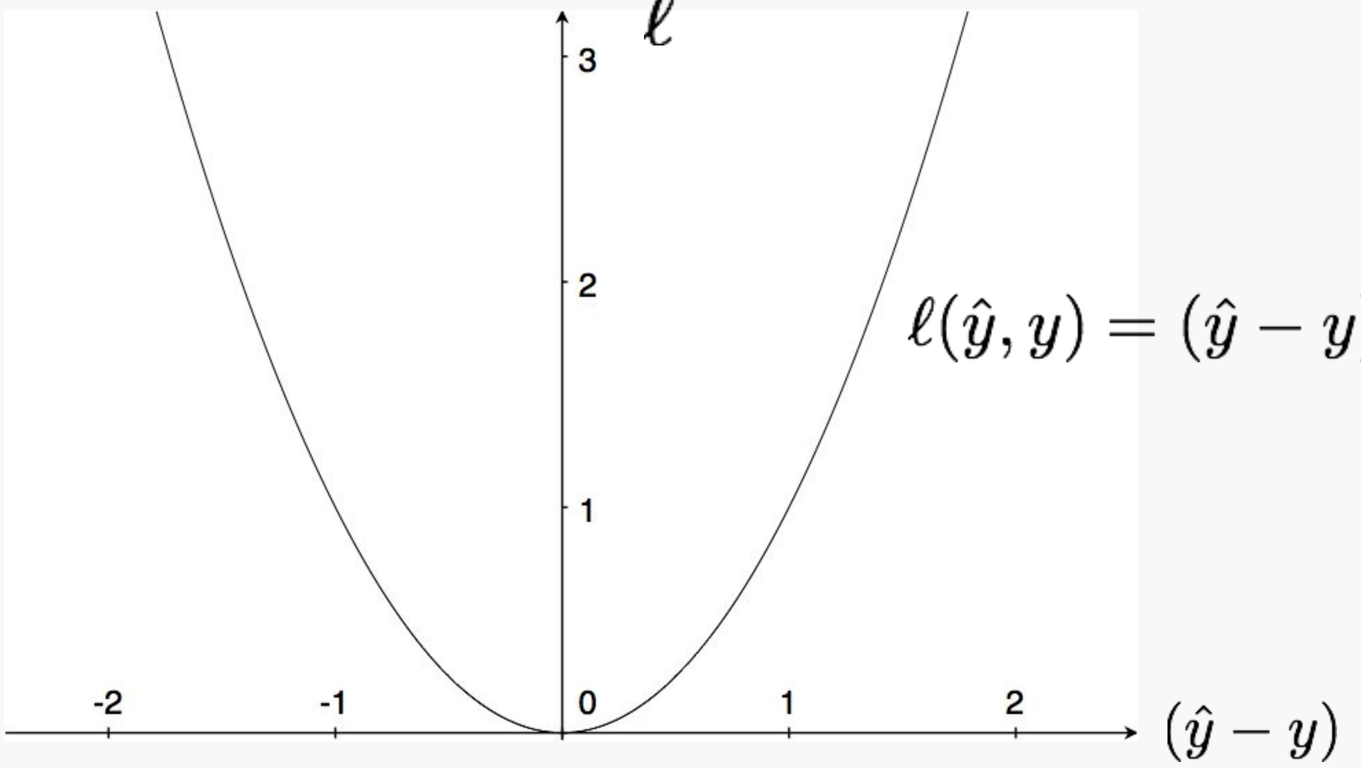
Before diving into gradient descent, we need to understand …

Loss Function
defines what is means to be
close to the true solution

YOU get to chose the loss function!
(some are better than others depending on what you want to do)

Squared Error (L2)

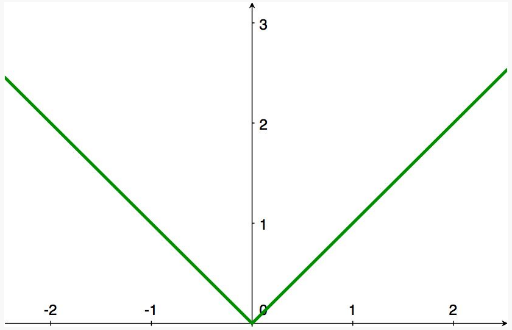
(a popular loss function) ((why?))



$$\ell(\hat{y}, y) = (\hat{y} - y)^2$$

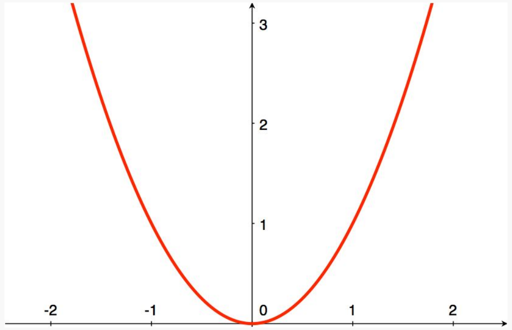
L1 Loss

$$\ell(\hat{y}, y) = |\hat{y} - y|$$



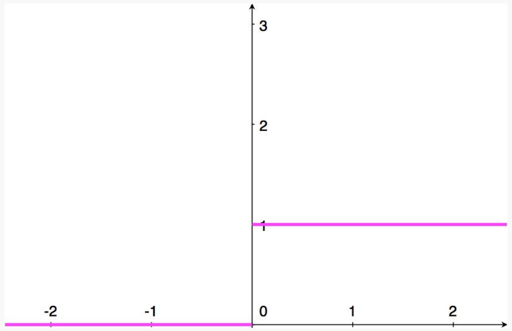
L2 Loss

$$\ell(\hat{y}, y) = (\hat{y} - y)^2$$



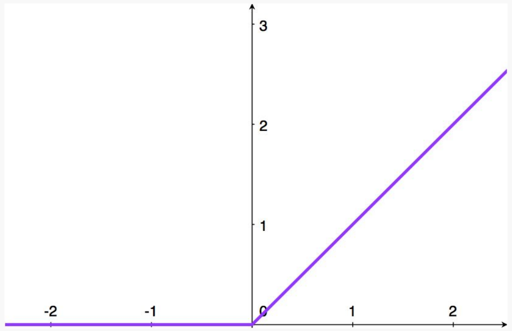
Zero-One Loss

$$\ell(\hat{y}, y) = \mathbf{1}[\hat{y} = y]$$



Hinge Loss

$$\ell(\hat{y}, y) = \max(0, 1 - y \cdot \hat{y})$$



back to the…

World's Smallest Perceptron!

$$x \xrightarrow{w} f \longrightarrow y$$

$$y = wx$$

(a.k.a. line equation, linear regression)

function of **ONE** parameter!

Learning a Perceptron

Given a set of samples and a Perceptron

$$\{x_i, y_i\}$$

$$y = f_{\text{PER}}(x; w)$$

what is this activation function?

Estimate the parameter of the Perceptron

$$w$$

Learning a Perceptron

Given a set of samples and a Perceptron

$$\{x_i, y_i\}$$

$$y = f_{\text{PER}}(x; w)$$

what is this activation function?

linear function! $f(x) = wx$

Estimate the parameter of the Perceptron

$$w$$

Learning Strategy

(gradient descent)

Given several examples

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$$

and a perceptron

$$\hat{y} = wx$$

Modify weight w such that $\hat{y}$ gets **'closer'** to y

perceptron parameter

perceptron output

true label

Let’s demystify this process first…

Code to train your perceptron:

Let's demystify this process first…

Code to train your perceptron:

$$\text{for } n = 1 \ldots N$$
$$\quad w = w + (y_n - \hat{y})x_i;$$

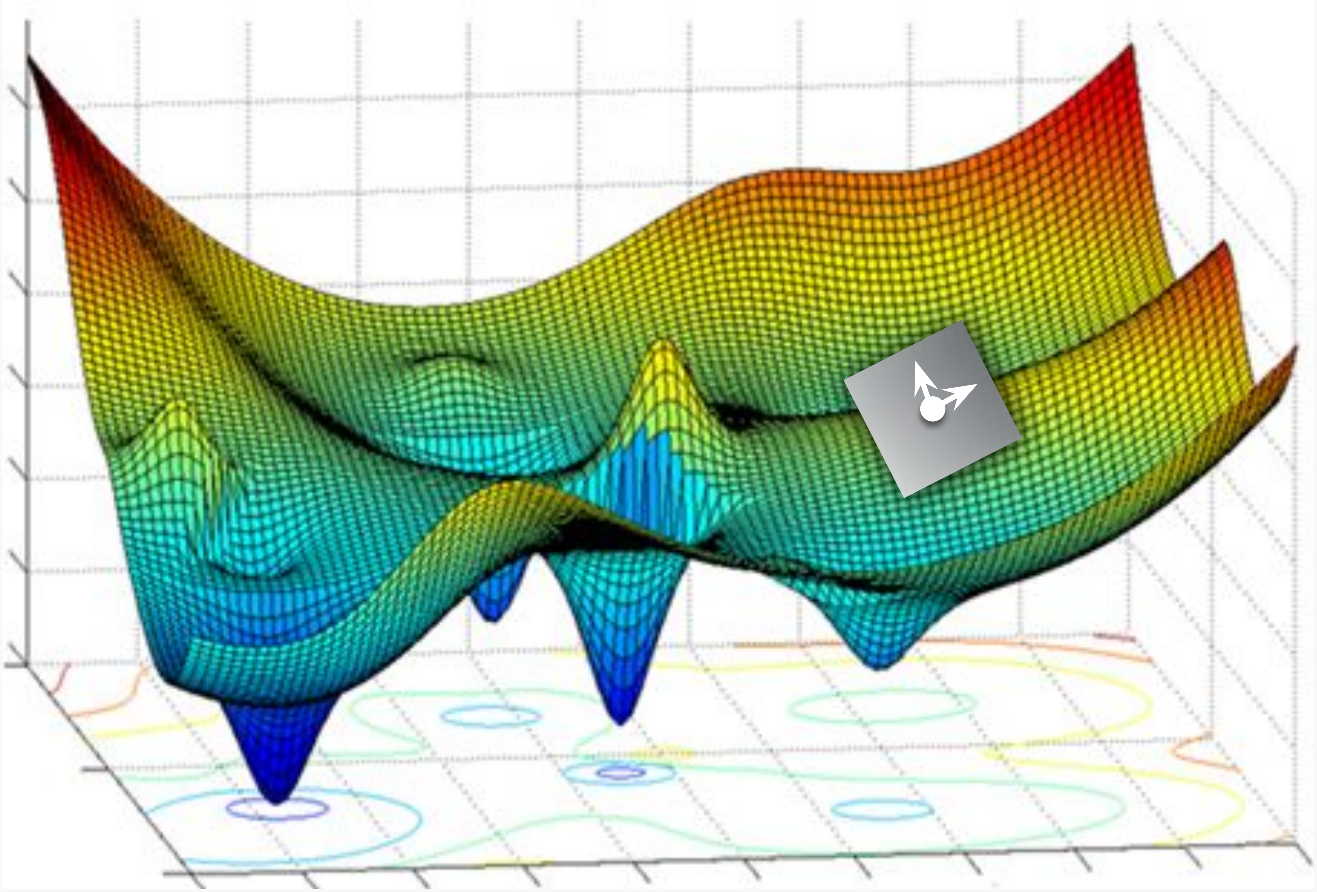
just one line of code!

Now where does this come from?

Gradient descent

(partial) derivatives tell us how much one variable affects another

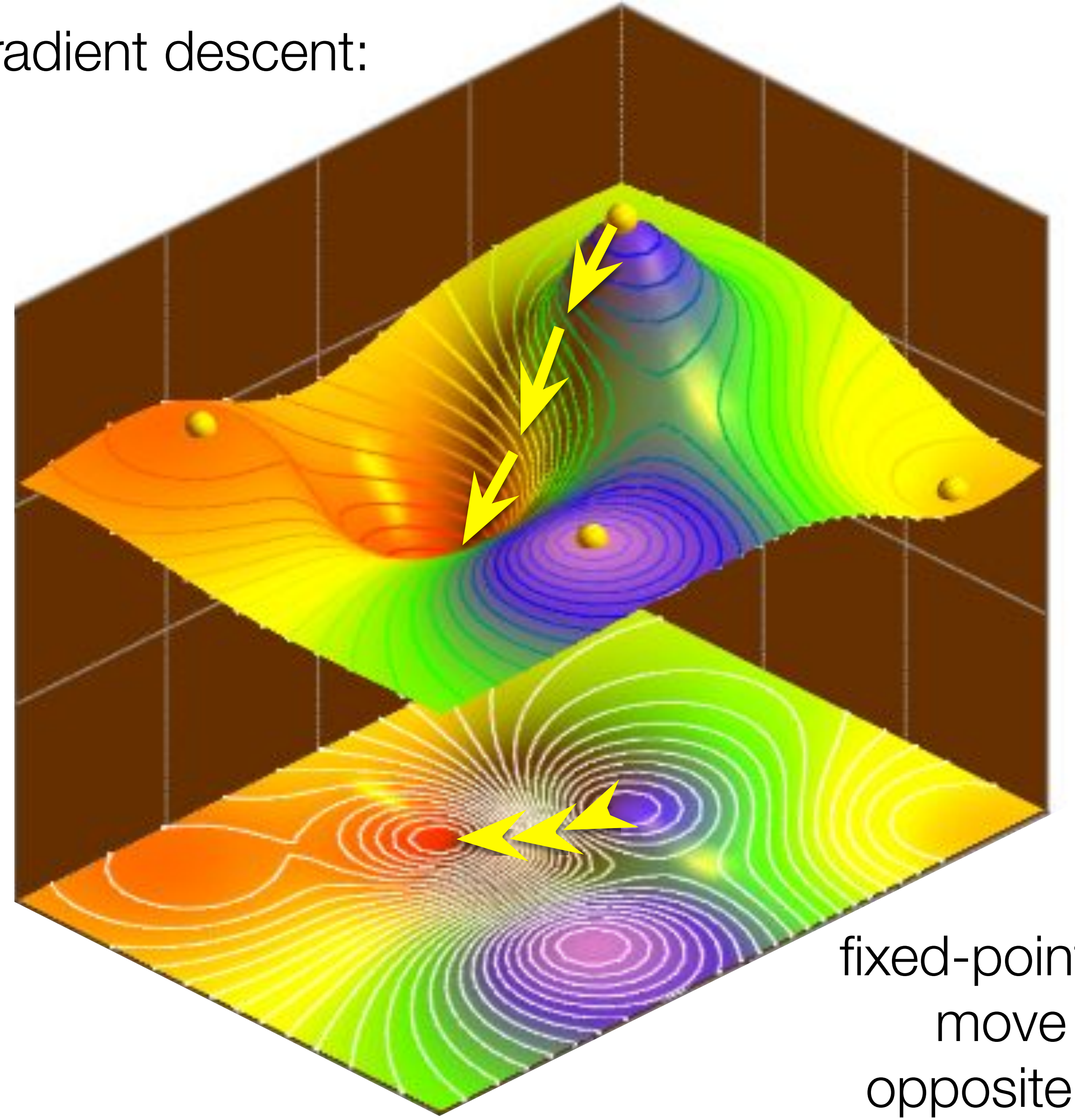
1. Slope of a function:



$$\frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} = \left[\frac{\partial f(\boldsymbol{x})}{\partial x}, \frac{\partial f(\boldsymbol{x})}{\partial y}\right]$$

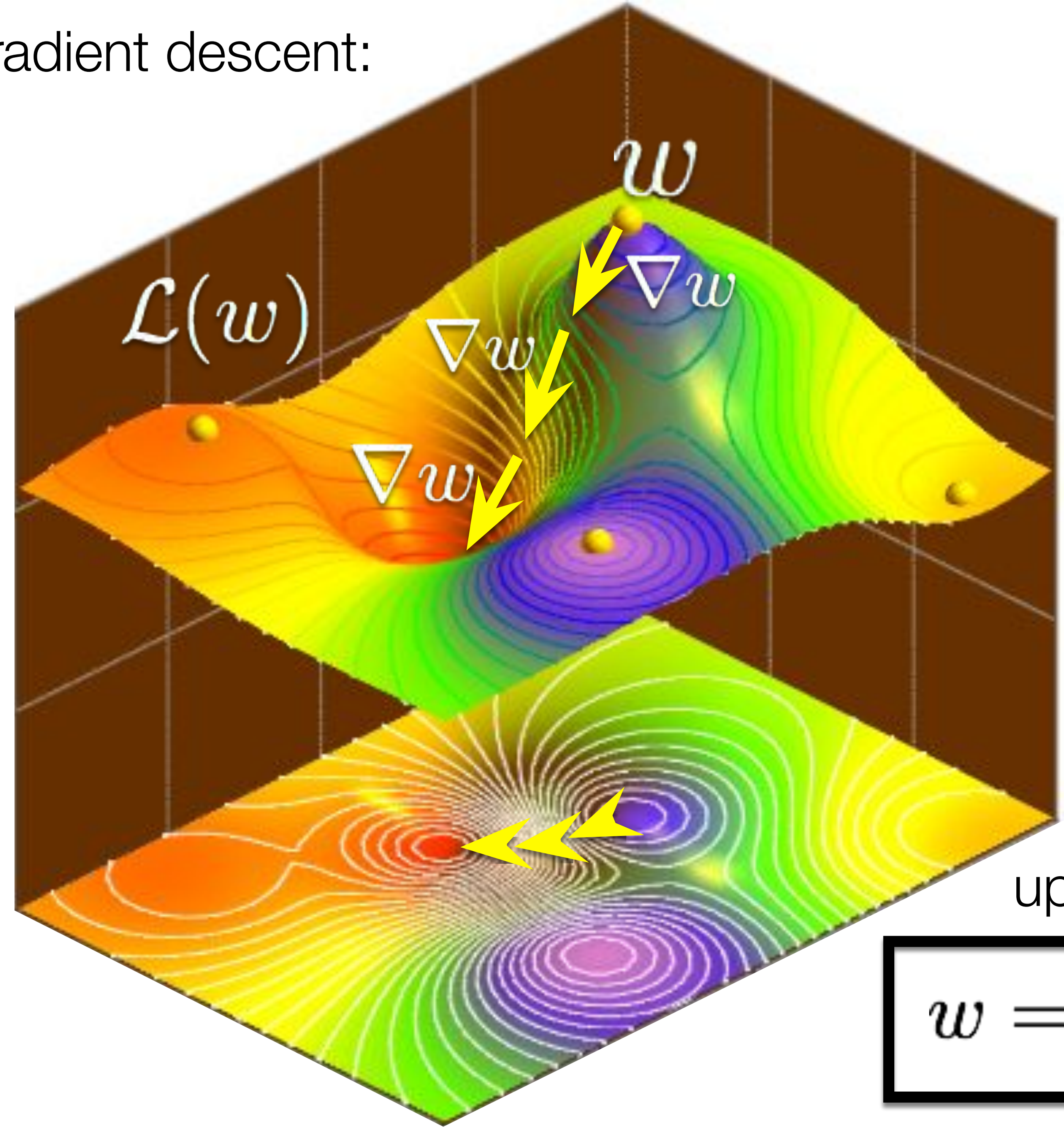
describes the slope around a point

Gradient descent:



Given a
fixed-point on a function,
move in the direction
opposite of the gradient

Gradient descent:



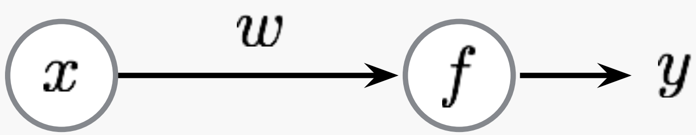
update rule:

$$\boxed{w = w - \nabla w}$$

Backpropagation

back to
the…

World’s Smallest Perceptron!



$$y = wx$$

(a.k.a. line equation, linear regression)

function of **ONE** parameter!

Training the world's smallest perceptron

$$\text{for } n = 1 \dots N$$
$$w = w + \underline{(y_n - \hat{y})x_i};$$

This is just gradient descent, that means...

this should be the gradient of the loss function

Now where does this come from?

$$\frac{d\mathcal{L}}{dw}$$

...is the rate at which **this** will change...

$$\mathcal{L} = \frac{1}{2}(y - \hat{y})^2$$

the loss function

... per unit change of **this**

$$y = wx$$

the weight parameter

Let's compute the derivative...

Compute the derivative

$$
\begin{aligned}
\frac{d\mathcal{L}}{dw} &= \frac{d}{dw}\left\{\frac{1}{2}(y-\hat{y})^2\right\} \\
&= -(y-\hat{y})\frac{dwx}{dw} \\
&= -(y-\hat{y})x = \nabla w
\end{aligned}
$$

just shorthand

That means the weight update for **gradient descent** is:

$$
\begin{aligned}
w &= w - \nabla w \\
&= w + (y-\hat{y})x
\end{aligned}
$$

move in direction of negative gradient

Gradient Descent **(world's smallest perceptron)**

For each sample

1. Predict
   a. Forward pass
   b. Compute Loss
2. Update
   a. Back Propagation
   b. Gradient update

$$\{x_i, y_i\}$$

$$\hat{y} = wx_i$$

$$\mathcal{L}_i = \frac{1}{2}(y_i - \hat{y})^2$$

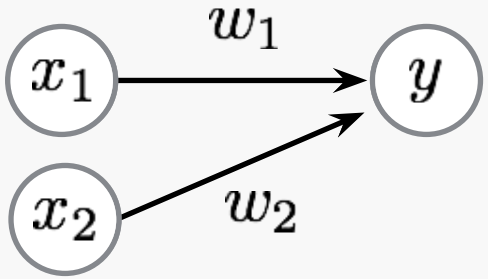
$$\frac{d\mathcal{L}_i}{dw} = -(y_i - \hat{y})x_i = \nabla w$$

$$w = w - \nabla w$$

Training the world's smallest perceptron

$$\text{for } n = 1 \ldots N$$
$$\quad w = w + (y_n - \hat{y})x_i;$$

world's (second) smallest **perceptron**!



function of **two** parameters!

Gradient Descent

For each sample $\{x_i, y_i\}$

1. Predict
   a. Forward pass
   b. Compute Loss
2. Update
   a. Back Propagation
   b. Gradient update

we just need to compute partial derivatives for this network

Derivative computation

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial w_1} &= \frac{\partial}{\partial w_1}\left\{\frac{1}{2}(y-\hat{y})^2\right\} \\ &= -(y-\hat{y})\frac{\partial \hat{y}}{\partial w_1} \\ &= -(y-\hat{y})\frac{\partial \sum_i w_i x_i}{\partial w_1} \\ &= -(y-\hat{y})\frac{\partial w_1 x_1}{\partial w_1} \\ &= -(y-\hat{y})x_1 = \nabla w_1\end{aligned}$$

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial w_2} &= \frac{\partial}{\partial w_2}\left\{\frac{1}{2}(y-\hat{y})^2\right\} \\ &= -(y-\hat{y})\frac{\partial \hat{y}}{\partial w_2} \\ &= -(y-\hat{y})\frac{\partial \sum_i w_i x_i}{\partial w_1} \\ &= -(y-\hat{y})\frac{\partial w_2 x_2}{\partial w_2} \\ &= -(y-\hat{y})x_2 = \nabla w_2\end{aligned}$$

Why do we have partial derivatives now?

Derivative computation

$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial}{\partial w_1}\left\{\frac{1}{2}(y-\hat{y})^2\right\}$$
$$= -(y-\hat{y})\frac{\partial \hat{y}}{\partial w_1}$$
$$= -(y-\hat{y})\frac{\partial \sum_i w_i x_i}{\partial w_1}$$
$$= -(y-\hat{y})\frac{\partial w_1 x_1}{\partial w_1}$$
$$\boxed{= -(y-\hat{y})x_1 = \nabla w_1}$$

$$\frac{\partial \mathcal{L}}{\partial w_2} = \frac{\partial}{\partial w_2}\left\{\frac{1}{2}(y-\hat{y})^2\right\}$$
$$= -(y-\hat{y})\frac{\partial \hat{y}}{\partial w_2}$$
$$= -(y-\hat{y})\frac{\partial \sum_i w_i x_i}{\partial w_1}$$
$$= -(y-\hat{y})\frac{\partial w_2 x_2}{\partial w_2}$$
$$\boxed{= -(y-\hat{y})x_2 = \nabla w_2}$$

Gradient Update

$$w_1 = w_1 - \eta \nabla w_1$$
$$\boxed{= w_1 + \eta(y-\hat{y})x_1}$$

$$w_2 = w_2 - \eta \nabla w_2$$
$$\boxed{= w_2 + \eta(y-\hat{y})x_2}$$

Gradient Descent

For each sample $\{x_i, y_i\}$

1. Predict

   a. Forward pass $\hat{y} = f_{\text{MLP}}(x_i; \theta)$

   b. Compute Loss $\mathcal{L}_i = \frac{1}{2}(y_i - \hat{y})$ (side computation to track loss. not needed for backprop)

2. Update

   two lines now

   a. Back Propagation

   $$\nabla w_{1i} = -(y_i - \hat{y})x_{1i}$$
   $$\nabla w_{2i} = -(y_i - \hat{y})x_{2i}$$

   b. Gradient update

   $$w_{1i} = w_{1i} + \eta(y - \hat{y})x_{1i}$$
   $$w_{2i} = w_{2i} + \eta(y - \hat{y})x_{2i}$$

(adjustable step size)

We haven’t seen a lot of ‘propagation’ yet because our perceptrons only had one layer…