

# **Stochastic Gradient Descent & Optimization Recap**

INFO-4604, Applied Machine Learning  
University of Colorado Boulder

**September 30, 2020**

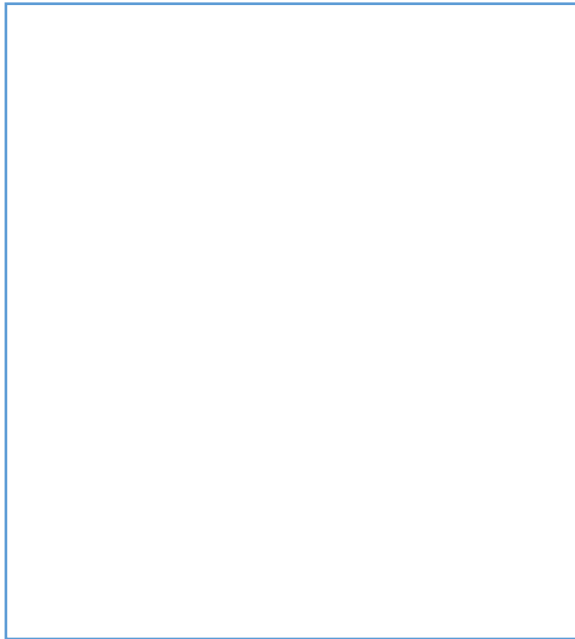
Prof. Abe Handler

# Where are we?

Reminder about what is going on over past two weeks

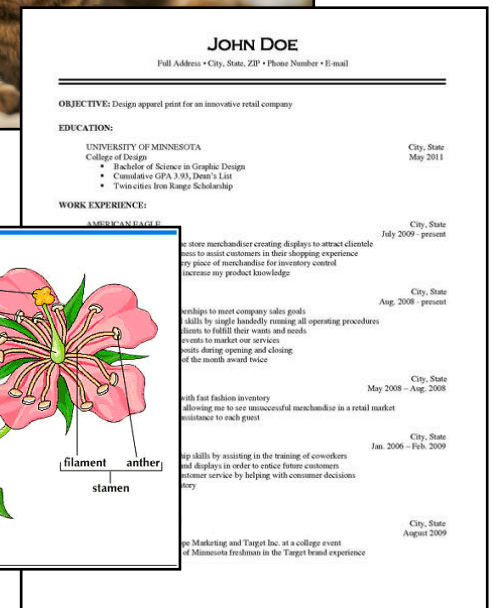
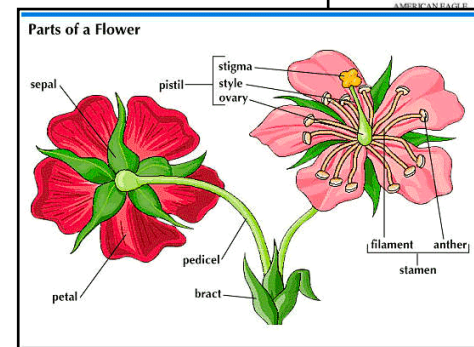
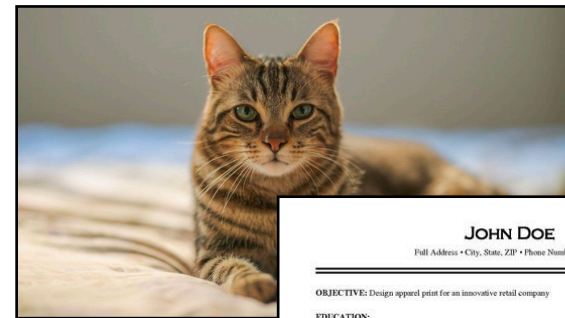
# Cast of characters ...

Features



$X$

Data



# Cast of characters ...

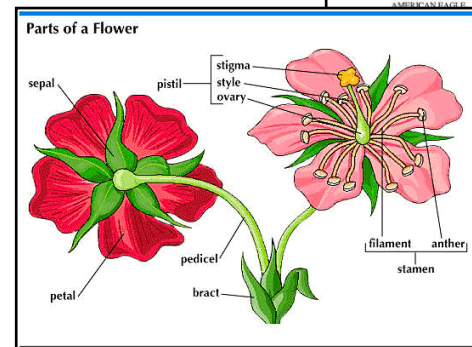
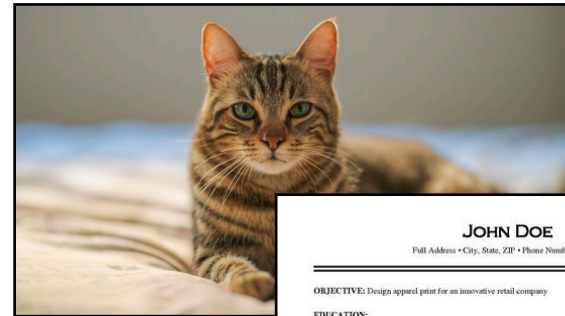
Features

Data

$x_i$

One data point

$X$



**JOHN DOE**  
Full Address • City, State, ZIP • Phone Number • E-mail

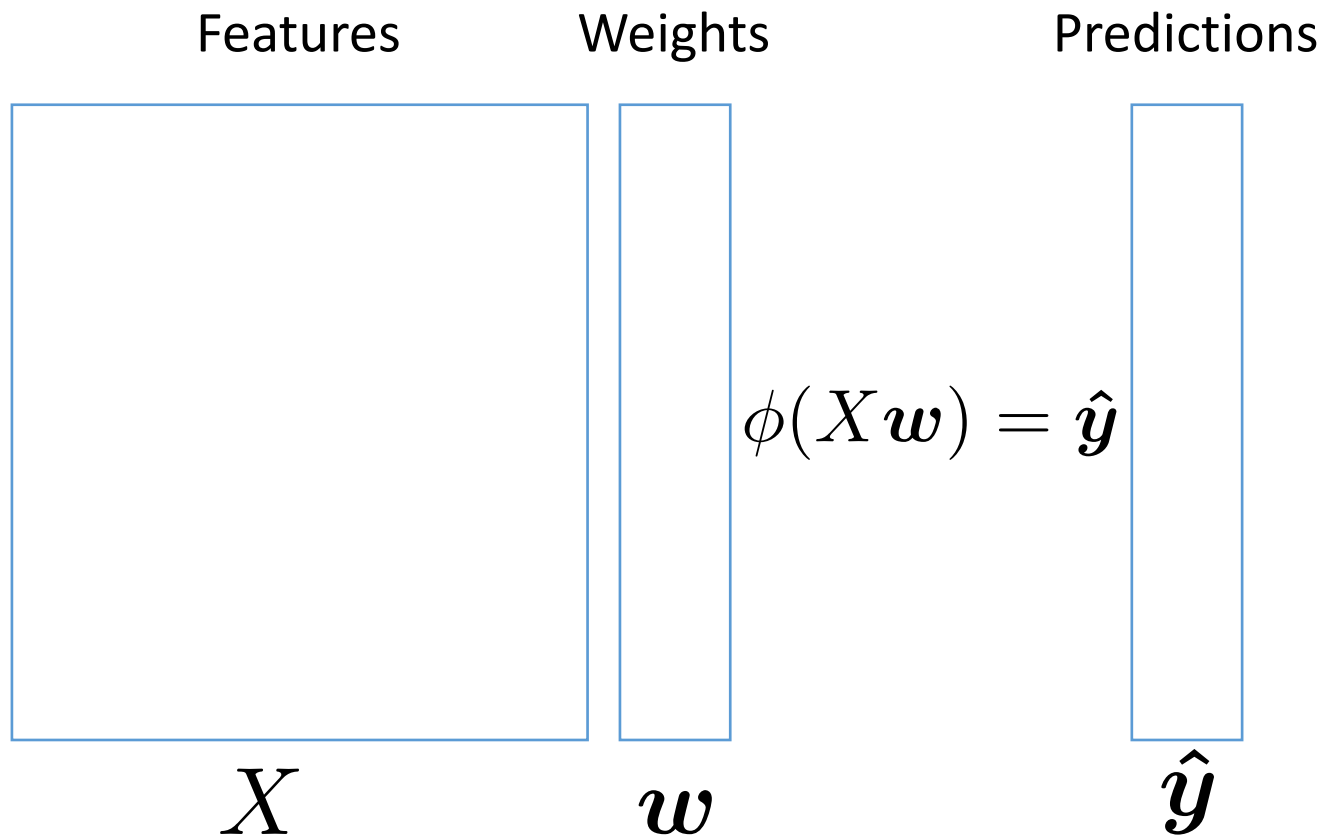
---

**OBJECTIVE:** Design apparel print for an innovative retail company

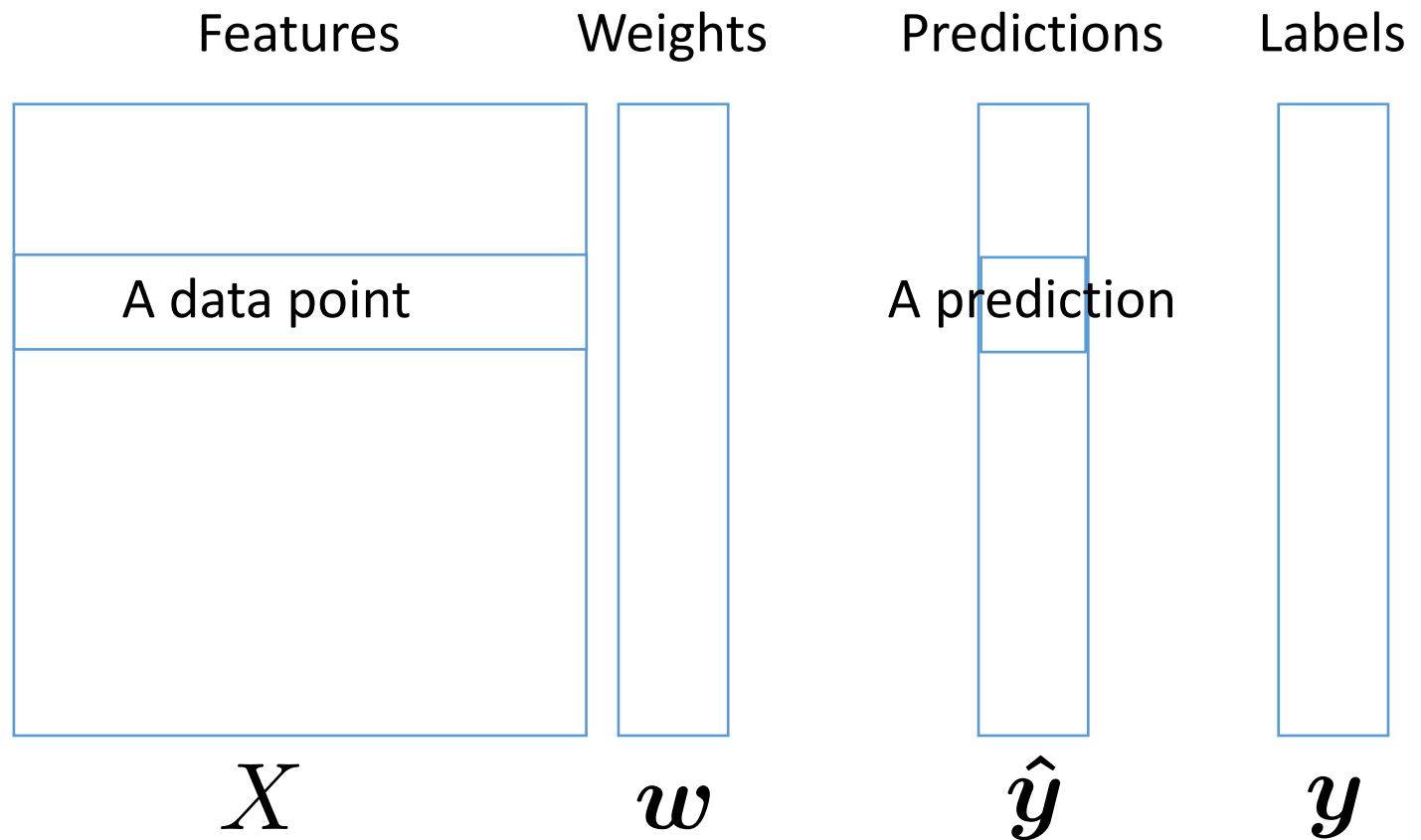
**EDUCATION:**  
UNIVERSITY OF MINNESOTA  
College of Design  
• Bachelor of Science in Graphic Design  
• Cumulative GPA 3.93, Dean's List  
• Twin cities Iron Range Scholarship

**WORK EXPERIENCE:**  
**AMERICAN EAGLE** City, State  
as store merchandiser creating displays to attract clientele  
assist to assist customers in their shopping experience  
every piece of merchandise for inventory control  
increase my product knowledge July 2009 - present  
City, State  
Aug. 2008 - present  
overlooks to meet company sales goals  
skills by single handily running all operating procedures  
desires to fulfill their wants and needs  
events to market our services  
month during opening and closing  
of the month award twice  
City, State  
May 2008 - Aug. 2008  
with fast fashion inventory  
allowing me to see unsuccessful merchandise in a retail market  
assistance to each guest  
City, State  
Jan. 2006 - Feb. 2009  
my skills by assisting in the training of co-workers  
and displays in order to entice future customers  
customer service by helping with consumer decisions  
they  
City, State  
August 2009  
ope Marketing and Target Inc. at a collage event  
of Minnesota freshmen in the Target brand experience

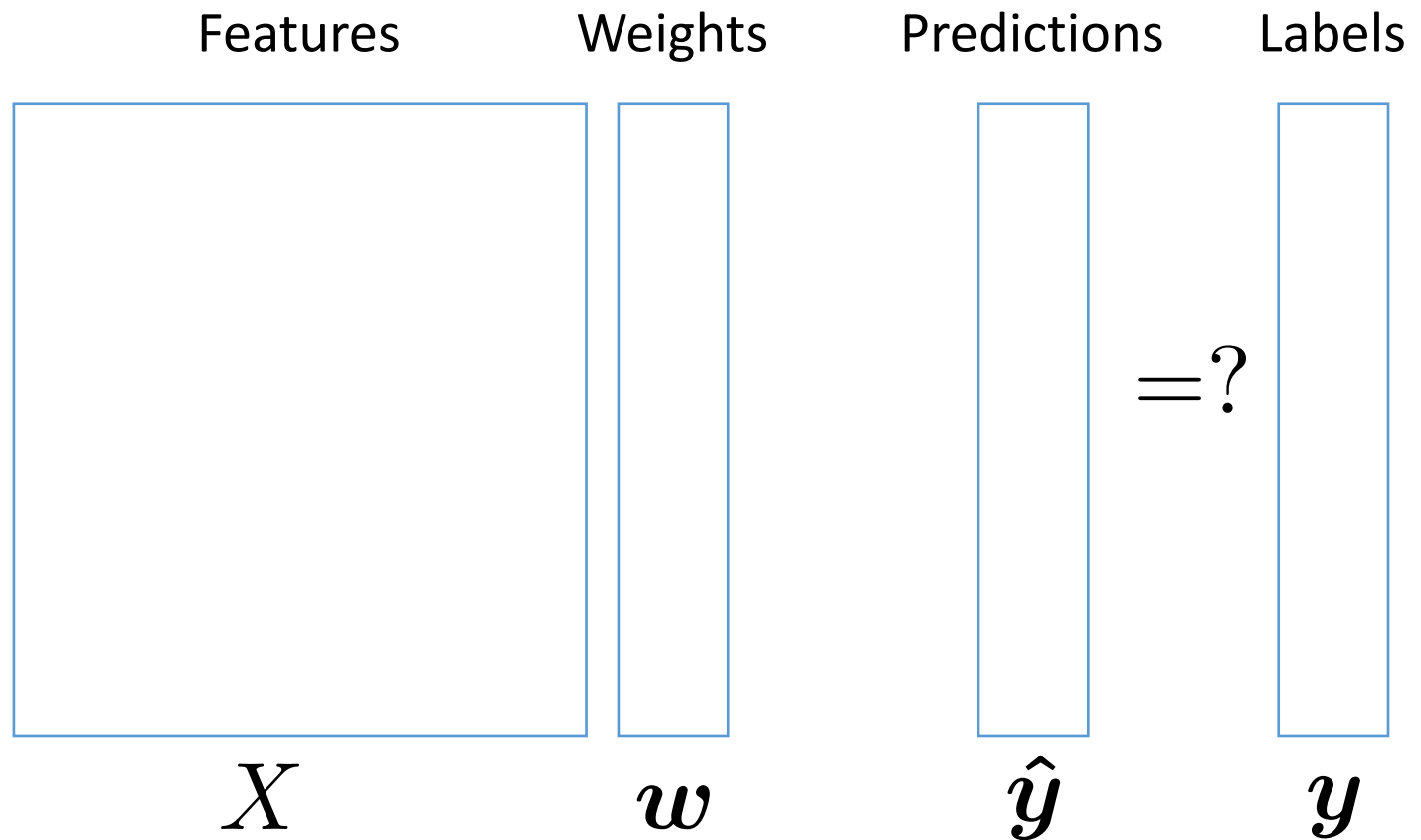
A linear combination of weights and features  
returns predictions



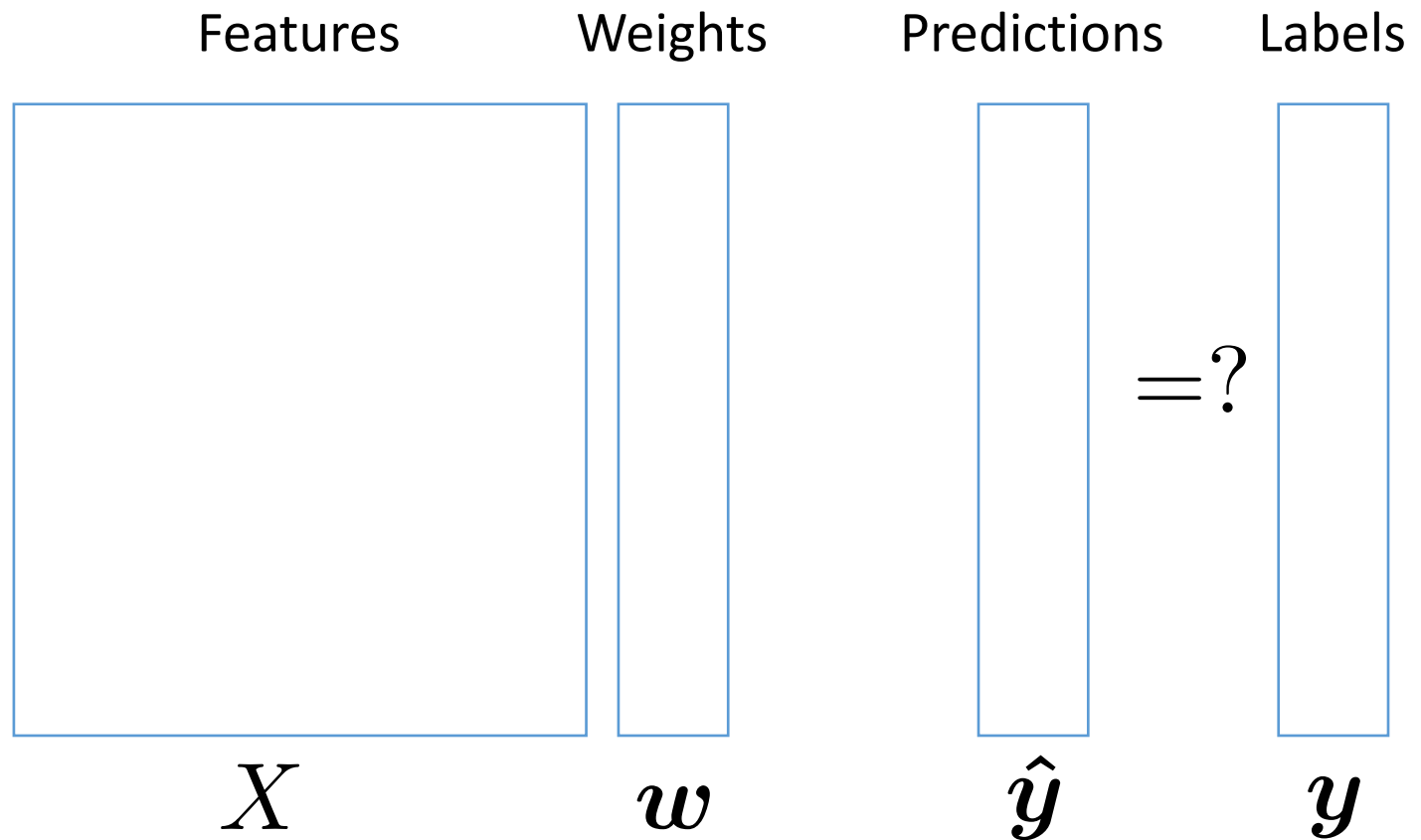
# Focus in on one single instance



# Are the predictions any good?

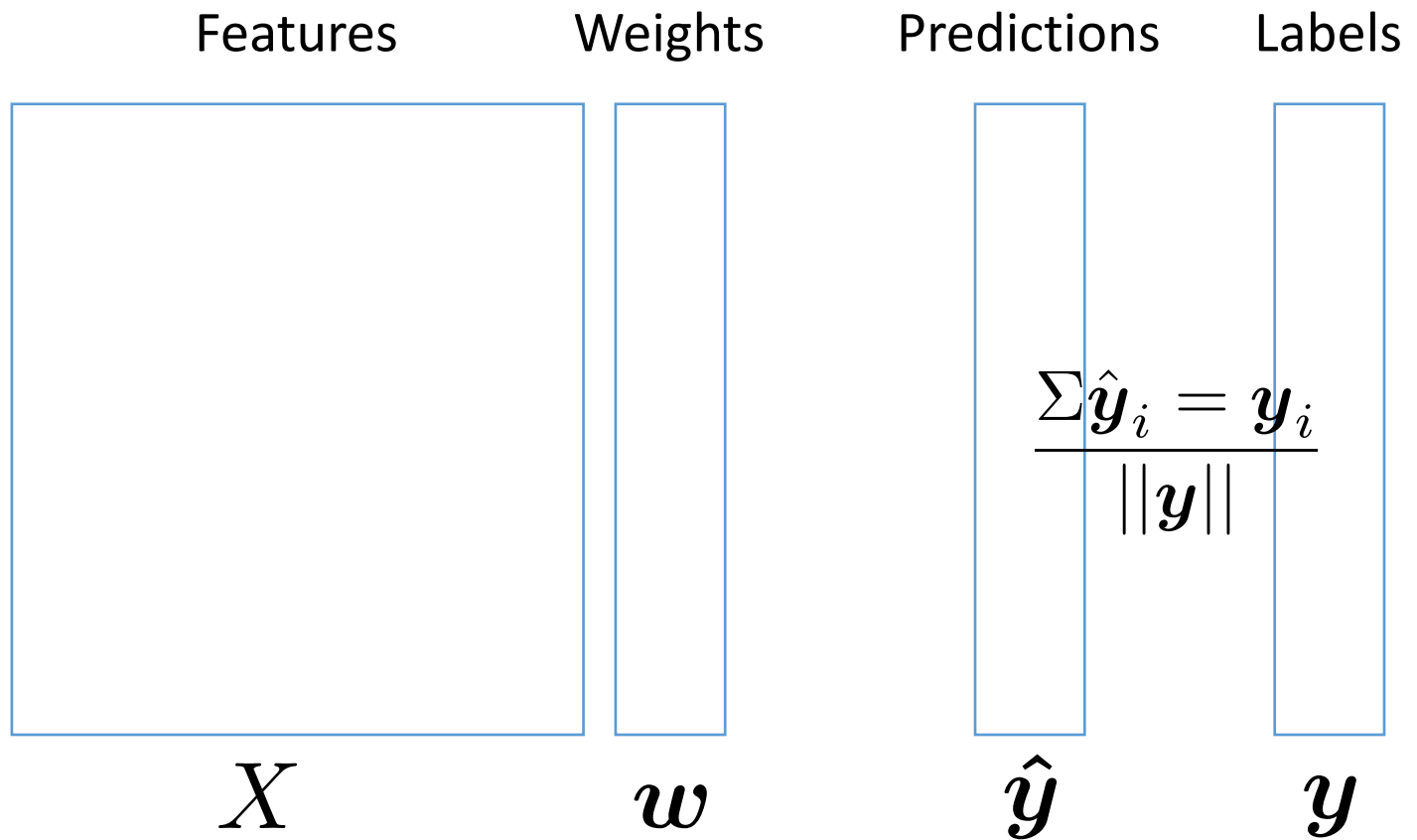


# Are the predictions any good?





# Accuracy



# Loss

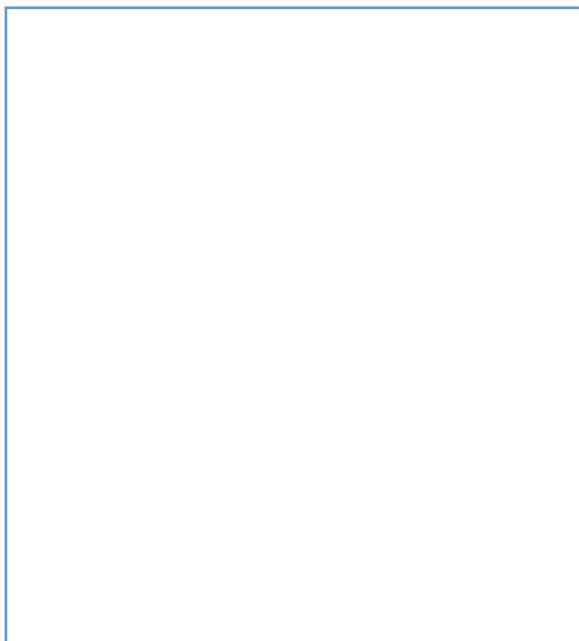
$$L(X, w, y)$$

Features

Weights

Predictions

Labels



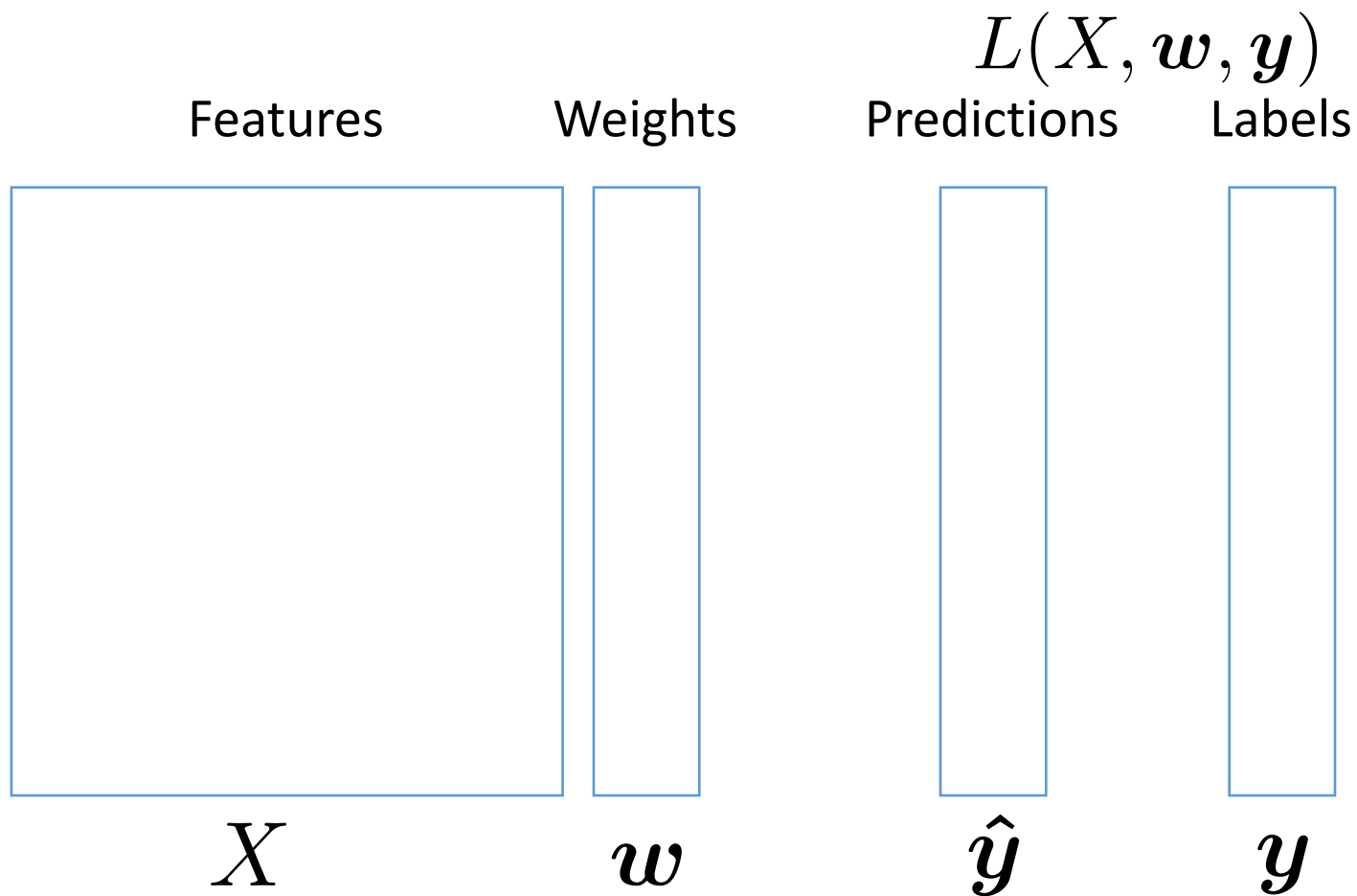
$X$

$w$

$\hat{y}$

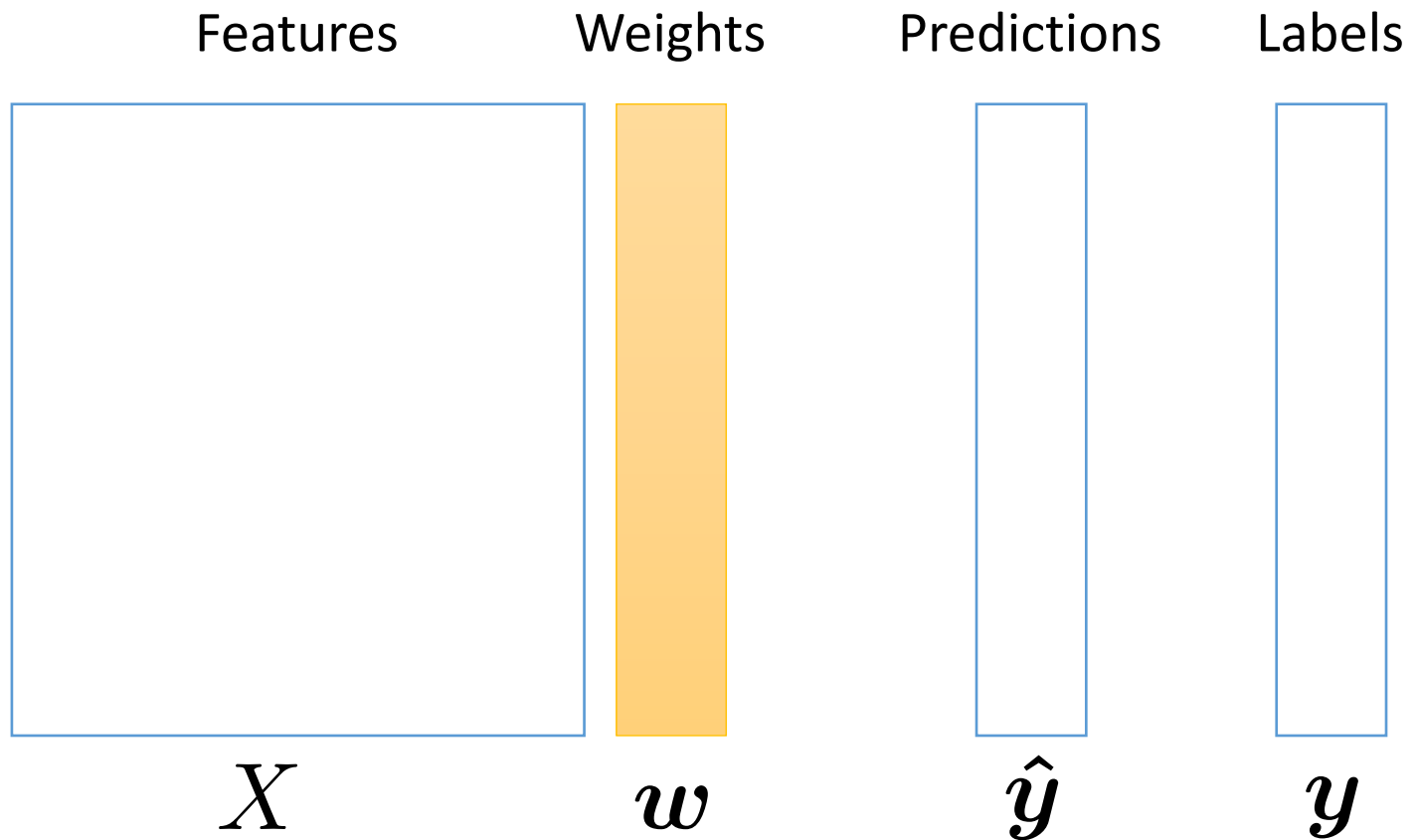
$y$

Loss: how far are the predictions from the labels?

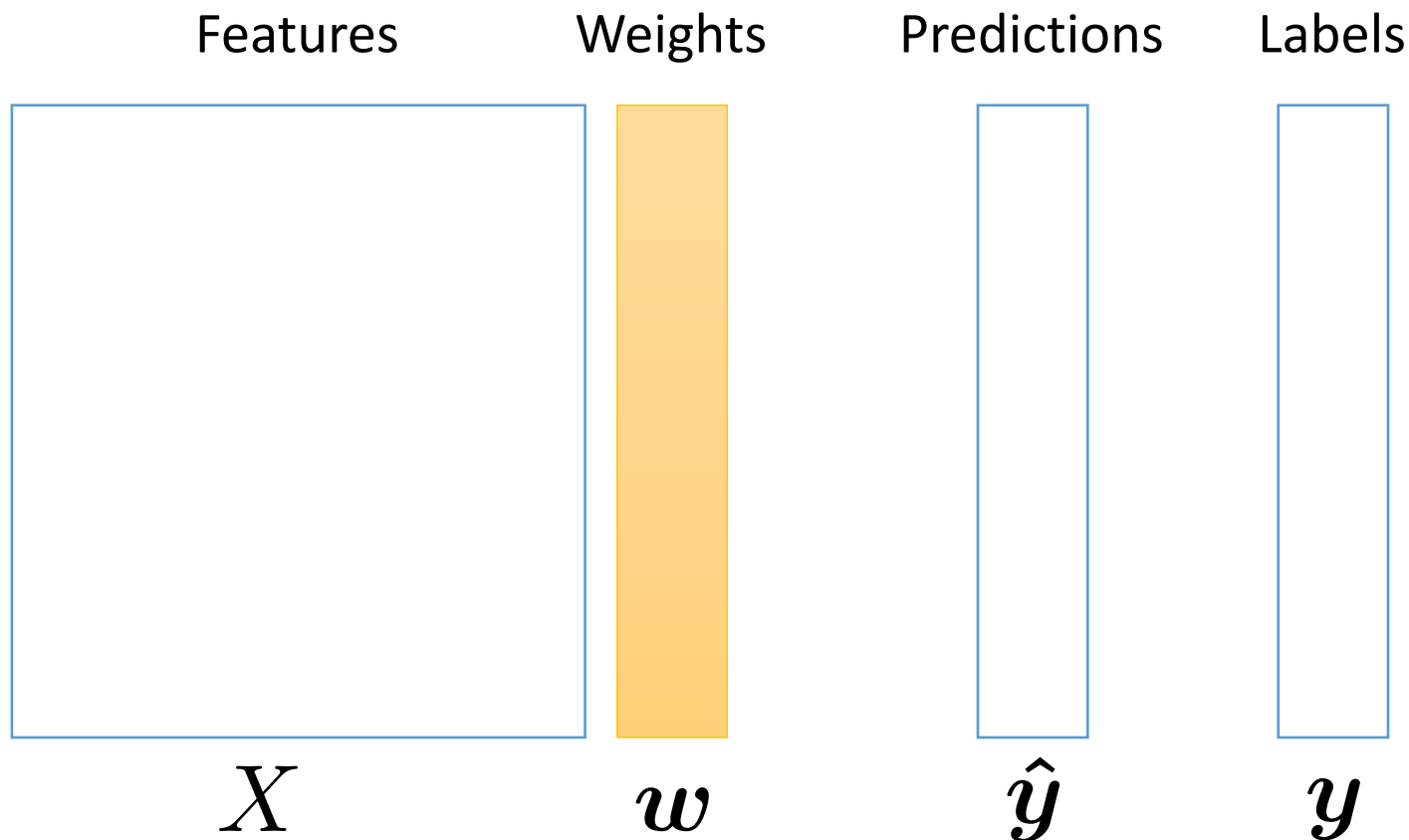


“Optimization”

“Optimize” (aka adjust) weights so predictions are more like the labels

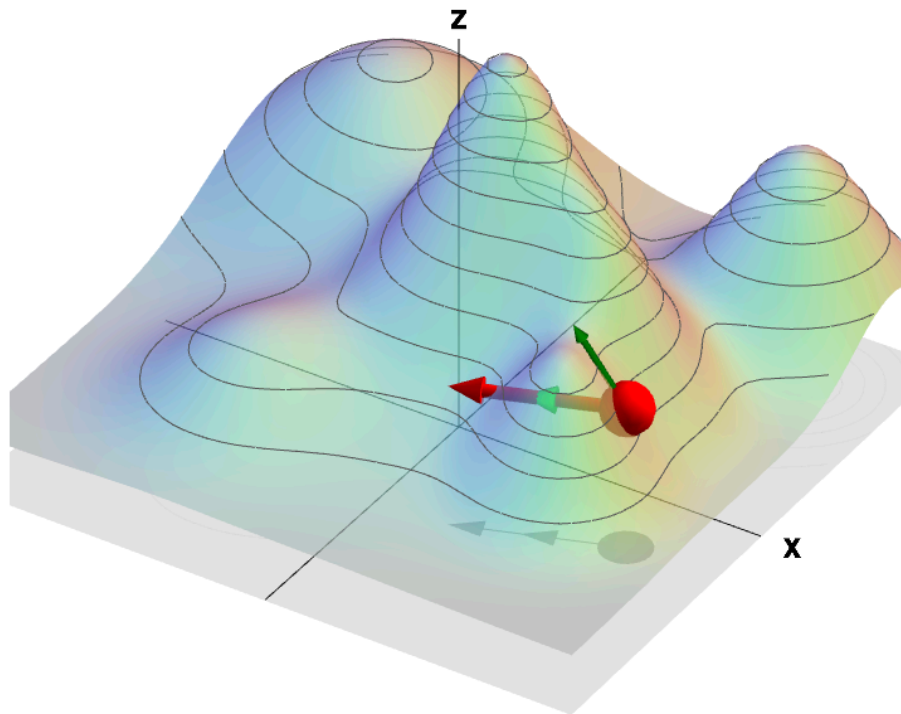


# Adjust weights to minimize loss

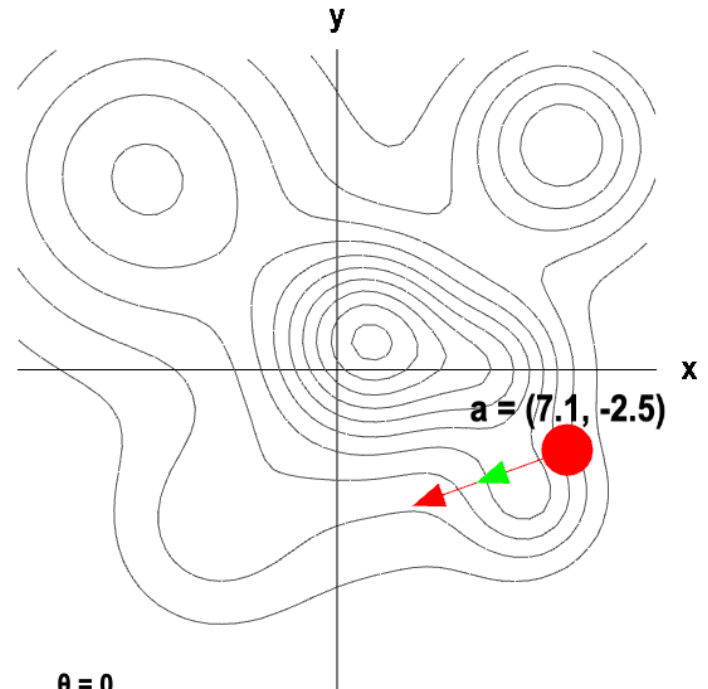


So far, we have seen one  
method for minimizing loss

# Gradient descent



$\theta = 0$   
 $u = (-0.94, -0.34)$   
 $a = (7.1, -2.5)$   
 $\nabla f(a) = (-1.59, -0.57)$   
 $D_u f(a) = 1.69$   
 $|\nabla f(a)| = 1.69$

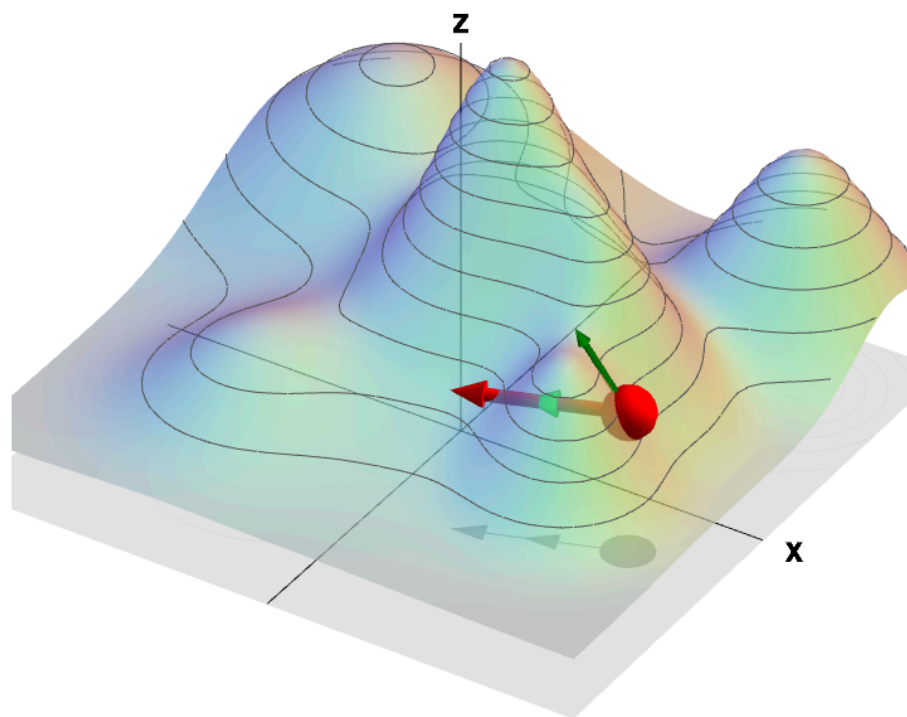


$\theta = 0$   
 $u = (-0.94, -0.34)$   
 $\nabla f(a) = (-1.59, -0.57)$   
 $D_u f(a) = 1.69$   
 $|\nabla f(a)| = 1.69$   
 $f(a) = 4.67$

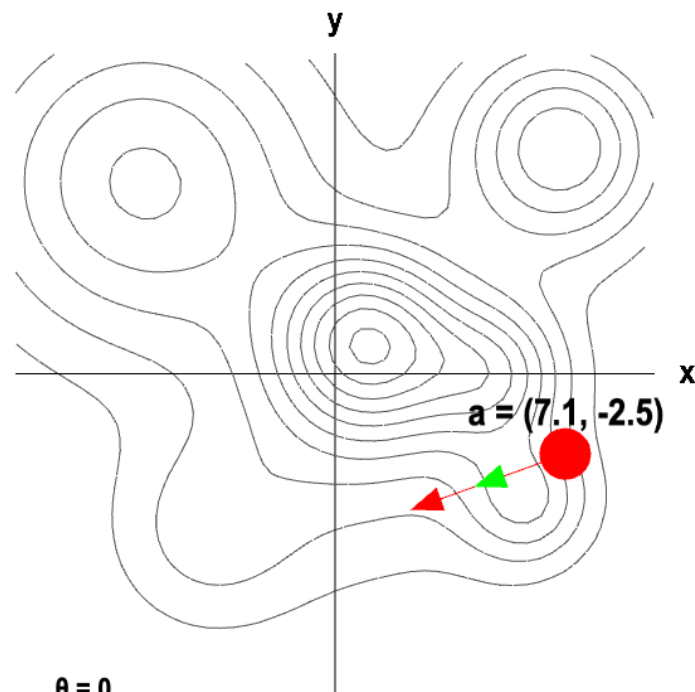
[https://mathinsight.org/applet/gradient\\_directional\\_derivative\\_mountain](https://mathinsight.org/applet/gradient_directional_derivative_mountain)



Reminder: this picture should be upside down



$\theta = 0$   
 $u = (-0.94, -0.34)$   
 $a = (7.1, -2.5)$   
 $\nabla f(a) = (-1.59, -0.57)$   
 $D_u f(a) = 1.69$   
 $|\nabla f(a)| = 1.69$



$\theta = 0$   
 $u = (-0.94, -0.34)$   
 $\nabla f(a) = (-1.59, -0.57)$   
 $D_u f(a) = 1.69$   
 $|\nabla f(a)| = 1.69$   
 $f(a) = 4.67$

[https://mathinsight.org/applet/gradient\\_directional\\_derivative\\_mountain](https://mathinsight.org/applet/gradient_directional_derivative_mountain)

# Gradient Descent

1. Initialize the parameters  $\mathbf{w}$  to some guess (usually all zeros, or random values)
2. Update the parameters:  
$$\mathbf{w} = \mathbf{w} - \eta \nabla L(\mathbf{w})$$
3. Repeat steps 2-3 until  $\nabla L(\mathbf{w})$  is close to zero.

# Computing the gradient

1. Initialize the parameters  $\mathbf{w}$  to some guess (usually all zeros, or random values)
2. Update the parameters:  
$$\mathbf{w} = \mathbf{w} - \eta \nabla L(\mathbf{w})$$
3. Repeat steps 2-3 until  $\nabla L(\mathbf{w})$  is close to zero.

# Computing the gradient

1. Initialize the parameters  $\mathbf{w}$  to some guess (usually all zeros, or random values)
2. Update the parameters:  
$$\mathbf{w} = \mathbf{w} - \eta \nabla L(\mathbf{w})$$

Computing the gradient is expensive!

$$\nabla L(\mathbf{w}) = dL_0/D\mathbf{w} + dL_1/D\mathbf{w} \dots dL_N/D\mathbf{w}$$

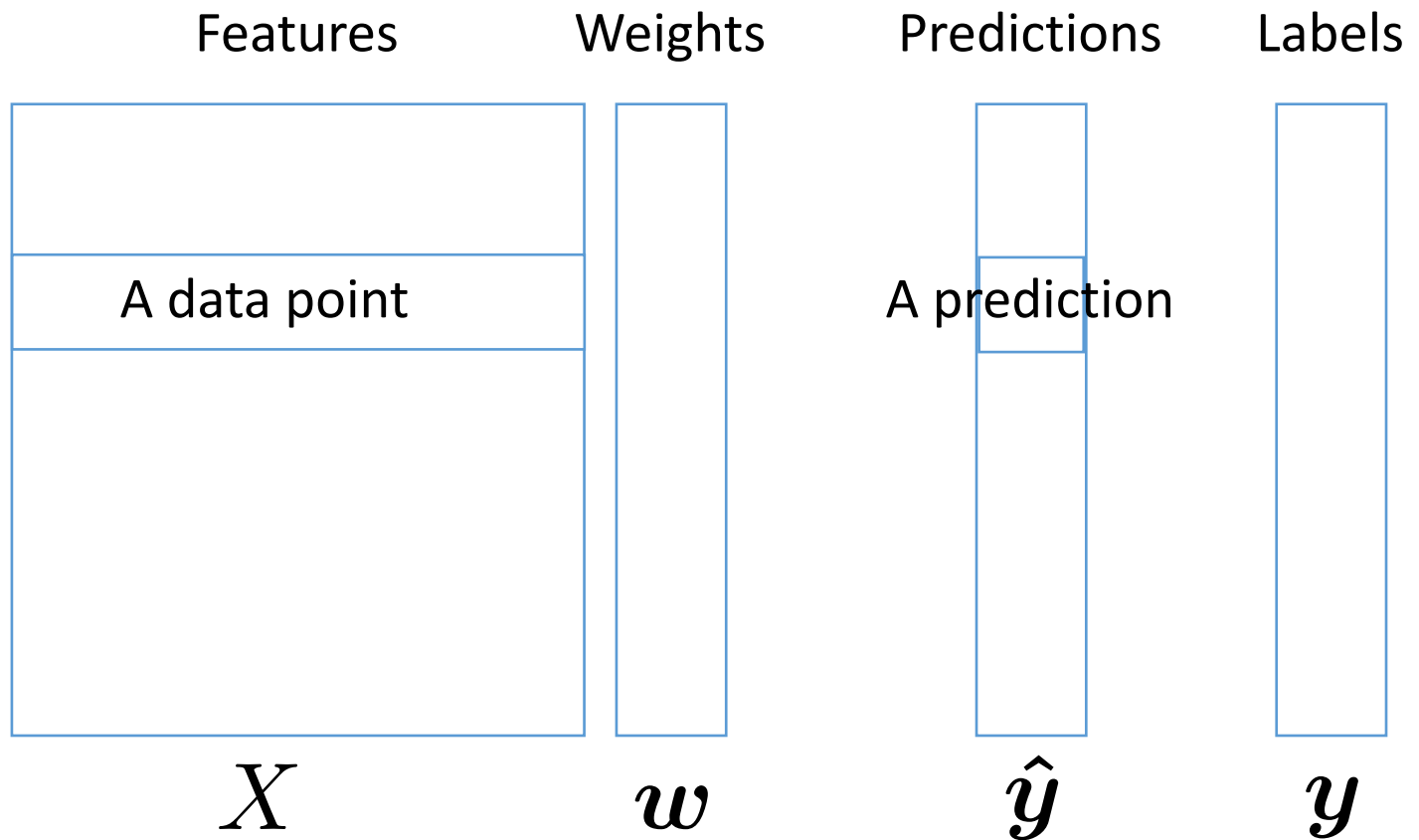
# Stochastic Gradient Descent

A variant of gradient descent makes updates using an approximate of the gradient that is only based on one instance at a time.

$$\nabla L(w) = dL_0/Dw + dL_1/Dw \dots dL_N/Dw$$

$$\nabla L(w) \approx dL_i/Dw$$

# Recall one instance



# Stochastic Gradient Descent

General algorithm for SGD:

1. Iterate through the instances in a random order
  - a) For each instance  $x_i$ , update the weights based on the gradient of the loss for that instance only:

$$\mathbf{w} = \mathbf{w} - \eta \nabla L_i(\mathbf{w}; \mathbf{x}_i)$$

The gradient for one instance's loss is an approximation to the true gradient

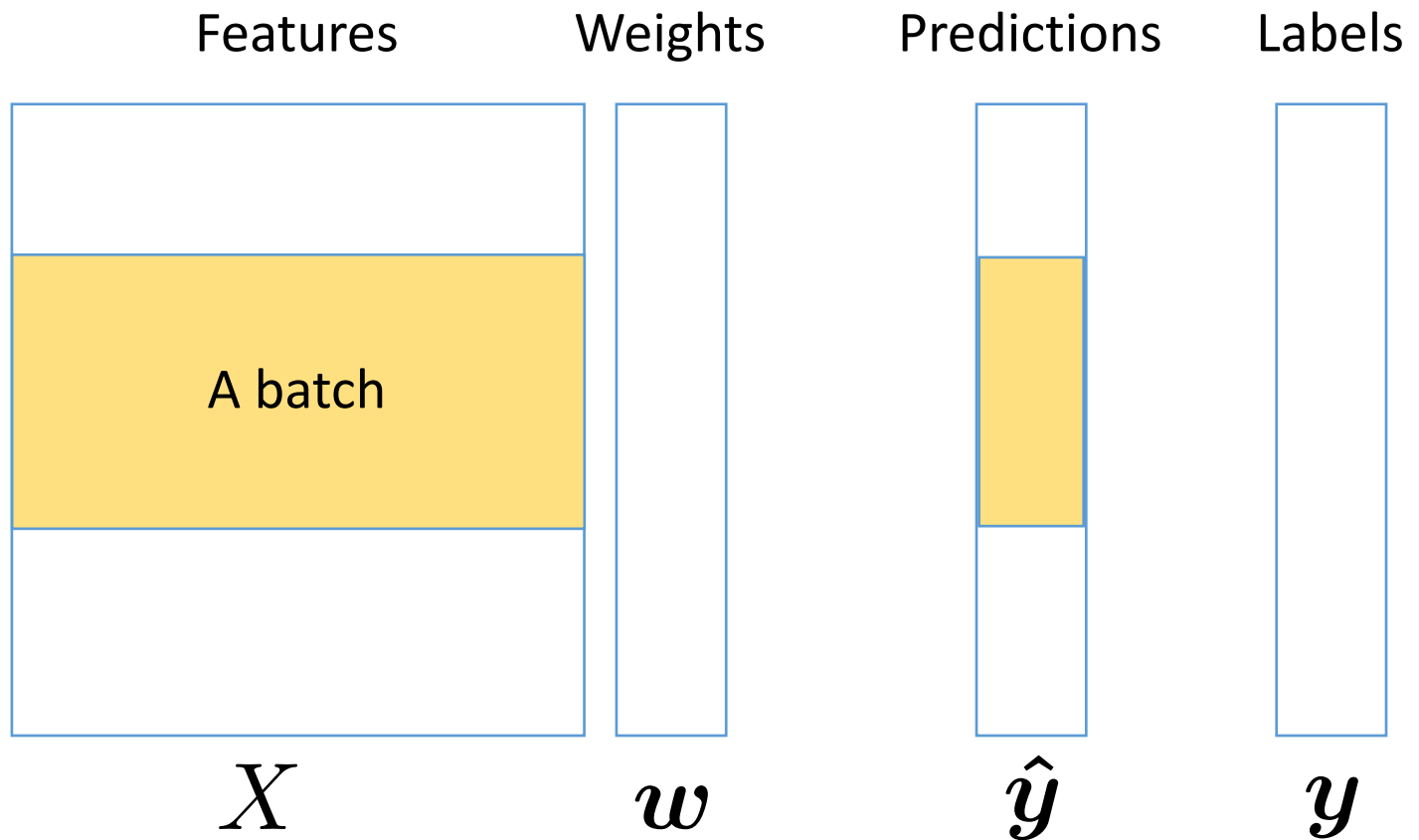
- stochastic = random  
The *expected* gradient is the true gradient

Like going from NY to LA asking for directions one person at a time...

- Dave Blei



# One point can be noisy



# Minibatch SGD

General algorithm for SGD:

1. Iterate through the instances in a random order
  - a) For each instance  $x_i$ , update the weights based on the gradient of the loss for **the batch**
  - b)  $\mathbf{w} = \mathbf{w} - \eta \nabla L_B(\mathbf{w}; \mathbf{x}_B)$

The gradient for one batch's loss is an approximation to the true gradient