Stochastic Gradient Descent & Optimization Recap

INFO-4604, Applied Machine Learning

University of Colorado Boulder

September 30, 2020

Prof. Abe Handler

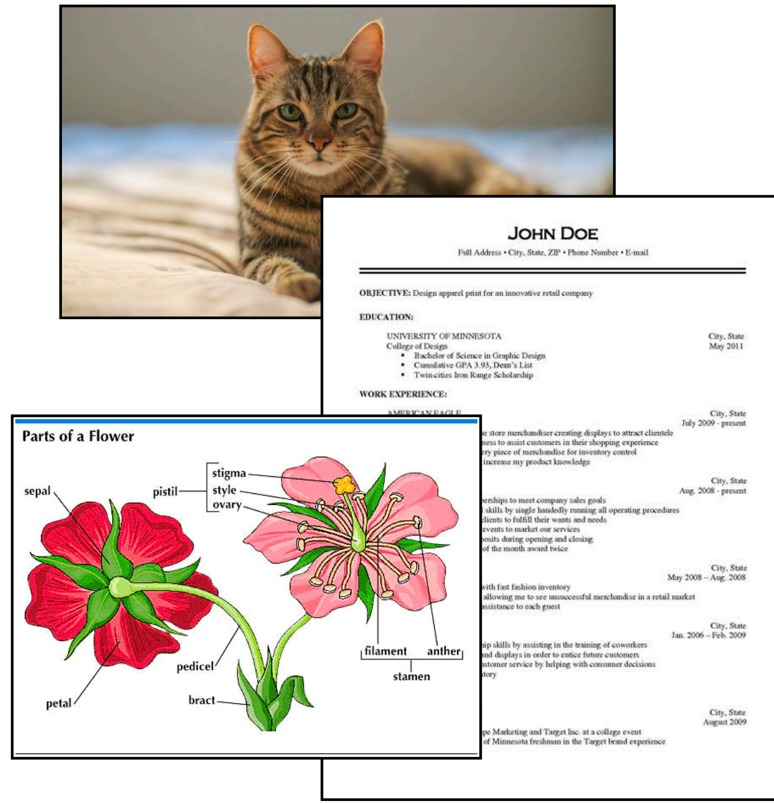
Where are we?

Reminder about what is going on over past two weeks

Cast of characters ...

Features

Data



X

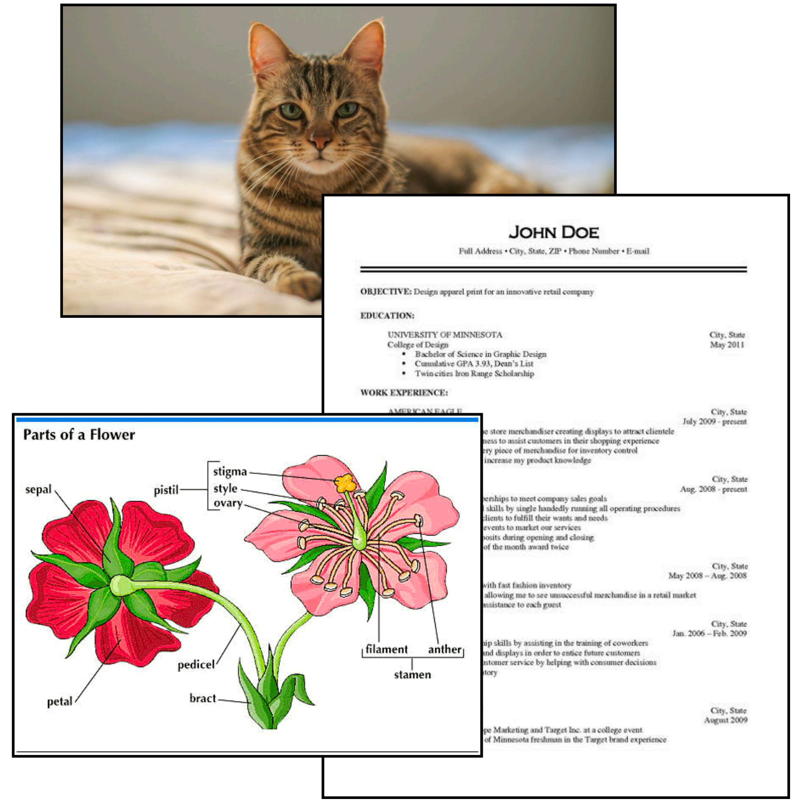
Cast of characters …

Features

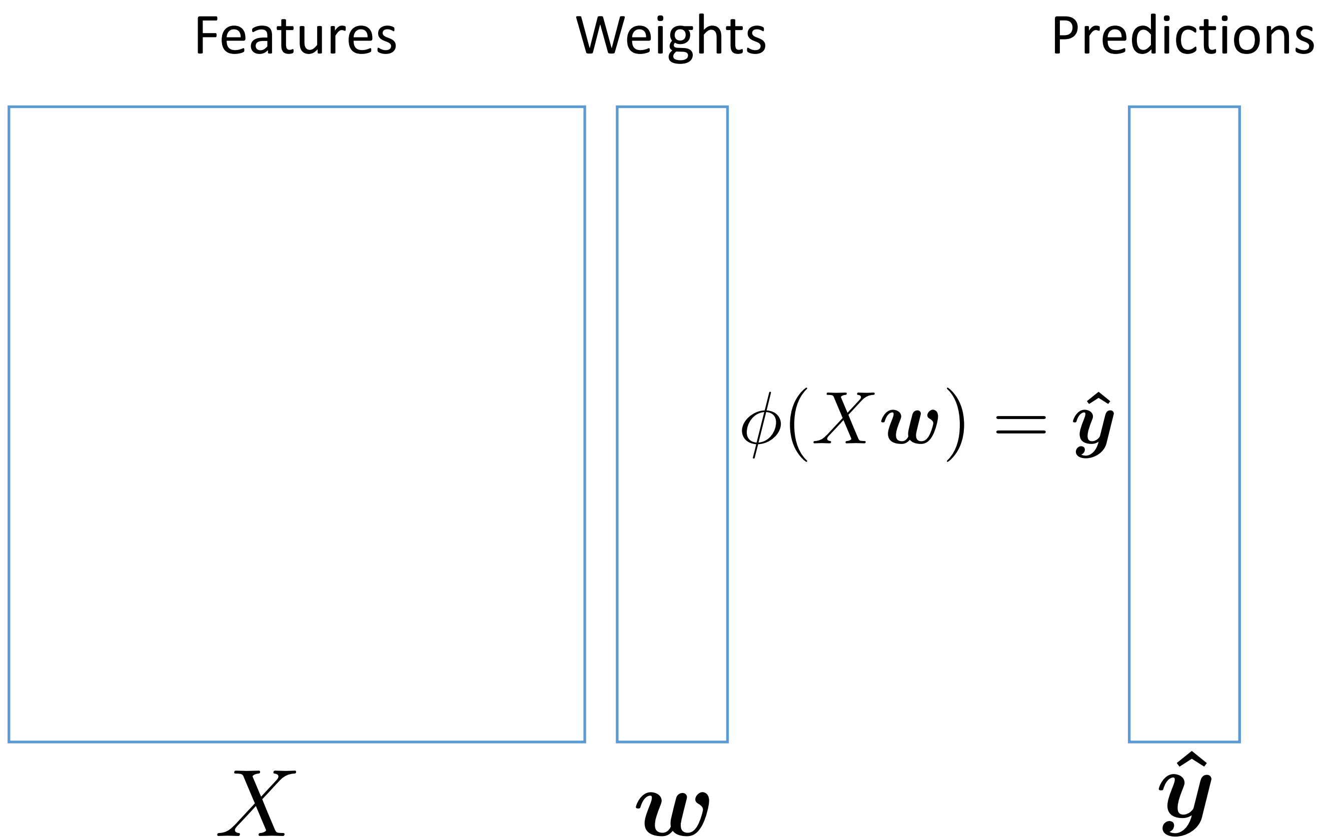
x_i One data point

X

Data

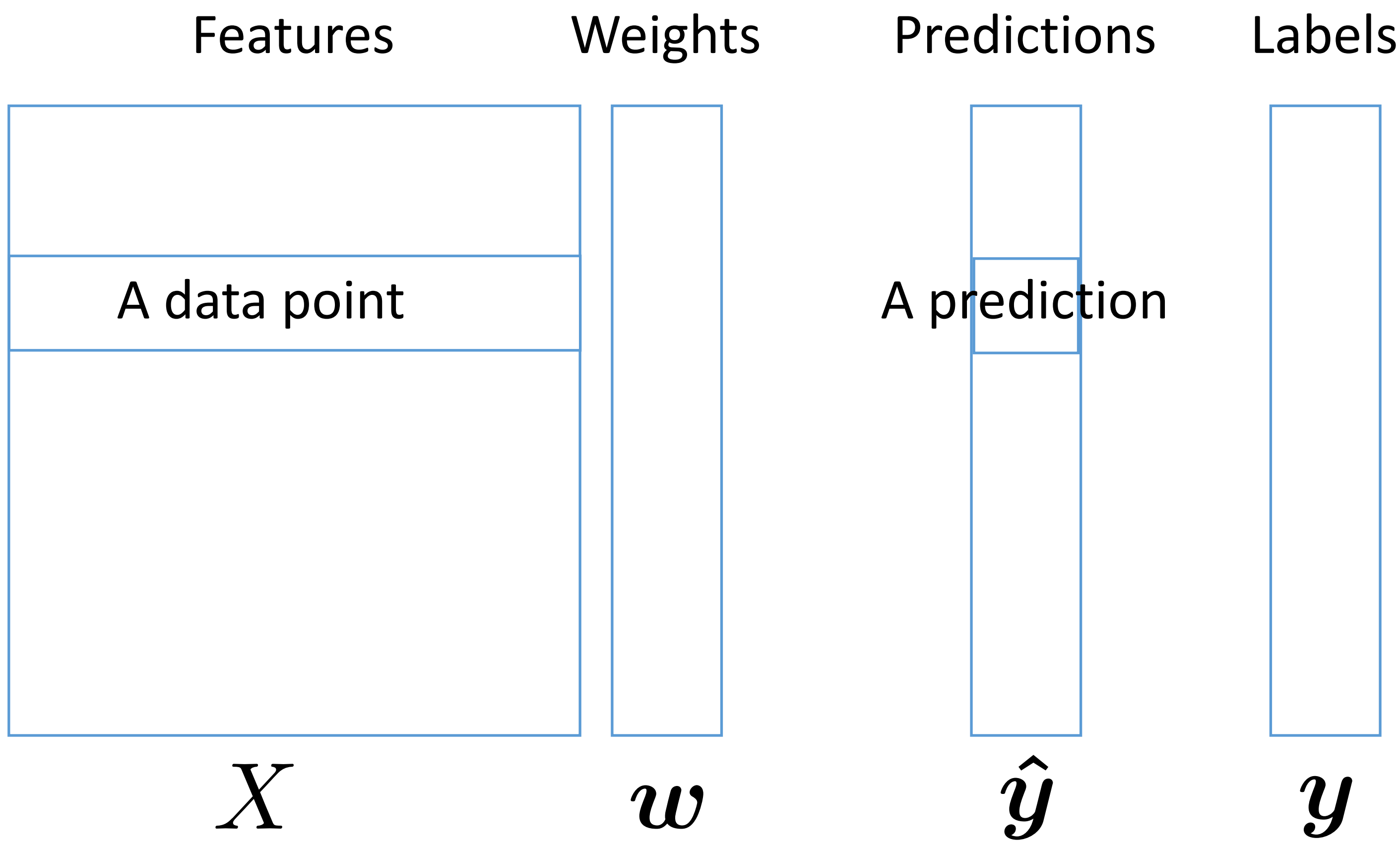


A linear combination of weights and features returns predictions

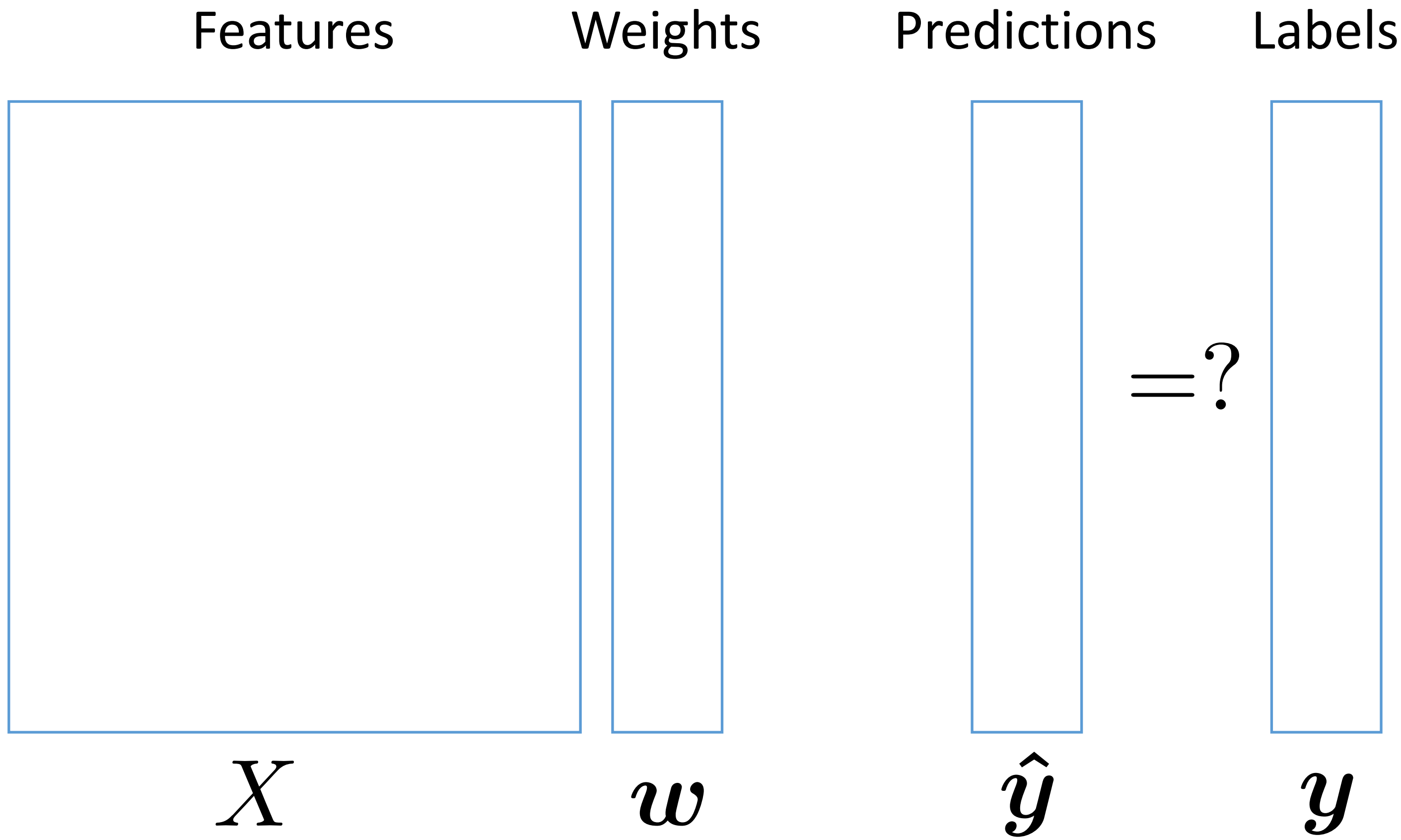


$$\phi(X\boldsymbol{w}) = \hat{\boldsymbol{y}}$$

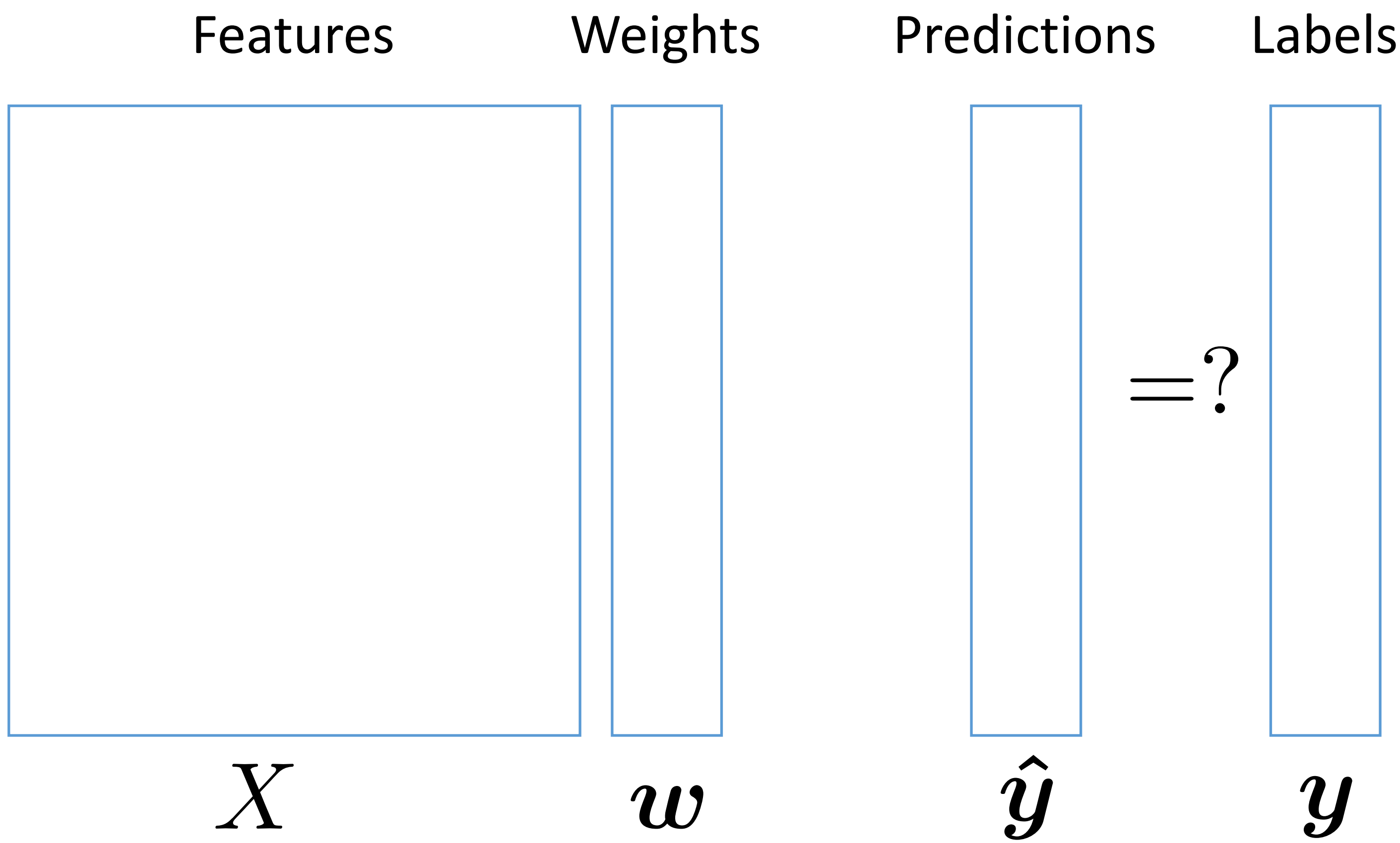
Focus in on one single instance



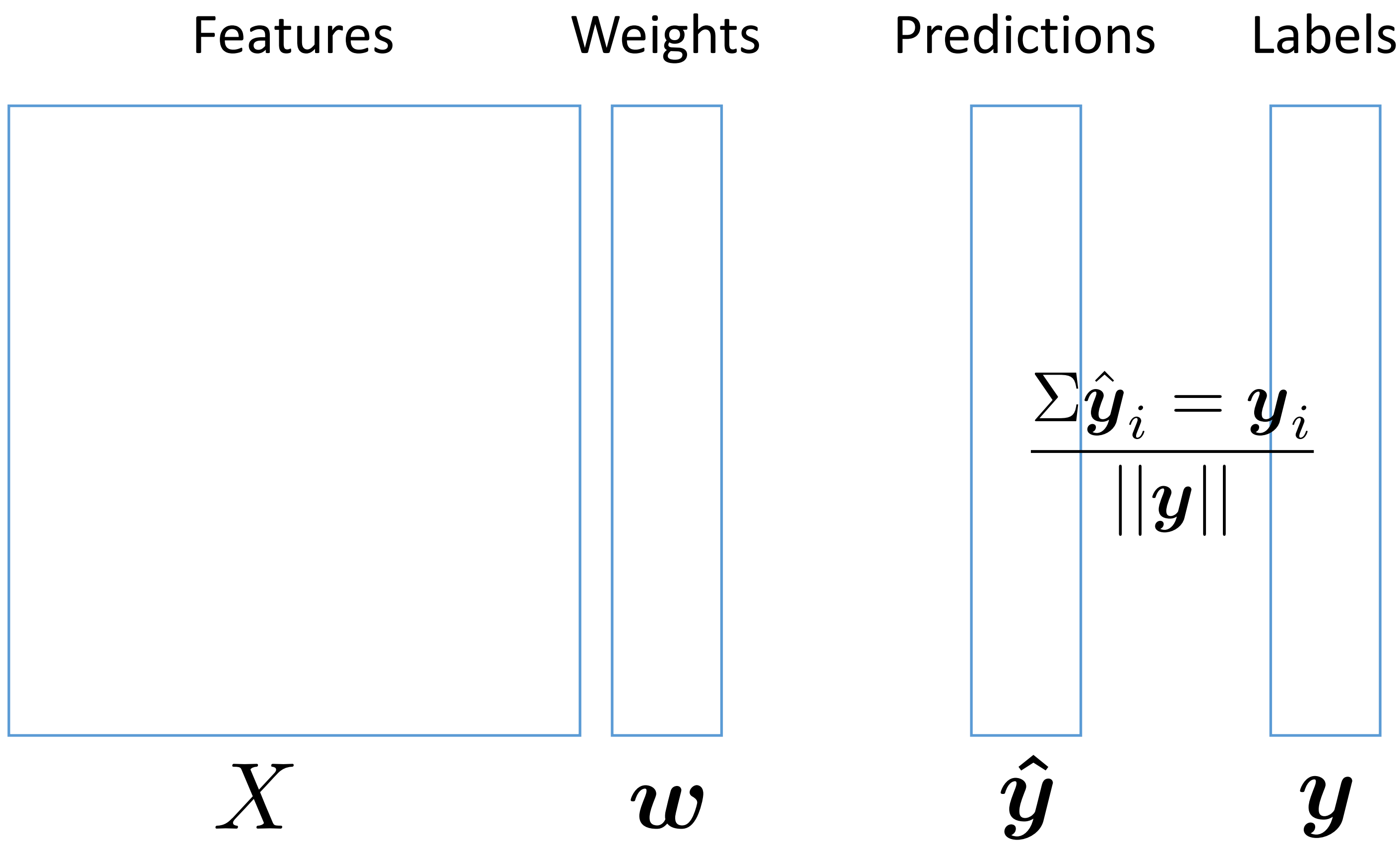
Are the predictions any good?



Are the predictions any good?



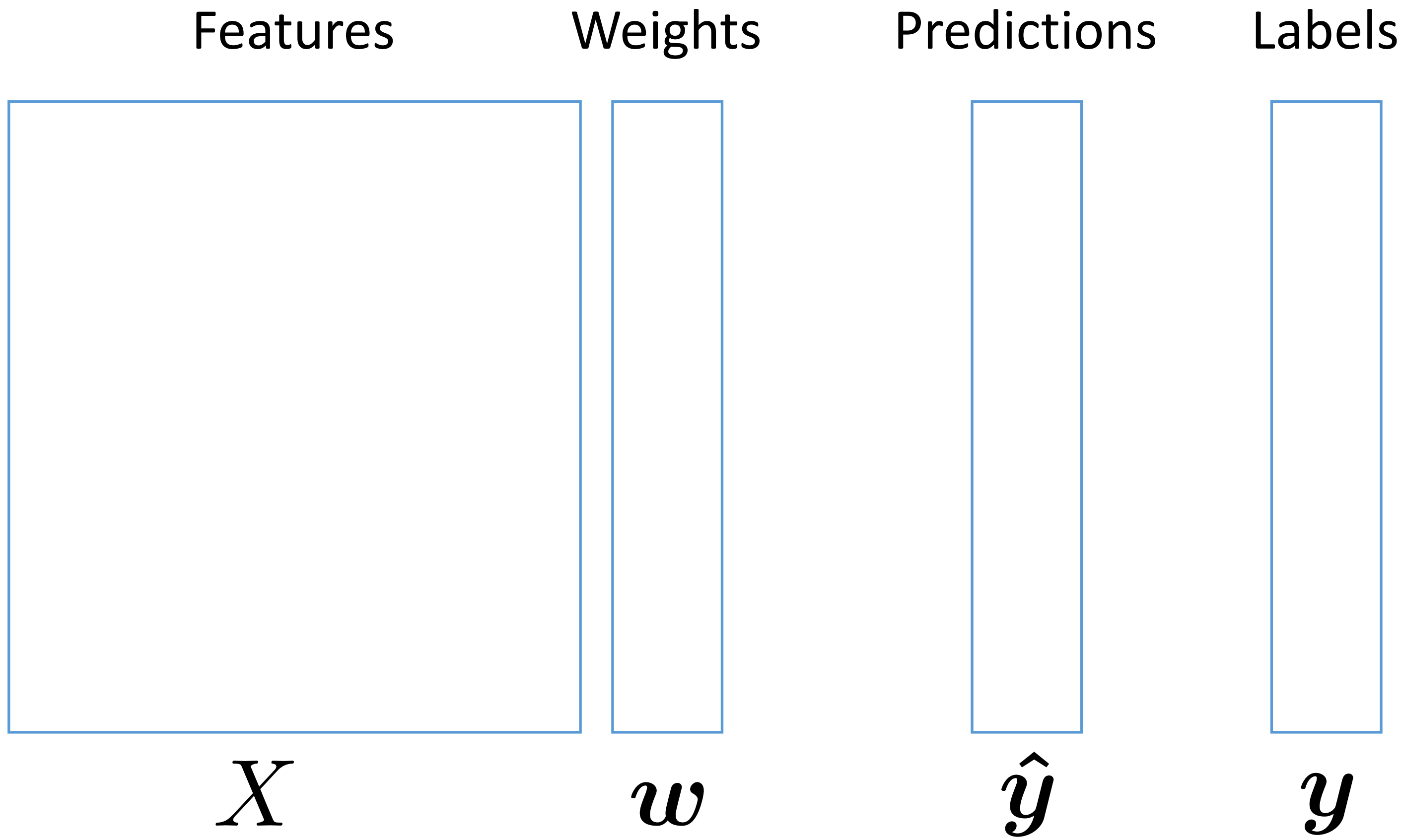
Accuracy



$$\frac{\Sigma \hat{\boldsymbol{y}}_i = \boldsymbol{y}_i}{||\boldsymbol{y}||}$$

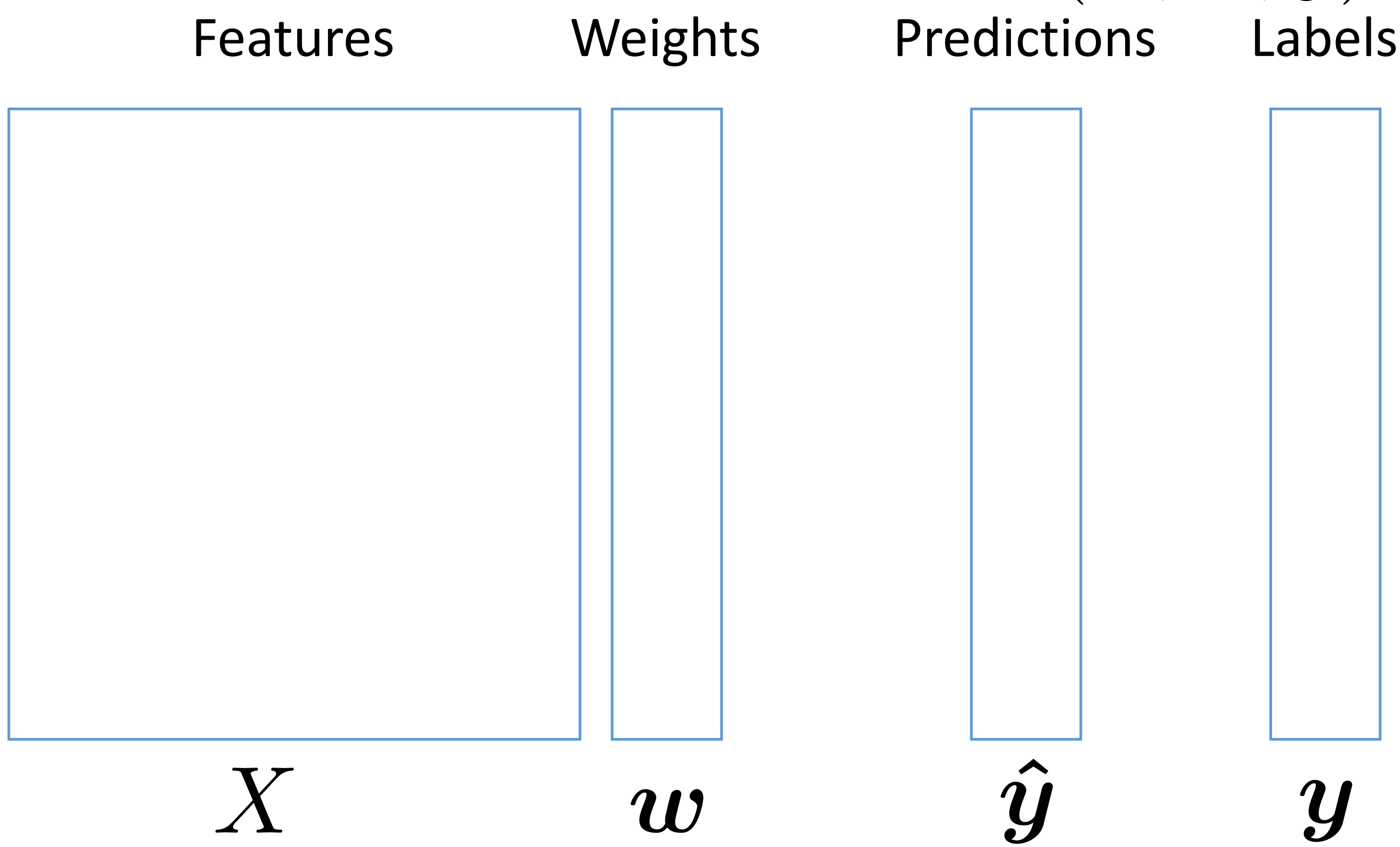
Loss

$$L(X, \boldsymbol{w}, \boldsymbol{y})$$



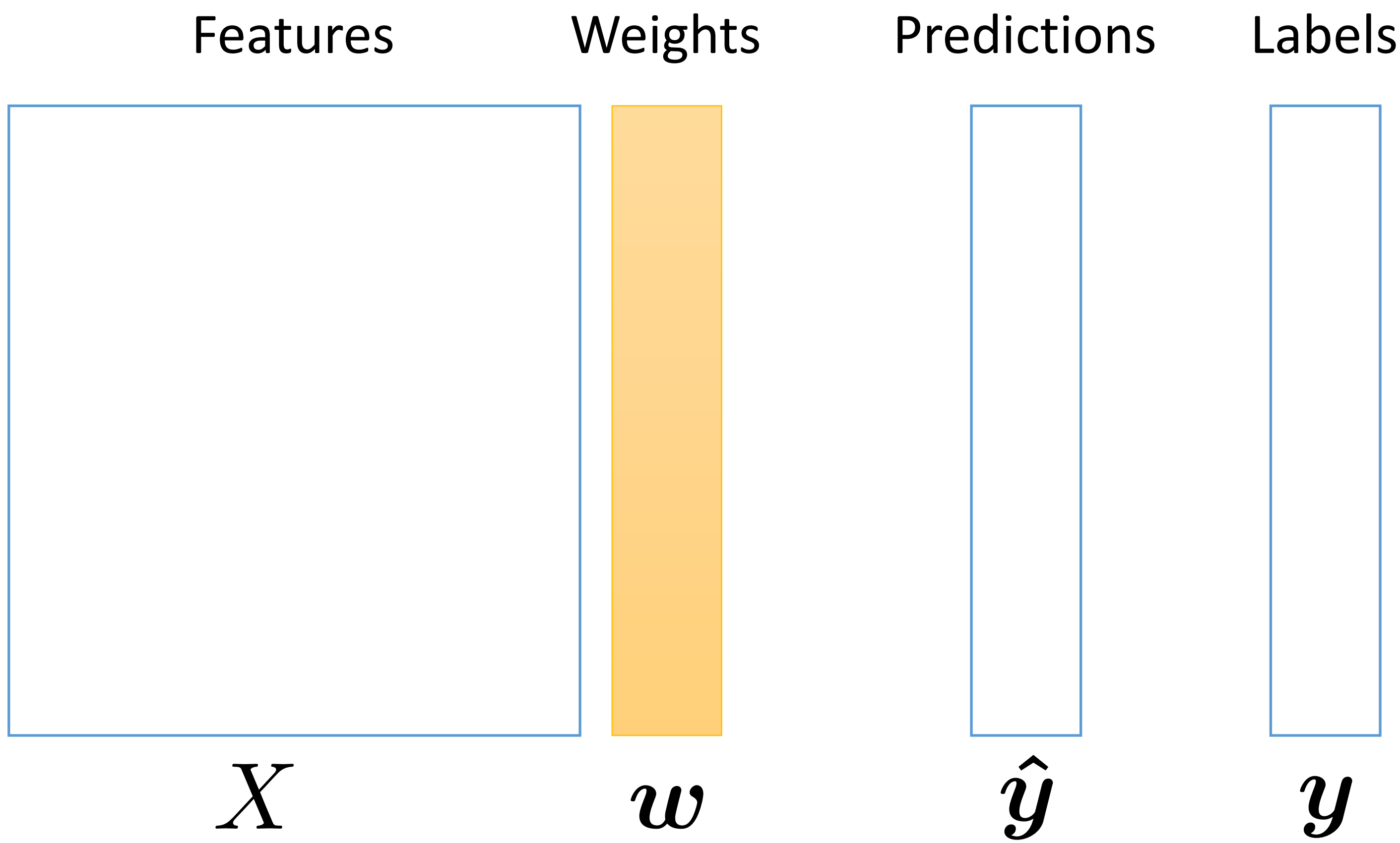
Loss: how far are the predictions from the labels?

$$L(X, \boldsymbol{w}, \boldsymbol{y})$$

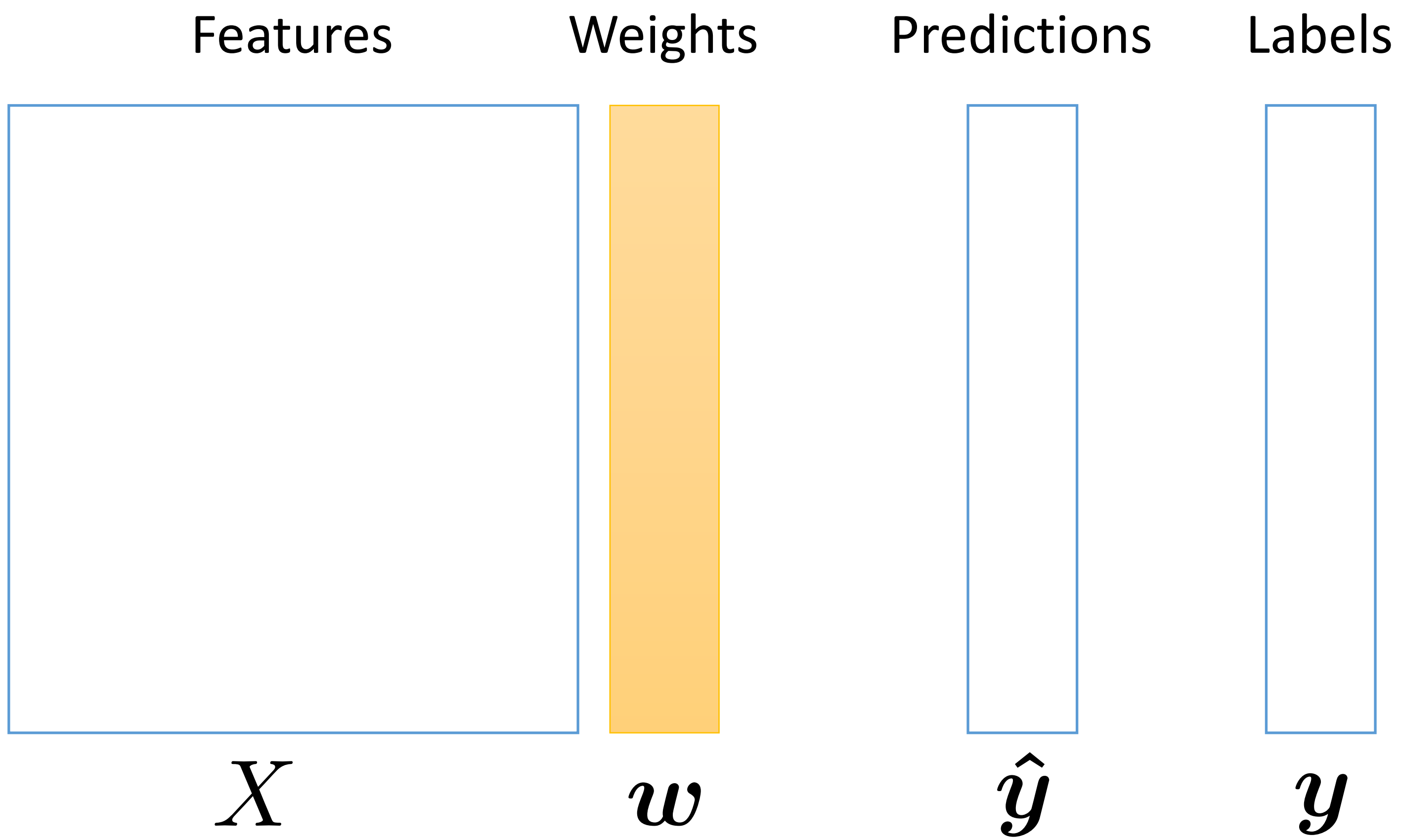


“Optimization”

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

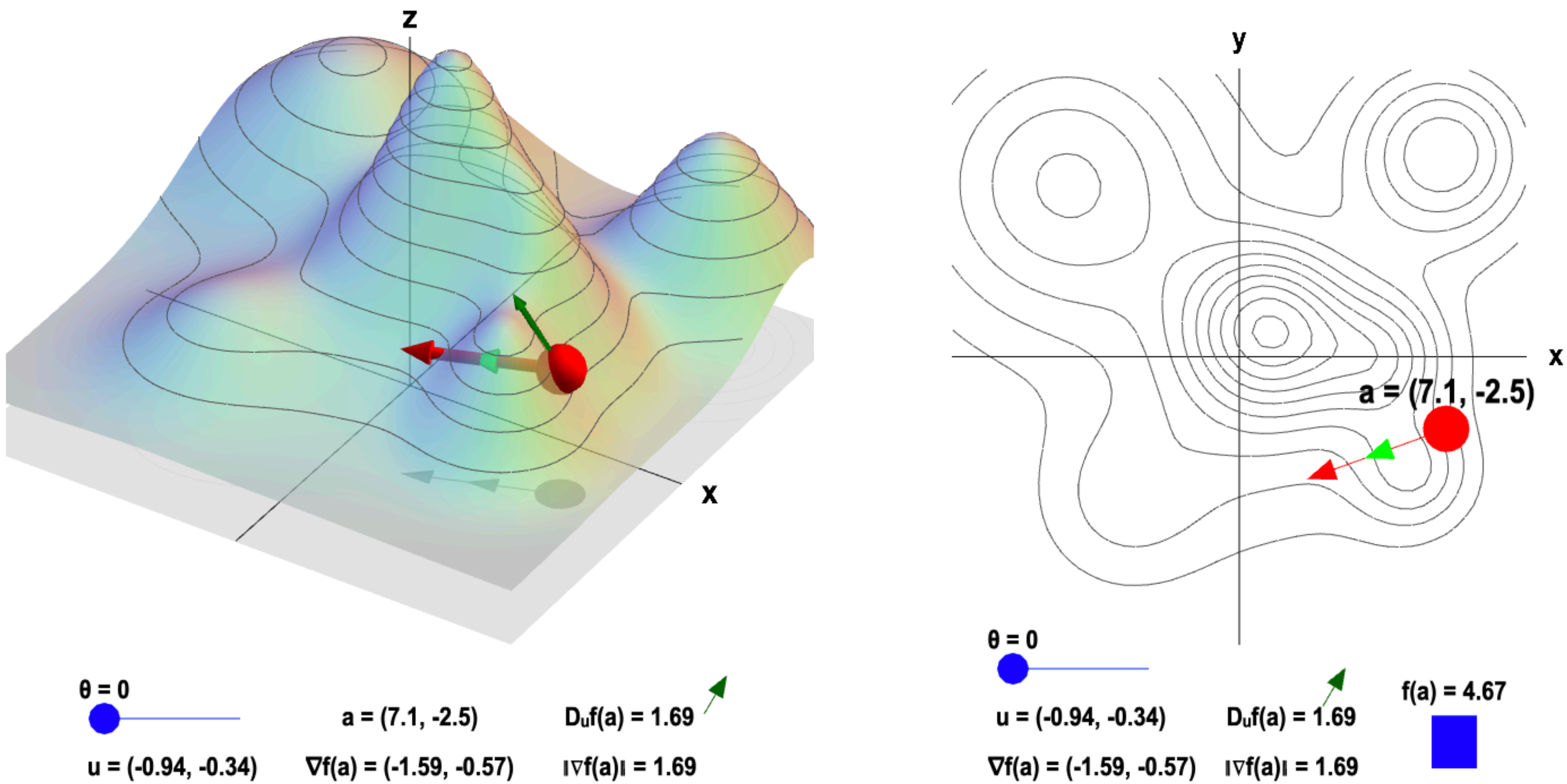


Adjust weights to minimize loss



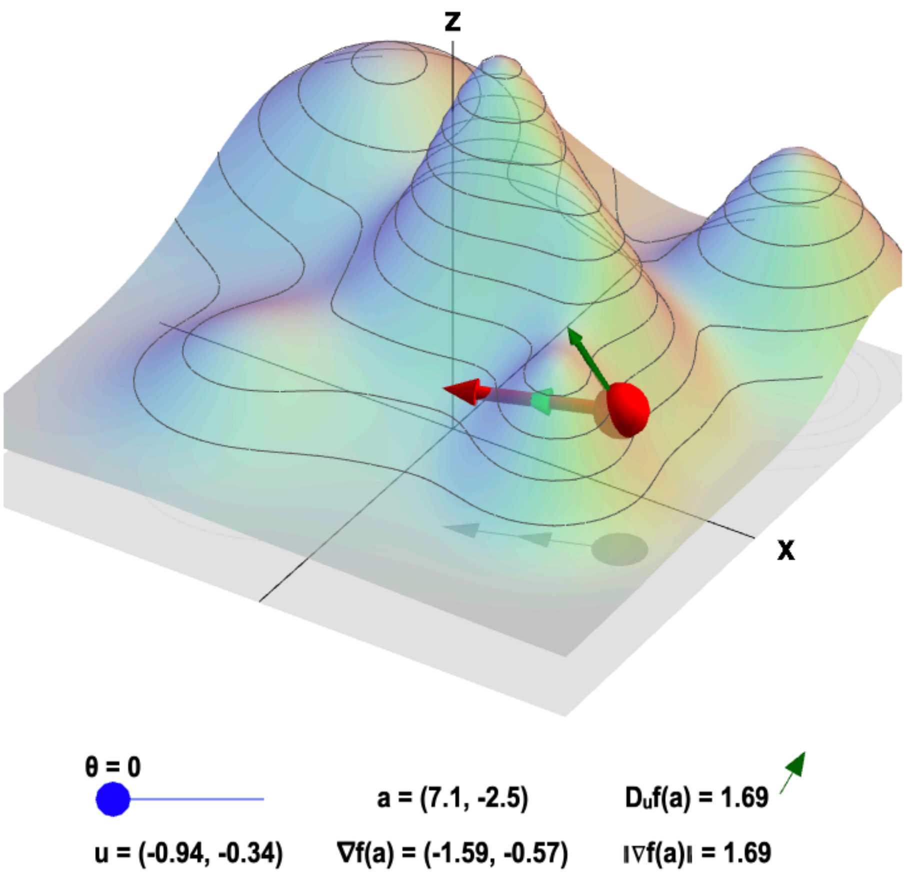
So far, we have seen one method for minimizing loss

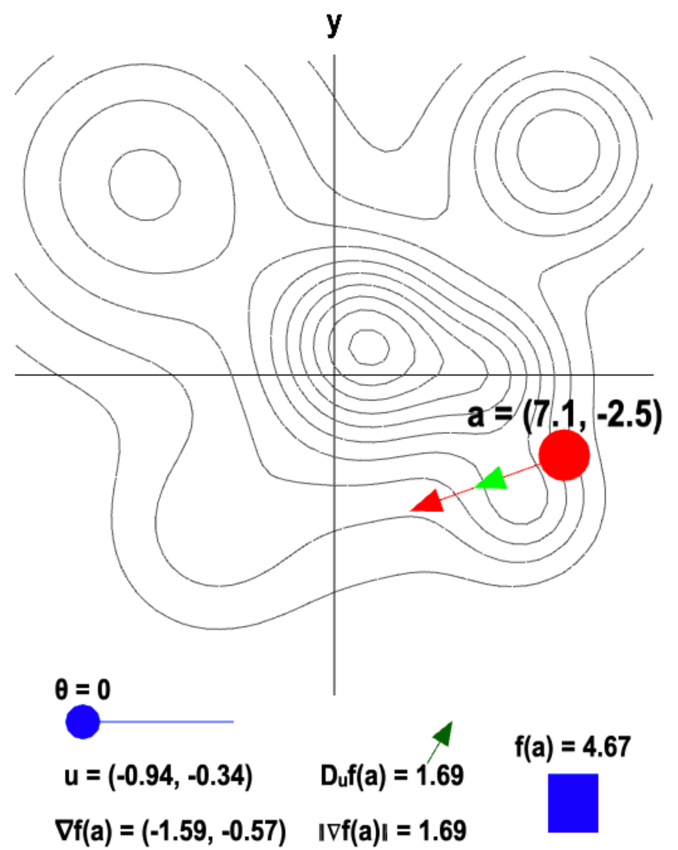
Gradient descent



https://mathinsight.org/applet/gradient_directional_derivative_mountain

Reminder: this picture should be upside down





https://mathinsight.org/applet/gradient_directional_derivative_mountain

Gradient Descent

1. Initialize the parameters **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 **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 **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!

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

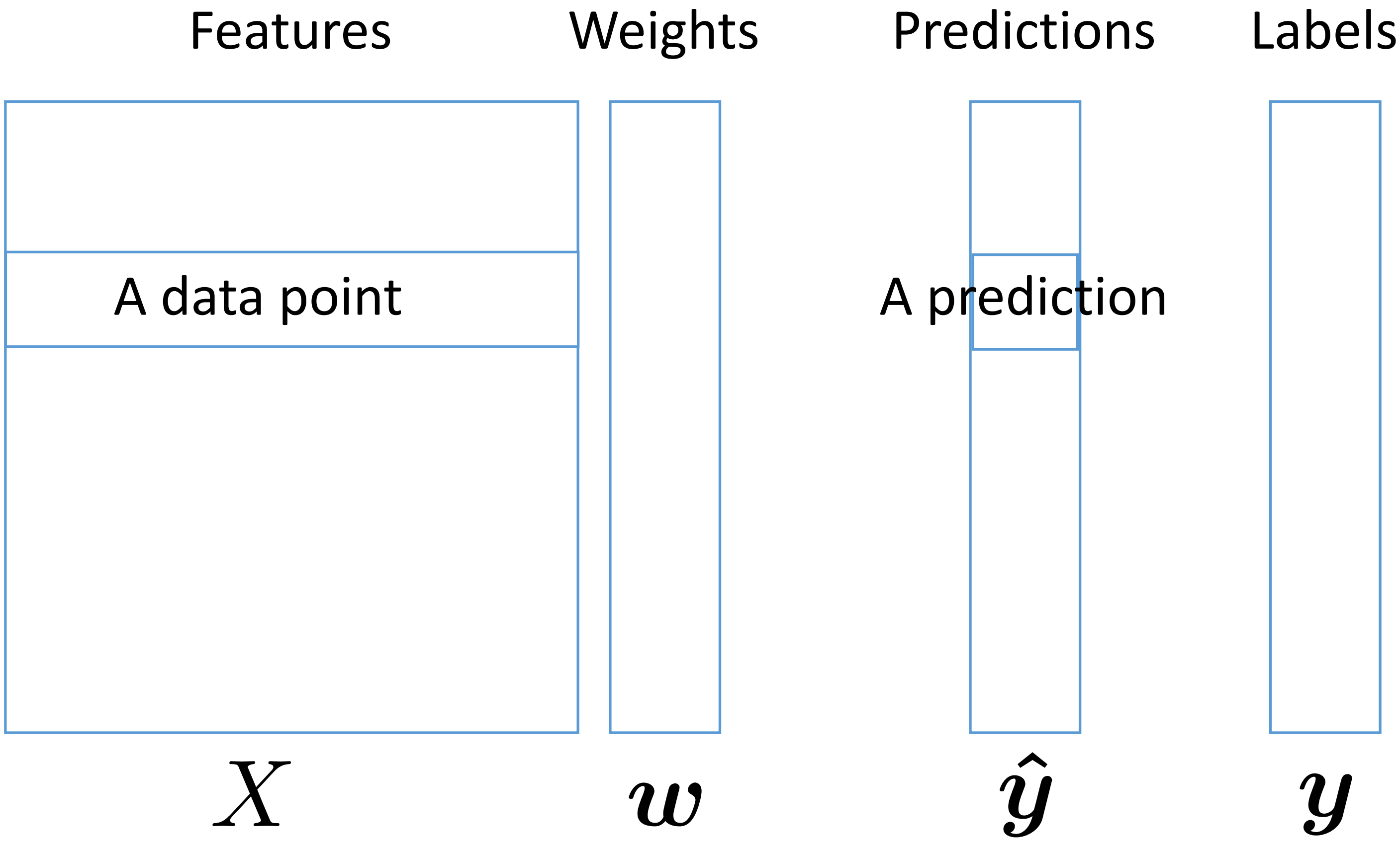
Stochastic Gradient Descent

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

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

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

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

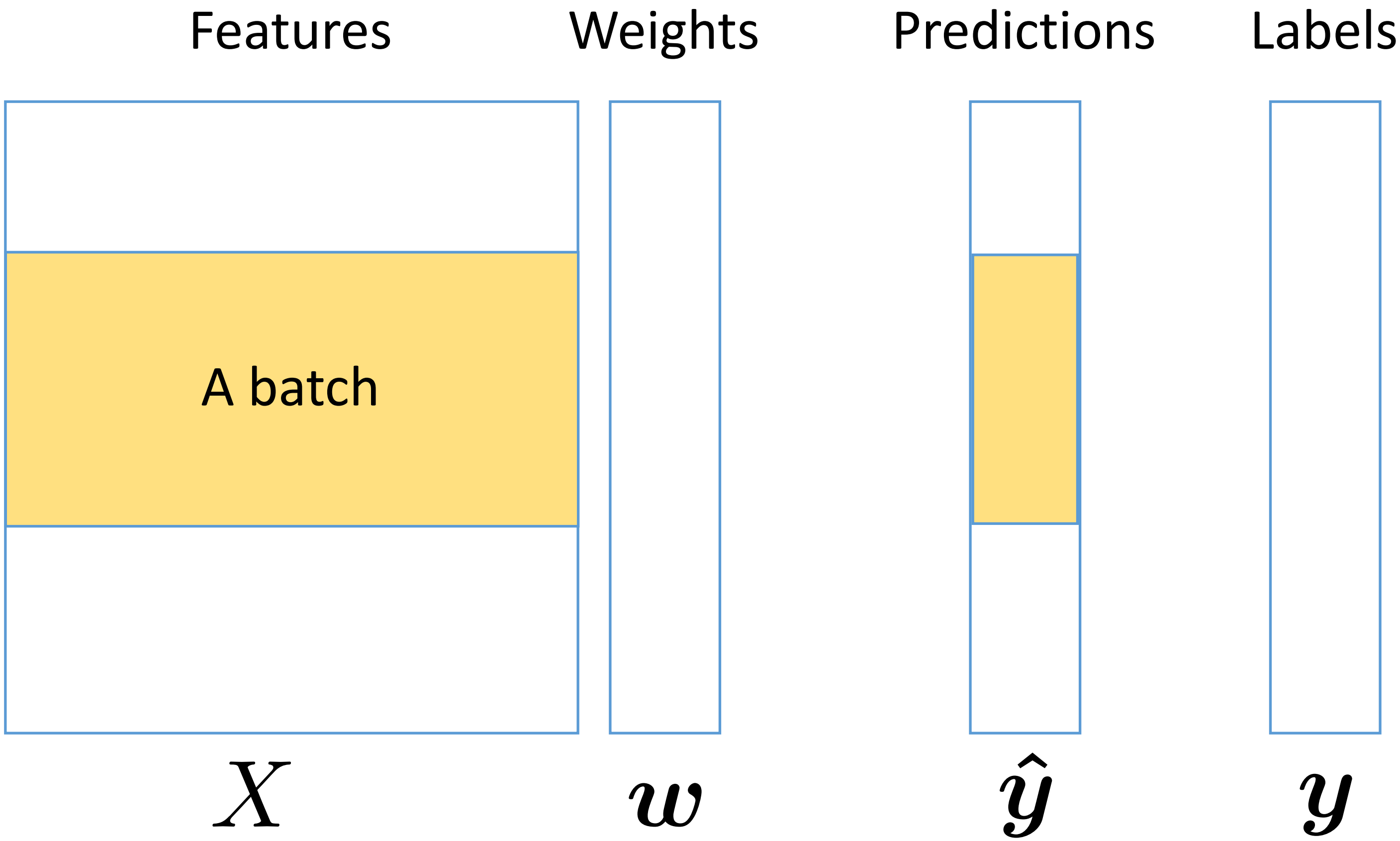
- stochastic = random
  The *expected* gradient is the true gradient

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

- Dave Blei

http://www.cs.columbia.edu/~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