Deep Learning Explained

Module 2: Logistic Regression

Sayan D. Pathak, Ph.D., Principal ML Scientist, Microsoft
Roland Fernandez, Senior Researcher, Microsoft

Module Outline

Application:

OCR with MNIST data

Model:

Logistic Regression

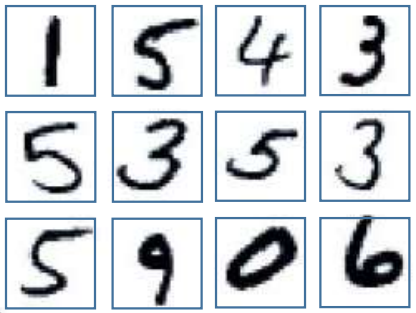
Concepts:

Loss, Minibatch

Train-Test-Predict workflow

MNIST Handwritten Digits (OCR)

Handwritten Digits

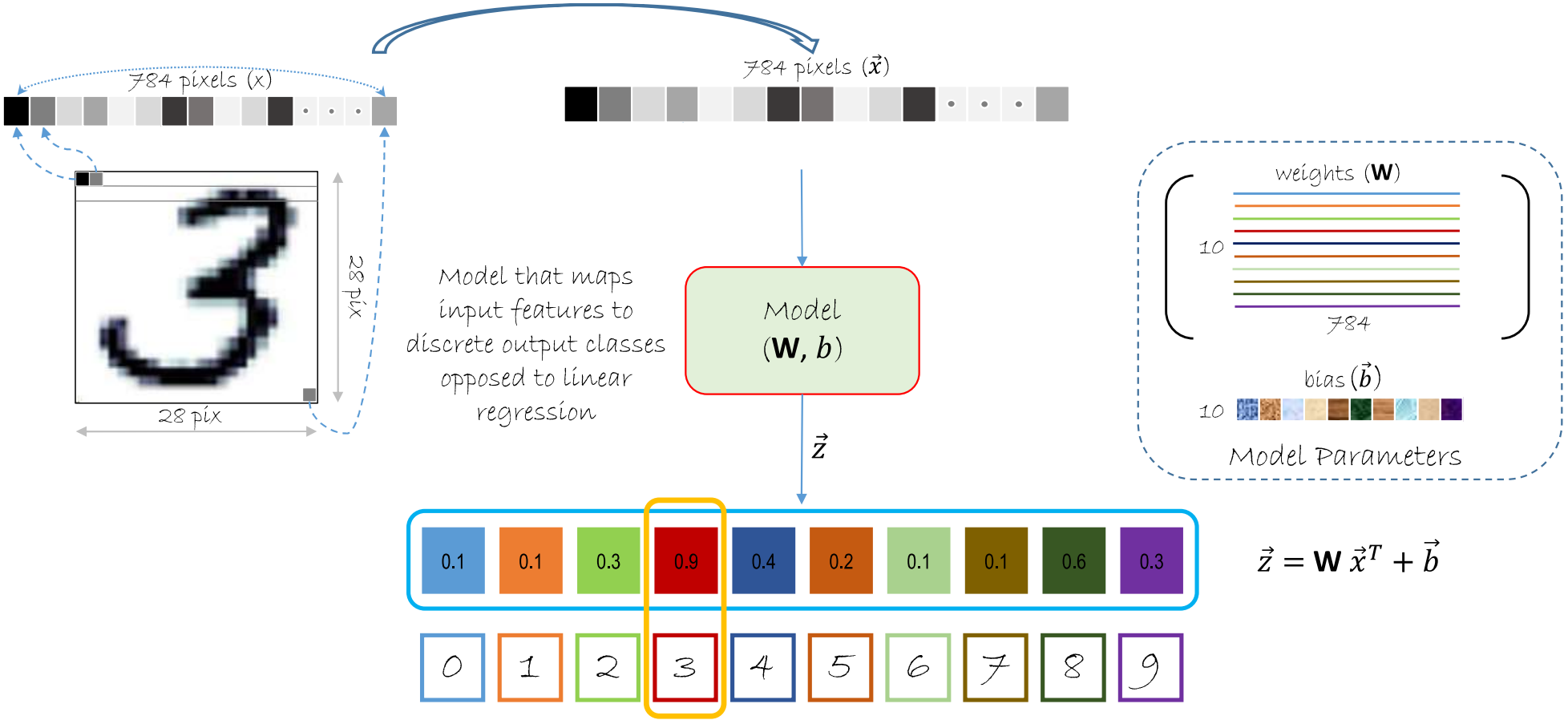


1 5 4 3
5 3 5 3
5 9 0 6

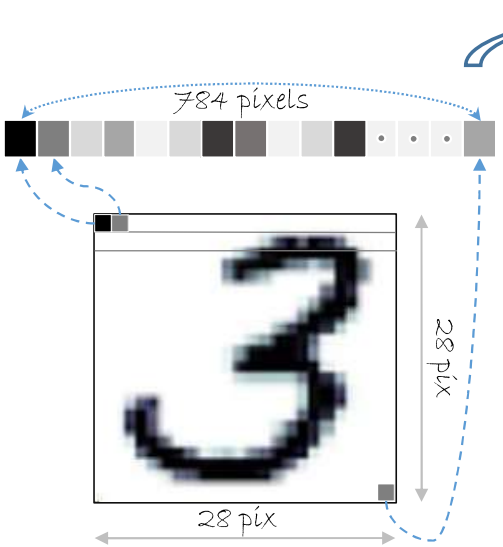
Corresponding Labels

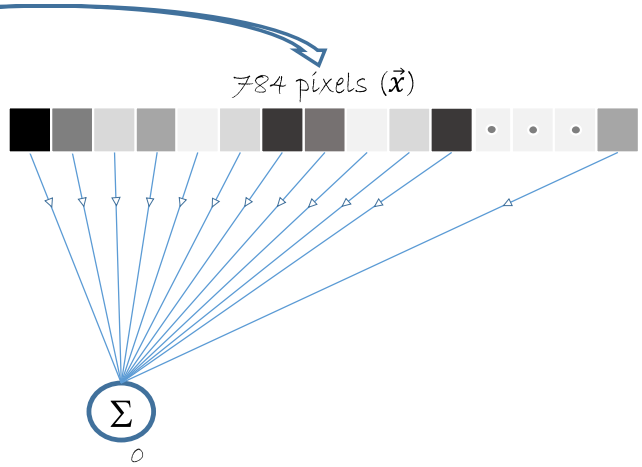
- Data set of hand written digits (0-9) with
  - ✓60,000 training images
  - ✓10,000 test images
- Each image is: 28 x 28 pixels

Logistic Regression



Logistic Regression



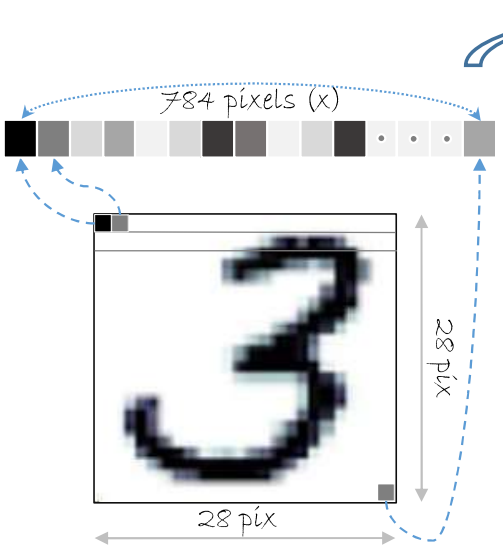


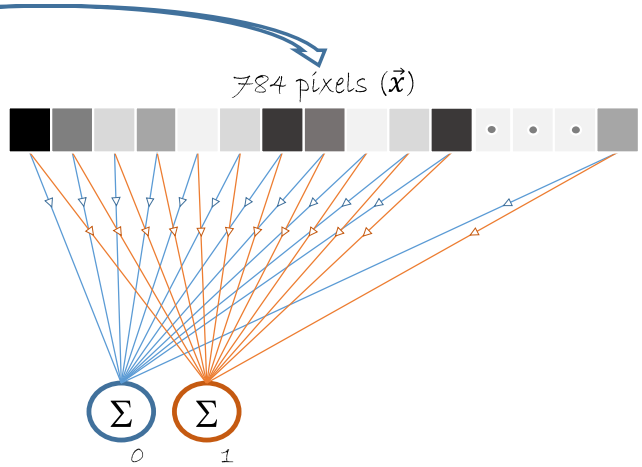
$\vec{w}_0$

Σ = Sum (weights x pixels) = $\vec{w}_0 \cdot \vec{x}^T$

784 784

Logistic Regression



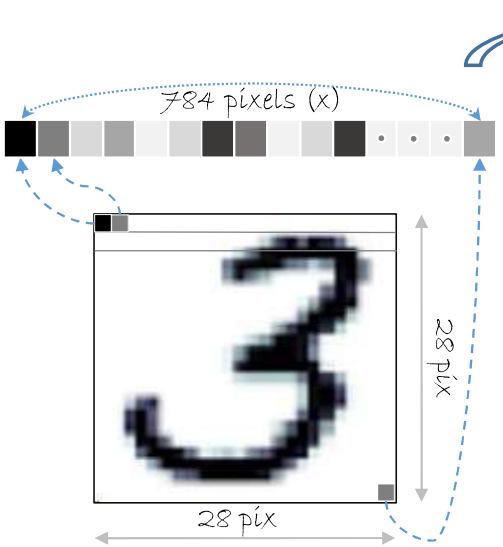


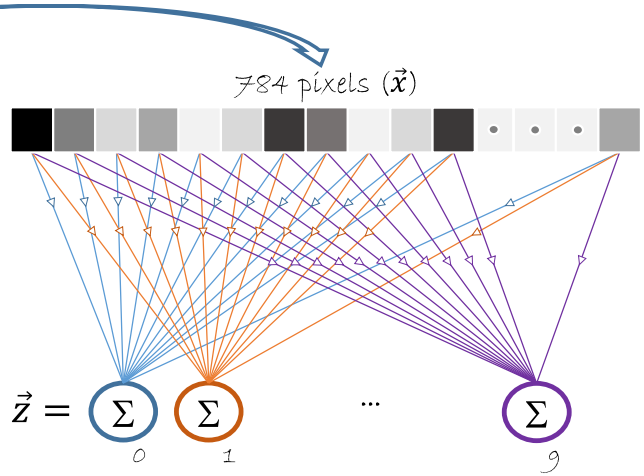
$\vec{w}_1$

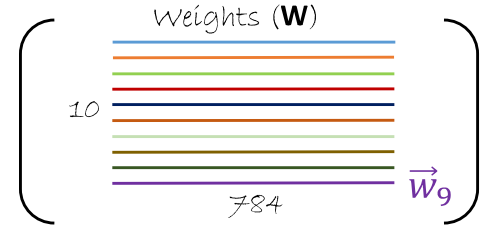
Σ = Sum (weights x pixels) = $\vec{w}_0 \cdot \vec{x}^T$
784 784

Σ = Sum (weights x pixels) = $\vec{w}_1 \cdot \vec{x}^T$
784 784

Logistic Regression







Σ = Sum (weights x pixels) = $\vec{w}_0 \cdot \vec{x}^T$
784 784

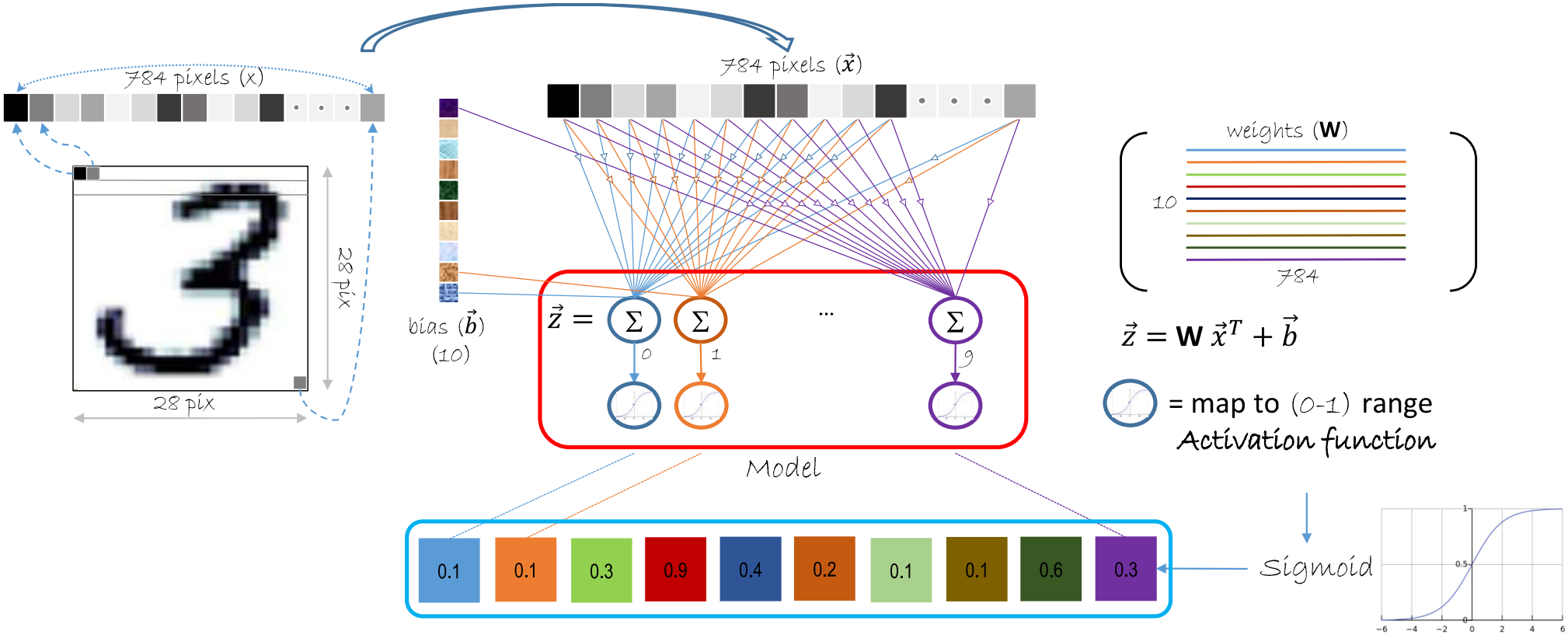
Σ = Sum (weights x pixels) = $\vec{w}_1 \cdot \vec{x}^T$
784 784

⋮

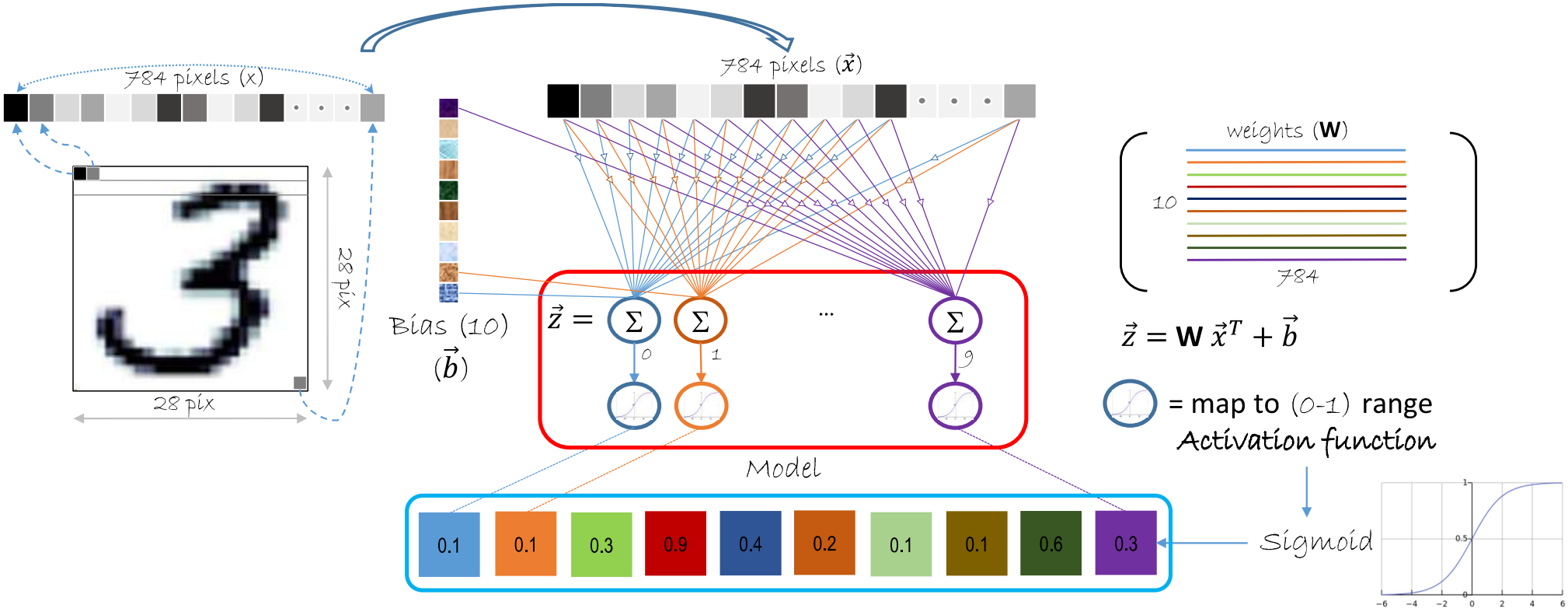
Σ = Sum (weights x pixels) = $\vec{w}_9 \cdot \vec{x}^T$
784 784

$$\vec{z} = \mathbf{W}\,\vec{x}^T$$

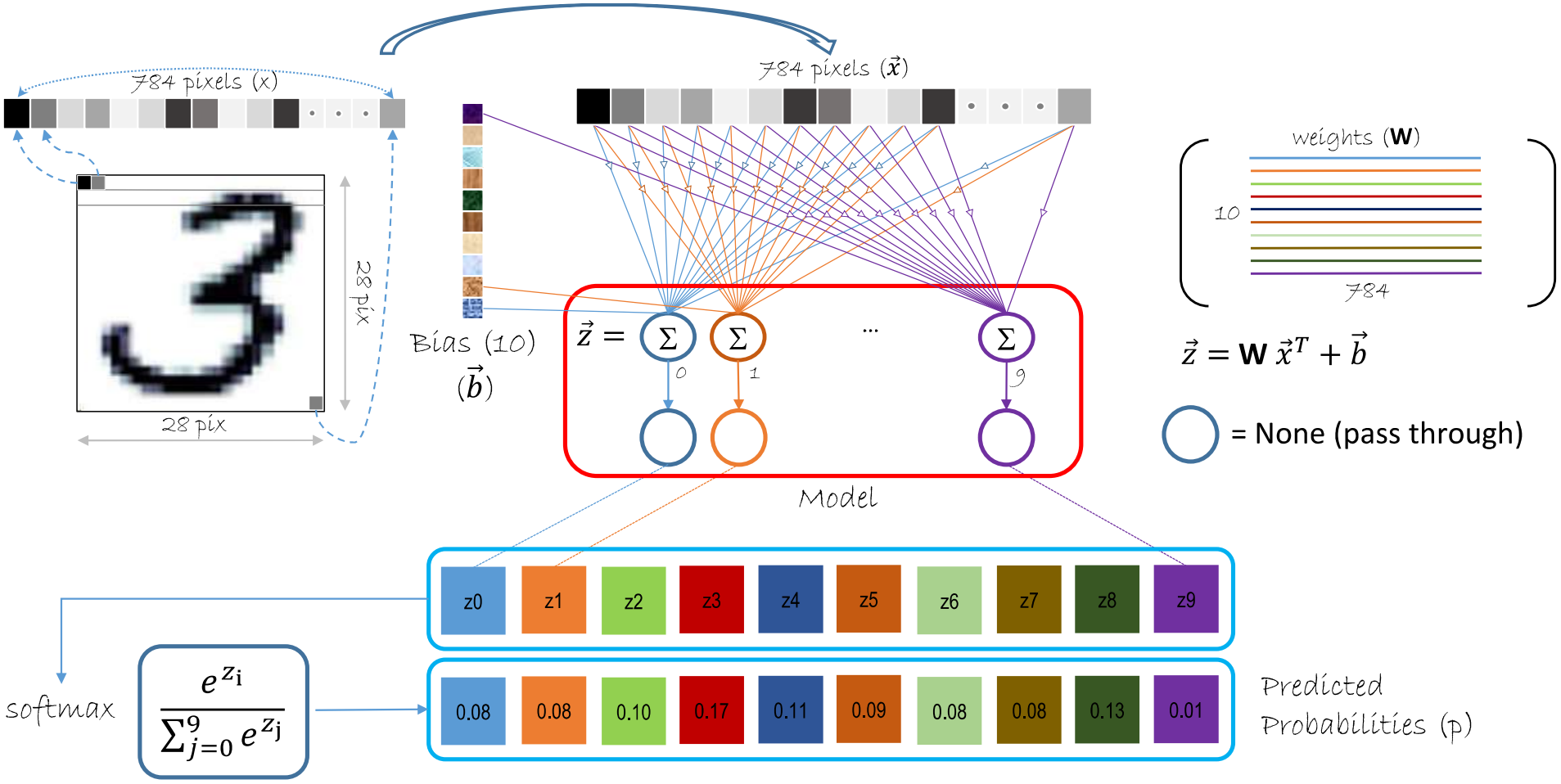
Logistic Regression



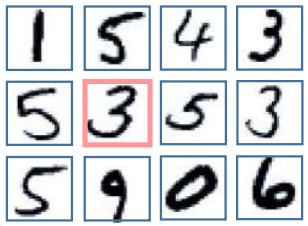
Logistic Regression



Logistic Regression with Softmax



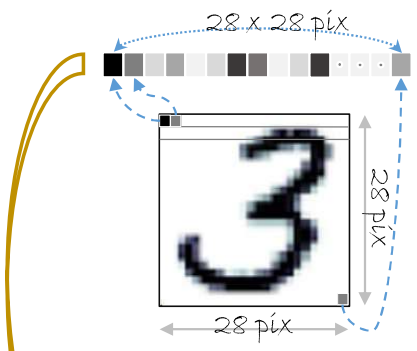
Loss Function



1 5 4 3
5 3 5 3
5 9 0 6

Label One-hot encoded (Y)

0	0	0	1	0	0	0	0	0	0



Loss functions

$$se = \sum_{j=0}^{9} (y_j - p_j)^2$$

Squared error

$$ce = -\sum_{j=0}^{9} y_j \, log(p_j)$$

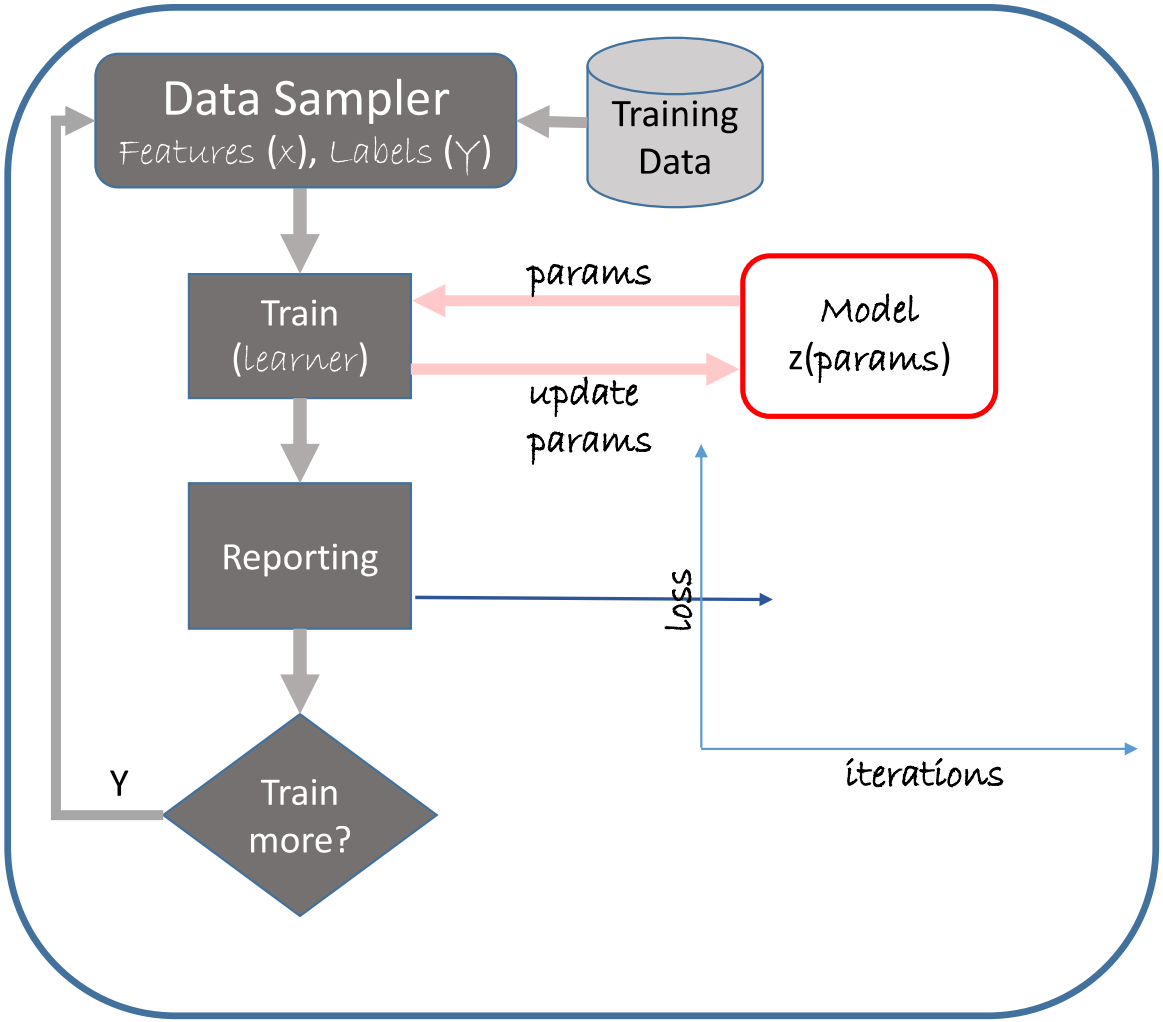
Cross entropy error

Model (w, b)

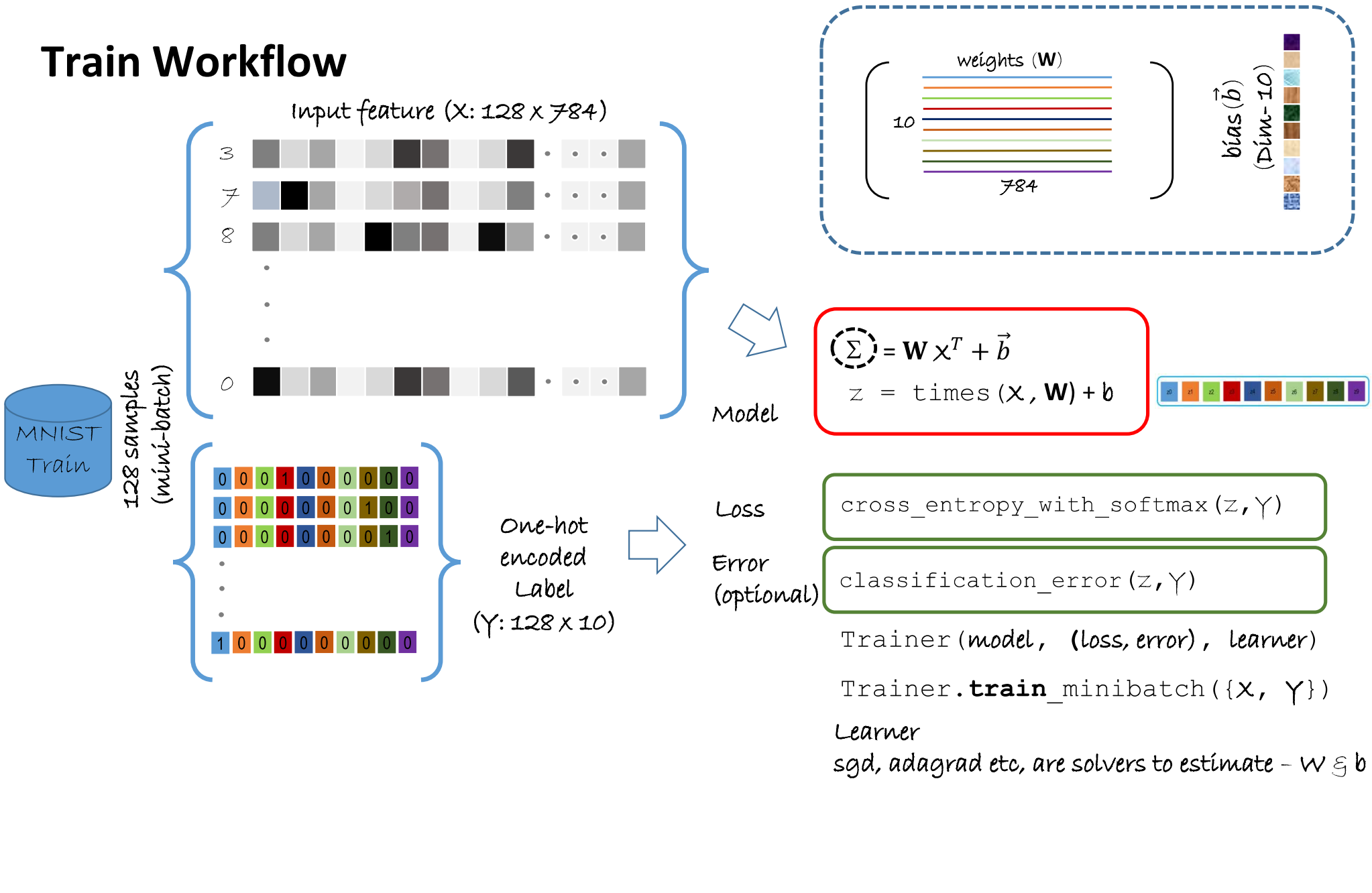
Predicted Probabilities (p)

0.08	0.08	0.10	0.17	0.11	0.09	0.08	0.08	0.13	0.01

Train Workflow



Train Workflow



Learn the weights: Learners / Optimizers / Solvers

For 1 sample:

$$\text{Loss } (L_i) = -\sum_{j=0}^{9} y_j^{(i)} log(p_j) \quad \text{where: } p_j = f(x^{(i)}; \theta)_j$$
$$\theta \in (w, b)$$

For all samples (m = 60000 images):

$$\text{Total loss} = \sum_{i=1}^{m} L_i \ (\theta; (x^{(i)}, y^{(i)}))$$

Convex function:

There is 1 and only 1 minimum

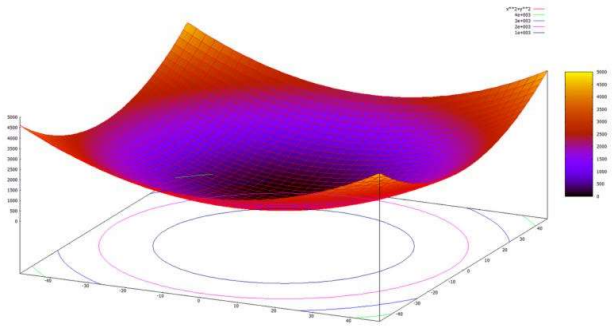
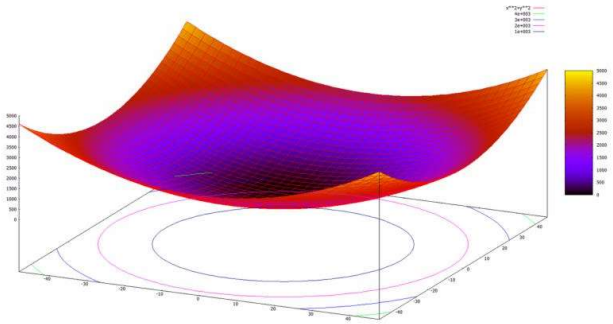
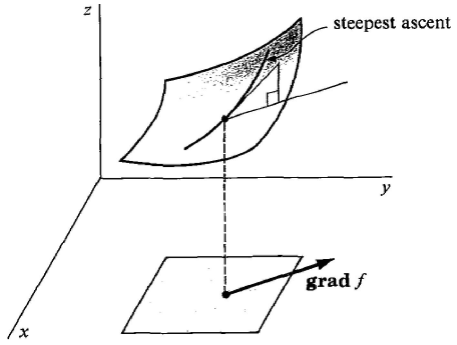
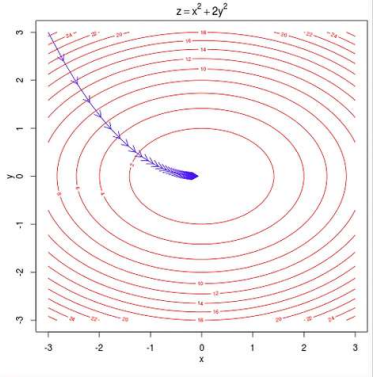


Fig: courtesy http://codingwiththomas.blogspot.com/2012/09/particle-swarm-optimization.html

Gradient Descent







$$\theta' = \theta - \mu \, grad\,(L;\theta)$$

Where:

θ = model parameter

μ = learning rate

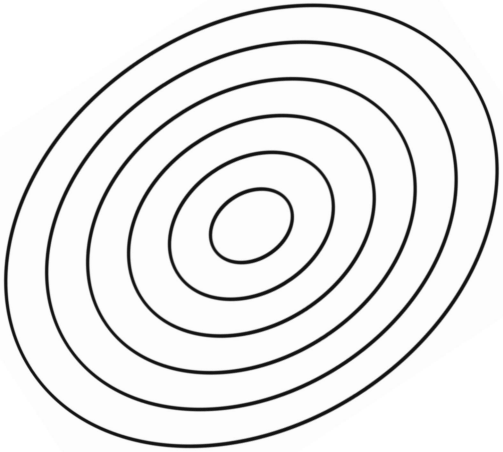
Computing "Total Loss" ($\sum_i^n L_i$) for large data set is expensive and often redundant

- refer to http://sebastianruder.com/optimizing-gradient-descent/ for details

Stochastic Gradient Descent (SGD)

SGD:

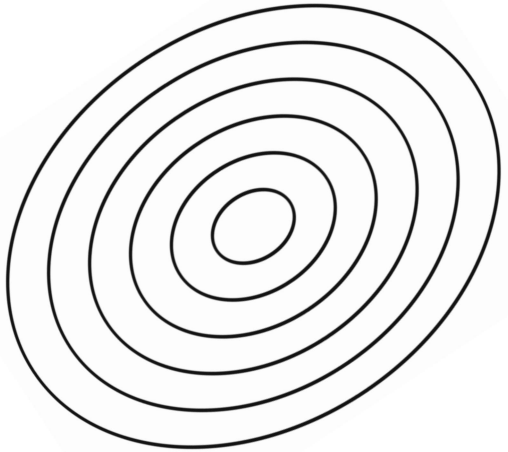
Update the parameters for each (data, label) pair



Mini-batch SGD:

Update the parameters for mini-batch set

Set of (data, label) pairs



refer to http://sebastianruder.com/optimizing-gradient-descent/ for details on different learners

Other learners

Momentum-SGD

Nestorov

Adagrad

Adsdelta

Adam

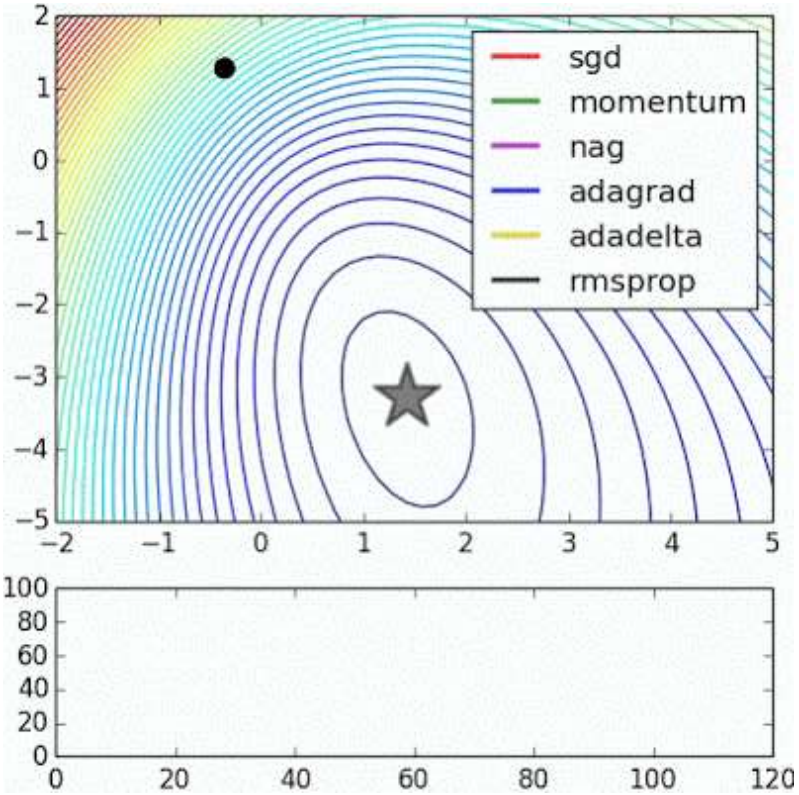
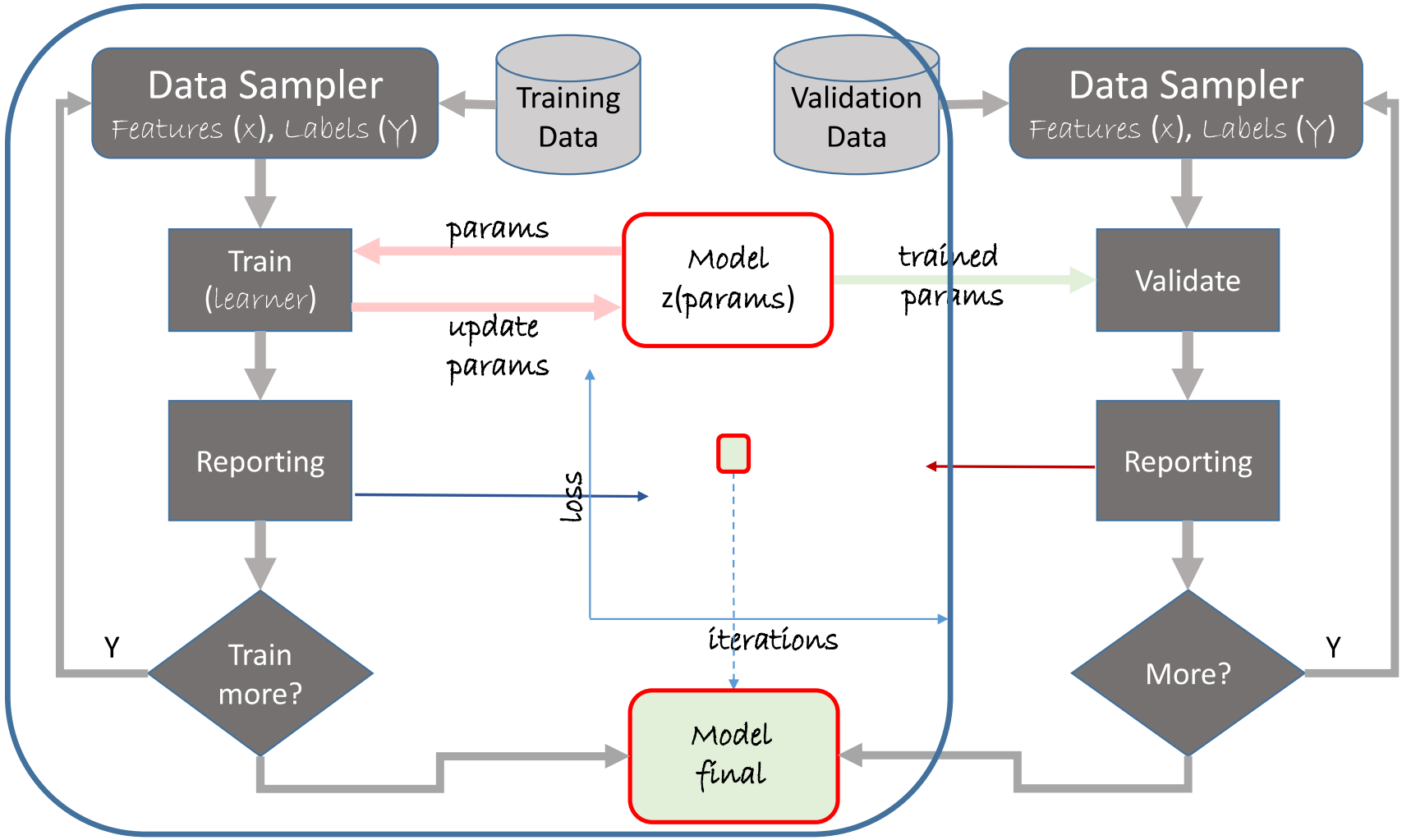


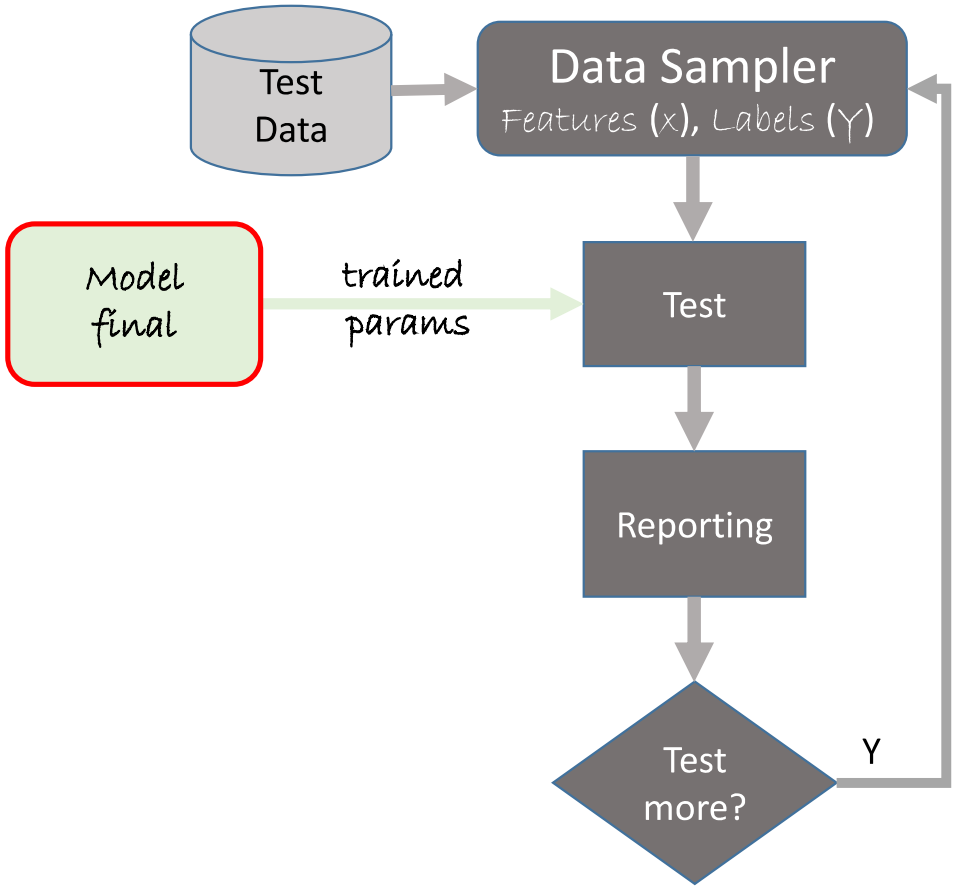
Image by: Alec Radford

Refer to http://sebastianruder.com/optimizing-gradient-descent/ for details on different learners

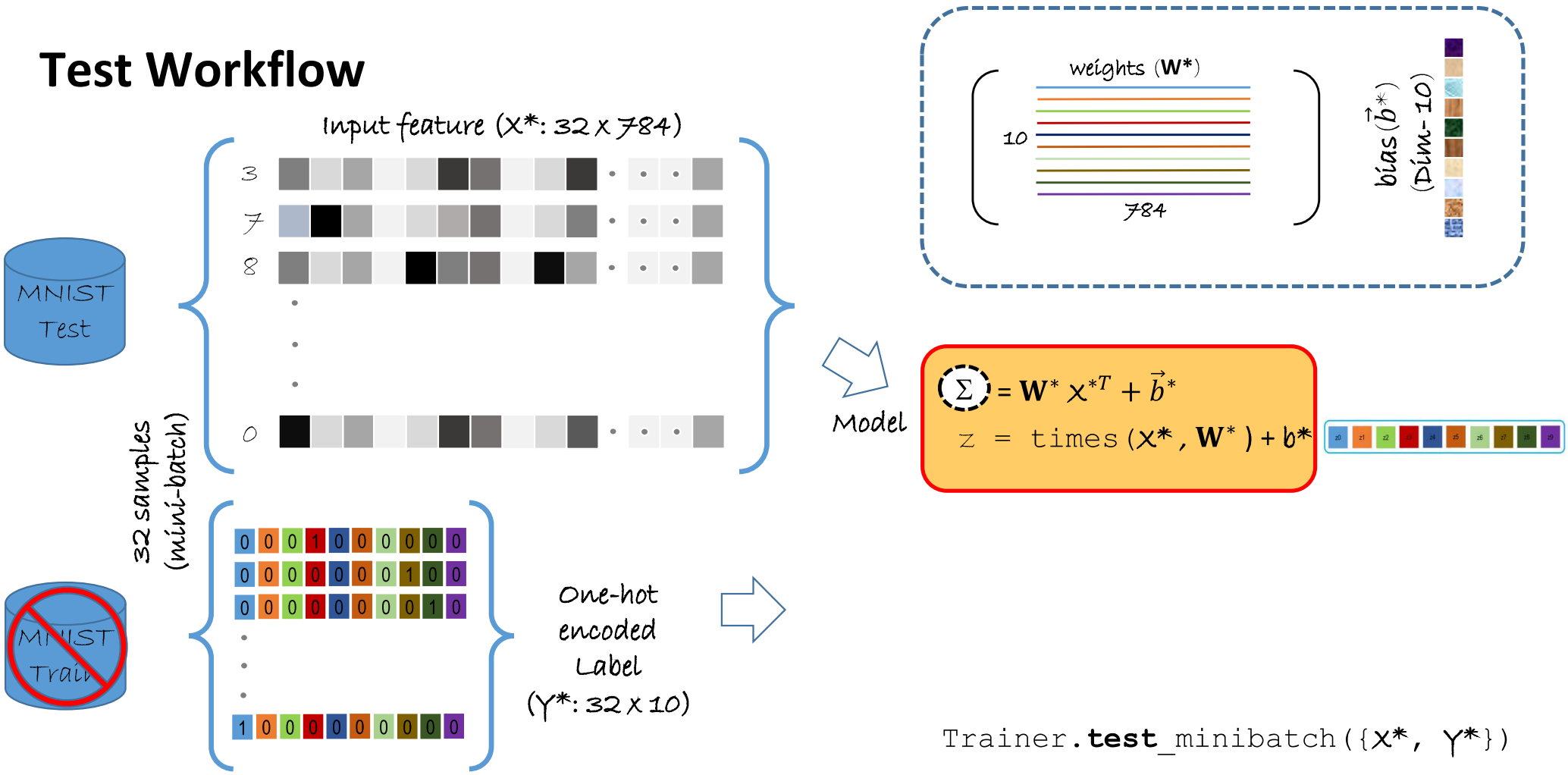
Validation Workflow



Test Workflow



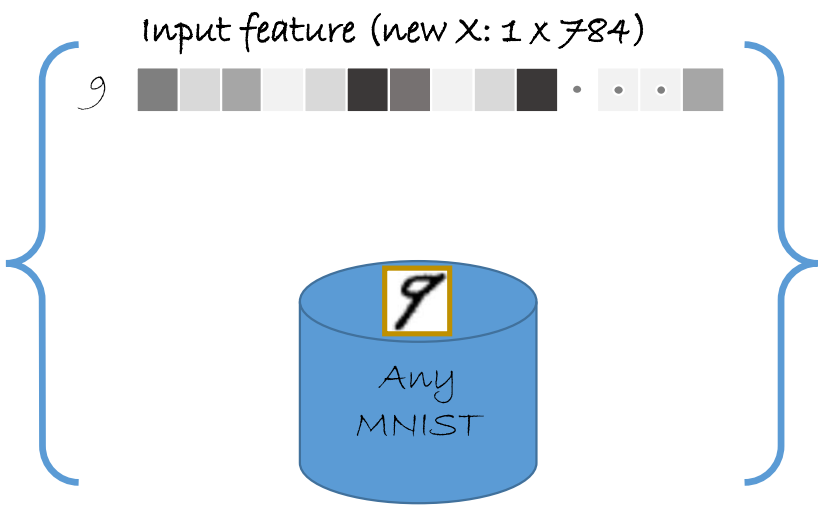
Test Workflow

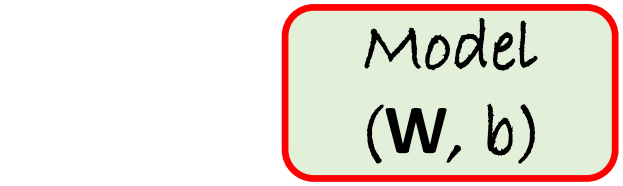



```
Trainer.test_minibatch({X*, Y*})
```

Returns the classification error as % incorrectly labeled MNIST image.

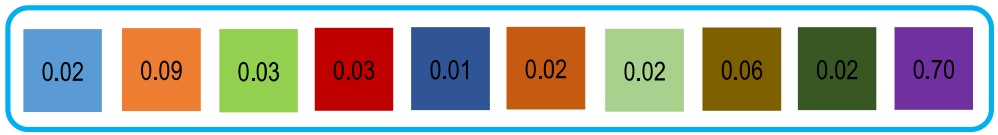
Prediction Workflow





Model.eval(new X)

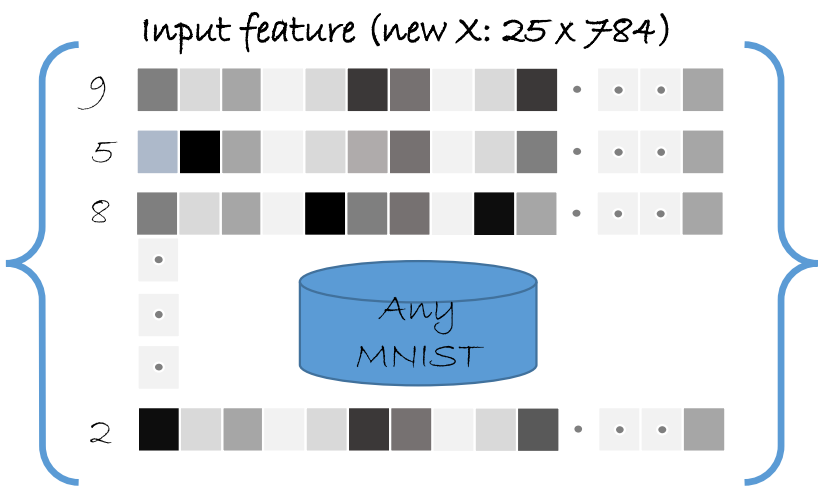
Predicted Softmax Probabilities (predicted_label)

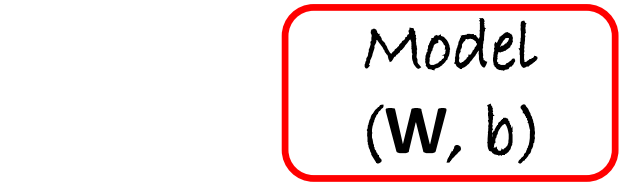


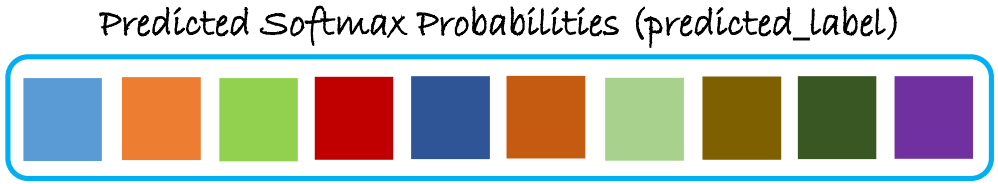
[numpy.argmax(predicted_label) for predicted_label in predicted_labels]

[9]

Prediction Workflow







[`numpy.argmax`(predicted_label) `for` predicted_label `in` predicted_labels]

[9, 5, 8, …, 2]