Lecture 5

Tian Han

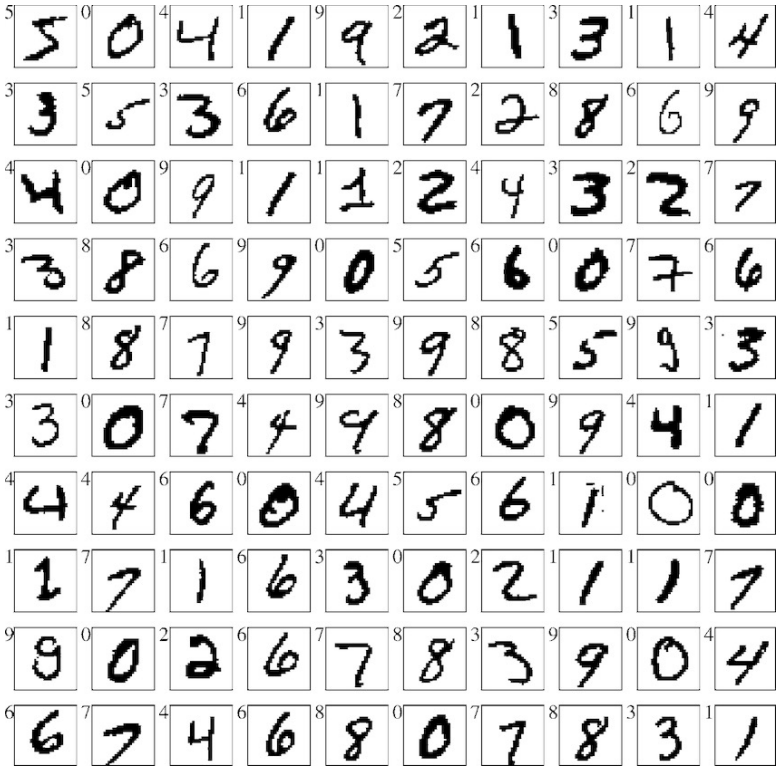
Outline

- Neural Networks
- Keras

Neural Networks: Basics

Revisit Softmax Classifier

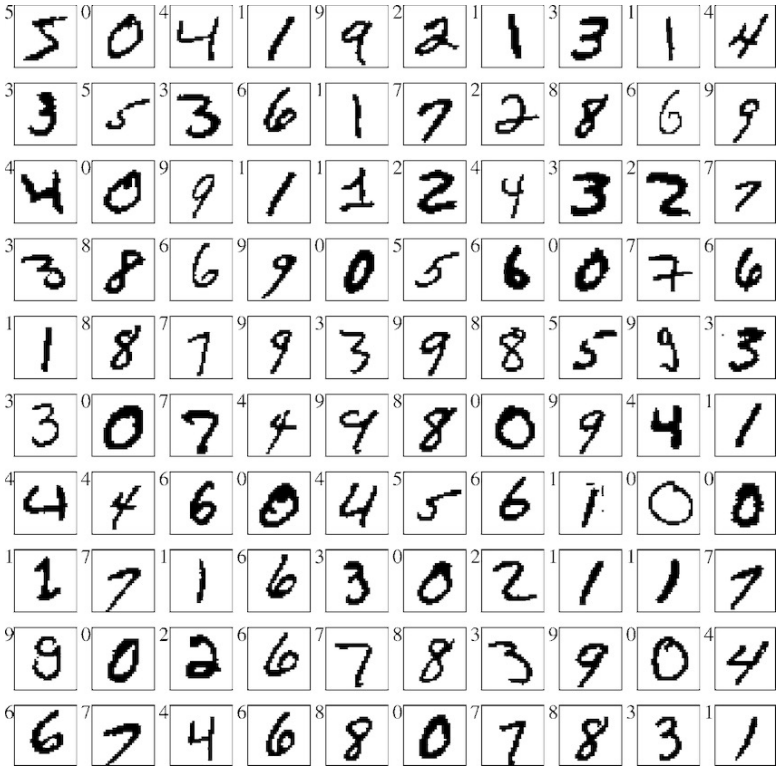
Train a Softmax Classifier



The MNIST Dataset

- $n = 60{,}000$ training samples $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)$.
- Each $\mathbf{x}_j$ is a 28×28 image.
- Each y_j is an integer in $\{0, 1, 2, \cdots, 9\}$.

Train a Softmax Classifier



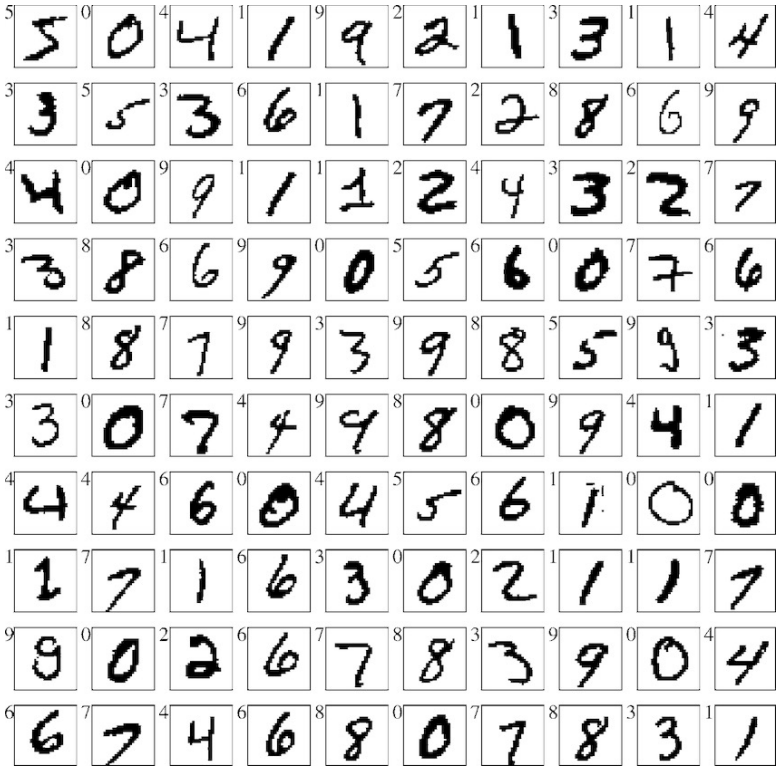
The MNIST Dataset

- $n = 60{,}000$ training samples $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)$.
- Each $\mathbf{x}_j$ is a 28×28 image.
- Each y_j is an integer in $\{0, 1, 2, \cdots, 9\}$.

Task: multi-class classification

- Given a 28×28 image, predict the digit.
- Learn a function $\mathbf{f}$: $\mathbb{R}^{28 \times 28} \mapsto \mathbb{R}^{10}$.
- The i-th entry of $\mathbf{f}(\mathbf{x})$ indicates how likely the image $\mathbf{x}$ is the digit i.

Train a Softmax Classifier

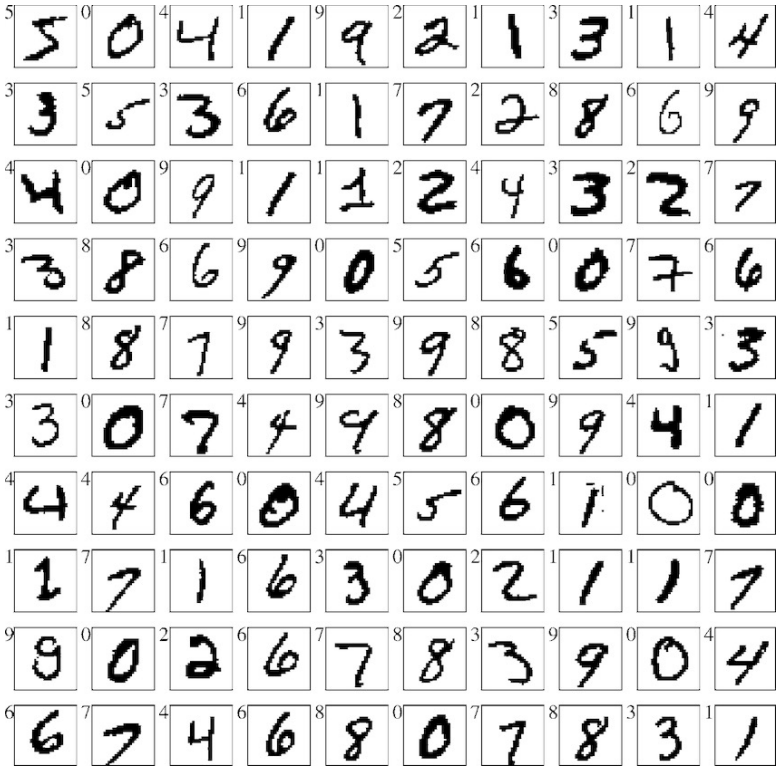


Linear model: softmax classifier

- Vectorize each 28×28 image to a 784-dim vector.
- Add a feature of all ones. (So $\mathbf{x}$ becomes 785-dim.)

basically, a single vector of all pixel data

Train a Softmax Classifier



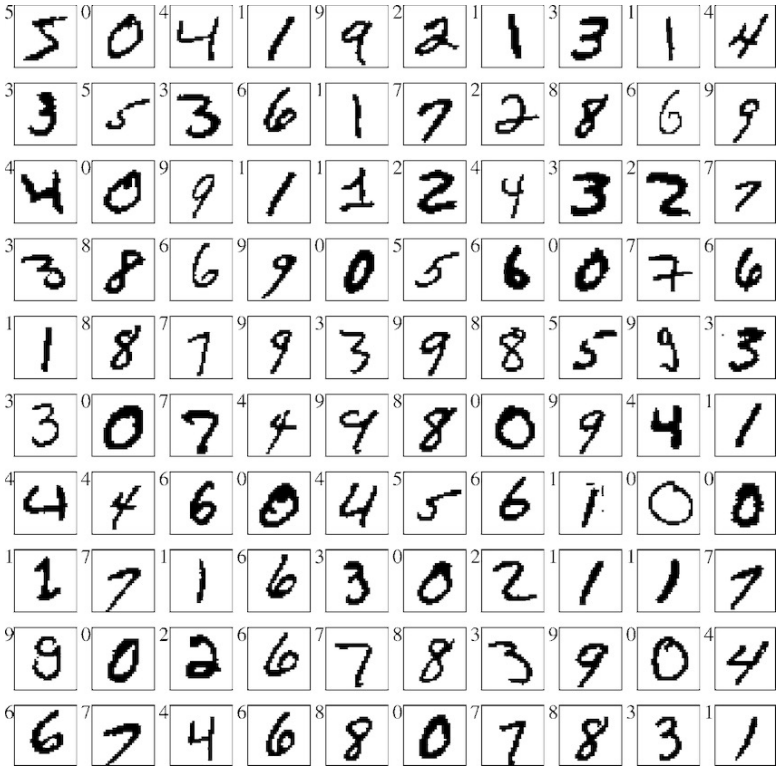
Linear model: softmax classifier

- Vectorize each 28×28 image to a 784-dim vector.
- Add a feature of all ones. (So $\mathbf{x}$ becomes 785-dim.)
- Let $\mathbf{W} \in \mathbb{R}^{10 \times 785}$ contain the parameters.
- Let $\mathbf{z} = \mathbf{W}\mathbf{x} \in \mathbb{R}^{10}$.
- Output a 10-dim vector:

$$\mathbf{f}(\mathbf{x}) = \mathrm{SoftMax}(\mathbf{z}).$$

ok so he added bias at the end

Train a Softmax Classifier



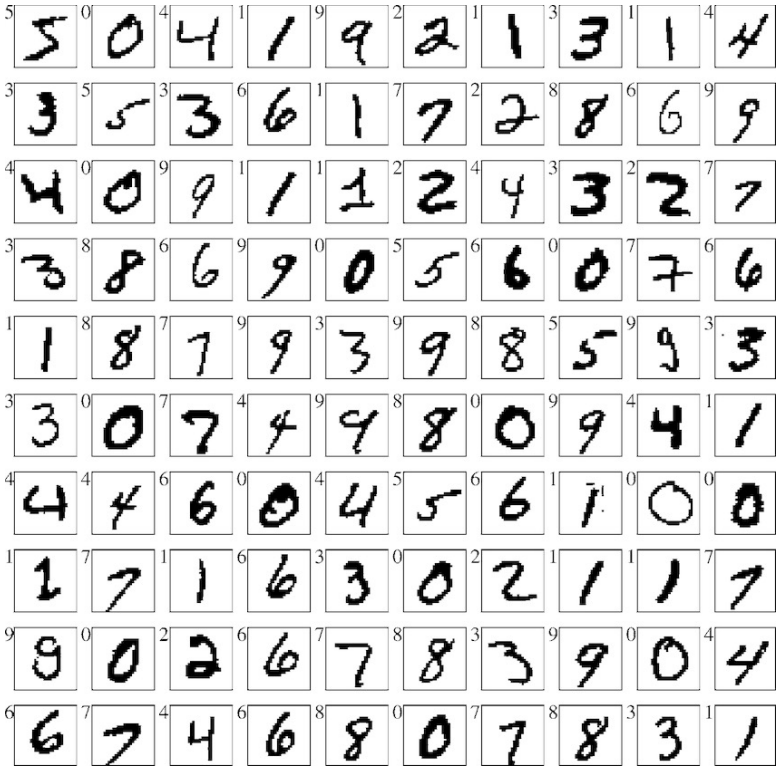
Linear model: softmax classifier

- Vectorize each 28×28 image to a 784-dim vector.
- Add a feature of all ones. (So $\mathbf{x}$ becomes 785-dim.)
- Let $\mathbf{W} \in \mathbb{R}^{10 \times 785}$ contain the parameters.
- Let $\mathbf{z} = \mathbf{W}\mathbf{x} \in \mathbb{R}^{10}$.
- Output a 10-dim vector:

$$\mathbf{f}(\mathbf{x}) = \mathrm{SoftMax}(\mathbf{z}).$$

$$\mathrm{SoftMax}(\mathbf{z}) = \frac{1}{\sum_{i=0}^{9} \exp(z_i)} [\exp(z_0), \cdots, \exp(z_9)]$$

Train a Softmax Classifier

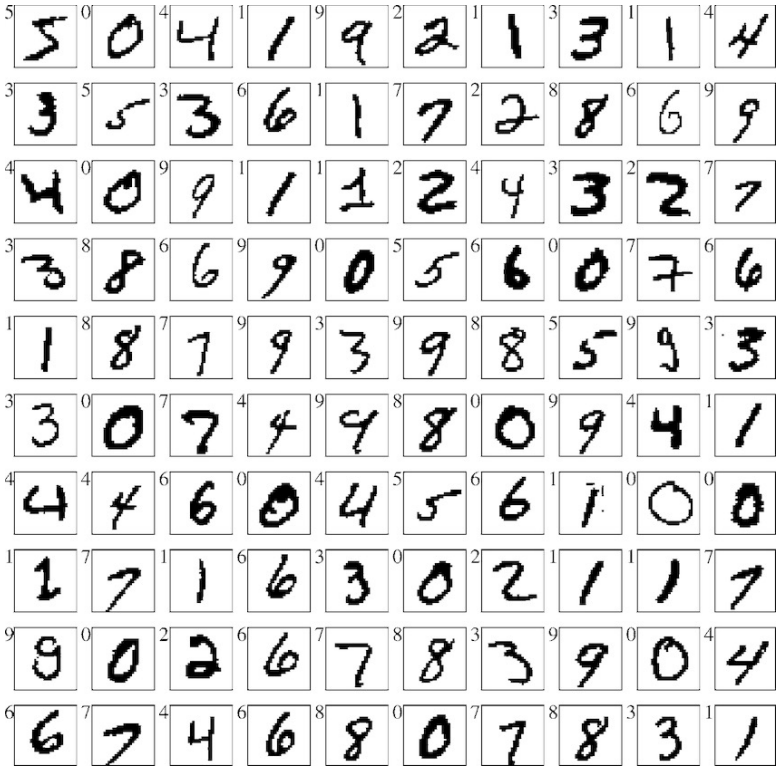


Learn $\mathbf{W} \in \mathbb{R}^{10\times 785}$ from the training data

- One-hot encode of the labels
  - Originally, a label is a scalar in $\{0, 1, 2, \cdots, 9\}$.
  - The one-hot encode $\mathbf{y}$ is a 10-dim vector $\{0,1\}^{10}$.
  - E.g., the one-hot encode of 2 is $[0, 0, 1, 0, 0, 0, 0, 0, 0, 0]$.

means 0 and 1 are the possibilities

Train a Softmax Classifier

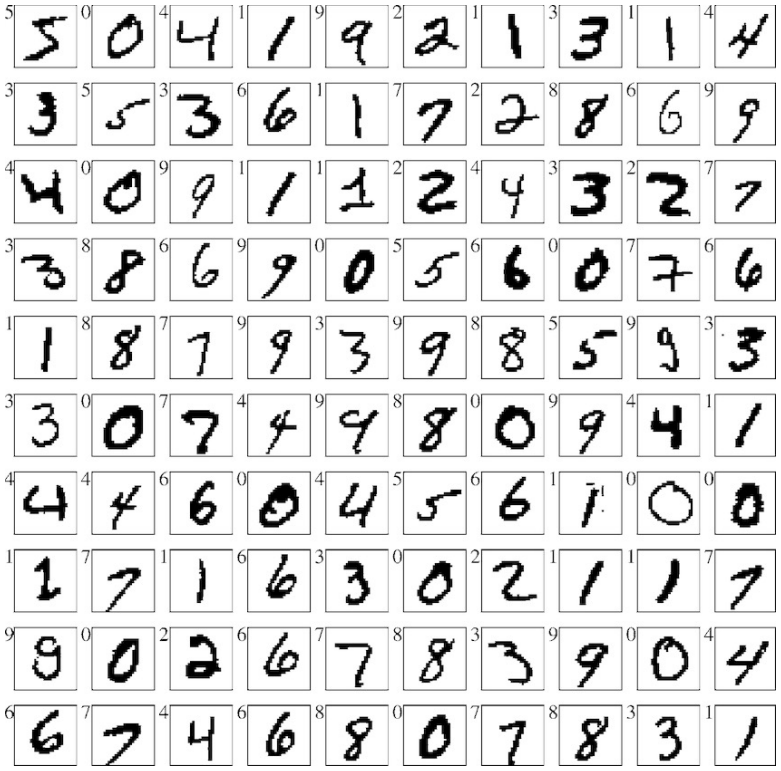


Learn $\mathbf{W} \in \mathbb{R}^{10\times 785}$ from the training data

- One-hot encode of the labels
  - Originally, a label is a scalar in $\{0, 1, 2, \cdots, 9\}$.
  - The one-hot encode $\mathbf{y}$ is a 10-dim vector $\{0,1\}^{10}$.
  - E.g., the one-hot encode of 2 is $[0, 0, 1, 0, 0, 0, 0, 0, 0, 0]$.
- Cross-entropy loss:

$$\text{CrossEntropy}(\mathbf{y}, \mathbf{f}) = -\sum_{i=0}^{9} y_i \cdot \log(f_i).$$

Train a Softmax Classifier



Learn $\mathbf{W} \in \mathbb{R}^{10\times 785}$ from the training data

- One-hot encode of the labels
  - Originally, a label is a scalar in $\{0, 1, 2, \cdots, 9\}$.
  - The one-hot encode $\mathbf{y}$ is a 10-dim vector $\{0,1\}^{10}$.
  - E.g., the one-hot encode of 2 is $[0, 0, 1, 0, 0, 0, 0, 0, 0, 0]$.
- Cross-entropy loss:

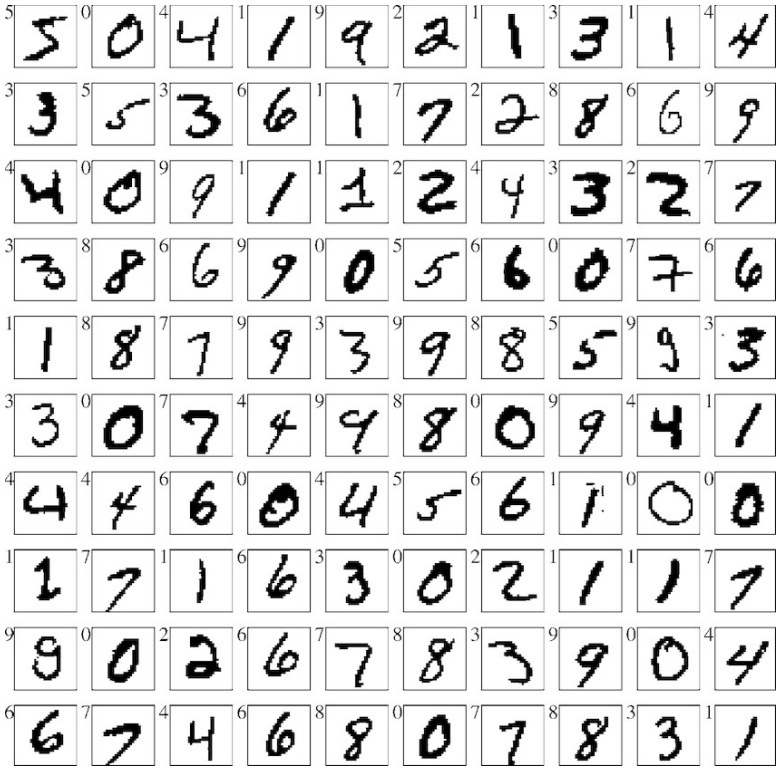
$$\text{CrossEntropy}(\mathbf{y}, \mathbf{f}) = -\sum_{i=0}^{9} y_i \cdot \log(f_i).$$

- Solve the optimization model:

$$\mathbf{W}^\star = \underset{\mathbf{W}}{\text{argmin}} \left\{ \frac{1}{n} \sum_{j=1}^{n} \text{CrossEntropy}\left(\mathbf{y}_j, \mathbf{f}(\mathbf{x}_j)\right) \right\}.$$

$\mathbf{W}$ is the parameter of $\mathbf{f}$

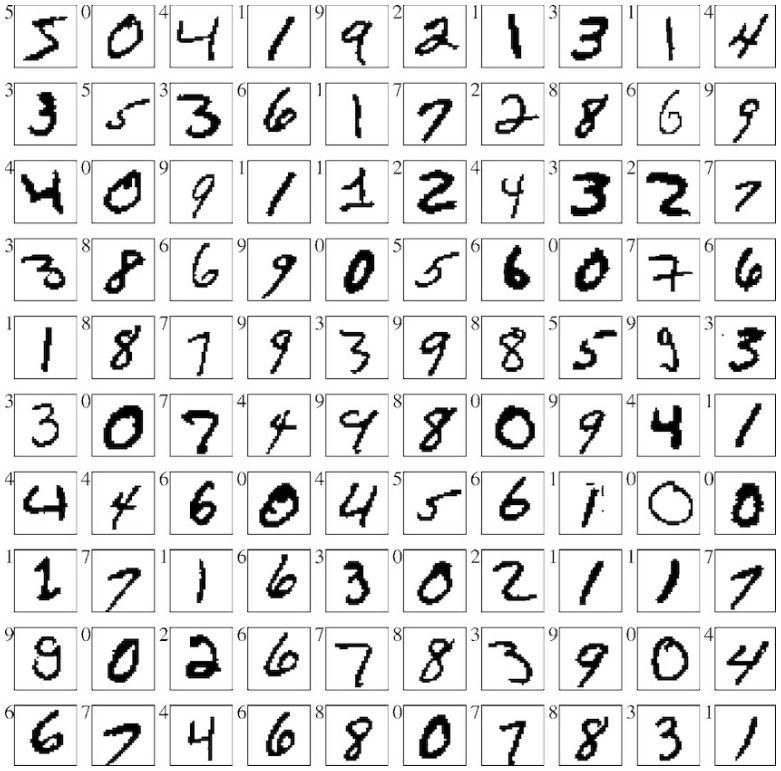
Train a Softmax Classifier



Make prediction for a test sample $\mathbf{x}'$

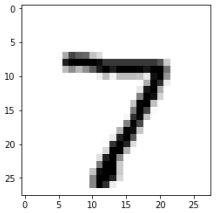
- Now we have $\mathbf{W}^\star \in \mathbb{R}^{10\times 785}$.
- For a test sample $\mathbf{x}'$, compute $\mathbf{z} = \mathbf{W}^\star\mathbf{x}' \in \mathbb{R}^{10}$.
- Make prediction by $\operatorname{argmax} \mathbf{z}$.
  - If the 7-th entry of $\mathbf{z}$ is the largest, then the model thinks the image is digit “7”.

Train a Softmax Classifier



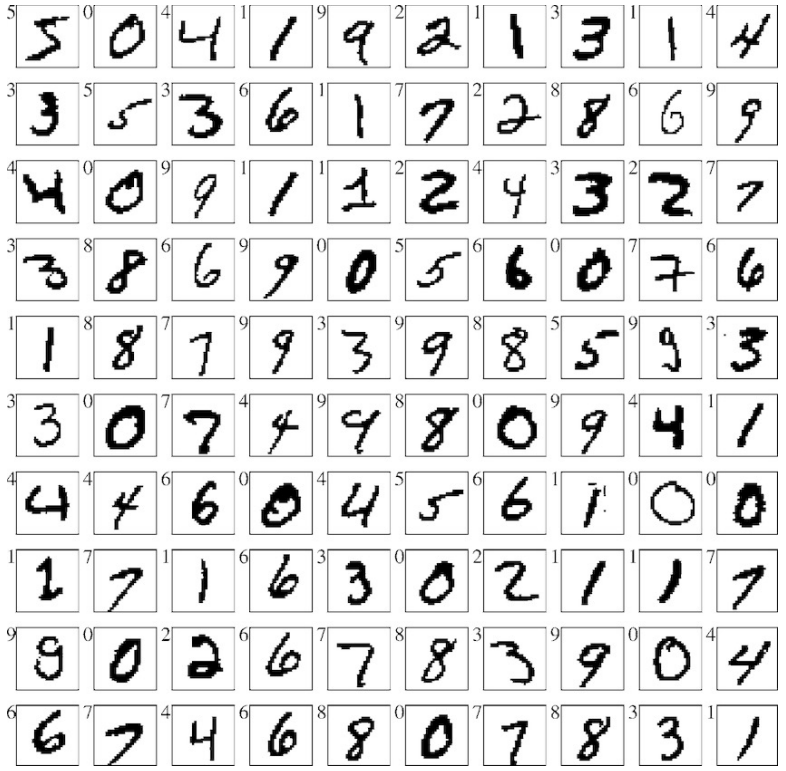
Make prediction for a test sample $\mathbf{x}'$

- Now we have $\mathbf{W}^\star \in \mathbb{R}^{10\times 785}$.
- For a test sample $\mathbf{x}'$, compute $\mathbf{z} = \mathbf{W}^\star\mathbf{x}' \in \mathbb{R}^{10}$.
- Make prediction by $\operatorname{argmax} \mathbf{z}$.
  - If the 7-th entry of $\mathbf{z}$ is the largest, then the model thinks the image is digit "7".



$\mathbf{z} = [-55.7, -141.4, 18.1, 188.3, -91.3, -26.8, -183.6, 411.2, -142.1, 96.2]$

Train a Softmax Classifier



Results

- The training set has $60{,}000$ samples.
- The test set has $10{,}000$ samples.

- The accuracy on the training set is 92.3%.
- The accuracy on the test set is 91.3%.

- Not too bad!
- The accuracy of a random guess is merely 10%.

Train a Softmax Classifier: Re-cap

Define a function $\mathbf{f}: \mathbb{R}^{785} \mapsto \mathbb{R}^{10}$:

- **Input**: vector $\mathbf{x} \in \mathbb{R}^{785}$.
- $\mathbf{z} = \mathbf{W}\,\mathbf{x} \in \mathbb{R}^{10}$.
- **Output**: $\mathbf{f}(\mathbf{x}) = \mathrm{SoftMax}(\mathbf{z})$.

Trainable parameters: $\mathbf{W} \in \mathbb{R}^{10\times 785}$

Train a Softmax Classifier: Re-cap

Define a function $\mathbf{f}: \mathbb{R}^{785} \mapsto \mathbb{R}^{10}$:

- **Input**: vector $\mathbf{x} \in \mathbb{R}^{785}$.
- $\mathbf{z} = \mathbf{W}\,\mathbf{x} \in \mathbb{R}^{10}$.
- **Output**: $\mathbf{f}(\mathbf{x}) = \mathrm{SoftMax}(\mathbf{z})$.

Trainable parameters: $\mathbf{W} \in \mathbb{R}^{10\times 785}$

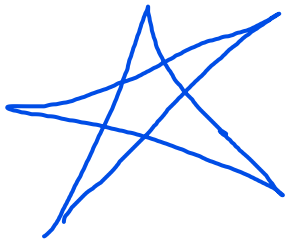
Train the function by empirical risk minimization (ERM):

- **Training set:** $(\mathbf{x}_1, \mathbf{y}_1), \cdots, (\mathbf{x}_n, \mathbf{y}_n) \in \mathbb{R}^{785} \times \mathbb{R}^{10}$.
- **Loss function:** $\mathrm{CrossEntropy}(\mathbf{y}, \mathbf{f}) = -\sum_{i=1}^{10} y_i \cdot \log(\mathbf{f}(\mathbf{x})_i)$.
- **Solve ERM:** $\underset{\mathbf{W}}{\mathrm{argmin}} \left\{ \frac{1}{n} \sum_{j=1}^{n} \mathrm{CrossEntropy}\left(\mathbf{y}_j, \mathbf{f}\left(\mathbf{x}_j\right)\right) \right\}$.

Train a Softmax Classifier: Re-cap

- **How to solve** $\underset{\mathbf{W}}{\operatorname{argmin}} \left\{ \frac{1}{n} \sum_{j=1}^{n} \text{CrossEntropy}\left(\mathbf{y}_j, \mathbf{f}\left(\mathbf{x}_j\right)\right) \right\}$?

- **Stochastic gradient descent (SGD) with momentum** repeats:
  1. Randomly pick j from $\{1, 2, \cdots, n\}$.
  2. Evaluate the gradient $\mathbf{G}_j = \frac{\partial \text{CrossEntropy}\left(\mathbf{y}_j, \mathbf{f}\left(\mathbf{x}_j\right)\right)}{\partial \mathbf{W}} \Big|_{\mathbf{W}=\mathbf{W}_{\text{old}}}$.
  3. Update the momentum: $\mathbf{V}_{\text{new}} = \beta \mathbf{V}_{\text{old}} + \mathbf{G}_j$.
  4. Update $\mathbf{W}$ by $\mathbf{W}_{\text{new}} \leftarrow \mathbf{W}_{\text{old}} - \alpha\, \mathbf{V}_{\text{new}}$.



Fully-Connected Neural Network (Multi-layer Perceptron)

Softmax Classifier

Define a function $\mathbf{f}: \mathbb{R}^{785} \mapsto \mathbb{R}^{10}$:

- **Input**: vector $\mathbf{x}^{(0)} \in \mathbb{R}^{785}$.
- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\,\mathbf{x}^{(0)} \in \mathbb{R}^{10}$.
- $\mathbf{x}^{(1)} = \mathrm{SoftMax}\left(\mathbf{z}^{(1)}\right) \in \mathbb{R}^{10}$.
- **Output**: $\mathbf{f}\left(\mathbf{x}^{(0)}\right) = \mathbf{x}^{(1)}$.

Trainable parameter:

- $\mathbf{W}^{(0)} \in \mathbb{R}^{10\times 785}$.

From Linear Model to Neural Network

Define a function $\mathbf{f}: \mathbb{R}^{785} \mapsto \mathbb{R}^{10}$:

- **Input**: vector $\mathbf{x}^{(0)} \in \mathbb{R}^{785}$.
- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\,\mathbf{x}^{(0)} \in \mathbb{R}^{d_1}$.
- $\mathbf{x}^{(1)} = \max\{\mathbf{0},\ \mathbf{z}^{(1)}\} \in \mathbb{R}^{d_1}$.

Trainable parameters:

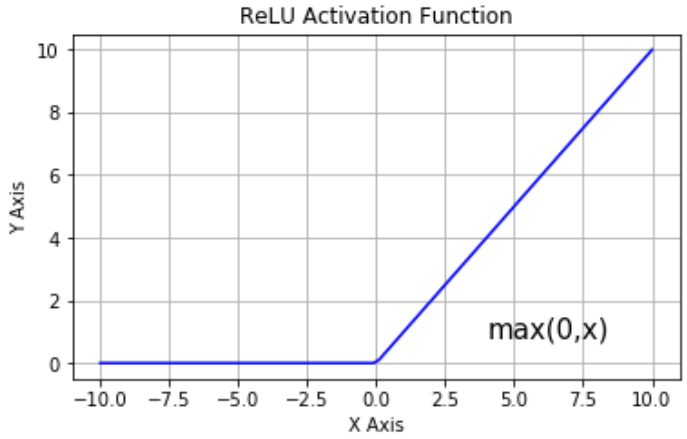
- $\mathbf{W}^{(0)} \in \mathbb{R}^{d_1 \times 785}$,

From Linear Model to Neural Network

Define a function $\mathbf{f}: \mathbb{R}^{785} \mapsto \mathbb{R}^{10}$:

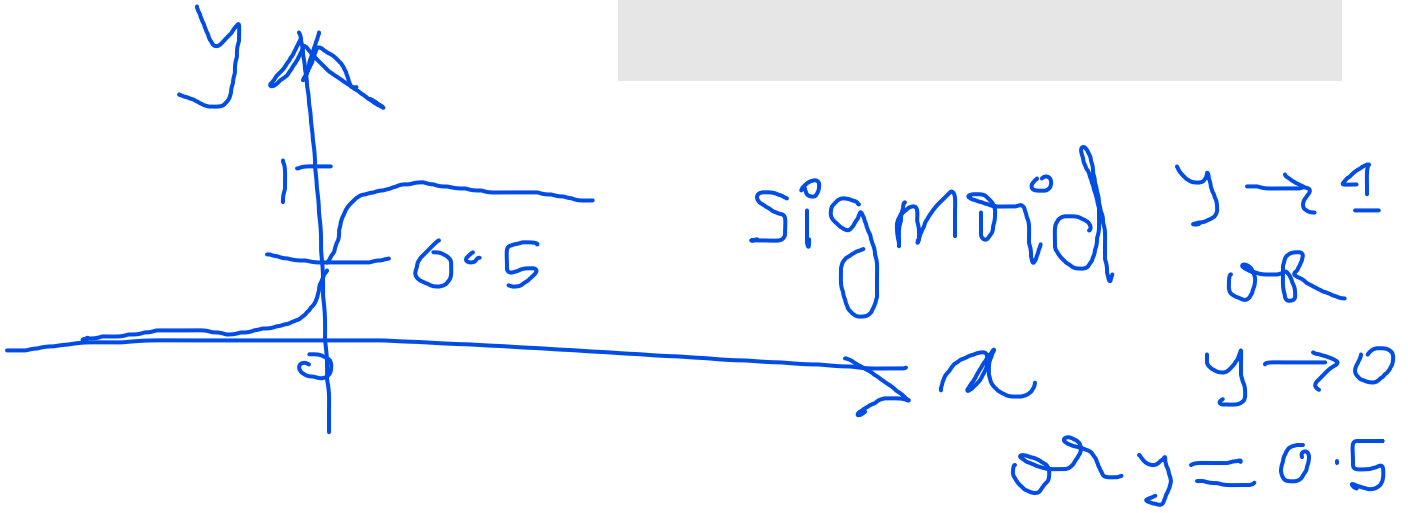
- **Input**: vector $\mathbf{x}^{(0)} \in \mathbb{R}^{785}$.
- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\,\mathbf{x}^{(0)} \in \mathbb{R}^{d_1}$.
- $\mathbf{x}^{(1)} = \max\{\mathbf{0},\ \mathbf{z}^{(1)}\} \in \mathbb{R}^{d_1}$.

ReLU (activation function)



Trainable parameters:

- $\mathbf{W}^{(0)} \in \mathbb{R}^{d_1 \times 785}$,



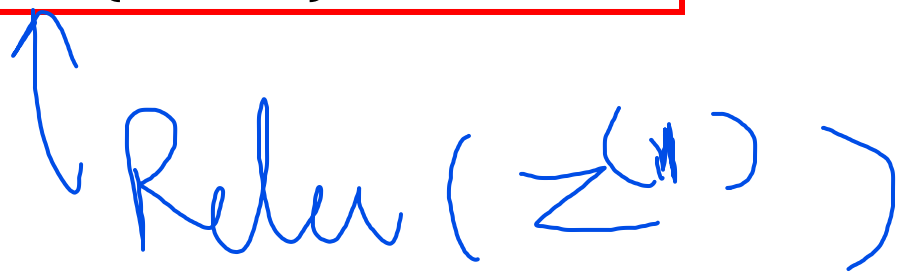
From Linear Model to Neural Network

Define a function $\mathbf{f}: \mathbb{R}^{785} \mapsto \mathbb{R}^{10}$:

- **Input**: vector $\mathbf{x}^{(0)} \in \mathbb{R}^{785}$.

- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\,\mathbf{x}^{(0)} \in \mathbb{R}^{d_1}$.
- $\mathbf{x}^{(1)} = \max\{\mathbf{0},\ \mathbf{z}^{(1)}\} \in \mathbb{R}^{d_1}$.

Hidden Layer 1



Trainable parameters:

- $\mathbf{W}^{(0)} \in \mathbb{R}^{d_1 \times 785}$,

From Linear Model to Neural Network

Define a function $\mathbf{f}: \mathbb{R}^{785} \mapsto \mathbb{R}^{10}$:

- **Input**: vector $\mathbf{x}^{(0)} \in \mathbb{R}^{785}$.

Input Layer

- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\,\mathbf{x}^{(0)} \in \mathbb{R}^{d_1}$.
- $\mathbf{x}^{(1)} = \max\{\mathbf{0},\ \mathbf{z}^{(1)}\} \in \mathbb{R}^{d_1}$.

Hidden Layer 1

- $\mathbf{z}^{(2)} = \mathbf{W}^{(1)}\,\mathbf{x}^{(1)} \in \mathbb{R}^{d_2}$.

Trainable parameters:

- $\mathbf{W}^{(0)} \in \mathbb{R}^{d_1 \times 785}$,
- $\mathbf{W}^{(1)} \in \mathbb{R}^{d_2 \times d_1}$,

It should be $\mathbf{z}^{(2)} = \mathbf{W}^{(1)}\,\mathbf{x}^{(1)} + \mathbf{b}^{(1)}$ in practice.
I leave out $\mathbf{b} \in \mathbb{R}^{d_2}$ in the slides for simplicity.

From Linear Model to Neural Network

Define a function $\mathbf{f}: \mathbb{R}^{785} \mapsto \mathbb{R}^{10}$:

- **Input**: vector $\mathbf{x}^{(0)} \in \mathbb{R}^{785}$.

- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\,\mathbf{x}^{(0)} \in \mathbb{R}^{d_1}$.
- $\mathbf{x}^{(1)} = \max\{\mathbf{0},\ \mathbf{z}^{(1)}\} \in \mathbb{R}^{d_1}$.

Hidden Layer 1

- $\mathbf{z}^{(2)} = \mathbf{W}^{(1)}\,\mathbf{x}^{(1)} \in \mathbb{R}^{d_2}$.
- $\mathbf{x}^{(2)} = \max\{\mathbf{0},\ \mathbf{z}^{(2)}\} \in \mathbb{R}^{d_2}$.

Hidden Layer 2

Trainable parameters:

- $\mathbf{W}^{(0)} \in \mathbb{R}^{d_1 \times 785}$,
- $\mathbf{W}^{(1)} \in \mathbb{R}^{d_2 \times d_1}$,

From Linear Model to Neural Network

Define a function $\mathbf{f}: \mathbb{R}^{785} \mapsto \mathbb{R}^{10}$:

- **Input**: vector $\mathbf{x}^{(0)} \in \mathbb{R}^{785}$.

- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\mathbf{x}^{(0)} \in \mathbb{R}^{d_1}$.
- $\mathbf{x}^{(1)} = \max\{\mathbf{0}, \mathbf{z}^{(1)}\} \in \mathbb{R}^{d_1}$.

Hidden Layer 1

ReLU

- $\mathbf{z}^{(2)} = \mathbf{W}^{(1)}\mathbf{x}^{(1)} \in \mathbb{R}^{d_2}$.
- $\mathbf{x}^{(2)} = \max\{\mathbf{0}, \mathbf{z}^{(2)}\} \in \mathbb{R}^{d_2}$.

Hidden Layer 2

ReLU

- $\mathbf{z}^{(3)} = \mathbf{W}^{(2)}\mathbf{x}^{(2)} \in \mathbb{R}^{10}$.
- $\mathbf{x}^{(3)} = \mathrm{SoftMax}(\mathbf{z}^{(3)}) \in \mathbb{R}^{10}$.

Output Layer

SoftMax

Trainable parameters:

- $\mathbf{W}^{(0)} \in \mathbb{R}^{d_1 \times 785}$,
- $\mathbf{W}^{(1)} \in \mathbb{R}^{d_2 \times d_1}$,
- $\mathbf{W}^{(2)} \in \mathbb{R}^{10 \times d_2}$.

From Linear Model to Neural Network

Define a function $\mathbf{f}: \mathbb{R}^{785} \mapsto \mathbb{R}^{10}$:

- **Input**: vector $\mathbf{x}^{(0)} \in \mathbb{R}^{785}$.

- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\,\mathbf{x}^{(0)} \in \mathbb{R}^{d_1}$.
- $\mathbf{x}^{(1)} = \max\{\mathbf{0},\ \mathbf{z}^{(1)}\} \in \mathbb{R}^{d_1}$.

Hidden Layer 1

- $\mathbf{z}^{(2)} = \mathbf{W}^{(1)}\,\mathbf{x}^{(1)} \in \mathbb{R}^{d_2}$.
- $\mathbf{x}^{(2)} = \max\{\mathbf{0},\ \mathbf{z}^{(2)}\} \in \mathbb{R}^{d_2}$.

Hidden Layer 2

- $\mathbf{z}^{(3)} = \mathbf{W}^{(2)}\,\mathbf{x}^{(2)} \in \mathbb{R}^{10}$.
- $\mathbf{x}^{(3)} = \mathrm{SoftMax}\left(\mathbf{z}^{(3)}\right) \in \mathbb{R}^{10}$.

Output Layer

- **Output**: $\mathbf{f}\left(\mathbf{x}^{(0)}\right) = \mathbf{x}^{(3)}$.

Trainable parameters:

- $\mathbf{W}^{(0)} \in \mathbb{R}^{d_1 \times 785}$,
- $\mathbf{W}^{(1)} \in \mathbb{R}^{d_2 \times d_1}$,
- $\mathbf{W}^{(2)} \in \mathbb{R}^{10 \times d_2}$.

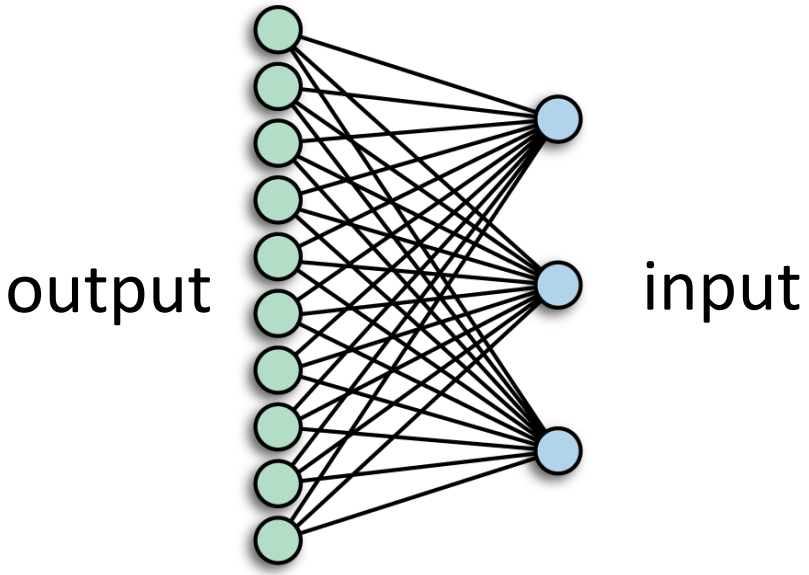
Fully-Connected Layer

Define a function $\mathbf{f}: \mathbb{R}^{785} \mapsto \mathbb{R}^{10}$:

- **Input:** vector $\mathbf{x}^{(0)} \in \mathbb{R}^{785}$.
- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\,\mathbf{x}^{(0)} \in \mathbb{R}^{d_1}$.
- $\mathbf{x}^{(1)} = \max\{\mathbf{0},\ \mathbf{z}^{(1)}\} \in \mathbb{R}^{d_1}$.
- $\mathbf{z}^{(2)} = \mathbf{W}^{(1)}\,\mathbf{x}^{(1)} \in \mathbb{R}^{d_2}$.
- $\mathbf{x}^{(2)} = \max\{\mathbf{0},\ \mathbf{z}^{(2)}\} \in \mathbb{R}^{d_2}$.
- $\mathbf{z}^{(3)} = \mathbf{W}^{(2)}\,\mathbf{x}^{(2)} \in \mathbb{R}^{10}$.
- $\mathbf{x}^{(3)} = \mathrm{SoftMax}(\mathbf{z}^{(3)}) \in \mathbb{R}^{10}$.
- **Output:** $\mathbf{f}(\mathbf{x}^{(0)}) = \mathbf{x}^{(3)}$.

"Fully-Connected" or "Dense" Layer

Each entry of $\mathbf{z}^{(1)}$ depends on (i.e., connected to) all the entries of $\mathbf{x}^{(0)}$.



Activation Functions

Define a function $\mathbf{f}: \mathbb{R}^{785} \mapsto \mathbb{R}^{10}$:

- **Input:** vector $\mathbf{x}^{(0)} \in \mathbb{R}^{785}$.
- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\,\mathbf{x}^{(0)} \in \mathbb{R}^{d_1}$.
- $\mathbf{x}^{(1)} = \max\{\mathbf{0},\ \mathbf{z}^{(1)}\} \in \mathbb{R}^{d_1}$. ReLU
- $\mathbf{z}^{(2)} = \mathbf{W}^{(1)}\,\mathbf{x}^{(1)} \in \mathbb{R}^{d_2}$.
- $\mathbf{x}^{(2)} = \max\{\mathbf{0},\ \mathbf{z}^{(2)}\} \in \mathbb{R}^{d_2}$. ReLU
- $\mathbf{z}^{(3)} = \mathbf{W}^{(2)}\,\mathbf{x}^{(2)} \in \mathbb{R}^{10}$.
- $\mathbf{x}^{(3)} = \mathrm{SoftMax}(\mathbf{z}^{(3)}) \in \mathbb{R}^{10}$. SoftMax
- **Output:** $\mathbf{f}(\mathbf{x}^{(0)}) = \mathbf{x}^{(3)}$.

Activation Functions

Define a function $\mathbf{f}: \mathbb{R}^{785} \mapsto \mathbb{R}^{10}$:

- **Input:** vector $\mathbf{x}^{(0)} \in \mathbb{R}^{785}$.
- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\,\mathbf{x}^{(0)} \in \mathbb{R}^{d_1}$.
- $\mathbf{x}^{(1)} = \max\{\mathbf{0},\ \mathbf{z}^{(1)}\} \in \mathbb{R}^{d_1}$.
- $\mathbf{z}^{(2)} = \mathbf{W}^{(1)}\,\mathbf{x}^{(1)} \in \mathbb{R}^{d_2}$.
- $\mathbf{x}^{(2)} = \max\{\mathbf{0},\ \mathbf{z}^{(2)}\} \in \mathbb{R}^{d_2}$.
- $\mathbf{z}^{(3)} = \mathbf{W}^{(2)}\,\mathbf{x}^{(2)} \in \mathbb{R}^{10}$.
- $\mathbf{x}^{(3)} = \mathrm{SoftMax}\left(\mathbf{z}^{(3)}\right) \in \mathbb{R}^{10}$.
- **Output:** $\mathbf{f}\left(\mathbf{x}^{(0)}\right) = \mathbf{x}^{(3)}$.

SoftMax

- Use SoftMax because this is a **multi-class classification** problem.

Activation Functions

Define a function $\mathbf{f}: \mathbb{R}^{785} \mapsto \mathbb{R}^{10}$:

- **Input:** vector $\mathbf{x}^{(0)} \in \mathbb{R}^{785}$.
- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\,\mathbf{x}^{(0)} \in \mathbb{R}^{d_1}$.
- $\mathbf{x}^{(1)} = \max\{\mathbf{0},\ \mathbf{z}^{(1)}\} \in \mathbb{R}^{d_1}$.
- $\mathbf{z}^{(2)} = \mathbf{W}^{(1)}\,\mathbf{x}^{(1)} \in \mathbb{R}^{d_2}$.
- $\mathbf{x}^{(2)} = \max\{\mathbf{0},\ \mathbf{z}^{(2)}\} \in \mathbb{R}^{d_2}$.
- $\mathbf{z}^{(3)} = \mathbf{W}^{(2)}\,\mathbf{x}^{(2)} \in \mathbb{R}^{10}$.
- $\mathbf{x}^{(3)} = \mathrm{SoftMax}\left(\mathbf{z}^{(3)}\right) \in \mathbb{R}^{10}$.
- **Output:** $\mathbf{f}\left(\mathbf{x}^{(0)}\right) = \mathbf{x}^{(3)}$.

SoftMax

- Use Sigmoid or tanh function for **binary classification problem**.
- **For regression**: same as linear activation
  - No activation function, if the labels are in $\mathbb{R}$.
  - Use ReLU if the targets are positive.
- Use SoftMax because this is a **multi-class classification** problem.

Activation Functions

Define a function $\mathbf{f}: \mathbb{R}^{785} \mapsto \mathbb{R}^{10}$:

- **Input:** vector $\mathbf{x}^{(0)} \in \mathbb{R}^{785}$.
- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\mathbf{x}^{(0)} \in \mathbb{R}^{d_1}$.
- $\mathbf{x}^{(1)} = \max\{\mathbf{0}, \mathbf{z}^{(1)}\} \in \mathbb{R}^{d_1}$. ReLU
- $\mathbf{z}^{(2)} = \mathbf{W}^{(1)}\mathbf{x}^{(1)} \in \mathbb{R}^{d_2}$.
- $\mathbf{x}^{(2)} = \max\{\mathbf{0}, \mathbf{z}^{(2)}\} \in \mathbb{R}^{d_2}$. ReLU
- $\mathbf{z}^{(3)} = \mathbf{W}^{(2)}\mathbf{x}^{(2)} \in \mathbb{R}^{10}$.
- $\mathbf{x}^{(3)} = \mathrm{SoftMax}(\mathbf{z}^{(3)}) \in \mathbb{R}^{10}$.
- **Output:** $\mathbf{f}(\mathbf{x}^{(0)}) = \mathbf{x}^{(3)}$.

Question: Why bothering using ReLU?

- Without the two activation functions, $\mathbf{z}^{(3)}$ would be a linear function of $\mathbf{x}^{(0)}$.
- A linear function can be represented by $\mathbf{z}^{(3)} = \mathbf{W}\mathbf{x}^{(0)}$.
- The neural network would be equally expressive as a linear model!!!

Gradient and Backpropagation

Train the Neural Network

Define a function $\mathbf{f}: \mathbb{R}^{785} \mapsto \mathbb{R}^{10}$:

- **Input**: vector $\mathbf{x}^{(0)} \in \mathbb{R}^{785}$.
- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\,\mathbf{x}^{(0)} \in \mathbb{R}^{d_1}$.
- $\mathbf{x}^{(1)} = \max\{\mathbf{0},\ \mathbf{z}^{(1)}\} \in \mathbb{R}^{d_1}$.
- $\mathbf{z}^{(2)} = \mathbf{W}^{(1)}\,\mathbf{x}^{(1)} \in \mathbb{R}^{d_2}$.
- $\mathbf{x}^{(2)} = \max\{\mathbf{0},\ \mathbf{z}^{(2)}\} \in \mathbb{R}^{d_2}$.
- $\mathbf{z}^{(3)} = \mathbf{W}^{(2)}\,\mathbf{x}^{(2)} \in \mathbb{R}^{10}$.
- $\mathbf{x}^{(3)} = \mathrm{SoftMax}\left(\mathbf{z}^{(3)}\right) \in \mathbb{R}^{10}$.
- **Output**: $\mathbf{f}\left(\mathbf{x}^{(0)}\right) = \mathbf{x}^{(3)}$.

Train the Neural Network

Define a function $\mathbf{f}: \mathbb{R}^{785} \mapsto \mathbb{R}^{10}$:

- **Input:** vector $\mathbf{x}^{(0)} \in \mathbb{R}^{785}$.
- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\,\mathbf{x}^{(0)} \in \mathbb{R}^{d_1}$.
- $\mathbf{x}^{(1)} = \max\{\mathbf{0},\ \mathbf{z}^{(1)}\} \in \mathbb{R}^{d_1}$.
- $\mathbf{z}^{(2)} = \mathbf{W}^{(1)}\,\mathbf{x}^{(1)} \in \mathbb{R}^{d_2}$.
- $\mathbf{x}^{(2)} = \max\{\mathbf{0},\ \mathbf{z}^{(2)}\} \in \mathbb{R}^{d_2}$.
- $\mathbf{z}^{(3)} = \mathbf{W}^{(2)}\,\mathbf{x}^{(2)} \in \mathbb{R}^{10}$.
- $\mathbf{x}^{(3)} = \mathrm{SoftMax}(\mathbf{z}^{(3)}) \in \mathbb{R}^{10}$.
- **Output:** $\mathbf{f}(\mathbf{x}^{(0)}) = \mathbf{x}^{(3)}$. $\hat{y}$

Build an optimization model:

$$\underset{\mathbf{W}^{(0)},\mathbf{W}^{(1)},\mathbf{W}^{(2)}}{\operatorname{argmin}} \left\{ \frac{1}{n}\sum_{j=1}^{n} \mathrm{Loss}\big(\mathbf{f}(\mathbf{x}_j),\ \mathbf{y}_j\big) \right\}$$

avg

E.g., the cross-entropy loss

Train the Neural Network

Define a function $\mathbf{f}: \mathbb{R}^{785} \mapsto \mathbb{R}^{10}$:

- **Input:** vector $\mathbf{x}^{(0)} \in \mathbb{R}^{785}$.
- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\,\mathbf{x}^{(0)} \in \mathbb{R}^{d_1}$.
- $\mathbf{x}^{(1)} = \max\{\mathbf{0},\ \mathbf{z}^{(1)}\} \in \mathbb{R}^{d_1}$.
- $\mathbf{z}^{(2)} = \mathbf{W}^{(1)}\,\mathbf{x}^{(1)} \in \mathbb{R}^{d_2}$.
- $\mathbf{x}^{(2)} = \max\{\mathbf{0},\ \mathbf{z}^{(2)}\} \in \mathbb{R}^{d_2}$.
- $\mathbf{z}^{(3)} = \mathbf{W}^{(2)}\,\mathbf{x}^{(2)} \in \mathbb{R}^{10}$.
- $\mathbf{x}^{(3)} = \mathrm{SoftMax}\left(\mathbf{z}^{(3)}\right) \in \mathbb{R}^{10}$.
- **Output:** $\mathbf{f}\left(\mathbf{x}^{(0)}\right) = \mathbf{x}^{(3)}$.

How to solve

$$\underset{\mathbf{W}^{(0)},\mathbf{W}^{(1)},\mathbf{W}^{(2)}}{\operatorname{argmin}} \left\{ \frac{1}{n}\sum_{j=1}^{n} \mathrm{Loss}\left(\mathbf{f}(\mathbf{x}_j),\ \mathbf{y}_j \right) \right\} ?$$

Stochastic gradient descent (SGD)

Train the Neural Network

Define a function $\mathbf{f}: \mathbb{R}^{785} \mapsto \mathbb{R}^{10}$:

- **Input:** vector $\mathbf{x}^{(0)} \in \mathbb{R}^{785}$.
- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\,\mathbf{x}^{(0)} \in \mathbb{R}^{d_1}$.
- $\mathbf{x}^{(1)} = \max\{\mathbf{0},\ \mathbf{z}^{(1)}\} \in \mathbb{R}^{d_1}$.
- $\mathbf{z}^{(2)} = \mathbf{W}^{(1)}\,\mathbf{x}^{(1)} \in \mathbb{R}^{d_2}$.
- $\mathbf{x}^{(2)} = \max\{\mathbf{0},\ \mathbf{z}^{(2)}\} \in \mathbb{R}^{d_2}$.
- $\mathbf{z}^{(3)} = \mathbf{W}^{(2)}\,\mathbf{x}^{(2)} \in \mathbb{R}^{10}$.
- $\mathbf{x}^{(3)} = \mathrm{SoftMax}\left(\mathbf{z}^{(3)}\right) \in \mathbb{R}^{10}$.
- **Output:** $\mathbf{f}\left(\mathbf{x}^{(0)}\right) = \mathbf{x}^{(3)}$.

How to solve

$$\underset{\mathbf{W}^{(0)},\mathbf{W}^{(1)},\mathbf{W}^{(2)}}{\operatorname{argmin}} \left\{ \frac{1}{n}\sum_{j=1}^{n} \mathrm{Loss}\left(\mathbf{f}(\mathbf{x}_j),\ \mathbf{y}_j \right) \right\} ?$$

→ **Stochastic gradient descent (SGD):**

- Randomly pick j from $\{1, 2, \cdots, n\}$.
- Compute the stochastic gradient w.r.t. $\mathbf{W}^{(0)}$ at the current iteration $\mathbf{W}_{\mathrm{old}}^{(0)}$:

$$\mathbf{g}_j^{(0)} = \frac{\partial\, \mathrm{Loss}\left(\mathbf{f}(\mathbf{x}_j), \mathbf{y}_j \right)}{\partial \mathbf{W}^{(0)}} \Big|_{\mathbf{W}^{(0)} = \mathbf{W}_{\mathrm{old}}^{(0)}}.$$

- Update $\mathbf{W}^{(0)}$: $\mathbf{W}_{\mathrm{new}}^{(0)} = \mathbf{W}_{\mathrm{old}}^{(0)} - \alpha\, \mathbf{g}_j^{(0)}$.
- Do the same for $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$.

Backpropagation

Define a function $\mathbf{f}: \mathbb{R}^{785} \mapsto \mathbb{R}^{10}$:

- **Input:** vector $\mathbf{x}^{(0)} \in \mathbb{R}^{785}$.
- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\,\mathbf{x}^{(0)} \in \mathbb{R}^{d_1}$.
- $\mathbf{x}^{(1)} = \max\{\mathbf{0},\ \mathbf{z}^{(1)}\} \in \mathbb{R}^{d_1}$.
- $\mathbf{z}^{(2)} = \mathbf{W}^{(1)}\,\mathbf{x}^{(1)} \in \mathbb{R}^{d_2}$.
- $\mathbf{x}^{(2)} = \max\{\mathbf{0},\ \mathbf{z}^{(2)}\} \in \mathbb{R}^{d_2}$.
- $\mathbf{z}^{(3)} = \mathbf{W}^{(2)}\,\mathbf{x}^{(2)} \in \mathbb{R}^{10}$.
- $\mathbf{x}^{(3)} = \mathrm{SoftMax}\left(\mathbf{z}^{(3)}\right) \in \mathbb{R}^{10}$.
- **Output:** $\mathbf{f}\left(\mathbf{x}^{(0)}\right) = \mathbf{x}^{(3)}$.

How to compute $\dfrac{\partial\, \mathrm{Loss}\left(\mathbf{f}(\mathbf{x}_j), \mathbf{y}_j\right)}{\partial \mathbf{W}^{(k)}}$ **?**

Backpropagation

Define a function $\mathbf{f}: \mathbb{R}^{785} \mapsto \mathbb{R}^{10}$:

- **Input**: vector $\mathbf{x}^{(0)} \in \mathbb{R}^{785}$.
- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\,\mathbf{x}^{(0)} \in \mathbb{R}^{d_1}$.
- $\mathbf{x}^{(1)} = \max\{\mathbf{0},\ \mathbf{z}^{(1)}\} \in \mathbb{R}^{d_1}$.
- $\mathbf{z}^{(2)} = \mathbf{W}^{(1)}\,\mathbf{x}^{(1)} \in \mathbb{R}^{d_2}$.
- $\mathbf{x}^{(2)} = \max\{\mathbf{0},\ \mathbf{z}^{(2)}\} \in \mathbb{R}^{d_2}$.
- $\mathbf{z}^{(3)} = \mathbf{W}^{(2)}\,\mathbf{x}^{(2)} \in \mathbb{R}^{10}$.
- $\mathbf{x}^{(3)} = \mathrm{SoftMax}\left(\mathbf{z}^{(3)}\right) \in \mathbb{R}^{10}$.
- **Output**: $\mathbf{f}\left(\mathbf{x}^{(0)}\right) = \mathbf{x}^{(3)}$.

How to compute $\dfrac{\partial\, \mathrm{Loss}\left(\mathbf{f}\left(\mathbf{x}_j\right), \mathbf{y}_j\right)}{\partial \mathbf{W}^{(k)}}$ **?**

Backpropagation:

- Denote $L = \mathrm{Loss}\left(\mathbf{f}\left(\mathbf{x}_j\right),\ \mathbf{y}_j\right)$.

Backpropagation

Define a function $\mathbf{f}: \mathbb{R}^{785} \mapsto \mathbb{R}^{10}$:

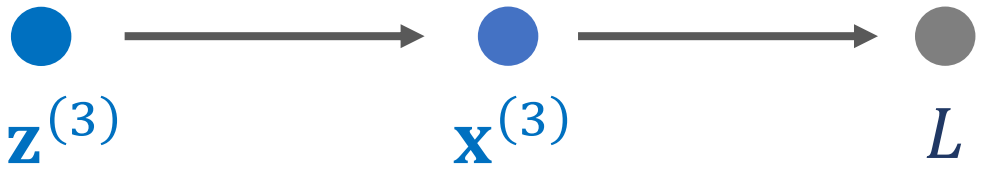
- Input: vector $\mathbf{x}^{(0)} \in \mathbb{R}^{785}$.
- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\,\mathbf{x}^{(0)} \in \mathbb{R}^{d_1}$.
- $\mathbf{x}^{(1)} = \max\{\mathbf{0},\ \mathbf{z}^{(1)}\} \in \mathbb{R}^{d_1}$.
- $\mathbf{z}^{(2)} = \mathbf{W}^{(1)}\,\mathbf{x}^{(1)} \in \mathbb{R}^{d_2}$.
- $\mathbf{x}^{(2)} = \max\{\mathbf{0},\ \mathbf{z}^{(2)}\} \in \mathbb{R}^{d_2}$.
- $\mathbf{z}^{(3)} = \mathbf{W}^{(2)}\,\mathbf{x}^{(2)} \in \mathbb{R}^{10}$.
- $\mathbf{x}^{(3)} = \mathrm{SoftMax}\left(\mathbf{z}^{(3)}\right) \in \mathbb{R}^{10}$.
- **Output**: $\mathbf{f}\left(\mathbf{x}^{(0)}\right) = \mathbf{x}^{(3)}$.

How to compute $\dfrac{\partial\, \mathrm{Loss}\left(\mathbf{f}\left(\mathbf{x}_j\right), \mathbf{y}_j\right)}{\partial \mathbf{W}^{(k)}}$ **?**

Backpropagation:

- Denote $L = \mathrm{Loss}\left(\mathbf{f}\left(\mathbf{x}_j\right),\ \mathbf{y}_j\right)$.
- Compute $\dfrac{\partial L}{\partial \mathbf{z}^{(3)}}$.

10-dim vector



Backpropagation

Define a function $\mathbf{f}: \mathbb{R}^{785} \mapsto \mathbb{R}^{10}$:

- Input: vector $\mathbf{x}^{(0)} \in \mathbb{R}^{785}$.
- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\,\mathbf{x}^{(0)} \in \mathbb{R}^{d_1}$.
- $\mathbf{x}^{(1)} = \max\{\mathbf{0},\ \mathbf{z}^{(1)}\} \in \mathbb{R}^{d_1}$.
- $\mathbf{z}^{(2)} = \mathbf{W}^{(1)}\,\mathbf{x}^{(1)} \in \mathbb{R}^{d_2}$.
- $\mathbf{x}^{(2)} = \max\{\mathbf{0},\ \mathbf{z}^{(2)}\} \in \mathbb{R}^{d_2}$.
- $\mathbf{z}^{(3)} = \mathbf{W}^{(2)}\,\mathbf{x}^{(2)} \in \mathbb{R}^{10}$.
- $\mathbf{x}^{(3)} = \mathrm{SoftMax}(\mathbf{z}^{(3)}) \in \mathbb{R}^{10}$.
- Output: $\mathbf{f}(\mathbf{x}^{(0)}) = \mathbf{x}^{(3)}$.

How to compute $\dfrac{\partial\, \mathrm{Loss}(\mathbf{f}(\mathbf{x}_j), \mathbf{y}_j)}{\partial \mathbf{W}^{(k)}}$ **?**

Backpropagation:

- Denote $L = \mathrm{Loss}(\mathbf{f}(\mathbf{x}_j),\ \mathbf{y}_j)$.
- Compute $\dfrac{\partial L}{\partial \mathbf{z}^{(3)}}$.

$\mathbf{z}^{(3)}$ is a function of $\mathbf{z}^{(2)}$ and $\mathbf{W}^{(2)}$.

Apply the chain rule.

Backpropagation

Define a function $\mathbf{f}: \mathbb{R}^{785} \mapsto \mathbb{R}^{10}$:

- Input: vector $\mathbf{x}^{(0)} \in \mathbb{R}^{785}$.
- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\,\mathbf{x}^{(0)} \in \mathbb{R}^{d_1}$.
- $\mathbf{x}^{(1)} = \max\{\mathbf{0},\ \mathbf{z}^{(1)}\} \in \mathbb{R}^{d_1}$.
- $\mathbf{z}^{(2)} = \mathbf{W}^{(1)}\,\mathbf{x}^{(1)} \in \mathbb{R}^{d_2}$.
- $\mathbf{x}^{(2)} = \max\{\mathbf{0},\ \mathbf{z}^{(2)}\} \in \mathbb{R}^{d_2}$.
- $\mathbf{z}^{(3)} = \mathbf{W}^{(2)}\,\mathbf{x}^{(2)} \in \mathbb{R}^{10}$.
- $\mathbf{x}^{(3)} = \mathrm{SoftMax}(\mathbf{z}^{(3)}) \in \mathbb{R}^{10}$.
- Output: $\mathbf{f}(\mathbf{x}^{(0)}) = \mathbf{x}^{(3)}$.

How to compute $\dfrac{\partial\, \mathrm{Loss}(\mathbf{f}(\mathbf{x}_j), \mathbf{y}_j)}{\partial \mathbf{W}^{(k)}}$ **?**

Backpropagation:

- Denote $L = \mathrm{Loss}(\mathbf{f}(\mathbf{x}_j),\ \mathbf{y}_j)$.
- Compute $\dfrac{\partial L}{\partial \mathbf{z}^{(3)}}$.
- $\dfrac{\partial L}{\partial \mathbf{z}^{(2)}} = \dfrac{\partial \mathbf{z}^{(3)}}{\partial \mathbf{z}^{(2)}} \dfrac{\partial L}{\partial \mathbf{z}^{(3)}}$, $\quad \dfrac{\partial L}{\partial \mathbf{W}^{(2)}} = \dfrac{\partial \mathbf{z}^{(3)}}{\partial \mathbf{W}^{(2)}} \dfrac{\partial L}{\partial \mathbf{z}^{(3)}}$,

$\mathbf{z}^{(3)}$ is a function of $\mathbf{z}^{(2)}$ and $\mathbf{W}^{(2)}$.

Apply the chain rule.

Backpropagation

Define a function $\mathbf{f}: \mathbb{R}^{785} \mapsto \mathbb{R}^{10}$:

- Input: vector $\mathbf{x}^{(0)} \in \mathbb{R}^{785}$.
- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\,\mathbf{x}^{(0)} \in \mathbb{R}^{d_1}$.
- $\mathbf{x}^{(1)} = \max\{\mathbf{0},\ \mathbf{z}^{(1)}\} \in \mathbb{R}^{d_1}$.
- $\mathbf{z}^{(2)} = \mathbf{W}^{(1)}\,\mathbf{x}^{(1)} \in \mathbb{R}^{d_2}$.
- $\mathbf{x}^{(2)} = \max\{\mathbf{0},\ \mathbf{z}^{(2)}\} \in \mathbb{R}^{d_2}$.
- $\mathbf{z}^{(3)} = \mathbf{W}^{(2)}\,\mathbf{x}^{(2)} \in \mathbb{R}^{10}$.
- $\mathbf{x}^{(3)} = \mathrm{SoftMax}(\mathbf{z}^{(3)}) \in \mathbb{R}^{10}$.
- Output: $\mathbf{f}(\mathbf{x}^{(0)}) = \mathbf{x}^{(3)}$.

How to compute $\dfrac{\partial\, \mathrm{Loss}(\mathbf{f}(\mathbf{x}_j), \mathbf{y}_j)}{\partial \mathbf{W}^{(k)}}$ **?**

Backpropagation:

- Denote $L = \mathrm{Loss}(\mathbf{f}(\mathbf{x}_j),\ \mathbf{y}_j)$.
- Compute $\dfrac{\partial L}{\partial \mathbf{z}^{(3)}}$.
- $\dfrac{\partial L}{\partial \mathbf{z}^{(2)}} = \dfrac{\partial \mathbf{z}^{(3)}}{\partial \mathbf{z}^{(2)}} \dfrac{\partial L}{\partial \mathbf{z}^{(3)}}$, $\quad \dfrac{\partial L}{\partial \mathbf{W}^{(2)}} = \dfrac{\partial \mathbf{z}^{(3)}}{\partial \mathbf{W}^{(2)}} \dfrac{\partial L}{\partial \mathbf{z}^{(3)}}$.

Then free $\dfrac{\partial L}{\partial \mathbf{z}^{(3)}}$ from memory.

Backpropagation

Define a function $\mathbf{f}: \mathbb{R}^{785} \mapsto \mathbb{R}^{10}$:

- Input: vector $\mathbf{x}^{(0)} \in \mathbb{R}^{785}$.
- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\mathbf{x}^{(0)} \in \mathbb{R}^{d_1}$.
- $\mathbf{x}^{(1)} = \max\{\mathbf{0}, \mathbf{z}^{(1)}\} \in \mathbb{R}^{d_1}$.
- $\mathbf{z}^{(2)} = \mathbf{W}^{(1)}\mathbf{x}^{(1)} \in \mathbb{R}^{d_2}$.
- $\mathbf{x}^{(2)} = \max\{\mathbf{0}, \mathbf{z}^{(2)}\} \in \mathbb{R}^{d_2}$.
- $\mathbf{z}^{(3)} = \mathbf{W}^{(2)}\mathbf{x}^{(2)} \in \mathbb{R}^{10}$.
- $\mathbf{x}^{(3)} = \mathrm{SoftMax}(\mathbf{z}^{(3)}) \in \mathbb{R}^{10}$.
- Output: $\mathbf{f}(\mathbf{x}^{(0)}) = \mathbf{x}^{(3)}$.

How to compute $\frac{\partial\, \mathrm{Loss}(\mathbf{f}(\mathbf{x}_j), \mathbf{y}_j)}{\partial \mathbf{W}^{(k)}}$?

Backpropagation:

- Denote $L = \mathrm{Loss}(\mathbf{f}(\mathbf{x}_j), \mathbf{y}_j)$.
- Compute $\frac{\partial L}{\partial \mathbf{z}^{(3)}}$.
- $\frac{\partial L}{\partial \mathbf{z}^{(2)}} = \frac{\partial \mathbf{z}^{(3)}}{\partial \mathbf{z}^{(2)}} \frac{\partial L}{\partial \mathbf{z}^{(3)}}$, $\boxed{\frac{\partial L}{\partial \mathbf{W}^{(2)}} = \frac{\partial \mathbf{z}^{(3)}}{\partial \mathbf{W}^{(2)}} \frac{\partial L}{\partial \mathbf{z}^{(3)}}}$.

Use it to update $\mathbf{W}^{(2)}$ (e.g., by SGD).

Backpropagation

Define a function $\mathbf{f}: \mathbb{R}^{785} \mapsto \mathbb{R}^{10}$:

- **Input:** vector $\mathbf{x}^{(0)} \in \mathbb{R}^{785}$.
- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\,\mathbf{x}^{(0)} \in \mathbb{R}^{d_1}$.
- $\mathbf{x}^{(1)} = \max\{\mathbf{0},\ \mathbf{z}^{(1)}\} \in \mathbb{R}^{d_1}$.
- $\mathbf{z}^{(2)} = \mathbf{W}^{(1)}\,\mathbf{x}^{(1)} \in \mathbb{R}^{d_2}$.
- $\mathbf{x}^{(2)} = \max\{\mathbf{0},\ \mathbf{z}^{(2)}\} \in \mathbb{R}^{d_2}$.
- $\mathbf{z}^{(3)} = \mathbf{W}^{(2)}\,\mathbf{x}^{(2)} \in \mathbb{R}^{10}$.
- $\mathbf{x}^{(3)} = \mathrm{SoftMax}(\mathbf{z}^{(3)}) \in \mathbb{R}^{10}$.
- **Output:** $\mathbf{f}(\mathbf{x}^{(0)}) = \mathbf{x}^{(3)}$.

How to compute $\dfrac{\partial\, \mathrm{Loss}(\mathbf{f}(\mathbf{x}_j), \mathbf{y}_j)}{\partial \mathbf{W}^{(k)}}$ **?**

Backpropagation:

- Denote $L = \mathrm{Loss}(\mathbf{f}(\mathbf{x}_j),\ \mathbf{y}_j)$.
- Compute $\dfrac{\partial L}{\partial \mathbf{z}^{(3)}}$.
- $\dfrac{\partial L}{\partial \mathbf{z}^{(2)}} = \dfrac{\partial \mathbf{z}^{(3)}}{\partial \mathbf{z}^{(2)}} \dfrac{\partial L}{\partial \mathbf{z}^{(3)}}$, $\dfrac{\partial L}{\partial \mathbf{W}^{(2)}} = \dfrac{\partial \mathbf{z}^{(3)}}{\partial \mathbf{W}^{(2)}} \dfrac{\partial L}{\partial \mathbf{z}^{(3)}}$.

Then free $\dfrac{\partial L}{\partial \mathbf{W}^{(2)}}$ from memory.

Backpropagation

Define a function $\mathbf{f}: \mathbb{R}^{785} \mapsto \mathbb{R}^{10}$:

- **Input:** vector $\mathbf{x}^{(0)} \in \mathbb{R}^{785}$.
- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\mathbf{x}^{(0)} \in \mathbb{R}^{d_1}$.
- $\mathbf{x}^{(1)} = \max\{\mathbf{0}, \mathbf{z}^{(1)}\} \in \mathbb{R}^{d_1}$.
- $\mathbf{z}^{(2)} = \mathbf{W}^{(1)}\mathbf{x}^{(1)} \in \mathbb{R}^{d_2}$.
- $\mathbf{x}^{(2)} = \max\{\mathbf{0}, \mathbf{z}^{(2)}\} \in \mathbb{R}^{d_2}$.
- $\mathbf{z}^{(3)} = \mathbf{W}^{(2)}\mathbf{x}^{(2)} \in \mathbb{R}^{10}$.
- $\mathbf{x}^{(3)} = \mathrm{SoftMax}(\mathbf{z}^{(3)}) \in \mathbb{R}^{10}$.
- **Output:** $\mathbf{f}(\mathbf{x}^{(0)}) = \mathbf{x}^{(3)}$.

How to compute $\dfrac{\partial\, \mathrm{Loss}(\mathbf{f}(\mathbf{x}_j), \mathbf{y}_j)}{\partial \mathbf{W}^{(k)}}$ **?**

Backpropagation:

- Denote $L = \mathrm{Loss}(\mathbf{f}(\mathbf{x}_j), \mathbf{y}_j)$.
- Compute $\dfrac{\partial L}{\partial \mathbf{z}^{(3)}}$.
- $\dfrac{\partial L}{\partial \mathbf{z}^{(2)}} = \dfrac{\partial \mathbf{z}^{(3)}}{\partial \mathbf{z}^{(2)}} \dfrac{\partial L}{\partial \mathbf{z}^{(3)}}$, $\dfrac{\partial L}{\partial \mathbf{W}^{(2)}} = \dfrac{\partial \mathbf{z}^{(3)}}{\partial \mathbf{W}^{(2)}} \dfrac{\partial L}{\partial \mathbf{z}^{(3)}}$.
- $\dfrac{\partial L}{\partial \mathbf{z}^{(1)}} = \dfrac{\partial \mathbf{z}^{(2)}}{\partial \mathbf{z}^{(1)}} \dfrac{\partial L}{\partial \mathbf{z}^{(2)}}$, $\dfrac{\partial L}{\partial \mathbf{W}^{(1)}} = \dfrac{\partial \mathbf{z}^{(2)}}{\partial \mathbf{W}^{(1)}} \dfrac{\partial L}{\partial \mathbf{z}^{(2)}}$,

Apply the chain rule again.

Backpropagation

Define a function $\mathbf{f}: \mathbb{R}^{785} \mapsto \mathbb{R}^{10}$:

- Input: vector $\mathbf{x}^{(0)} \in \mathbb{R}^{785}$.
- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\,\mathbf{x}^{(0)} \in \mathbb{R}^{d_1}$.
- $\mathbf{x}^{(1)} = \max\{\mathbf{0},\ \mathbf{z}^{(1)}\} \in \mathbb{R}^{d_1}$.
- $\mathbf{z}^{(2)} = \mathbf{W}^{(1)}\,\mathbf{x}^{(1)} \in \mathbb{R}^{d_2}$.
- $\mathbf{x}^{(2)} = \max\{\mathbf{0},\ \mathbf{z}^{(2)}\} \in \mathbb{R}^{d_2}$.
- $\mathbf{z}^{(3)} = \mathbf{W}^{(2)}\,\mathbf{x}^{(2)} \in \mathbb{R}^{10}$.
- $\mathbf{x}^{(3)} = \mathrm{SoftMax}(\mathbf{z}^{(3)}) \in \mathbb{R}^{10}$.
- Output: $\mathbf{f}(\mathbf{x}^{(0)}) = \mathbf{x}^{(3)}$.

How to compute $\dfrac{\partial\, \mathrm{Loss}(\mathbf{f}(\mathbf{x}_j), \mathbf{y}_j)}{\partial \mathbf{W}^{(k)}}$?

Backpropagation:

- Denote $L = \mathrm{Loss}(\mathbf{f}(\mathbf{x}_j),\ \mathbf{y}_j)$.
- Compute $\dfrac{\partial L}{\partial \mathbf{z}^{(3)}}$.
- $\dfrac{\partial L}{\partial \mathbf{z}^{(2)}} = \dfrac{\partial \mathbf{z}^{(3)}}{\partial \mathbf{z}^{(2)}} \dfrac{\partial L}{\partial \mathbf{z}^{(3)}}$, $\quad \dfrac{\partial L}{\partial \mathbf{W}^{(2)}} = \dfrac{\partial \mathbf{z}^{(3)}}{\partial \mathbf{W}^{(2)}} \dfrac{\partial L}{\partial \mathbf{z}^{(3)}}$.
- $\dfrac{\partial L}{\partial \mathbf{z}^{(1)}} = \dfrac{\partial \mathbf{z}^{(2)}}{\partial \mathbf{z}^{(1)}} \dfrac{\partial L}{\partial \mathbf{z}^{(2)}}$, $\quad \dfrac{\partial L}{\partial \mathbf{W}^{(1)}} = \dfrac{\partial \mathbf{z}^{(2)}}{\partial \mathbf{W}^{(1)}} \dfrac{\partial L}{\partial \mathbf{z}^{(2)}}$.

Free $\dfrac{\partial L}{\partial \mathbf{z}^{(2)}}$ from memory.

Backpropagation

Define a function $\mathbf{f}: \mathbb{R}^{785} \mapsto \mathbb{R}^{10}$:

- **Input:** vector $\mathbf{x}^{(0)} \in \mathbb{R}^{785}$.
- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\mathbf{x}^{(0)} \in \mathbb{R}^{d_1}$.
- $\mathbf{x}^{(1)} = \max\{\mathbf{0}, \mathbf{z}^{(1)}\} \in \mathbb{R}^{d_1}$.
- $\mathbf{z}^{(2)} = \mathbf{W}^{(1)}\mathbf{x}^{(1)} \in \mathbb{R}^{d_2}$.
- $\mathbf{x}^{(2)} = \max\{\mathbf{0}, \mathbf{z}^{(2)}\} \in \mathbb{R}^{d_2}$.
- $\mathbf{z}^{(3)} = \mathbf{W}^{(2)}\mathbf{x}^{(2)} \in \mathbb{R}^{10}$.
- $\mathbf{x}^{(3)} = \mathrm{SoftMax}(\mathbf{z}^{(3)}) \in \mathbb{R}^{10}$.
- **Output:** $\mathbf{f}(\mathbf{x}^{(0)}) = \mathbf{x}^{(3)}$.

How to compute $\frac{\partial\, \mathrm{Loss}(\mathbf{f}(\mathbf{x}_j), \mathbf{y}_j)}{\partial \mathbf{W}^{(k)}}$ **?**

Backpropagation:

- Denote $L = \mathrm{Loss}(\mathbf{f}(\mathbf{x}_j), \mathbf{y}_j)$.
- Compute $\frac{\partial L}{\partial \mathbf{z}^{(3)}}$.
- $\frac{\partial L}{\partial \mathbf{z}^{(2)}} = \frac{\partial \mathbf{z}^{(3)}}{\partial \mathbf{z}^{(2)}}\frac{\partial L}{\partial \mathbf{z}^{(3)}}$, $\frac{\partial L}{\partial \mathbf{W}^{(2)}} = \frac{\partial \mathbf{z}^{(3)}}{\partial \mathbf{W}^{(2)}}\frac{\partial L}{\partial \mathbf{z}^{(3)}}$.
- $\frac{\partial L}{\partial \mathbf{z}^{(1)}} = \frac{\partial \mathbf{z}^{(2)}}{\partial \mathbf{z}^{(1)}}\frac{\partial L}{\partial \mathbf{z}^{(2)}}$, $\boxed{\frac{\partial L}{\partial \mathbf{W}^{(1)}} = \frac{\partial \mathbf{z}^{(2)}}{\partial \mathbf{W}^{(1)}}\frac{\partial L}{\partial \mathbf{z}^{(2)}}}$.

Use it to update $\mathbf{W}^{(1)}$ (e.g., by SGD).

Backpropagation

Define a function $\mathbf{f}: \mathbb{R}^{785} \mapsto \mathbb{R}^{10}$:

- Input: vector $\mathbf{x}^{(0)} \in \mathbb{R}^{785}$.
- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\,\mathbf{x}^{(0)} \in \mathbb{R}^{d_1}$.
- $\mathbf{x}^{(1)} = \max\{\mathbf{0},\ \mathbf{z}^{(1)}\} \in \mathbb{R}^{d_1}$.
- $\mathbf{z}^{(2)} = \mathbf{W}^{(1)}\,\mathbf{x}^{(1)} \in \mathbb{R}^{d_2}$.
- $\mathbf{x}^{(2)} = \max\{\mathbf{0},\ \mathbf{z}^{(2)}\} \in \mathbb{R}^{d_2}$.
- $\mathbf{z}^{(3)} = \mathbf{W}^{(2)}\,\mathbf{x}^{(2)} \in \mathbb{R}^{10}$.
- $\mathbf{x}^{(3)} = \mathrm{SoftMax}(\mathbf{z}^{(3)}) \in \mathbb{R}^{10}$.
- Output: $\mathbf{f}(\mathbf{x}^{(0)}) = \mathbf{x}^{(3)}$.

How to compute $\frac{\partial\, \mathrm{Loss}(\mathbf{f}(\mathbf{x}_j), \mathbf{y}_j)}{\partial \mathbf{W}^{(k)}}$?

Backpropagation:

- Denote $L = \mathrm{Loss}(\mathbf{f}(\mathbf{x}_j),\ \mathbf{y}_j)$.
- Compute $\frac{\partial L}{\partial \mathbf{z}^{(3)}}$.
- $\frac{\partial L}{\partial \mathbf{z}^{(2)}} = \frac{\partial \mathbf{z}^{(3)}}{\partial \mathbf{z}^{(2)}} \frac{\partial L}{\partial \mathbf{z}^{(3)}}$, $\quad \frac{\partial L}{\partial \mathbf{W}^{(2)}} = \frac{\partial \mathbf{z}^{(3)}}{\partial \mathbf{W}^{(2)}} \frac{\partial L}{\partial \mathbf{z}^{(3)}}$.
- $\frac{\partial L}{\partial \mathbf{z}^{(1)}} = \frac{\partial \mathbf{z}^{(2)}}{\partial \mathbf{z}^{(1)}} \frac{\partial L}{\partial \mathbf{z}^{(2)}}$, $\quad \frac{\partial L}{\partial \mathbf{W}^{(1)}} = \frac{\partial \mathbf{z}^{(2)}}{\partial \mathbf{W}^{(1)}} \frac{\partial L}{\partial \mathbf{z}^{(2)}}$.

Free $\frac{\partial L}{\partial \mathbf{W}^{(1)}}$ from memory.

Backpropagation

Define a function $\mathbf{f}: \mathbb{R}^{785} \mapsto \mathbb{R}^{10}$:

- **Input:** vector $\mathbf{x}^{(0)} \in \mathbb{R}^{785}$.
- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\mathbf{x}^{(0)} \in \mathbb{R}^{d_1}$.
- $\mathbf{x}^{(1)} = \max\{\mathbf{0}, \mathbf{z}^{(1)}\} \in \mathbb{R}^{d_1}$.
- $\mathbf{z}^{(2)} = \mathbf{W}^{(1)}\mathbf{x}^{(1)} \in \mathbb{R}^{d_2}$.
- $\mathbf{x}^{(2)} = \max\{\mathbf{0}, \mathbf{z}^{(2)}\} \in \mathbb{R}^{d_2}$.
- $\mathbf{z}^{(3)} = \mathbf{W}^{(2)}\mathbf{x}^{(2)} \in \mathbb{R}^{10}$.
- $\mathbf{x}^{(3)} = \mathrm{SoftMax}(\mathbf{z}^{(3)}) \in \mathbb{R}^{10}$.
- **Output:** $\mathbf{f}(\mathbf{x}^{(0)}) = \mathbf{x}^{(3)}$.

How to compute $\frac{\partial\, \mathrm{Loss}(\mathbf{f}(\mathbf{x}_j), \mathbf{y}_j)}{\partial \mathbf{W}^{(k)}}$ **?**

Backpropagation:

- Denote $L = \mathrm{Loss}(\mathbf{f}(\mathbf{x}_j), \mathbf{y}_j)$.
- Compute $\frac{\partial L}{\partial \mathbf{z}^{(3)}}$.
- $\frac{\partial L}{\partial \mathbf{z}^{(2)}} = \frac{\partial \mathbf{z}^{(3)}}{\partial \mathbf{z}^{(2)}} \frac{\partial L}{\partial \mathbf{z}^{(3)}}$, $\quad \frac{\partial L}{\partial \mathbf{W}^{(2)}} = \frac{\partial \mathbf{z}^{(3)}}{\partial \mathbf{W}^{(2)}} \frac{\partial L}{\partial \mathbf{z}^{(3)}}$.
- $\frac{\partial L}{\partial \mathbf{z}^{(1)}} = \frac{\partial \mathbf{z}^{(2)}}{\partial \mathbf{z}^{(1)}} \frac{\partial L}{\partial \mathbf{z}^{(2)}}$, $\quad \frac{\partial L}{\partial \mathbf{W}^{(1)}} = \frac{\partial \mathbf{z}^{(2)}}{\partial \mathbf{W}^{(1)}} \frac{\partial L}{\partial \mathbf{z}^{(2)}}$.
- $\frac{\partial L}{\partial \mathbf{W}^{(0)}} = \frac{\partial \mathbf{z}^{(1)}}{\partial \mathbf{W}^{(0)}} \frac{\partial L}{\partial \mathbf{z}^{(1)}}$.

Apply the chain rule again.

Backpropagation

Define a function $\mathbf{f}: \mathbb{R}^{785} \mapsto \mathbb{R}^{10}$:

- Input: vector $\mathbf{x}^{(0)} \in \mathbb{R}^{785}$.
- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\,\mathbf{x}^{(0)} \in \mathbb{R}^{d_1}$.
- $\mathbf{x}^{(1)} = \max\{\mathbf{0},\ \mathbf{z}^{(1)}\} \in \mathbb{R}^{d_1}$.
- $\mathbf{z}^{(2)} = \mathbf{W}^{(1)}\,\mathbf{x}^{(1)} \in \mathbb{R}^{d_2}$.
- $\mathbf{x}^{(2)} = \max\{\mathbf{0},\ \mathbf{z}^{(2)}\} \in \mathbb{R}^{d_2}$.
- $\mathbf{z}^{(3)} = \mathbf{W}^{(2)}\,\mathbf{x}^{(2)} \in \mathbb{R}^{10}$.
- $\mathbf{x}^{(3)} = \mathrm{SoftMax}\left(\mathbf{z}^{(3)}\right) \in \mathbb{R}^{10}$.
- Output: $\mathbf{f}\left(\mathbf{x}^{(0)}\right) = \mathbf{x}^{(3)}$.

How to compute $\dfrac{\partial\, \mathrm{Loss}\left(\mathbf{f}(\mathbf{x}_j), \mathbf{y}_j\right)}{\partial \mathbf{W}^{(k)}}$ **?**

Backpropagation:

- Denote $L = \mathrm{Loss}\left(\mathbf{f}(\mathbf{x}_j),\ \mathbf{y}_j\right)$.
- Compute $\dfrac{\partial L}{\partial \mathbf{z}^{(3)}}$.
- $\dfrac{\partial L}{\partial \mathbf{z}^{(2)}} = \dfrac{\partial \mathbf{z}^{(3)}}{\partial \mathbf{z}^{(2)}} \dfrac{\partial L}{\partial \mathbf{z}^{(3)}}$, $\quad \dfrac{\partial L}{\partial \mathbf{W}^{(2)}} = \dfrac{\partial \mathbf{z}^{(3)}}{\partial \mathbf{W}^{(2)}} \dfrac{\partial L}{\partial \mathbf{z}^{(3)}}$.
- $\dfrac{\partial L}{\partial \mathbf{z}^{(1)}} = \dfrac{\partial \mathbf{z}^{(2)}}{\partial \mathbf{z}^{(1)}} \dfrac{\partial L}{\partial \mathbf{z}^{(2)}}$,

Free $\dfrac{\partial L}{\partial \mathbf{z}^{(1)}}$ from memory.

- $\dfrac{\partial L}{\partial \mathbf{W}^{(0)}} = \dfrac{\partial \mathbf{z}^{(1)}}{\partial \mathbf{W}^{(0)}} \dfrac{\partial L}{\partial \mathbf{z}^{(1)}}$.

Backpropagation

Define a function $\mathbf{f}: \mathbb{R}^{785} \mapsto \mathbb{R}^{10}$:

- Input: vector $\mathbf{x}^{(0)} \in \mathbb{R}^{785}$.
- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\mathbf{x}^{(0)} \in \mathbb{R}^{d_1}$.
- $\mathbf{x}^{(1)} = \max\{\mathbf{0}, \mathbf{z}^{(1)}\} \in \mathbb{R}^{d_1}$.
- $\mathbf{z}^{(2)} = \mathbf{W}^{(1)}\mathbf{x}^{(1)} \in \mathbb{R}^{d_2}$.
- $\mathbf{x}^{(2)} = \max\{\mathbf{0}, \mathbf{z}^{(2)}\} \in \mathbb{R}^{d_2}$.
- $\mathbf{z}^{(3)} = \mathbf{W}^{(2)}\mathbf{x}^{(2)} \in \mathbb{R}^{10}$.
- $\mathbf{x}^{(3)} = \mathrm{SoftMax}(\mathbf{z}^{(3)}) \in \mathbb{R}^{10}$.
- Output: $\mathbf{f}(\mathbf{x}^{(0)}) = \mathbf{x}^{(3)}$.

How to compute $\frac{\partial\, \mathrm{Loss}(\mathbf{f}(\mathbf{x}_j), \mathbf{y}_j)}{\partial \mathbf{W}^{(k)}}$?

Backpropagation:

- Denote $L = \mathrm{Loss}(\mathbf{f}(\mathbf{x}_j), \mathbf{y}_j)$.
- Compute $\frac{\partial L}{\partial \mathbf{z}^{(3)}}$.
- $\frac{\partial L}{\partial \mathbf{z}^{(2)}} = \frac{\partial \mathbf{z}^{(3)}}{\partial \mathbf{z}^{(2)}}\frac{\partial L}{\partial \mathbf{z}^{(3)}}$, $\frac{\partial L}{\partial \mathbf{W}^{(2)}} = \frac{\partial \mathbf{z}^{(3)}}{\partial \mathbf{W}^{(2)}}\frac{\partial L}{\partial \mathbf{z}^{(3)}}$.
- $\frac{\partial L}{\partial \mathbf{z}^{(1)}} = \frac{\partial \mathbf{z}^{(2)}}{\partial \mathbf{z}^{(1)}}\frac{\partial L}{\partial \mathbf{z}^{(2)}}$, $\frac{\partial L}{\partial \mathbf{W}^{(1)}} = \frac{\partial \mathbf{z}^{(2)}}{\partial \mathbf{W}^{(1)}}\frac{\partial L}{\partial \mathbf{z}^{(2)}}$.
- $\frac{\partial L}{\partial \mathbf{W}^{(0)}} = \frac{\partial \mathbf{z}^{(1)}}{\partial \mathbf{W}^{(0)}}\frac{\partial L}{\partial \mathbf{z}^{(1)}}$. Use it to update $\mathbf{W}^{(0)}$.

Backpropagation: Example

1D Example

Define a function $f: \mathbb{R} \mapsto \mathbb{R}$

- **Input**: scalar $x^{(0)}$.
- $z^{(1)} = w^{(0)} x^{(0)}$.
- $x^{(1)} = \max\{0, z^{(1)}\}$.
- $z^{(2)} = w^{(1)} x^{(1)}$.
- $x^{(2)} = \max\{0, z^{(2)}\}$.
- $z^{(3)} = w^{(2)} x^{(2)}$.
- **Output**: $f(x^{(0)}) = z^{(3)}$.

1D Example

Define a function $f: \mathbb{R} \mapsto \mathbb{R}$

- **Input**: scalar $x^{(0)}$.
- $z^{(1)} = w^{(0)} x^{(0)}$.
- $x^{(1)} = \max\{0, z^{(1)}\}$.
- $z^{(2)} = w^{(1)} x^{(1)}$.
- $x^{(2)} = \max\{0, z^{(2)}\}$.
- $z^{(3)} = w^{(2)} x^{(2)}$.
- **Output**: $f(x^{(0)}) = z^{(3)}$.

Random sampling:

- Randomly sample j from $\{1, 2, \cdots, n\}$.

Forward pass (input ➔ output):

- Take x_j as input ($x^{(0)} = x_j$).
- Compute each layer $z^{(1)}, x^{(1)}, z^{(2)}, x^{(2)}, z^{(3)}$.

1D Example

Define a function $f: \mathbb{R} \mapsto \mathbb{R}$

- **Input:** scalar $x^{(0)}$.
- $z^{(1)} = w^{(0)} x^{(0)}$.
- $x^{(1)} = \max\{0, z^{(1)}\}$.
- $z^{(2)} = w^{(1)} x^{(1)}$.
- $x^{(2)} = \max\{0, z^{(2)}\}$.
- $z^{(3)} = w^{(2)} x^{(2)}$.
- **Output:** $f(x^{(0)}) = z^{(3)}$.

Backpropagation:

- Loss: $L = \frac{1}{2}(f(x_j) - y_j)^2$.

1D Example

Define a function $f: \mathbb{R} \mapsto \mathbb{R}$

- **Input:** scalar $x^{(0)}$.
- $z^{(1)} = w^{(0)} x^{(0)}$.
- $x^{(1)} = \max\{0, z^{(1)}\}$.
- $z^{(2)} = w^{(1)} x^{(1)}$.
- $x^{(2)} = \max\{0, z^{(2)}\}$.
- $z^{(3)} = w^{(2)} x^{(2)}$.
- **Output:** $f(x^{(0)}) = z^{(3)}$.

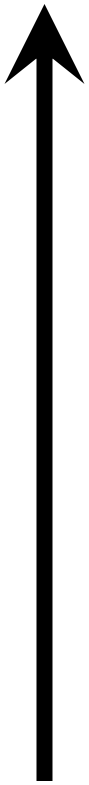
Backpropagation:

- Loss: $L = \frac{1}{2}(f(x_j) - y_j)^2 = \frac{1}{2}(z^{(3)} - y_j)^2$.
- $\frac{\partial L}{\partial z^{(3)}} = z^{(3)} - y_j$.

1D Example

Define a function $f: \mathbb{R} \mapsto \mathbb{R}$

- **Input:** scalar $x^{(0)}$.
- $z^{(1)} = w^{(0)} x^{(0)}$.
- $x^{(1)} = \max\{0, z^{(1)}\}$.
- $z^{(2)} = w^{(1)} x^{(1)}$.
- $x^{(2)} = \max\{0, z^{(2)}\}$.
- $z^{(3)} = w^{(2)} x^{(2)}$.
- **Output:** $f(x^{(0)}) = z^{(3)}$.



Backpropagation:

- Loss: $L = \frac{1}{2}(f(x_j) - y_j)^2$.
- $\frac{\partial L}{\partial z^{(3)}} = z^{(3)} - y_j$.

The value of $z^{(3)}$ is known (after the forward pass).

1D Example

Define a function $f: \mathbb{R} \mapsto \mathbb{R}$

- Input: scalar $x^{(0)}$.
- $z^{(1)} = w^{(0)} x^{(0)}$.
- $x^{(1)} = \max\{0, z^{(1)}\}$.
- $z^{(2)} = w^{(1)} x^{(1)}$.
- $x^{(2)} = \max\{0, z^{(2)}\}$.
- $z^{(3)} = w^{(2)} x^{(2)}$.
- **Output**: $f(x^{(0)}) = z^{(3)}$.

Backpropagation:

- Loss: $L = \frac{1}{2}(f(x_j) - y_j)^2$.
- $\frac{\partial L}{\partial z^{(3)}} = z^{(3)} - y_j$.

The value of $z^{(3)}$ is known (after the forward pass).

Thus the value of $\frac{\partial L}{\partial z^{(3)}}$ is known.

1D Example

Define a function $f: \mathbb{R} \mapsto \mathbb{R}$

- **Input:** scalar $x^{(0)}$.
- $z^{(1)} = w^{(0)} x^{(0)}$.
- $x^{(1)} = \max\{0, z^{(1)}\}$.
- $z^{(2)} = w^{(1)} x^{(1)}$.
- $x^{(2)} = \max\{0, z^{(2)}\}$.
- $z^{(3)} = w^{(2)} x^{(2)}$.
- **Output:** $f(x^{(0)}) = z^{(3)}$.

Backpropagation:

- Loss: $L = \frac{1}{2}(f(x_j) - y_j)^2$.
- $\frac{\partial L}{\partial z^{(3)}} = z^{(3)} - y_j$.
- $\frac{\partial L}{\partial z^{(2)}} = \boxed{\frac{\partial z^{(3)}}{\partial z^{(2)}}} \frac{\partial L}{\partial z^{(3)}}$

$$\frac{\partial z^{(3)}}{\partial x^{(2)}} = w^{(2)}, \qquad \frac{\partial x^{(2)}}{\partial z^{(2)}} = \begin{cases} 1, & \text{if } z^{(2)} > 0; \\ 0, & \text{else.} \end{cases}$$

1D Example

Define a function $f: \mathbb{R} \mapsto \mathbb{R}$

- **Input:** scalar $x^{(0)}$.
- $z^{(1)} = w^{(0)} x^{(0)}$.
- $x^{(1)} = \max\{0, z^{(1)}\}$.
- $z^{(2)} = w^{(1)} x^{(1)}$.
- $x^{(2)} = \max\{0, z^{(2)}\}$.
- $z^{(3)} = w^{(2)} x^{(2)}$.
- **Output:** $f(x^{(0)}) = z^{(3)}$.

Backpropagation:

- Loss: $L = \frac{1}{2}(f(x_j) - y_j)^2$.
- $\frac{\partial L}{\partial z^{(3)}} = z^{(3)} - y_j$.
- $\frac{\partial L}{\partial z^{(2)}} = \frac{\partial z^{(3)}}{\partial z^{(2)}} \frac{\partial L}{\partial z^{(3)}}$

$$\frac{\partial z^{(3)}}{\partial x^{(2)}} = w^{(2)}, \qquad \frac{\partial x^{(2)}}{\partial z^{(2)}} = \begin{cases} 1, & \text{if } z^{(2)} > 0; \\ 0, & \text{else.} \end{cases}$$

$$\frac{\partial z^{(3)}}{\partial z^{(2)}} = \frac{\partial z^{(3)}}{\partial x^{(2)}} \frac{\partial x^{(2)}}{\partial z^{(2)}} = \begin{cases} w^{(2)}, & \text{if } z^{(2)} > 0; \\ 0, & \text{else.} \end{cases}$$

1D Example

Define a function $f: \mathbb{R} \mapsto \mathbb{R}$

- **Input:** scalar $x^{(0)}$.
- $z^{(1)} = w^{(0)} x^{(0)}$.
- $x^{(1)} = \max\{0, z^{(1)}\}$.
- $z^{(2)} = w^{(1)} x^{(1)}$.
- $x^{(2)} = \max\{0, z^{(2)}\}$.
- $z^{(3)} = w^{(2)} x^{(2)}$.
- **Output:** $f(x^{(0)}) = z^{(3)}$.

Backpropagation:

- Loss: $L = \frac{1}{2}(f(x_j) - y_j)^2$.
- $\frac{\partial L}{\partial z^{(3)}} = z^{(3)} - y_j$.
- $\frac{\partial L}{\partial z^{(2)}} = \frac{\partial z^{(3)}}{\partial z^{(2)}} \frac{\partial L}{\partial z^{(3)}}$, $\frac{\partial L}{\partial w^{(2)}} = \boxed{\frac{\partial z^{(3)}}{\partial w^{(2)}}} \frac{\partial L}{\partial z^{(3)}}$.

$\frac{\partial z^{(3)}}{\partial w^{(2)}} = x^{(2)}$.

1D Example

Define a function $f: \mathbb{R} \mapsto \mathbb{R}$

- **Input:** scalar $x^{(0)}$.
- $z^{(1)} = w^{(0)} x^{(0)}$.
- $x^{(1)} = \max\{0, z^{(1)}\}$.
- $z^{(2)} = w^{(1)} x^{(1)}$.
- $x^{(2)} = \max\{0, z^{(2)}\}$.
- $z^{(3)} = w^{(2)} x^{(2)}$.
- **Output:** $f(x^{(0)}) = z^{(3)}$.

Backpropagation:

- Loss: $L = \frac{1}{2}(f(x_j) - y_j)^2$.
- $\frac{\partial L}{\partial z^{(3)}} = z^{(3)} - y_j$.
- $\frac{\partial L}{\partial z^{(2)}} = \frac{\partial z^{(3)}}{\partial z^{(2)}} \frac{\partial L}{\partial z^{(3)}}$, $\frac{\partial L}{\partial w^{(2)}} = \frac{\partial z^{(3)}}{\partial w^{(2)}} \frac{\partial L}{\partial z^{(3)}}$.

Free $\frac{\partial L}{\partial z^{(3)}}$ from memory.

1D Example

Define a function $f: \mathbb{R} \mapsto \mathbb{R}$

- **Input:** scalar $x^{(0)}$.
- $z^{(1)} = w^{(0)} x^{(0)}$.
- $x^{(1)} = \max\{0, z^{(1)}\}$.
- $z^{(2)} = w^{(1)} x^{(1)}$.
- $x^{(2)} = \max\{0, z^{(2)}\}$.
- $z^{(3)} = w^{(2)} x^{(2)}$.
- **Output:** $f(x^{(0)}) = z^{(3)}$.

Backpropagation:

- Loss: $L = \frac{1}{2}(f(x_j) - y_j)^2$.
- $\frac{\partial L}{\partial z^{(3)}} = z^{(3)} - y_j$.
- $\frac{\partial L}{\partial z^{(2)}} = \frac{\partial z^{(3)}}{\partial z^{(2)}} \frac{\partial L}{\partial z^{(3)}}$, $\boxed{\frac{\partial L}{\partial w^{(2)}}} = \frac{\partial z^{(3)}}{\partial w^{(2)}} \frac{\partial L}{\partial z^{(3)}}$.

Update $w^{(2)}$: $w^{(2)} \leftarrow w^{(2)} - \alpha \frac{\partial L}{\partial w^{(2)}}$.

1D Example

Define a function $f: \mathbb{R} \mapsto \mathbb{R}$

- **Input:** scalar $x^{(0)}$.
- $z^{(1)} = w^{(0)} x^{(0)}$.
- $x^{(1)} = \max\{0, z^{(1)}\}$.
- $z^{(2)} = w^{(1)} x^{(1)}$.
- $x^{(2)} = \max\{0, z^{(2)}\}$.
- $z^{(3)} = w^{(2)} x^{(2)}$.
- **Output:** $f(x^{(0)}) = z^{(3)}$.

Backpropagation:

- Loss: $L = \frac{1}{2}(f(x_j) - y_j)^2$.
- $\frac{\partial L}{\partial z^{(3)}} = z^{(3)} - y_j$.
- $\frac{\partial L}{\partial z^{(2)}} = \frac{\partial z^{(3)}}{\partial z^{(2)}} \frac{\partial L}{\partial z^{(3)}}$, $\frac{\partial L}{\partial w^{(2)}} = \frac{\partial z^{(3)}}{\partial w^{(2)}} \frac{\partial L}{\partial z^{(3)}}$.

Free $\frac{\partial z^{(3)}}{\partial w^{(2)}}$ from memory.

1D Example

Define a function $f: \mathbb{R} \mapsto \mathbb{R}$

- **Input:** scalar $x^{(0)}$.
- $z^{(1)} = w^{(0)} x^{(0)}$.
- $x^{(1)} = \max\{0,\ z^{(1)}\}$.
- $z^{(2)} = w^{(1)} x^{(1)}$.
- $x^{(2)} = \max\{0,\ z^{(2)}\}$.
- $z^{(3)} = w^{(2)} x^{(2)}$.
- **Output:** $f(x^{(0)}) = z^{(3)}$.

Backpropagation:

- Loss: $L = \frac{1}{2}(f(x_j) - y_j)^2$.
- $\frac{\partial L}{\partial z^{(3)}} = z^{(3)} - y_j$.
- $\frac{\partial L}{\partial z^{(2)}} = \frac{\partial z^{(3)}}{\partial z^{(2)}} \frac{\partial L}{\partial z^{(3)}}$, $\frac{\partial L}{\partial w^{(2)}} = \frac{\partial z^{(3)}}{\partial w^{(2)}} \frac{\partial L}{\partial z^{(3)}}$.
- $\frac{\partial L}{\partial z^{(1)}} = \boxed{\frac{\partial z^{(2)}}{\partial z^{(1)}}} \frac{\partial L}{\partial z^{(2)}}$

$$\boxed{\frac{\partial z^{(2)}}{\partial z^{(1)}}} = \frac{\partial z^{(2)}}{\partial x^{(1)}} \frac{\partial x^{(1)}}{\partial z^{(1)}} = \begin{cases} w^{(1)}, & \text{if } z^{(1)} > 0; \\ 0, & \text{else.} \end{cases}$$

1D Example

Define a function $f: \mathbb{R} \mapsto \mathbb{R}$

- **Input:** scalar $x^{(0)}$.
- $z^{(1)} = w^{(0)} x^{(0)}$.
- $x^{(1)} = \max\{0,\ z^{(1)}\}$.
- $z^{(2)} = w^{(1)} x^{(1)}$.
- $x^{(2)} = \max\{0,\ z^{(2)}\}$.
- $z^{(3)} = w^{(2)} x^{(2)}$.
- **Output:** $f(x^{(0)}) = z^{(3)}$.

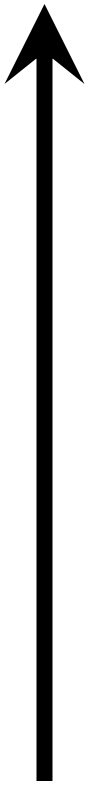
Backpropagation:

- Loss: $L = \frac{1}{2}(f(x_j) - y_j)^2$.
- $\frac{\partial L}{\partial z^{(3)}} = z^{(3)} - y_j$.
- $\frac{\partial L}{\partial z^{(2)}} = \frac{\partial z^{(3)}}{\partial z^{(2)}} \frac{\partial L}{\partial z^{(3)}}$, $\frac{\partial L}{\partial w^{(2)}} = \frac{\partial z^{(3)}}{\partial w^{(2)}} \frac{\partial L}{\partial z^{(3)}}$.
- $\frac{\partial L}{\partial z^{(1)}} = \frac{\partial z^{(2)}}{\partial z^{(1)}} \frac{\partial L}{\partial z^{(2)}}$, $\frac{\partial L}{\partial w^{(1)}} = \boxed{\frac{\partial z^{(2)}}{\partial w^{(1)}}} \frac{\partial L}{\partial z^{(2)}}$.

$$\frac{\partial z^{(2)}}{\partial w^{(1)}} = x^{(1)}.$$

1D Example

Define a function $f: \mathbb{R} \mapsto \mathbb{R}$



- **Input:** scalar $x^{(0)}$.
- $z^{(1)} = w^{(0)} x^{(0)}$.
- $x^{(1)} = \max\{0,\ z^{(1)}\}$.
- $z^{(2)} = w^{(1)} x^{(1)}$.
- $x^{(2)} = \max\{0,\ z^{(2)}\}$.
- $z^{(3)} = w^{(2)} x^{(2)}$.
- **Output:** $f(x^{(0)}) = z^{(3)}$.

Backpropagation:

- Loss: $L = \frac{1}{2}(f(x_j) - y_j)^2$.
- $\frac{\partial L}{\partial z^{(3)}} = z^{(3)} - y_j$.
- $\frac{\partial L}{\partial z^{(2)}} = \frac{\partial z^{(3)}}{\partial z^{(2)}} \frac{\partial L}{\partial z^{(3)}}$, $\frac{\partial L}{\partial w^{(2)}} = \frac{\partial z^{(3)}}{\partial w^{(2)}} \frac{\partial L}{\partial z^{(3)}}$.
- $\frac{\partial L}{\partial z^{(1)}} = \frac{\partial z^{(2)}}{\partial z^{(1)}} \frac{\partial L}{\partial z^{(2)}}$, $\frac{\partial L}{\partial w^{(1)}} = \frac{\partial z^{(2)}}{\partial w^{(1)}} \frac{\partial L}{\partial z^{(2)}}$.

Free $\frac{\partial L}{\partial z^{(2)}}$ from memory.

1D Example

Define a function $f: \mathbb{R} \mapsto \mathbb{R}$

- **Input:** scalar $x^{(0)}$.
- $z^{(1)} = w^{(0)} x^{(0)}$.
- $x^{(1)} = \max\{0, z^{(1)}\}$.
- $z^{(2)} = w^{(1)} x^{(1)}$.
- $x^{(2)} = \max\{0, z^{(2)}\}$.
- $z^{(3)} = w^{(2)} x^{(2)}$.
- **Output:** $f(x^{(0)}) = z^{(3)}$.

Backpropagation:

- Loss: $L = \frac{1}{2}(f(x_j) - y_j)^2$.
- $\frac{\partial L}{\partial z^{(3)}} = z^{(3)} - y_j$.
- $\frac{\partial L}{\partial z^{(2)}} = \frac{\partial z^{(3)}}{\partial z^{(2)}} \frac{\partial L}{\partial z^{(3)}}$, $\frac{\partial L}{\partial w^{(2)}} = \frac{\partial z^{(3)}}{\partial w^{(2)}} \frac{\partial L}{\partial z^{(3)}}$.
- $\frac{\partial L}{\partial z^{(1)}} = \frac{\partial z^{(2)}}{\partial z^{(1)}} \frac{\partial L}{\partial z^{(2)}}$, $\boxed{\frac{\partial L}{\partial w^{(1)}}} = \frac{\partial z^{(2)}}{\partial w^{(1)}} \frac{\partial L}{\partial z^{(2)}}$.

Update $w^{(1)}$: $w^{(1)} \leftarrow w^{(1)} - \alpha \frac{\partial L}{\partial w^{(1)}}$.

1D Example

Define a function $f: \mathbb{R} \mapsto \mathbb{R}$

- **Input:** scalar $x^{(0)}$.
- $z^{(1)} = w^{(0)} x^{(0)}$.
- $x^{(1)} = \max\{0, z^{(1)}\}$.
- $z^{(2)} = w^{(1)} x^{(1)}$.
- $x^{(2)} = \max\{0, z^{(2)}\}$.
- $z^{(3)} = w^{(2)} x^{(2)}$.
- **Output:** $f(x^{(0)}) = z^{(3)}$.

Backpropagation:

- Loss: $L = \frac{1}{2}(f(x_j) - y_j)^2$.
- $\frac{\partial L}{\partial z^{(3)}} = z^{(3)} - y_j$.
- $\frac{\partial L}{\partial z^{(2)}} = \frac{\partial z^{(3)}}{\partial z^{(2)}} \frac{\partial L}{\partial z^{(3)}}$, $\frac{\partial L}{\partial w^{(2)}} = \frac{\partial z^{(3)}}{\partial w^{(2)}} \frac{\partial L}{\partial z^{(3)}}$.
- $\frac{\partial L}{\partial z^{(1)}} = \frac{\partial z^{(2)}}{\partial z^{(1)}} \frac{\partial L}{\partial z^{(2)}}$, $\frac{\partial L}{\partial w^{(1)}} = \frac{\partial z^{(2)}}{\partial w^{(1)}} \frac{\partial L}{\partial z^{(2)}}$.

Free $\frac{\partial L}{\partial w^{(1)}}$ from memory.

1D Example

Define a function $f: \mathbb{R} \mapsto \mathbb{R}$

- **Input:** scalar $x^{(0)}$.
- $z^{(1)} = w^{(0)} x^{(0)}$.
- $x^{(1)} = \max\{0, z^{(1)}\}$.
- $z^{(2)} = w^{(1)} x^{(1)}$.
- $x^{(2)} = \max\{0, z^{(2)}\}$.
- $z^{(3)} = w^{(2)} x^{(2)}$.
- **Output:** $f(x^{(0)}) = z^{(3)}$.

Backpropagation:

- Loss: $L = \frac{1}{2}(f(x_j) - y_j)^2$.
- $\frac{\partial L}{\partial z^{(3)}} = z^{(3)} - y_j$.
- $\frac{\partial L}{\partial z^{(2)}} = \frac{\partial z^{(3)}}{\partial z^{(2)}} \frac{\partial L}{\partial z^{(3)}}$, $\frac{\partial L}{\partial w^{(2)}} = \frac{\partial z^{(3)}}{\partial w^{(2)}} \frac{\partial L}{\partial z^{(3)}}$.
- $\frac{\partial L}{\partial z^{(1)}} = \frac{\partial z^{(2)}}{\partial z^{(1)}} \frac{\partial L}{\partial z^{(2)}}$, $\frac{\partial L}{\partial w^{(1)}} = \frac{\partial z^{(2)}}{\partial w^{(1)}} \frac{\partial L}{\partial z^{(2)}}$.
- $\frac{\partial L}{\partial w^{(0)}} = \frac{\partial z^{(1)}}{\partial w^{(0)}} \frac{\partial L}{\partial z^{(1)}}$.

$$\frac{\partial z^{(1)}}{\partial w^{(0)}} = x^{(0)}.$$

1D Example

Define a function $f: \mathbb{R} \mapsto \mathbb{R}$

- **Input:** scalar $x^{(0)}$.
- $z^{(1)} = w^{(0)} x^{(0)}$.
- $x^{(1)} = \max\{0, z^{(1)}\}$.
- $z^{(2)} = w^{(1)} x^{(1)}$.
- $x^{(2)} = \max\{0, z^{(2)}\}$.
- $z^{(3)} = w^{(2)} x^{(2)}$.
- **Output:** $f(x^{(0)}) = z^{(3)}$.

Backpropagation:

- Loss: $L = \frac{1}{2}(f(x_j) - y_j)^2$.
- $\frac{\partial L}{\partial z^{(3)}} = z^{(3)} - y_j$.
- $\frac{\partial L}{\partial z^{(2)}} = \frac{\partial z^{(3)}}{\partial z^{(2)}} \frac{\partial L}{\partial z^{(3)}}$, $\frac{\partial L}{\partial w^{(2)}} = \frac{\partial z^{(3)}}{\partial w^{(2)}} \frac{\partial L}{\partial z^{(3)}}$.
- $\frac{\partial L}{\partial z^{(1)}} = \frac{\partial z^{(2)}}{\partial z^{(1)}} \frac{\partial L}{\partial z^{(2)}}$, $\frac{\partial L}{\partial w^{(1)}} = \frac{\partial z^{(2)}}{\partial w^{(1)}} \frac{\partial L}{\partial z^{(2)}}$.
- $\frac{\partial L}{\partial w^{(0)}} = \frac{\partial z^{(1)}}{\partial w^{(0)}} \frac{\partial L}{\partial z^{(1)}}$.

Free $\frac{\partial L}{\partial z^{(1)}}$ from memory.

1D Example

Define a function $f: \mathbb{R} \mapsto \mathbb{R}$

- **Input:** scalar $x^{(0)}$.
- $z^{(1)} = w^{(0)} x^{(0)}$.
- $x^{(1)} = \max\{0, z^{(1)}\}$.
- $z^{(2)} = w^{(1)} x^{(1)}$.
- $x^{(2)} = \max\{0, z^{(2)}\}$.
- $z^{(3)} = w^{(2)} x^{(2)}$.
- **Output:** $f(x^{(0)}) = z^{(3)}$.

Backpropagation:

- Loss: $L = \frac{1}{2}(f(x_j) - y_j)^2$.
- $\frac{\partial L}{\partial z^{(3)}} = z^{(3)} - y_j$.
- $\frac{\partial L}{\partial z^{(2)}} = \frac{\partial z^{(3)}}{\partial z^{(2)}} \frac{\partial L}{\partial z^{(3)}}$, $\frac{\partial L}{\partial w^{(2)}} = \frac{\partial z^{(3)}}{\partial w^{(2)}} \frac{\partial L}{\partial z^{(3)}}$.
- $\frac{\partial L}{\partial z^{(1)}} = \frac{\partial z^{(2)}}{\partial z^{(1)}} \frac{\partial L}{\partial z^{(2)}}$, $\frac{\partial L}{\partial w^{(1)}} = \frac{\partial z^{(2)}}{\partial w^{(1)}} \frac{\partial L}{\partial z^{(2)}}$.
- $\boxed{\frac{\partial L}{\partial w^{(0)}}} = \frac{\partial z^{(1)}}{\partial w^{(0)}} \frac{\partial L}{\partial z^{(1)}}$.

Update $w^{(0)}$: $w^{(0)} \leftarrow w^{(0)} - \alpha \frac{\partial L}{\partial w^{(0)}}$.

1D Example

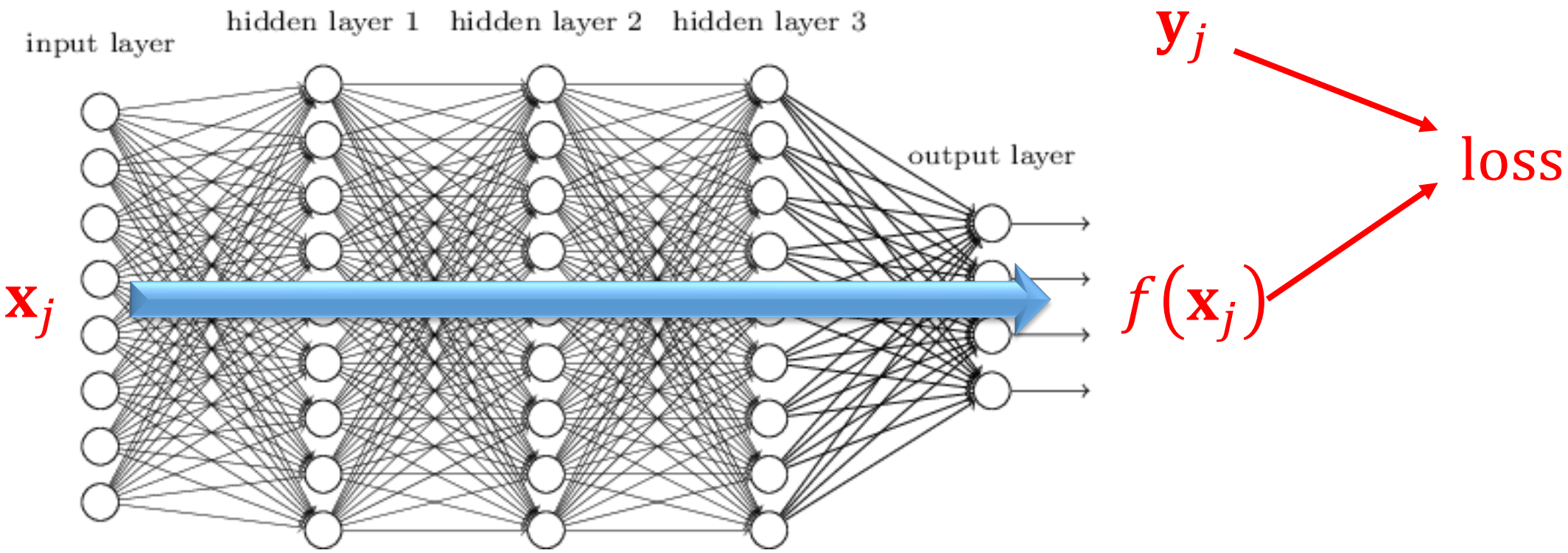
One iteration:

1. Randomly sample j from $\{1, 2, \cdots, n\}$.
2. **Forward pass**: take x_j as input ($x^{(0)} = x_j$), compute each layer $z^{(1)}, x^{(1)}, z^{(2)}, x^{(2)}, z^{(3)}$.
3. **Backward pass**:
   i. Compute the derivatives $\frac{\partial L}{\partial z^{(3)}}, \frac{\partial L}{\partial w^{(2)}}, \frac{\partial L}{\partial z^{(2)}}, \frac{\partial L}{\partial w^{(1)}}, \frac{\partial L}{\partial z^{(1)}}, \frac{\partial L}{\partial w^{(0)}}$.
   ii. Update $w^{(k)}$ using $\frac{\partial L}{\partial w^{(k)}}$.

Summary of Backpropagation

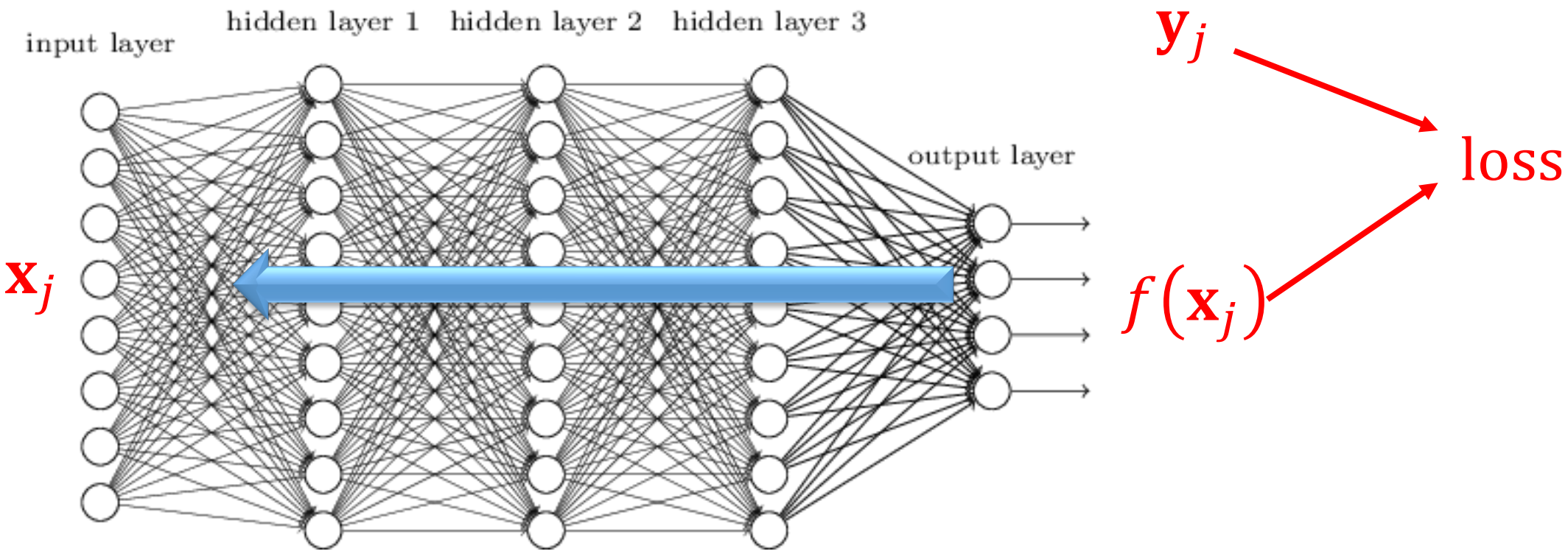
Backpropagation: Take-Home Message

1. Randomly pick a sample $(\mathbf{x}_j, \mathbf{y}_j)$.
2. Run a forward pass (from the input $\mathbf{x}^{(0)}$ to the prediction).



Backpropagation: Take-Home Message

1. Randomly pick a sample $(\mathbf{x}_j, \mathbf{y}_j)$.
2. Run a forward pass (from the input $\mathbf{x}^{(0)}$ to the prediction).
3. Run a backward pass (from the loss to $\mathbf{W}^{(0)}$).



Backpropagation: Take-Home Message

1. Randomly pick a sample $(\mathbf{x}_j, \mathbf{y}_j)$.
2. Run a forward pass (from the input $\mathbf{x}^{(0)}$ to the prediction).
3. Run a backward pass (from the loss to $\mathbf{W}^{(0)}$).

Get the derivatives (stochastic gradients):

$$\frac{\partial \text{Loss}(\mathbf{f}(\mathbf{x}_j), \mathbf{y}_j)}{\partial \mathbf{W}^{(2)}}, \quad \frac{\partial \text{Loss}(\mathbf{f}(\mathbf{x}_j), \mathbf{y}_j)}{\partial \mathbf{W}^{(1)}}, \quad \frac{\partial \text{Loss}(\mathbf{f}(\mathbf{x}_j), \mathbf{y}_j)}{\partial \mathbf{W}^{(0)}}.$$

Backpropagation: Take-Home Message

1. Randomly pick a sample $(\mathbf{x}_j, \mathbf{y}_j)$.
2. Run a forward pass (from the input $\mathbf{x}^{(0)}$ to the prediction).
3. Run a backward pass (from the loss to $\mathbf{W}^{(0)}$).

Get the derivatives (stochastic gradients):

$$\frac{\partial \text{Loss}(\mathbf{f}(\mathbf{x}_j), \mathbf{y}_j)}{\partial \mathbf{W}^{(2)}}, \quad \frac{\partial \text{Loss}(\mathbf{f}(\mathbf{x}_j), \mathbf{y}_j)}{\partial \mathbf{W}^{(1)}}, \quad \frac{\partial \text{Loss}(\mathbf{f}(\mathbf{x}_j), \mathbf{y}_j)}{\partial \mathbf{W}^{(0)}}.$$

Update $\mathbf{W}^{(0)}, \mathbf{W}^{(1)}, \mathbf{W}^{(2)}$ using the derivatives.

Mini-Batch

1. Randomly pick a sample $(\mathbf{x}_j, \mathbf{y}_j)$.
2. Run a forward pass (from the input $\mathbf{x}^{(0)}$ to the prediction).
3. Run a backward pass (from the loss to $\mathbf{W}^{(0)}$).

Several random samples.

Get the derivatives (stochastic gradients):

$$\frac{\partial \text{Loss}(\mathbf{f}(\mathbf{x}_j), \mathbf{y}_j)}{\partial \mathbf{W}^{(2)}}, \quad \frac{\partial \text{Loss}(\mathbf{f}(\mathbf{x}_j), \mathbf{y}_j)}{\partial \mathbf{W}^{(1)}}, \quad \frac{\partial \text{Loss}(\mathbf{f}(\mathbf{x}_j), \mathbf{y}_j)}{\partial \mathbf{W}^{(0)}}.$$

$$\frac{1}{|\mathcal{J}|}\sum_{j\in\mathcal{J}} \frac{\partial \text{Loss}(\mathbf{f}(\mathbf{x}_j), \mathbf{y}_j)}{\partial \mathbf{W}^{(2)}}, \quad \frac{1}{|\mathcal{J}|}\sum_{j\in\mathcal{J}} \frac{\partial \text{Loss}(\mathbf{f}(\mathbf{x}_j), \mathbf{y}_j)}{\partial \mathbf{W}^{(1)}}, \quad \frac{1}{|\mathcal{J}|}\sum_{j\in\mathcal{J}} \frac{\partial \text{Loss}(\mathbf{f}(\mathbf{x}_j), \mathbf{y}_j)}{\partial \mathbf{W}^{(0)}}.$$

Mini-batch should always be used! Set batch size $|\mathcal{J}|$ to 16, 32, 64, …

Mini-Batch

SGD: $\text{BatchSize} = 1$.

- Per-iteration cost is low.
- Lots of iterations to converge.

Mini-Batch: $\text{BatchSize} > 1$.

- Better than the other two, if BatchSize is properly set.

Full Gradient: $\text{BatchSize} = n$.

- Per-iteration cost is n times higher than SGD.
- Convex problem: less number of iterations.
- Neural network: it doesn't work!

First-Order Optimization

- First-order optimization: update the parameters using gradient.
- Gradient descent algorithm (including SGD, mini-batch SGD, and full gradient descent, conjugate gradient) are typical 1st-order algorithms.
- Other 1st-order algorithms: SGD with momentum, AdaGrad, RMSprop…
- See the blogs:
  - http://ruder.io/optimizing-gradient-descent/
  - https://distill.pub/2017/momentum/

Summary of FC Neural Network

Build a Fully-Connected Neural Network

- Network structure

  Number of layers Width of each layer Activation functions

Build a Fully-Connected Neural Network

- Network structure

Number of layers | Width of each layer | Activation functions

Example:

- **Input**: vector $\mathbf{x}^{(0)} \in \mathbb{R}^{785}$. Input layer
- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\,\mathbf{x}^{(0)} \in \mathbb{R}^{d_1}$. Hidden Layer 1
- $\mathbf{x}^{(1)} = \max\{\mathbf{0},\ \mathbf{z}^{(1)}\} \in \mathbb{R}^{d_1}$.
- $\mathbf{z}^{(2)} = \mathbf{W}^{(1)}\,\mathbf{x}^{(1)} \in \mathbb{R}^{d_2}$. Hidden Layer 2
- $\mathbf{x}^{(2)} = \max\{\mathbf{0},\ \mathbf{z}^{(2)}\} \in \mathbb{R}^{d_2}$.
- $\mathbf{z}^{(3)} = \mathbf{W}^{(2)}\,\mathbf{x}^{(2)} \in \mathbb{R}^{10}$. Output layer
- **Output**: $\mathbf{f}(\mathbf{x}^{(0)}) = \mathrm{SoftMax}(\mathbf{z}^{(2)})$.

- Three layers (2 hidden and 1 output).
  - Input layer doesn't count (no parameter).

Build a Fully-Connected Neural Network

- Network structure

Number of layers | Width of each layer | Activation functions

Example:

- **Input**: vector $\mathbf{x}^{(0)} \in \mathbb{R}^{785}$.
- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\,\mathbf{x}^{(0)} \in \mathbb{R}^{d_1}$.
- $\mathbf{x}^{(1)} = \max\{\mathbf{0},\ \mathbf{z}^{(1)}\} \in \mathbb{R}^{d_1}$.
- $\mathbf{z}^{(2)} = \mathbf{W}^{(1)}\,\mathbf{x}^{(1)} \in \mathbb{R}^{d_2}$.
- $\mathbf{x}^{(2)} = \max\{\mathbf{0},\ \mathbf{z}^{(2)}\} \in \mathbb{R}^{d_2}$.
- $\mathbf{z}^{(3)} = \mathbf{W}^{(2)}\,\mathbf{x}^{(2)} \in \mathbb{R}^{10}$.
- **Output**: $\mathbf{f}(\mathbf{x}^{(0)}) = \mathrm{SoftMax}(\mathbf{z}^{(2)})$.

- Three layers (2 hidden and 1 output).
  - Input layer doesn't count (no parameter).
- Width of each layer:
  - Layer 1: d_1,
  - Layer 2: d_2,
  - Output layer: 10.

Build a Fully-Connected Neural Network

- Network structure

Number of layers | Width of each layer | Activation functions

Example:

- **Input**: vector $\mathbf{x}^{(0)} \in \mathbb{R}^{785}$.
- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\,\mathbf{x}^{(0)} \in \mathbb{R}^{d_1}$.
- $\mathbf{x}^{(1)} = \max\{\mathbf{0},\ \mathbf{z}^{(1)}\} \in \mathbb{R}^{d_1}$.
- $\mathbf{z}^{(2)} = \mathbf{W}^{(1)}\,\mathbf{x}^{(1)} \in \mathbb{R}^{d_2}$.
- $\mathbf{x}^{(2)} = \max\{\mathbf{0},\ \mathbf{z}^{(2)}\} \in \mathbb{R}^{d_2}$.
- $\mathbf{z}^{(3)} = \mathbf{W}^{(2)}\,\mathbf{x}^{(2)} \in \mathbb{R}^{10}$.
- **Output**: $\mathbf{f}(\mathbf{x}^{(0)}) = \mathrm{SoftMax}(\mathbf{z}^{(2)})$.

- Three layers (2 hidden and 1 output).
  - Input layer doesn't count (no parameter).
- Width of each layer:
  - Layer 1: d_1,
  - Layer 2: d_2,
  - Output layer: 10.
- Activation functions:
  - Layer 1: ReLU,
  - Layer 2: ReLU,
  - Output layer: Softmax.

Build a Fully-Connected Neural Network

- Network structure

Number of layers | Width of each layer | Activation functions

Example:

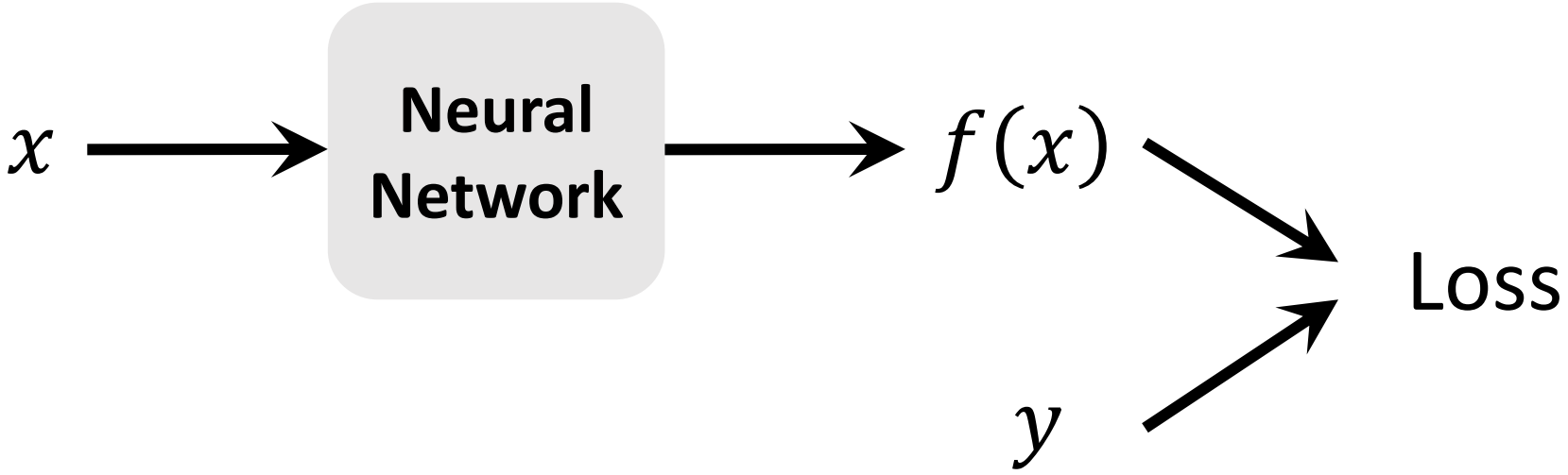
- **Input**: vector $\mathbf{x}^{(0)} \in \mathbb{R}^{785}$.
- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\,\mathbf{x}^{(0)} \in \mathbb{R}^{d_1}$.
- $\mathbf{x}^{(1)} = \max\{\mathbf{0},\ \mathbf{z}^{(1)}\} \in \mathbb{R}^{d_1}$.
- $\mathbf{z}^{(2)} = \mathbf{W}^{(1)}\,\mathbf{x}^{(1)} \in \mathbb{R}^{d_2}$.
- $\mathbf{x}^{(2)} = \max\{\mathbf{0},\ \mathbf{z}^{(2)}\} \in \mathbb{R}^{d_2}$.
- $\mathbf{z}^{(3)} = \mathbf{W}^{(2)}\,\mathbf{x}^{(2)} \in \mathbb{R}^{10}$.
- **Output**: $\mathbf{f}(\mathbf{x}^{(0)}) = \mathrm{SoftMax}(\mathbf{z}^{(2)})$.

- Trainable parameters:
  - $\mathbf{W}^{(0)} \in \mathbb{R}^{d_1 \times 785}$,
  - $\mathbf{W}^{(1)} \in \mathbb{R}^{d_2 \times d_1}$,
  - $\mathbf{W}^{(2)} \in \mathbb{R}^{10 \times d_2}$.

- Number of parameters:
  - $785d_1 + d_1d_2 + 10d_2$.

Build a Fully-Connected Neural Network

- Network structure
  - Number of layers
  - Width of each layer
  - Activation functions
- Loss functions
  - Cross-entropy for classification
  - L1 or squared L2 for regression (the labels are continuous)



Build a Fully-Connected Neural Network

- Network structure

  Number of layers | Width of each layer | Activation functions

- Loss functions

  Cross-entropy for classification

  L1 or squared L2 for regression (the labels are continuous)

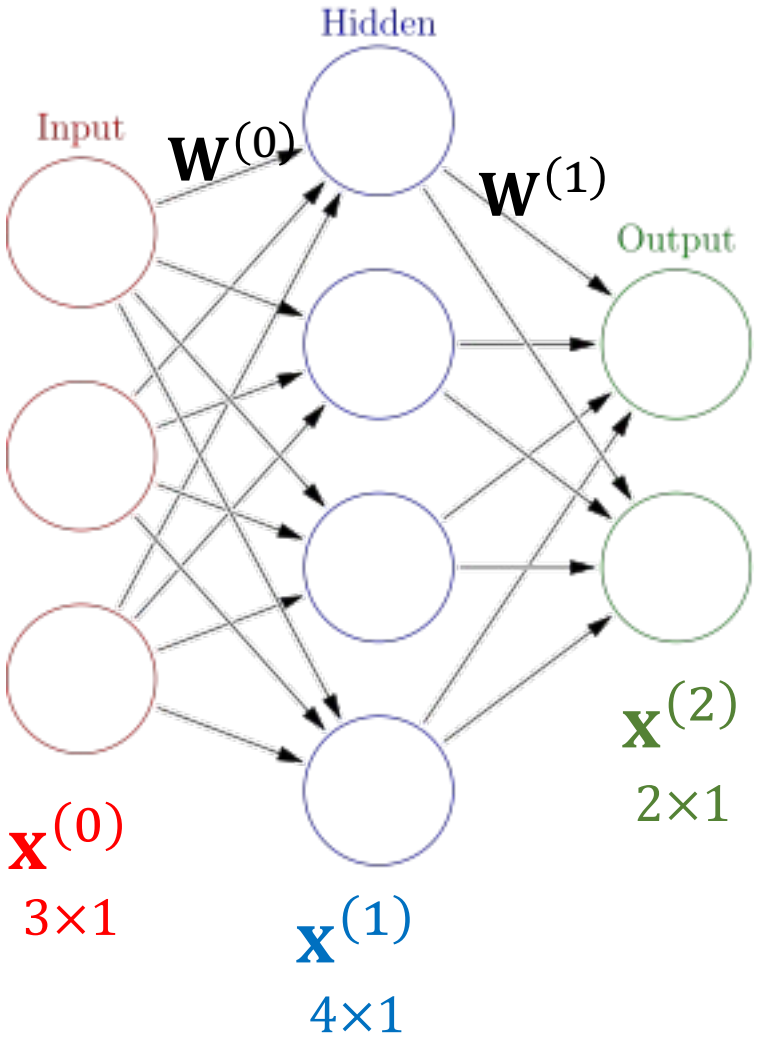
- Compute gradient: by a forward pass and a backward pass

  Automatically performed by many deep learning systems | Batch size

Build a Fully-Connected Neural Network

- Network structure
  - Number of layers
  - Width of each layer
  - Activation functions
- Loss functions
  - Cross-entropy for classification
  - L1 or squared L2 for regression (the labels are continuous)
- Compute gradient: by a forward pass and a backward pass
  - Automatically performed by many deep learning systems
  - Batch size
- Choose an optimization algorithm (and tune its hyper-parameters)
  - SGD
  - SGD with momentum
  - AdaGrad
  - RMSprop

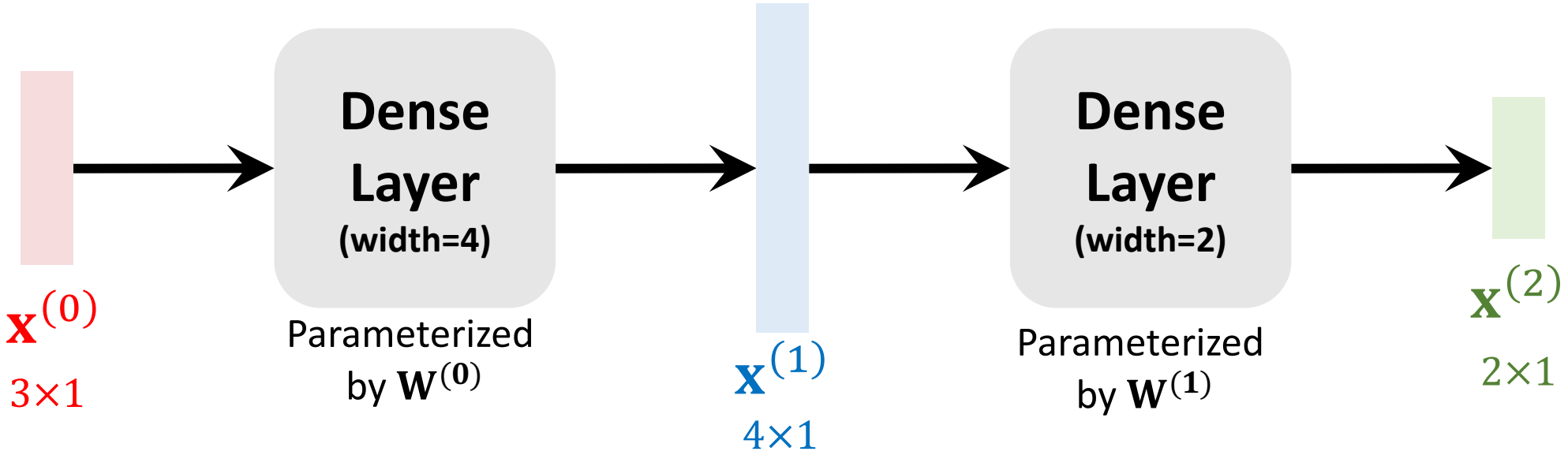
Representing a Neural Network



- A node denotes one entry of vector $\mathbf{x}$.
- A dense layer is parameterized by a matrix $\mathbf{W}$.
- An edge denotes one entry of parameter matrix $\mathbf{W}$.

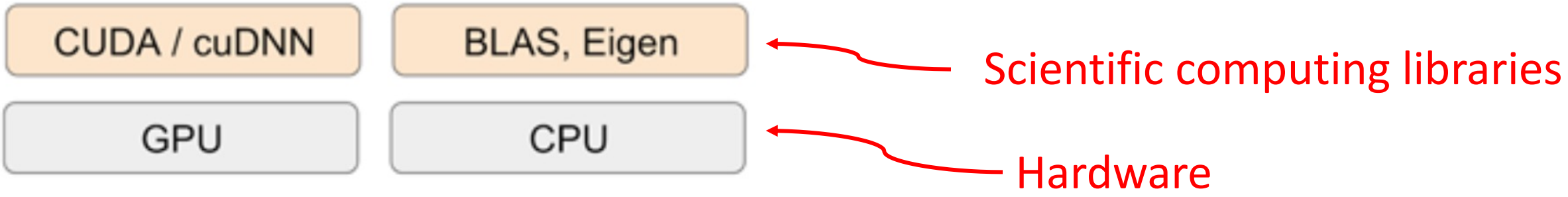
Representing a Neural Network

Equivalent representation:



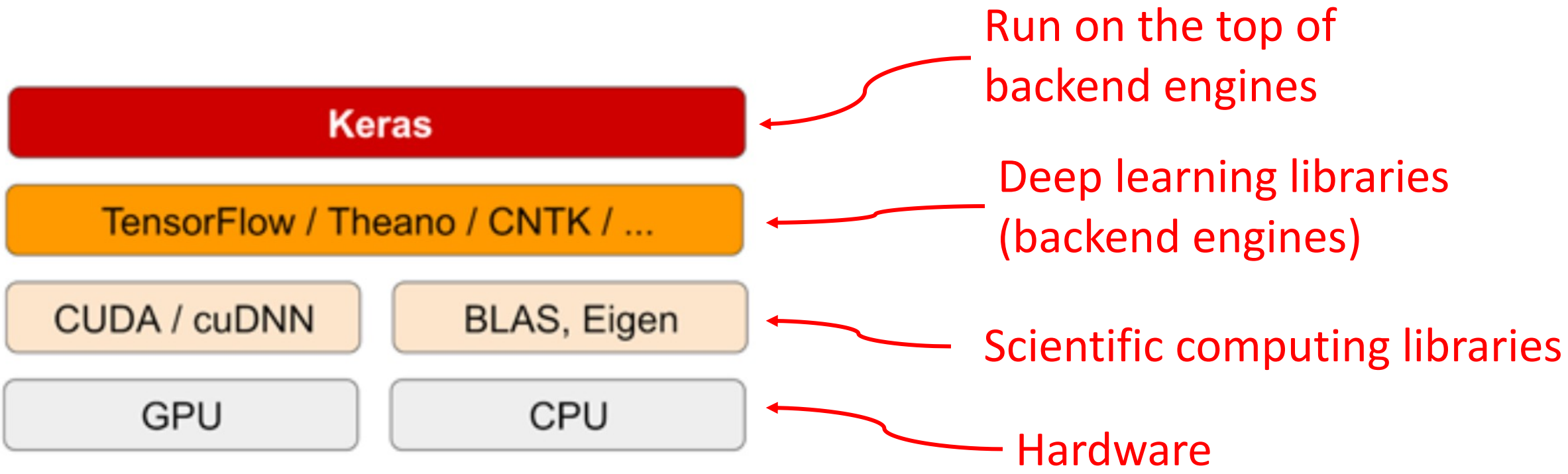
Train a Neural Networks using Keras

Deep Learning Systems



Deep Learning Systems

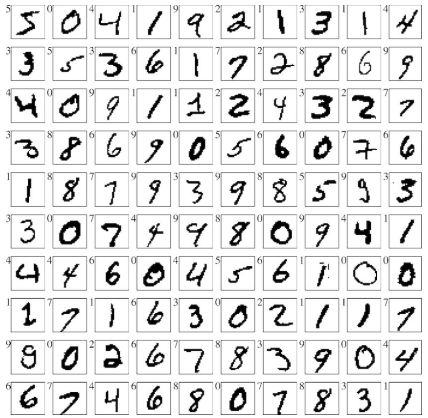
Deep Learning Systems



Implement a Softmax Classifier Using Keras

Softmax Classifier

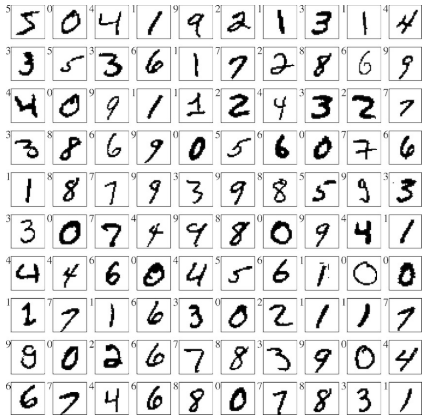
The MNIST Dataset



- $n = 60{,}000$ training samples $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)$.
- Each $\mathbf{x}_j$ is a 28×28 image.
- Each y_j is an integer in $\{0, 1, 2, \cdots, 9\}$.

Softmax Classifier

The MNIST Dataset



- $n = 60{,}000$ training samples $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)$.
- Each $\mathbf{x}_j$ is a 28×28 image.
- Each y_j is an integer in $\{0, 1, 2, \cdots, 9\}$.

Softmax classifier ($\mathbf{f}: \mathbb{R}^{784} \mapsto \mathbb{R}^{10}$) for MNIST:

- **Input**: vector $\mathbf{x} \in \mathbb{R}^{784}$.
- $\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b} \in \mathbb{R}^{10}$.
- **Output**: $\mathbf{f}(\mathbf{x}) = \mathrm{SoftMax}(\mathbf{z})$.

Trainable parameters:

- $\mathbf{W} \in \mathbb{R}^{10 \times 784}$
- $\mathbf{b} \in \mathbb{R}^{10}$

1. Load and Process the MNIST Dataset

Load data

```
from keras.datasets import mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()

print('Shape of x_train: ' + str(x_train.shape))
print('Shape of x_test: ' + str(x_test.shape))
print('Shape of y_train: ' + str(y_train.shape))
print('Shape of y_test: ' + str(y_test.shape))
```

```
Shape of x_train: (60000, 28, 28)
Shape of x_test: (10000, 28, 28)
Shape of y_train: (60000,)
Shape of y_test: (10000,)
```

1. Load and Process the MNIST Dataset

Vectorization: convert the 28×28 images to 784-dim vectors.

```
x_train_vec = x_train.reshape(60000, 784)
x_test_vec = x_test.reshape(10000, 784)

print('Shape of x_train_vec is ' + str(x_train_vec.shape))
```

```
Shape of x_train_vec is (60000, 784)
```

1. Load and Process the MNIST Dataset

One-hot encode: convert the labels (an integer in $\{0, 1, \cdots, 9\}$) to 10-dim vectors.

```
def to_one_hot(labels, dimension=10):
    results = np.zeros((len(labels), dimension))
    for i, label in enumerate(labels):
        results[i, label] = 1.
    return results

y_train_vec = to_one_hot(y_train)
y_test_vec = to_one_hot(y_test)
print('Shape of y_train_vec is ' + str(y_train_vec.shape))
```

```
Shape of y_train_vec is (60000, 10)
```

1. Load and Process the MNIST Dataset

Partition the **training** set to **training** and **validation** sets.

1. Load and Process the MNIST Dataset

Partition the **training** set to **training** and **validation** sets.

1. Load and Process the MNIST Dataset

```
rand_indices = np.random.permutation(60000)
train_indices = rand_indices[0:50000]
valid_indices = rand_indices[50000:60000]

x_valid_vec = x_train_vec[valid_indices, :]
y_valid_vec = y_train_vec[valid_indices, :]

x_train_vec = x_train_vec[train_indices, :]
y_train_vec = y_train_vec[train_indices, :]

print('Shape of x_valid_vec: ' + str(x_valid_vec.shape))
print('Shape of y_valid_vec: ' + str(y_valid_vec.shape))
print('Shape of x_train_vec: ' + str(x_train_vec.shape))
print('Shape of y_train_vec: ' + str(y_train_vec.shape))
```

```
Shape of x_valid_vec: (10000, 784)
Shape of y_valid_vec: (10000, 10)
Shape of x_train_vec: (50000, 784)
Shape of y_train_vec: (50000, 10)
```

2. Build the Softmax Classifier

```
from keras import models
from keras import layers

model = models.Sequential()
model.add(layers.Dense(10, activation='softmax', input_shape=(784,)))
```

$$\text{SoftMax}(\mathbf{W}\,\mathbf{x} + \mathbf{b})$$

width of the layer
(output shape)

2. Build the Softmax Classifier

```
from keras import models
from keras import layers

model = models.Sequential()
model.add(layers.Dense(10, activation='softmax', input_shape=(784,)))

# print the summary of the model.
model.summary()
```

$$\text{SoftMax}(\mathbf{W}\,\mathbf{x} + \mathbf{b})$$

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 10)                7850
=================================================================
Total params: 7,850
Trainable params: 7,850
Non-trainable params: 0
_________________________________________________________________
```

3. Train the Softmax Classifier

Specify: optimization algorithm, learning rate (LR), loss function, and metric.

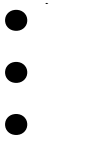
```
from keras import optimizers
model.compile(optimizers.RMSprop(lr=0.0001),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

3. Train the Softmax Classifier

Specify: batch size and number of epochs.

```
history = model.fit(x_train_vec, y_train_vec,
                    batch_size=128, epochs=50,
                    validation_data=(x_valid_vec, y_valid_vec))
```

```
Train on 50000 samples, validate on 10000 samples
Epoch 1/50
50000/50000 [==============================] - 1s 19us/step - loss: 11.2824 - acc: 0.2816 - val_loss: 8.7464 - val_acc: 0.4398
Epoch 2/50
50000/50000 [==============================] - 1s 13us/step - loss: 7.1525 - acc: 0.5384 - val_loss: 5.7830 - val_acc: 0.6221
Epoch 3/50
50000/50000 [==============================] - 1s 13us/step - loss: 5.1013 - acc: 0.6661 - val_loss: 4.5210 - val_acc: 0.7036
Epoch 4/50
50000/50000 [==============================] - 1s 13us/step - loss: 4.0385 - acc: 0.7339 - val_loss: 3.7206 - val_acc: 0.7533
```



```
Epoch 49/50
50000/50000 [==============================] - 1s 13us/step - loss: 1.1430 - acc: 0.9228 - val_loss: 1.3567 - val_acc: 0.9088
Epoch 50/50
50000/50000 [==============================] - 1s 14us/step - loss: 1.1407 - acc: 0.9230 - val_loss: 1.3526 - val_acc: 0.9077
```

3. Train the Softmax Classifier

Epoch: One pass over the n training samples.

Iteration: One update of the model parameters.

- Suppose we use mini-batch SGD and set the batch size to b.
- In the t-th iteration,
  - use a batch of b samples to evaluate the gradient, $\bar{\mathbf{g}}_t$;
  - update the model parameters using $\bar{\mathbf{g}}_t$.
- $\frac{n}{b}$ *iterations* is equivalent to *one epoch*.

4. Examine the Results

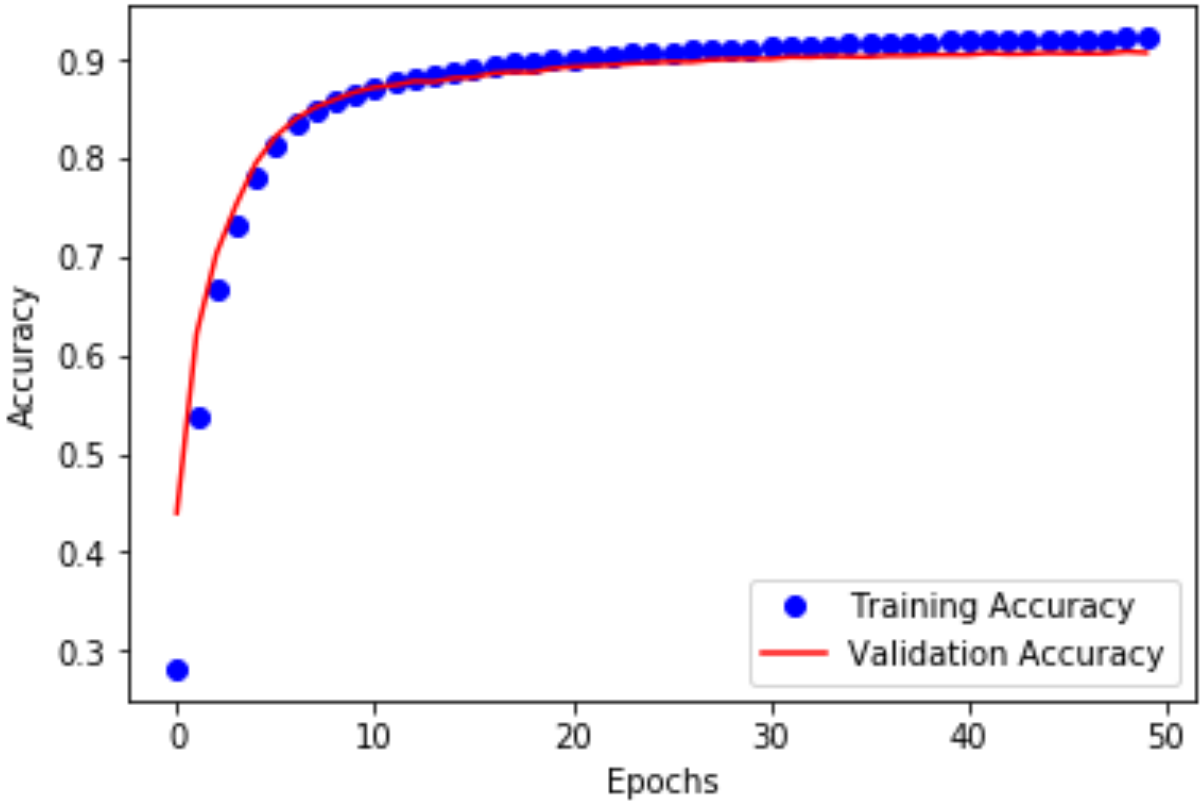
Plot the ***accuracy*** against ***epochs*** (1 epoch = 1 pass over the data).

```
import matplotlib.pyplot as plt
%matplotlib inline

epochs = range(50) # 50 is the number of epochs
train_acc = history.history['acc']
valid_acc = history.history['val_acc']
plt.plot(epochs, train_acc, 'bo', label='Training Accuracy')
plt.plot(epochs, valid_acc, 'r', label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

4. Examine the Results

Plot the ***accuracy*** against ***epochs*** (1 epoch = 1 pass over the data).



Why using the validation set?

- Tune the hyper-parameters, e.g., learning rate, regularization.
- Check if overfitting or underfitting.

4. Examine the Results

Evaluate the model on the ***test*** set. (So far, the model has not seen the ***test*** set.)

```
loss_and_acc = model.evaluate(x_test_vec, y_test_vec)
print('loss = ' + str(loss_and_acc[0]))
print('accuracy = ' + str(loss_and_acc[1]))
```

```
10000/10000 [==============================] - 0s 20us/step
loss = 1.2781493856459185
accuracy = 0.9131
```

4. Examine the Results

- Training accuracy: 92.3%
- Validation accuracy: 90.8%
- Test accuracy: 91.3%

Not bad! A random guess has only 10% accuracy.

Implement a Neural Network Using Keras

Full-Connected Neural Network

- **Input**: vector $\mathbf{x}^{(0)} \in \mathbb{R}^{784}$.

- $\mathbf{z}^{(1)} = \mathbf{W}^{(0)}\,\mathbf{x}^{(0)} + \mathbf{b}^{(0)} \in \mathbb{R}^{d_1}$.
- $\mathbf{x}^{(1)} = \max\{\mathbf{0},\ \mathbf{z}^{(1)}\} \in \mathbb{R}^{d_1}$.

Hidden Layer 1

- $\mathbf{z}^{(2)} = \mathbf{W}^{(1)}\,\mathbf{x}^{(1)} + \mathbf{b}^{(1)} \in \mathbb{R}^{d_2}$.
- $\mathbf{x}^{(2)} = \max\{\mathbf{0},\ \mathbf{z}^{(2)}\} \in \mathbb{R}^{d_2}$.

Hidden Layer 2

- $\mathbf{z}^{(3)} = \mathbf{W}^{(2)}\,\mathbf{x}^{(2)} + \mathbf{b}^{(2)} \in \mathbb{R}^{10}$.

Output Layer

- **Output**: $\mathrm{SoftMax}\left(\mathbf{z}^{(3)}\right)$.

Trainable parameters:

- $\mathbf{W}^{(0)} \in \mathbb{R}^{d_1 \times 784}$,
- $\mathbf{W}^{(1)} \in \mathbb{R}^{d_2 \times d_1}$,
- $\mathbf{W}^{(2)} \in \mathbb{R}^{10 \times d_2}$,
- $\mathbf{b}^{(0)} \in \mathbb{R}^{d_1}$,
- $\mathbf{b}^{(1)} \in \mathbb{R}^{d_1}$,
- $\mathbf{b}^{(2)} \in \mathbb{R}^{10}$.

2. Build the Neural Network

```
from keras import models
from keras import layers

d1 = 500 # width of the 1st hidden layer
d2 = 500 # width of the 2nd hidden layer

model = models.Sequential()
model.add(layers.Dense(d1, activation='relu', input_shape=(784,)))
model.add(layers.Dense(d2, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
```

2. Build the Neural Network

```
# print the summary of the model.
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_4 (Dense)              (None, 500)               392500    
_________________________________________________________________
dense_5 (Dense)              (None, 500)               250500    
_________________________________________________________________
dense_6 (Dense)              (None, 10)                5010      
=================================================================
Total params: 648,010
Trainable params: 648,010
Non-trainable params: 0
```

3. Train the Neural Network

Specify: optimization algorithm, learning rate (LR), loss function, and metric.

```
from keras import optimizers
model.compile(optimizers.RMSprop(lr=0.0001),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

3. Train the Neural Network

Specify: batch size and number of epochs.

```
history = model.fit(x_train_vec, y_train_vec,
                    batch_size=128, epochs=50,
                    validation_data=(x_valid_vec, y_valid_vec))
```

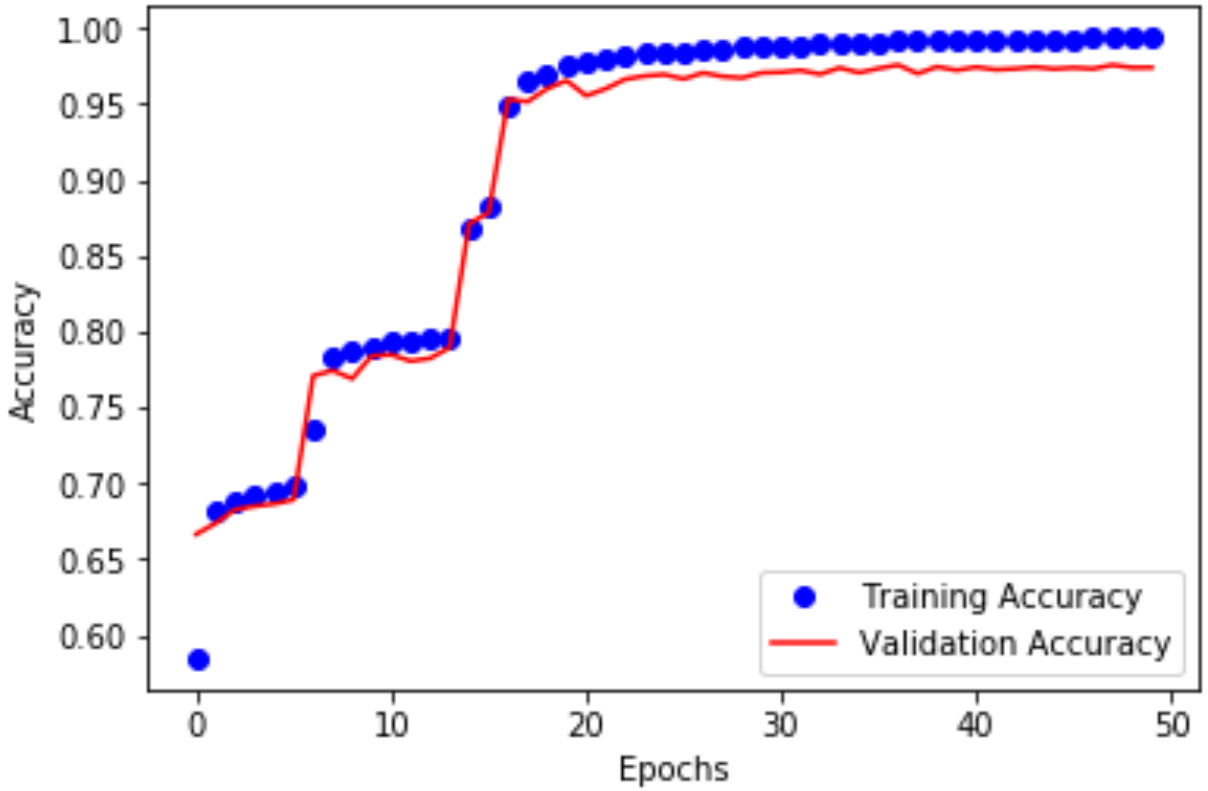
```
Train on 50000 samples, validate on 10000 samples
Epoch 1/50
50000/50000 [==============================] - 5s 92us/step - loss: 6.5323 - acc: 0.5850 - val_loss: 5.2988 - val_acc: 0.6666
Epoch 2/50
50000/50000 [==============================] - 4s 84us/step - loss: 5.0559 - acc: 0.6817 - val_loss: 5.1904 - val_acc: 0.6738
Epoch 3/50
50000/50000 [==============================] - 3s 64us/step - loss: 4.9762 - acc: 0.6881 - val_loss: 5.0556 - val_acc: 0.6828
```

-
-
-

```
Epoch 49/50
50000/50000 [==============================] - 5s 93us/step - loss: 0.0912 - acc: 0.9934 - val_loss: 0.3404 - val_acc: 0.9738
Epoch 50/50
50000/50000 [==============================] - 4s 79us/step - loss: 0.0920 - acc: 0.9934 - val_loss: 0.3327 - val_acc: 0.9739
```

4. Examine the Results

Plot the ***accuracy*** against ***epochs*** (1 epoch = 1 pass over the data).



4. Examine the Results

Evaluate the model on the ***test*** set. (So far, the model has not seen the ***test*** set.)

```
loss_and_acc = model.evaluate(x_test_vec, y_test_vec)
print('loss = ' + str(loss_and_acc[0]))
print('accuracy = ' + str(loss_and_acc[1]))
```

```
10000/10000 [==============================] - 0s 43us/step
loss = 0.3648844491034643
accuracy = 0.9724
```

4. Examine the Results

- Training accuracy: 99.3%
- Validation accuracy: 97.4%
- Test accuracy: 97.2%

Much better than the linear softmax classifier (91.3% accuracy).

Summary

Train a Neural Network for Digit Recognition

1. Examine and process the data.
   - Reshape images to vectors. (Matrix to vector.)
   - One-hot encode the labels. (Integer to vector.)
   - Partition to training and validation sets.

Train a Neural Network for Digit Recognition

1. Examine and process the data.
2. Build the network (just like Lego.)

Train a Neural Network for Digit Recognition

1. Examine and process the data.
2. Build the network (just like Lego.)
3. Compile the model (specify loss function, optimization algorithm, and evaluation metrics.)

Train a Neural Network for Digit Recognition

1. Examine and process the data.
2. Build the network (just like Lego.)
3. Compile the model (specify loss function, optimization algorithm, and evaluation metrics.)
4. Fit the model on training data.
   - Record training and validation accuracies.
   - Use the validation accuracies to tune the hyper-parameter.

Train a Neural Network for Digit Recognition

1. Examine and process the data.
2. Build the network (just like Lego.)
3. Compile the model (specify loss function, optimization algorithm, and evaluation metrics.)
4. Fit the model on training data.
5. Make predictions for test data.