Neural networks

STAT 4710

November 16, 2023

Where we are

✓ **Unit 1:** R for data mining

✓ **Unit 2:** Prediction fundamentals

✓ **Unit 3:** Regression-based methods

✓ **Unit 4:** Tree-based methods

Unit 5: Deep learning

Lecture 1: Deep learning preliminaries

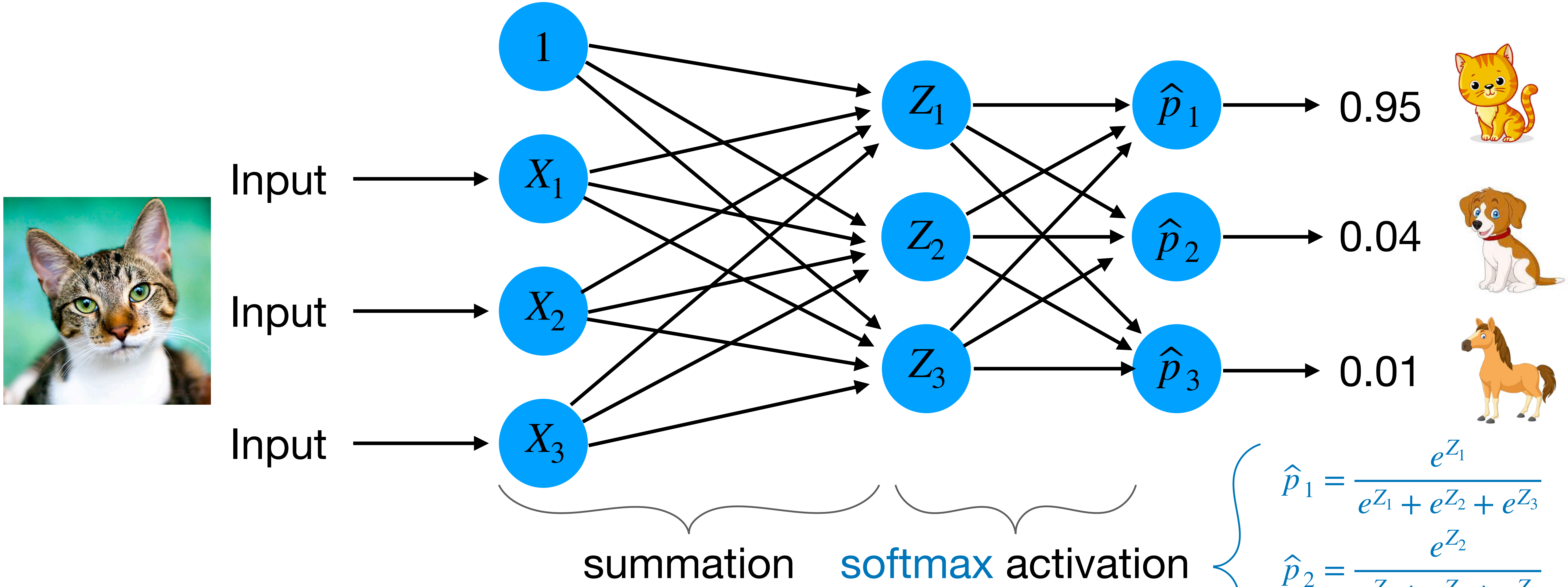
Lecture 2: Neural networks

Lecture 3: Deep learning for images

Lecture 4: Deep learning for text

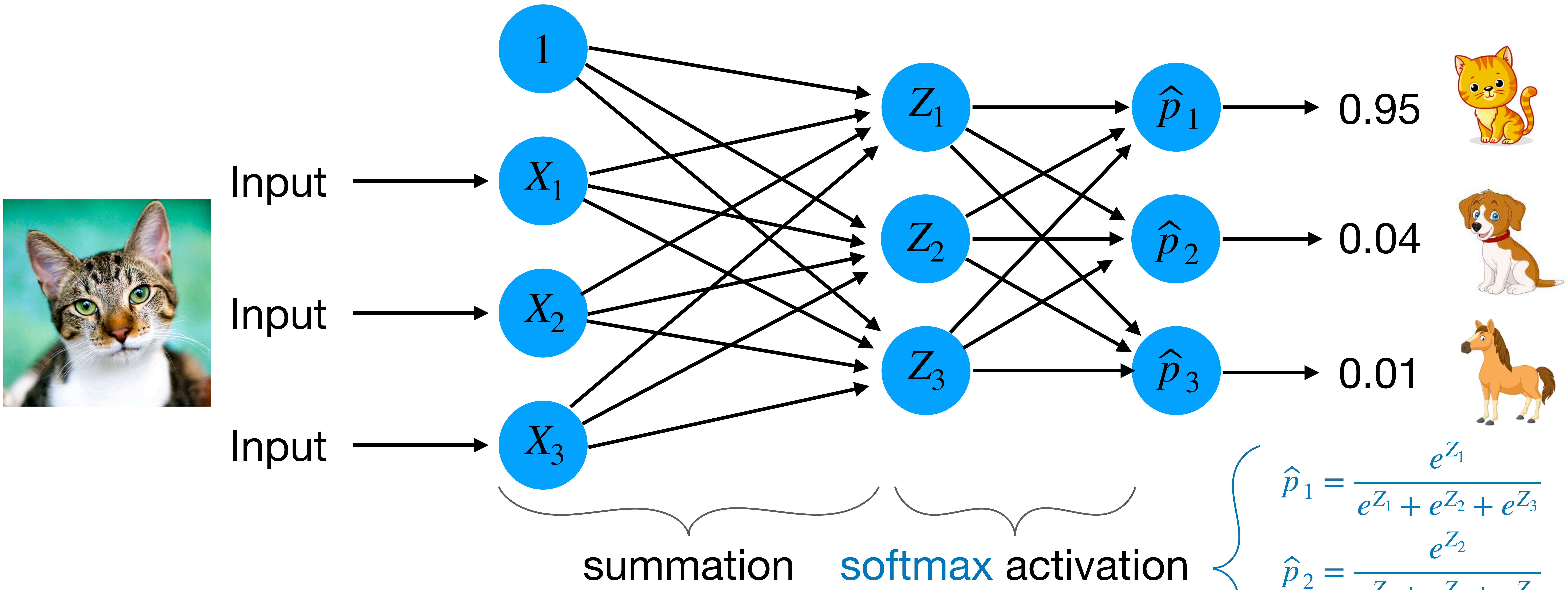
Lecture 5: Unit review and quiz in class

Recall: Multi-class logistic model



$$\begin{cases} \hat{p}_1 = \dfrac{e^{Z_1}}{e^{Z_1} + e^{Z_2} + e^{Z_3}} \\ \hat{p}_2 = \dfrac{e^{Z_2}}{e^{Z_1} + e^{Z_2} + e^{Z_3}} \\ \hat{p}_3 = \dfrac{e^{Z_3}}{e^{Z_1} + e^{Z_2} + e^{Z_3}} \end{cases}$$

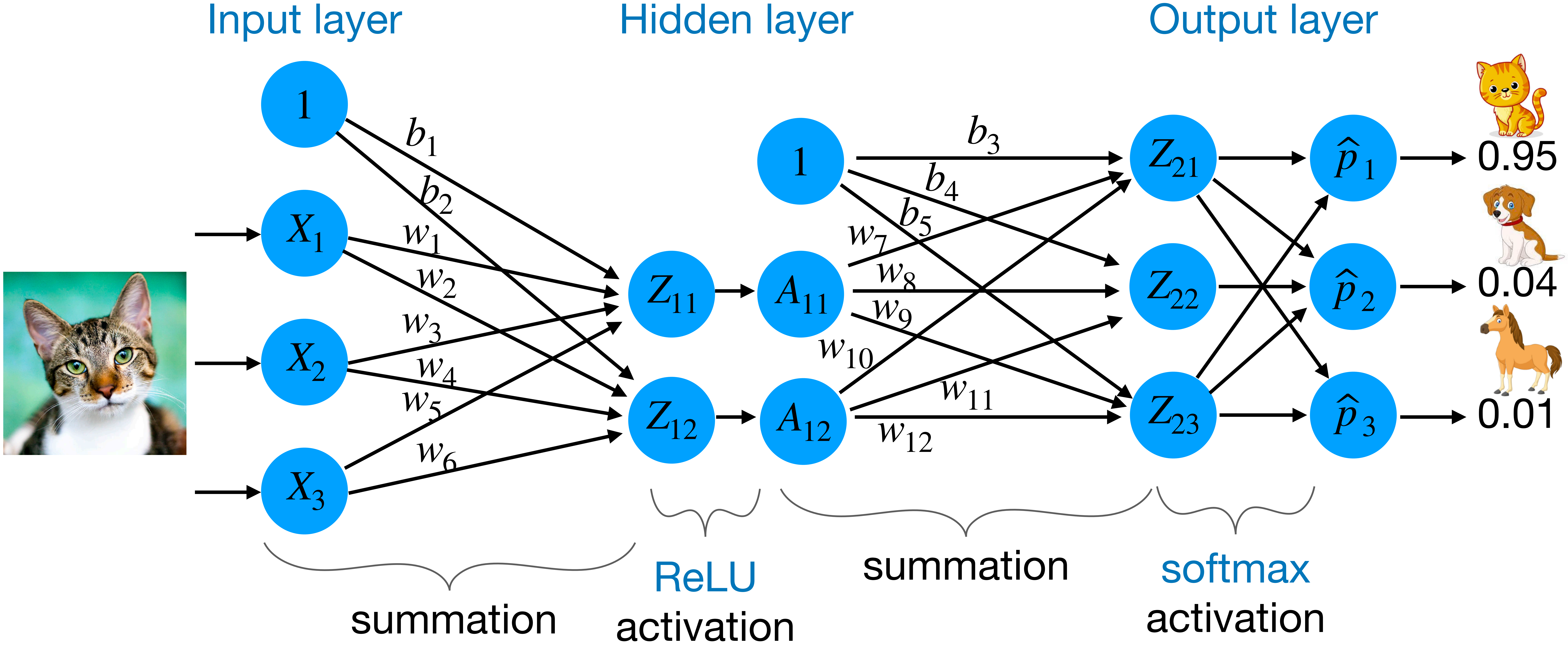
Recall: Multi-class logistic model



$$\begin{cases} \hat{p}_1 = \dfrac{e^{Z_1}}{e^{Z_1} + e^{Z_2} + e^{Z_3}} \\ \hat{p}_2 = \dfrac{e^{Z_2}}{e^{Z_1} + e^{Z_2} + e^{Z_3}} \\ \hat{p}_3 = \dfrac{e^{Z_3}}{e^{Z_1} + e^{Z_2} + e^{Z_3}} \end{cases}$$

This simple architecture directly uses raw pixel features.
Intuitively, a good classifier would pick up on cat’s eyes, ears, etc.

One-hidden-layer neural network

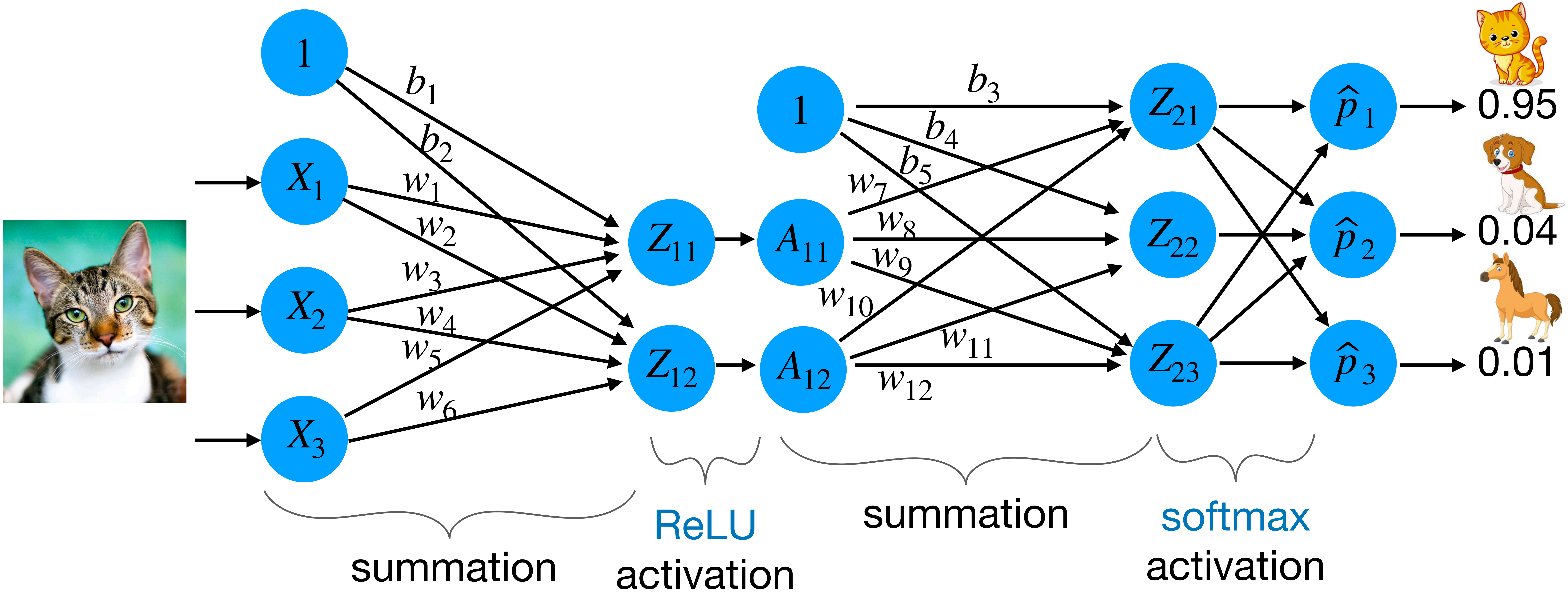


One-hidden-layer neural network

Input layer

Hidden layer

Output layer



summation

ReLU activation

summation

softmax activation

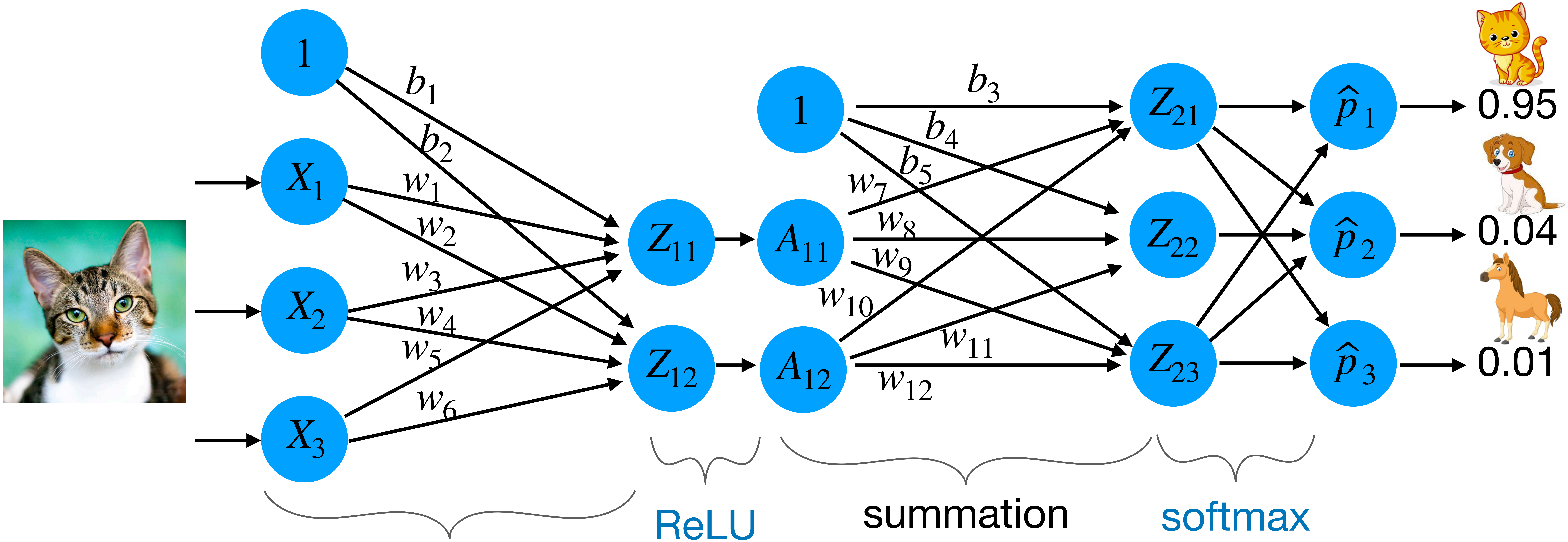
E.g., $Z_{12} = b_2 + w_2X_1 + w_4X_2 + w_6X_3$

One-hidden-layer neural network

Input layer

Hidden layer

Output layer

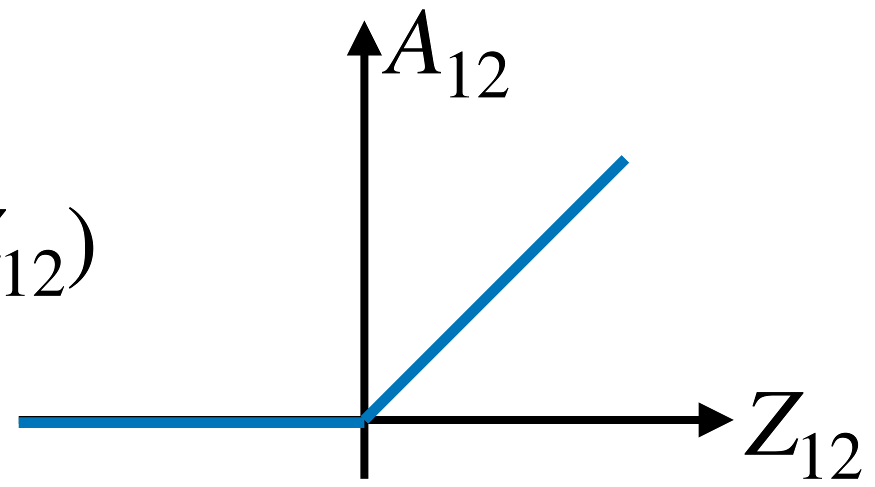


summation

E.g., $Z_{12} = b_2 + w_2X_1 + w_4X_2 + w_6X_3$

ReLU activation

E.g., $A_{12} = \max(0, Z_{12})$



summation

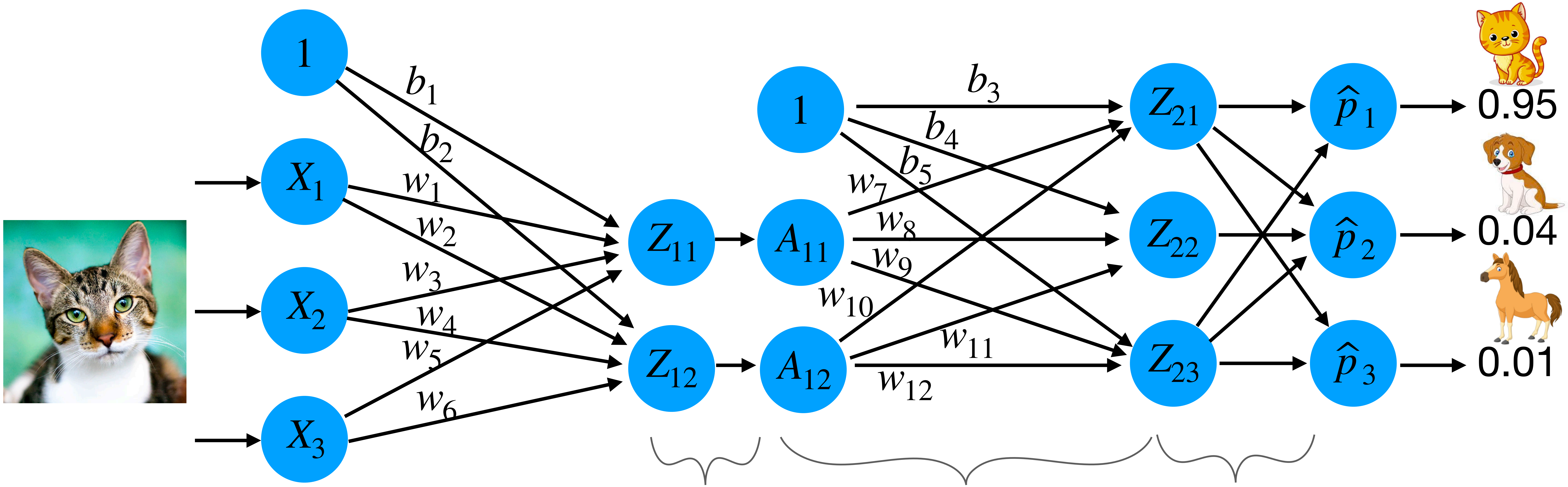
softmax activation

One-hidden-layer neural network

Input layer

Hidden layer

Output layer



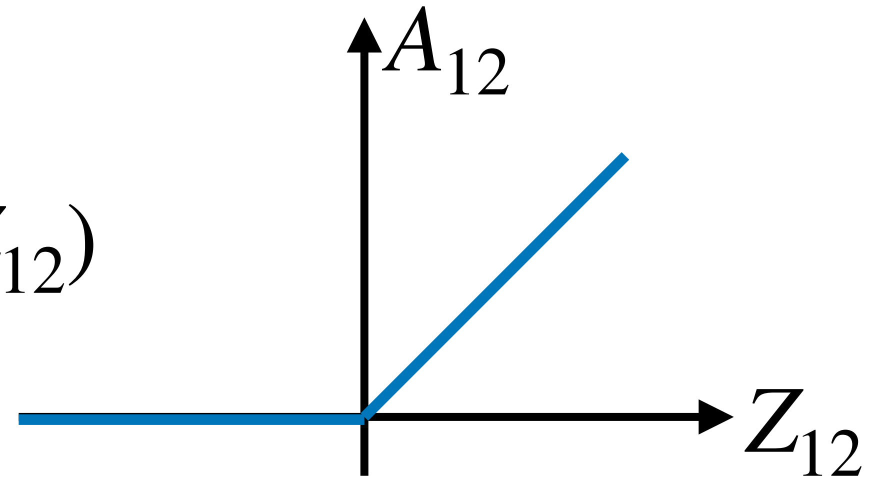
summation

E.g., $Z_{12} = b_2 + w_2X_1 + w_4X_2 + w_6X_3$

ReLU

activation

E.g., $A_{12} = \max(0, Z_{12})$



summation

softmax

activation

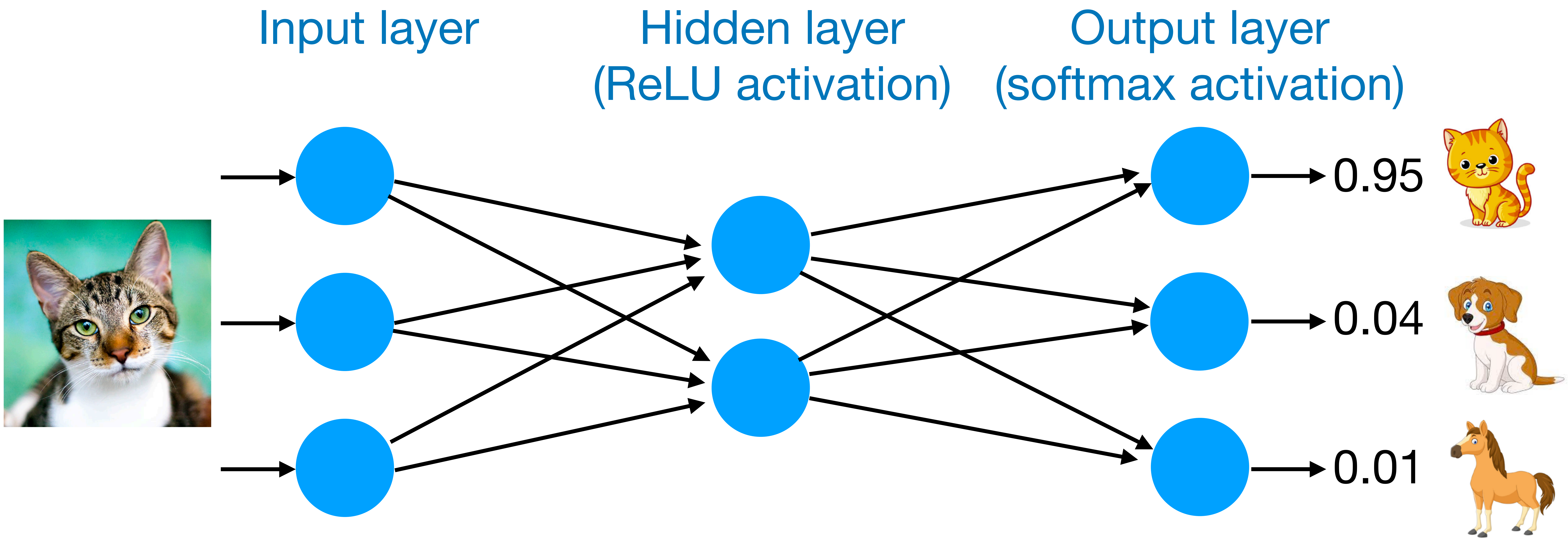
Terminology:

b_j: biases

w_j: weights

One-hidden-layer neural network

Concise representation



Input layer

Hidden layer
(ReLU activation)

Output layer
(softmax activation)

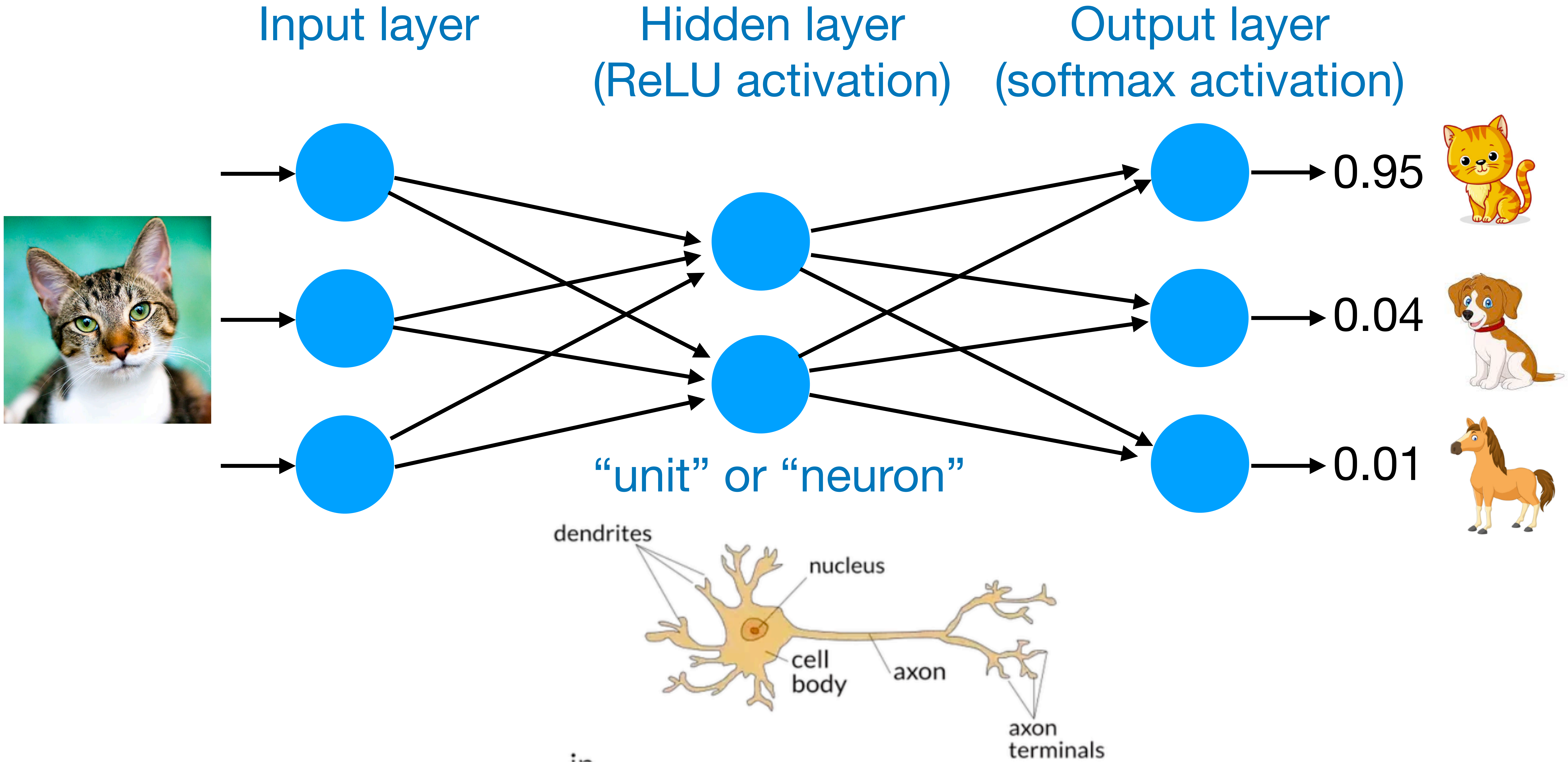
One-hidden-layer neural network

Concise representation

Input layer

Hidden layer
(ReLU activation)

Output layer
(softmax activation)



"unit" or "neuron"

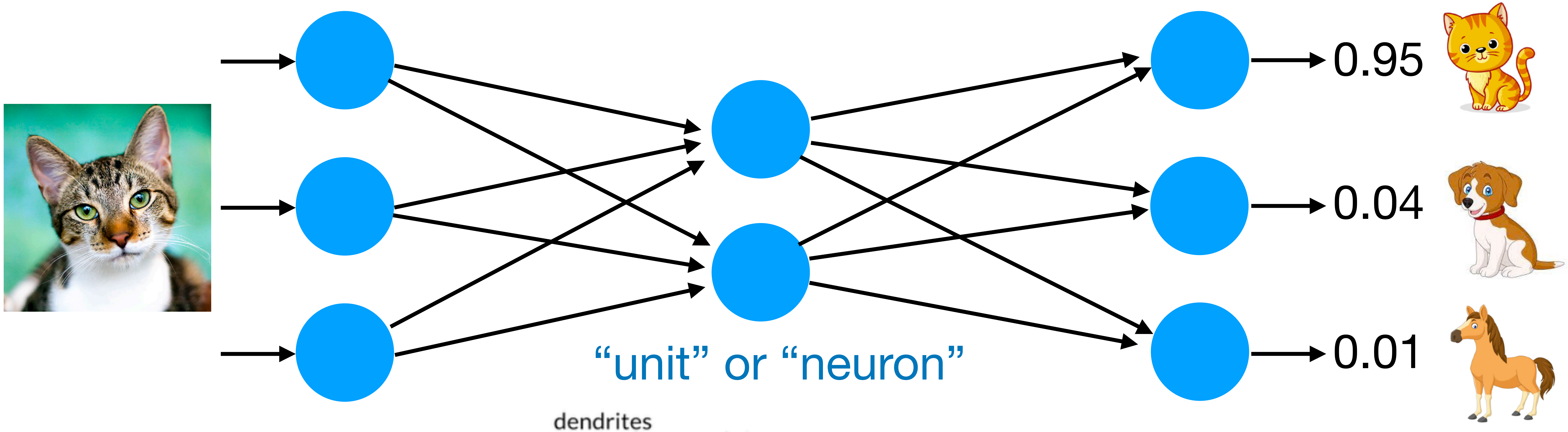
One-hidden-layer neural network

Concise representation

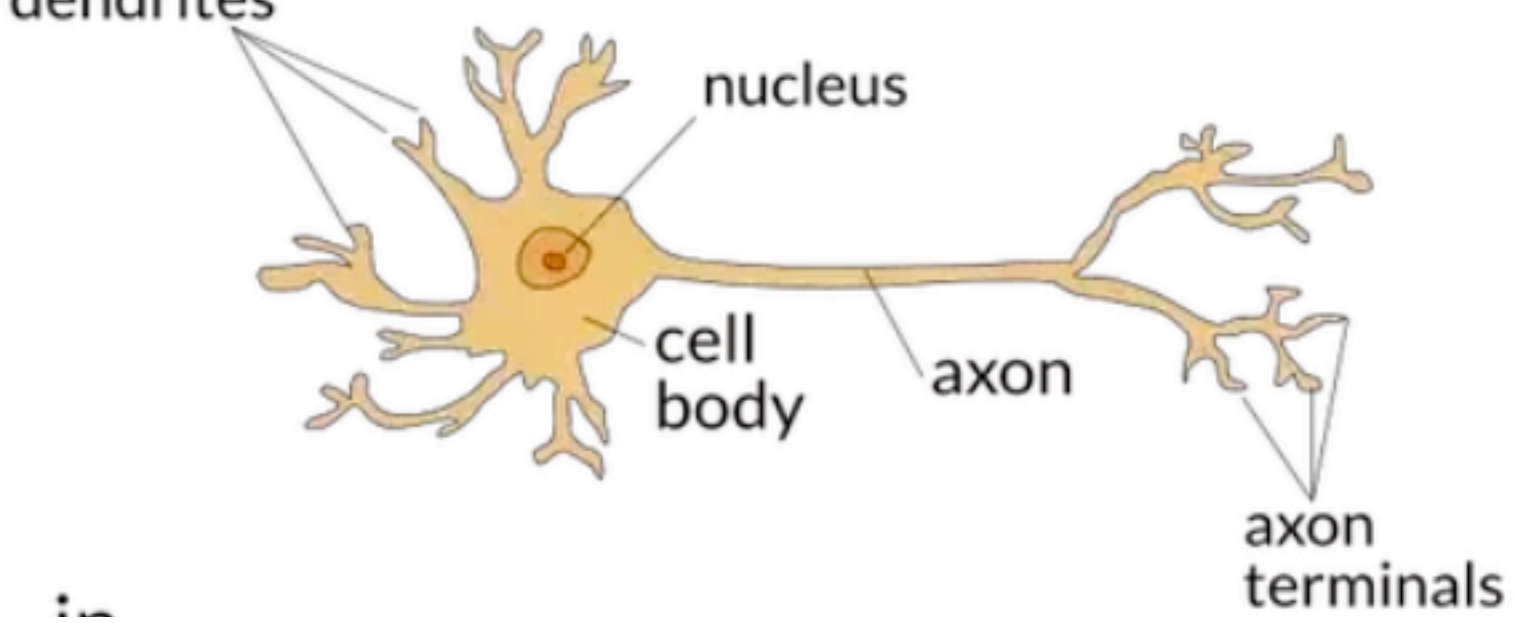
Input layer

Hidden layer
(ReLU activation)

Output layer
(softmax activation)



“unit” or “neuron”

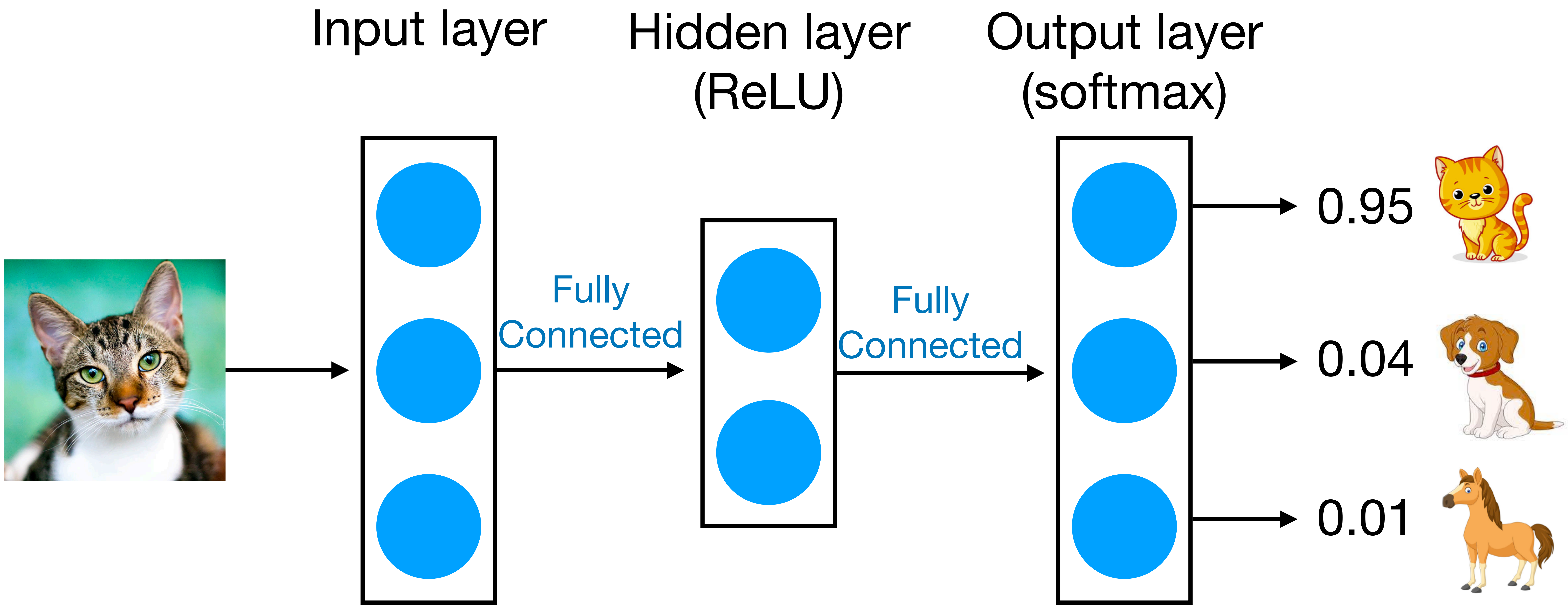


Notes:

- Bias terms left implicit.
- Fully connected architecture.
- Hidden layer: learned features.

One-hidden-layer neural network

Even more concise representation



Input layer

Hidden layer (ReLU)

Output layer (softmax)

Fully Connected

Fully Connected

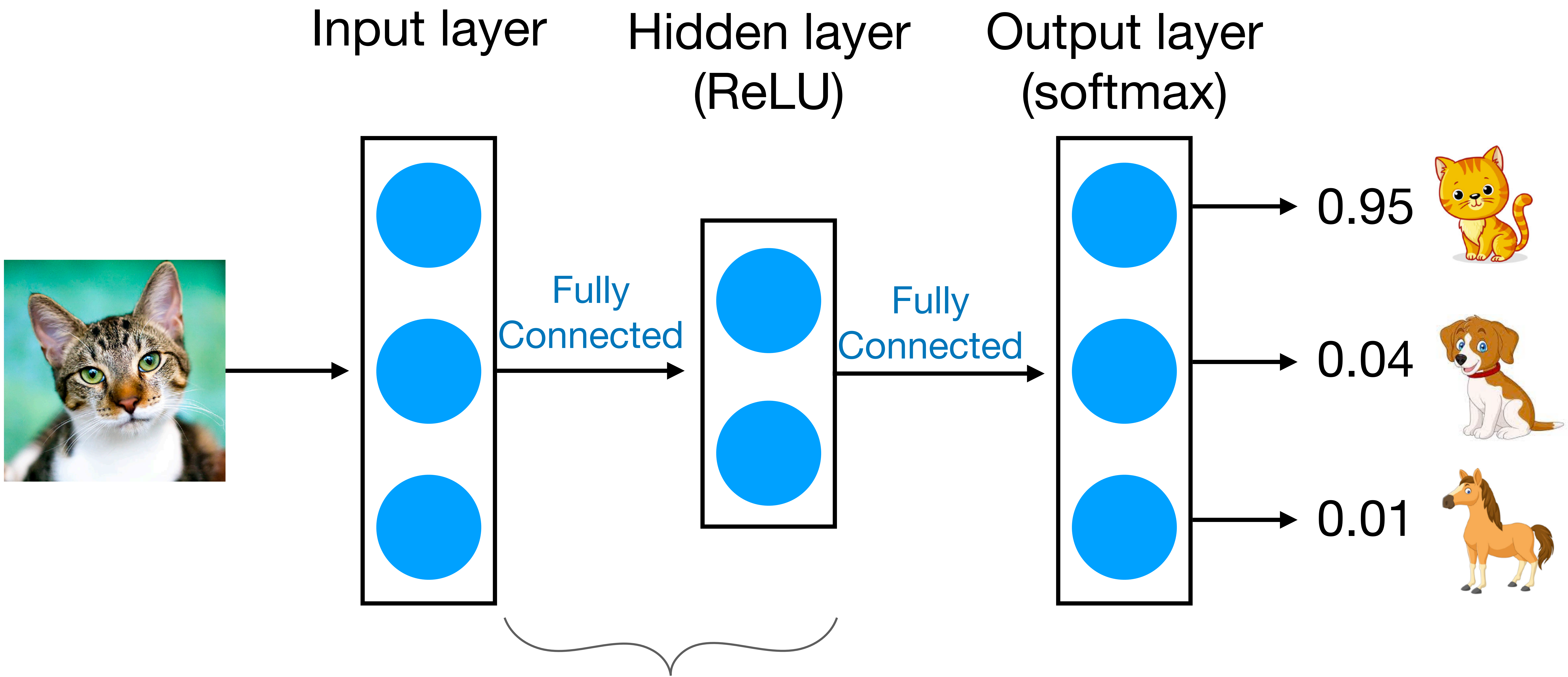
0.95

0.04

0.01

One-hidden-layer neural network

Even more concise representation



Input layer

Hidden layer (ReLU)

Output layer (softmax)

Fully Connected

Fully Connected

0.95

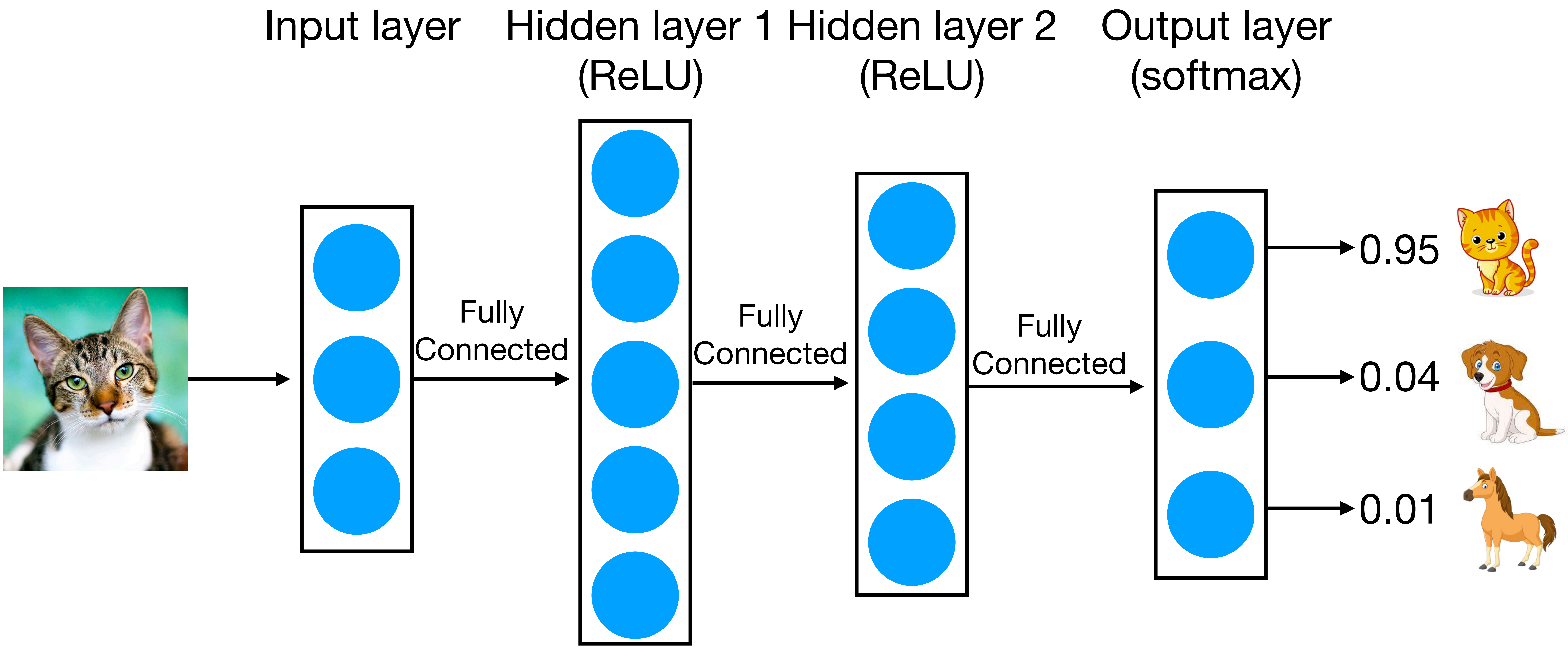
0.04

0.01

Number of parameters in FC layer =

(# units in previous layer + 1) × (# units in this layer)

Multi-layer fully connected neural networks



Input layer

Hidden layer 1 (ReLU)

Hidden layer 2 (ReLU)

Output layer (softmax)

Fully Connected

Fully Connected

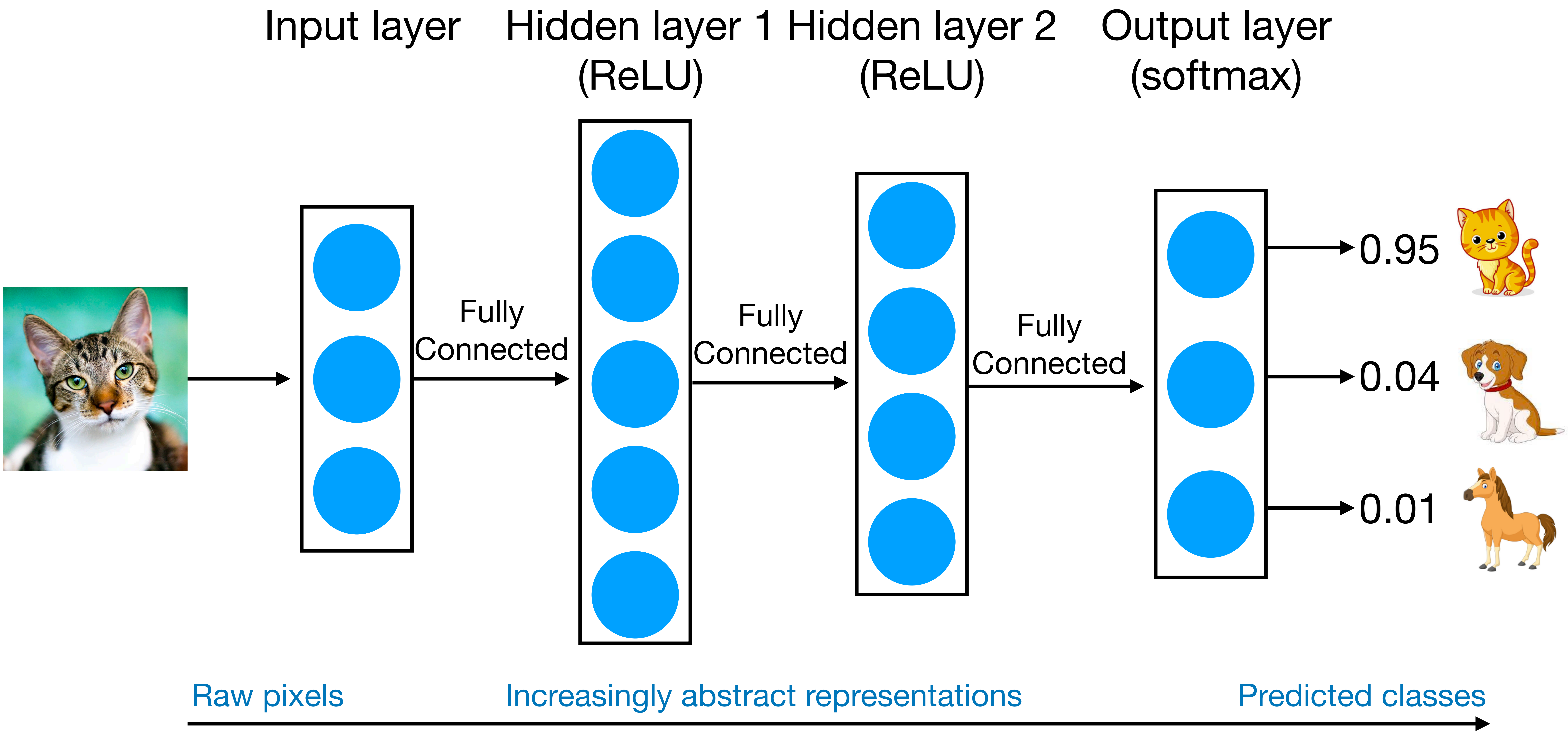
Fully Connected

0.95

0.04

0.01

Multi-layer fully connected neural networks



Input layer

Hidden layer 1 (ReLU)

Hidden layer 2 (ReLU)

Output layer (softmax)

Fully Connected

Fully Connected

Fully Connected

0.95

0.04

0.01

Raw pixels

Increasingly abstract representations

Predicted classes

Training deep learning models

Training deep learning models

Training a neural network amounts to solving our usual optimization problem:

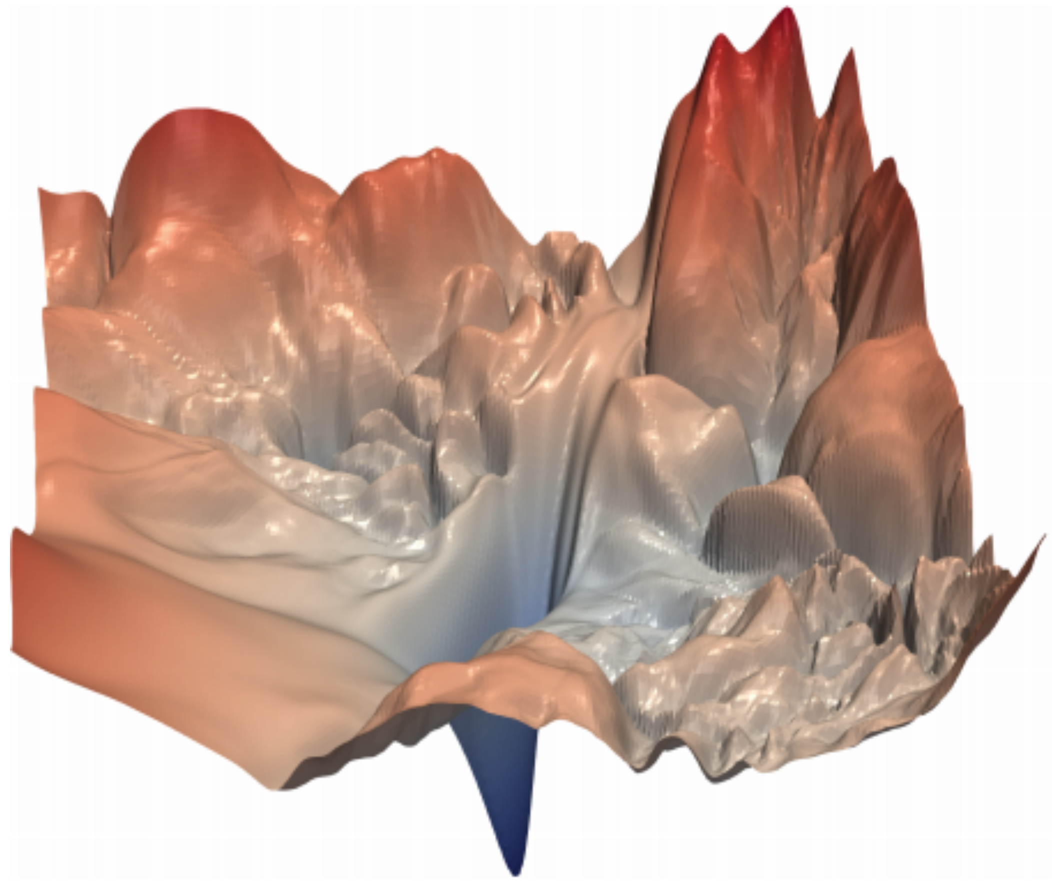
$$\widehat{\beta} = \arg\min_{\beta} \frac{1}{n}\sum_{i=1}^{n} L(Y_i, f_\beta(X_i)) + \lambda \cdot \text{penalty}(\beta) = \arg\min_{\beta} F(\beta)\,.$$

Training deep learning models

Training a neural network amounts to solving our usual optimization problem:

$$\widehat{\beta} = \arg\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} L(Y_i, f_\beta(X_i)) + \lambda \cdot \mathsf{penalty}(\beta) = \arg\min_{\beta} F(\beta) \,.$$

Unfortunately, neural networks have non-convex objective functions, causing issues discussed in the previous lecture.



https://arxiv.org/abs/1712.09913

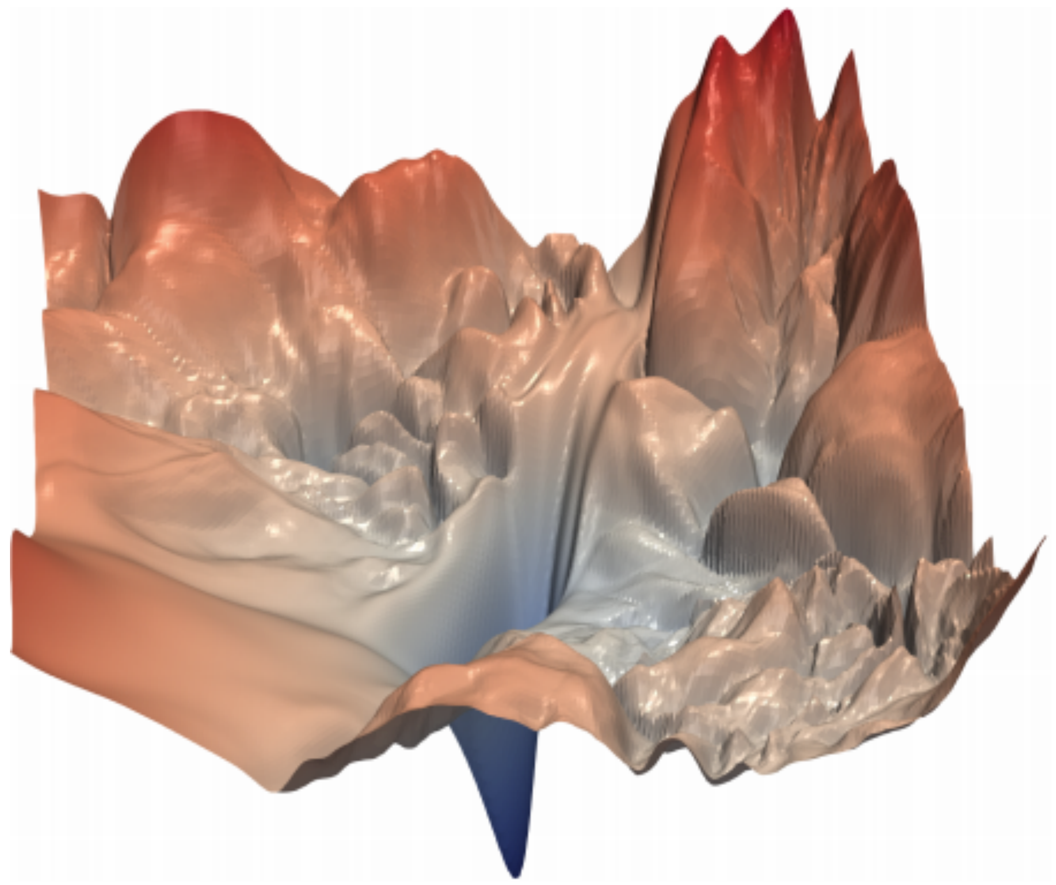
Training deep learning models

Training a neural network amounts to solving our usual optimization problem:

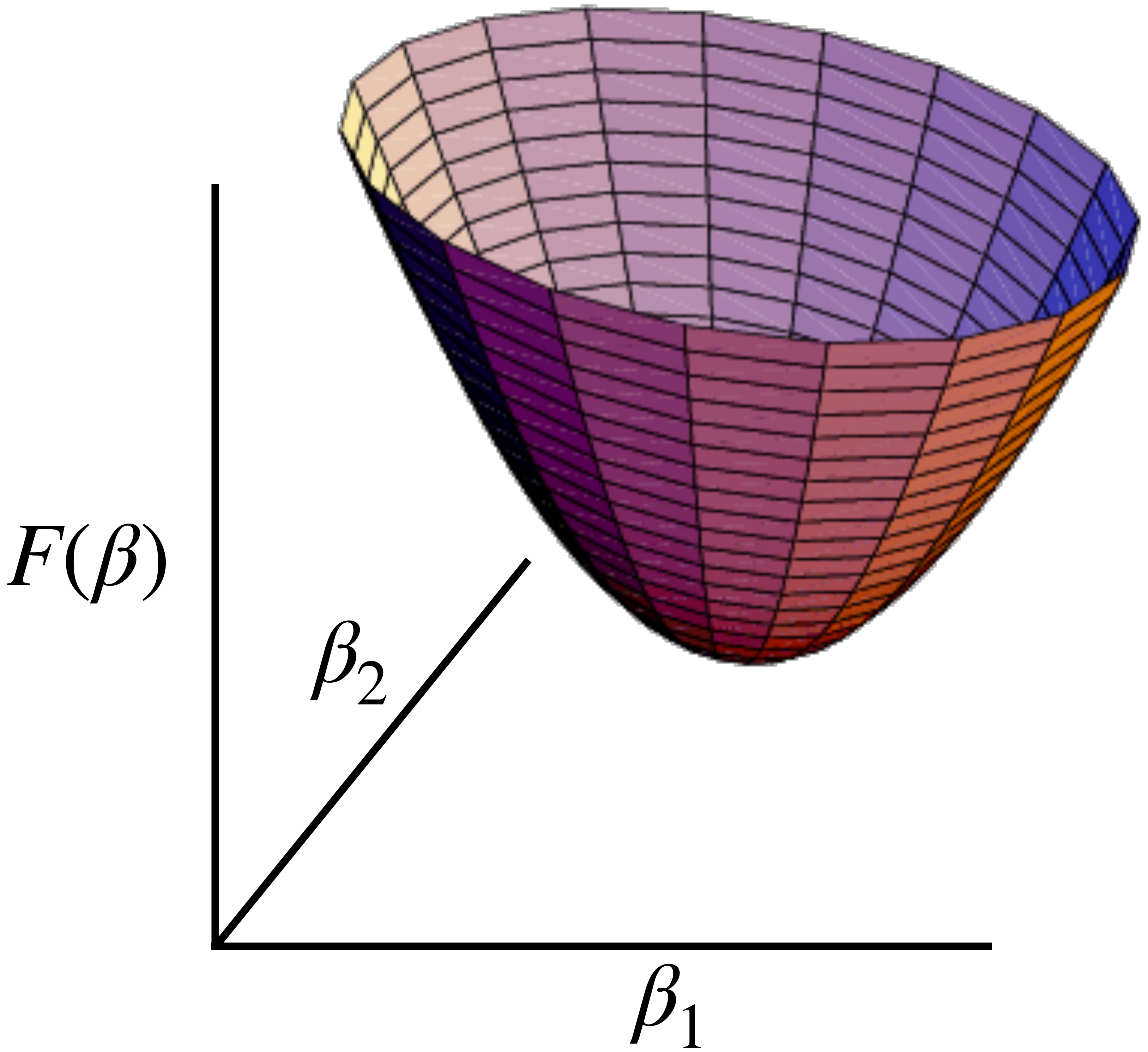
$$\widehat{\beta} = \arg\min_{\beta} \frac{1}{n}\sum_{i=1}^{n} L(Y_i, f_\beta(X_i)) + \lambda \cdot \text{penalty}(\beta) = \arg\min_{\beta} \; F(\beta)\,.$$

Unfortunately, neural networks have non-convex objective functions, causing issues discussed in the previous lecture.

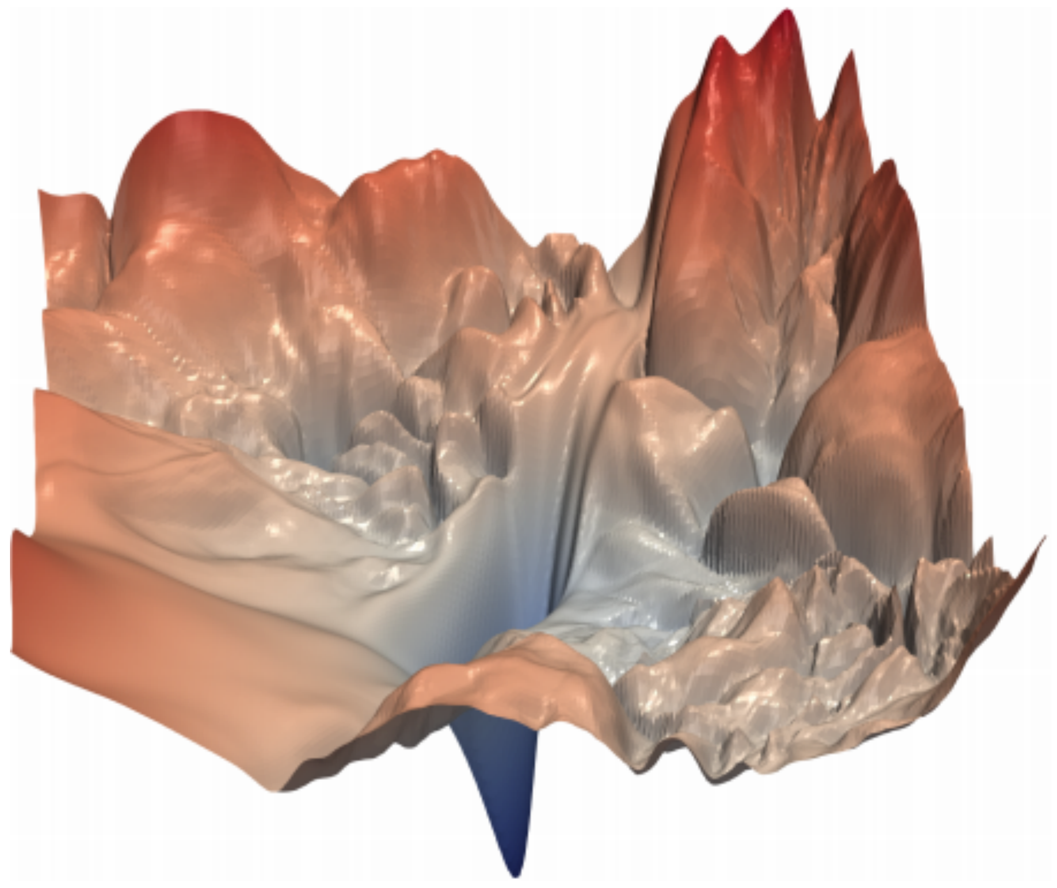
In practice, deep learning models are trained using stochastic gradient descent, which is a variant of gradient descent.



https://arxiv.org/abs/1712.09913



Training deep learning models



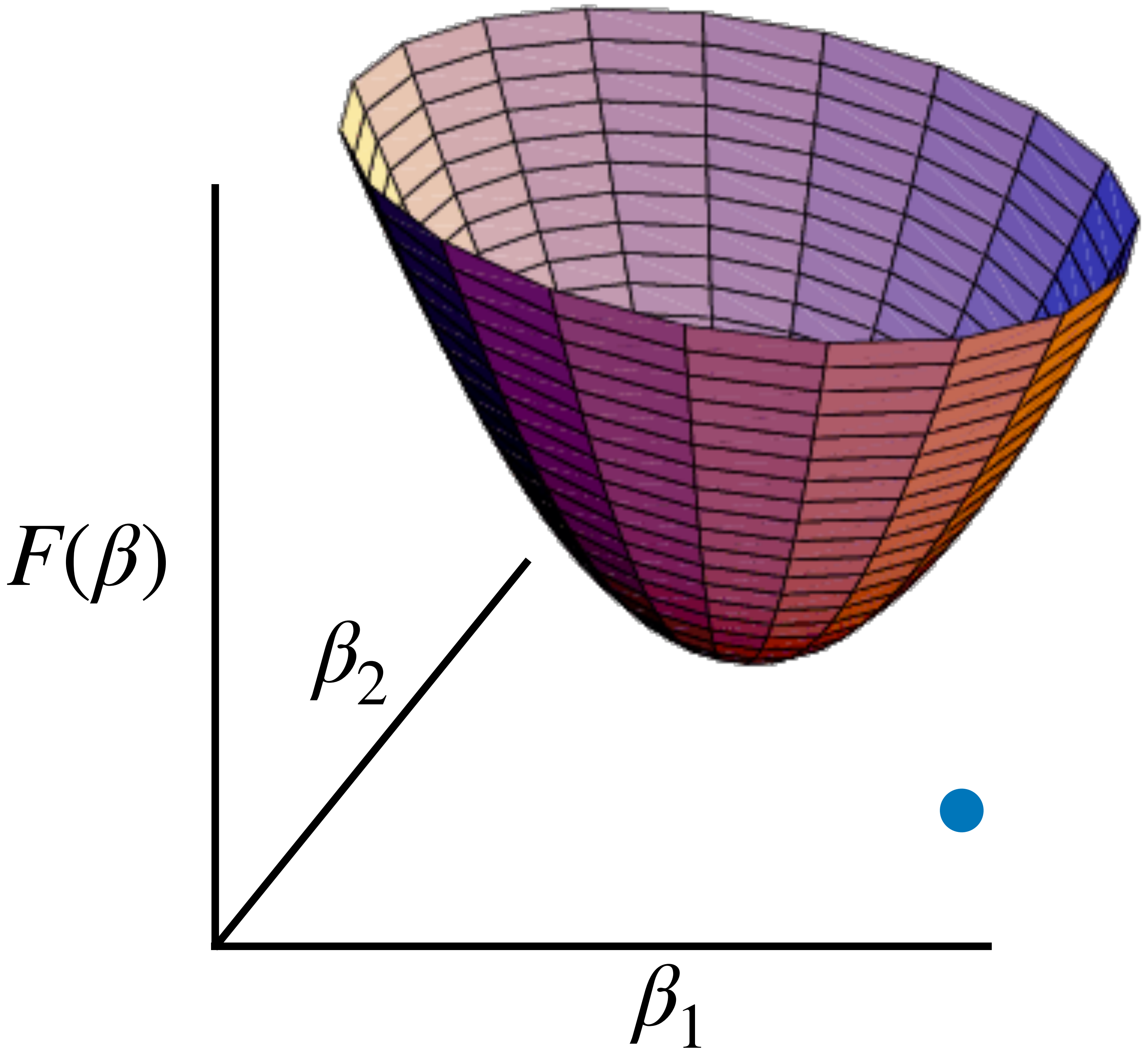
https://arxiv.org/abs/1712.09913

Training a neural network amounts to solving our usual optimization problem:

$$\widehat{\beta} = \arg\min_{\beta} \frac{1}{n}\sum_{i=1}^{n} L(Y_i, f_\beta(X_i)) + \lambda \cdot \text{penalty}(\beta) = \arg\min_{\beta}\, F(\beta)\,.$$

Unfortunately, neural networks have non-convex objective functions, causing issues discussed in the previous lecture.

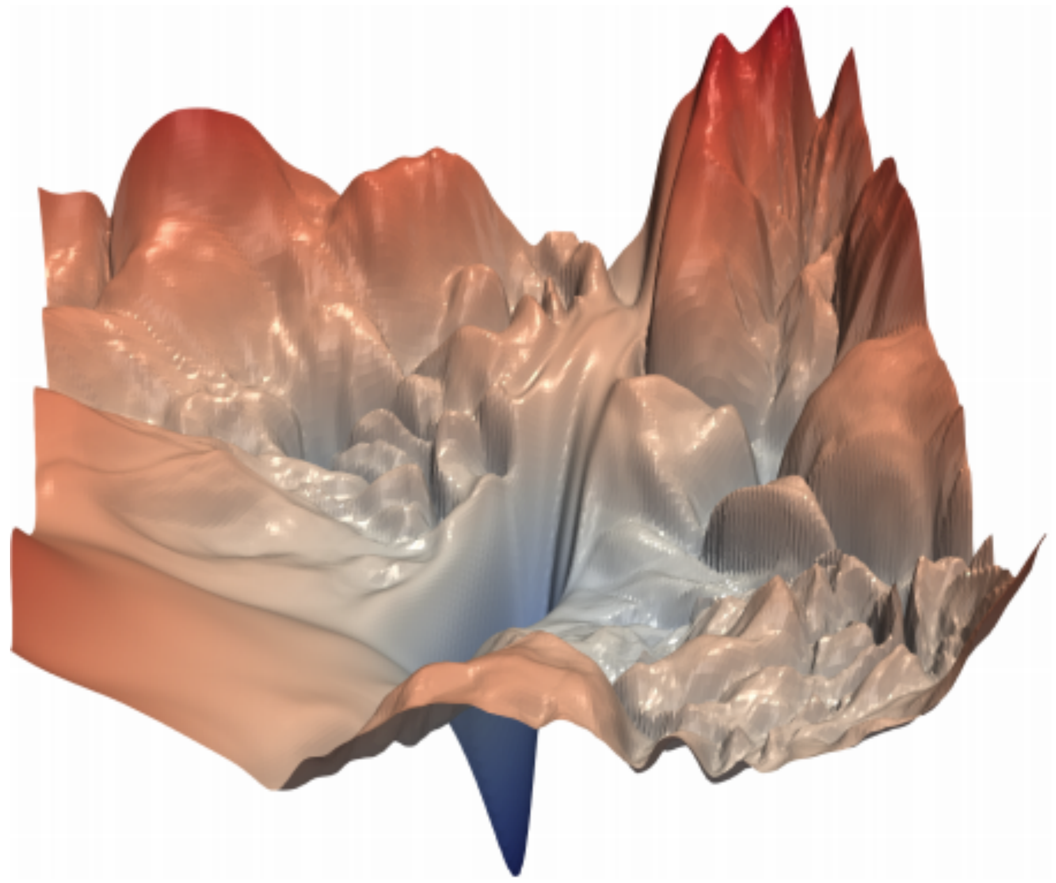
In practice, deep learning models are trained using stochastic gradient descent, which is a variant of gradient descent.



Training deep learning models

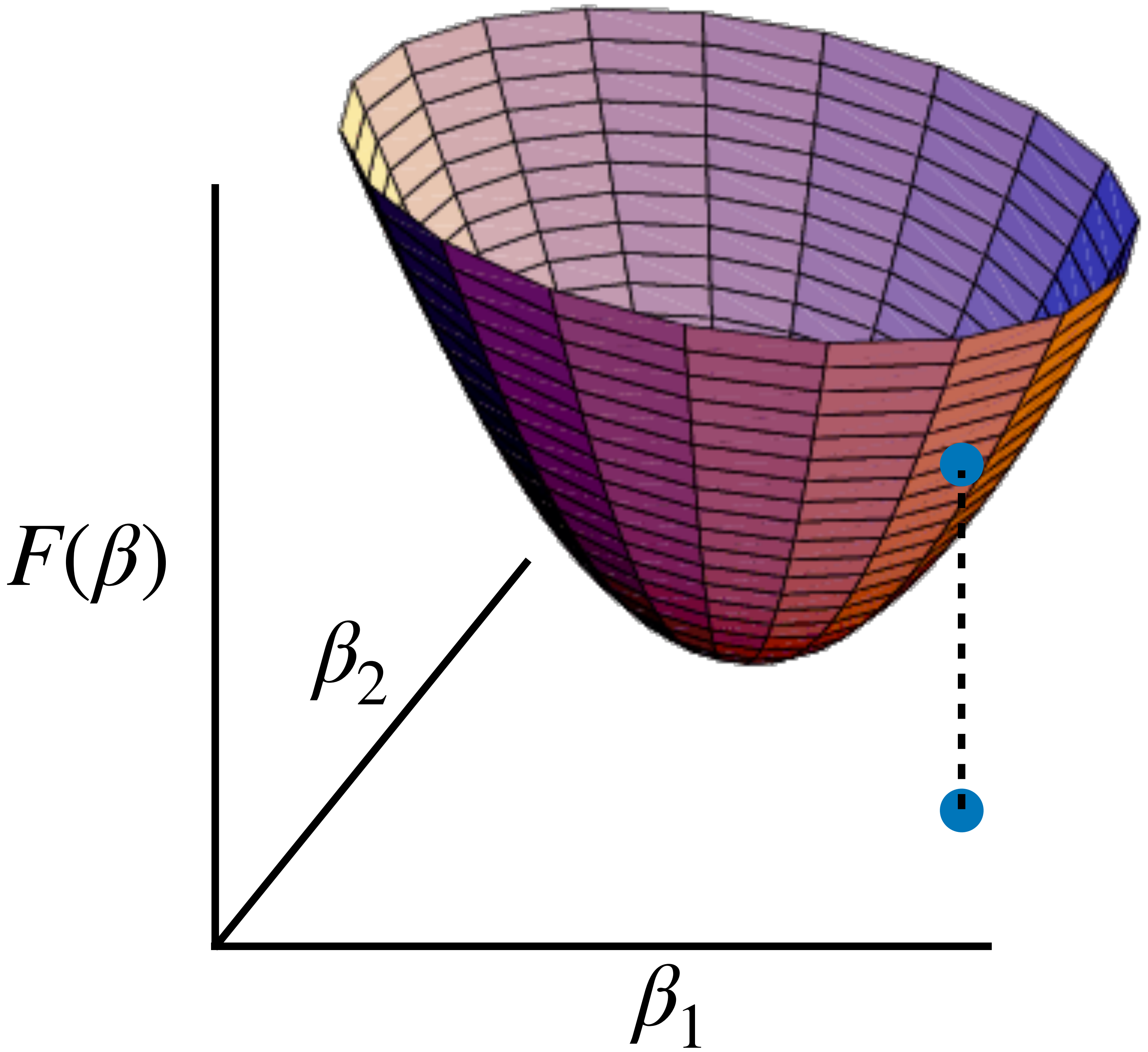
Training a neural network amounts to solving our usual optimization problem:

$$\widehat{\beta} = \arg\min_{\beta} \frac{1}{n}\sum_{i=1}^{n} L(Y_i, f_\beta(X_i)) + \lambda \cdot \text{penalty}(\beta) = \arg\min_{\beta} F(\beta)\,.$$



https://arxiv.org/abs/1712.09913

Unfortunately, neural networks have non-convex objective functions, causing issues discussed in the previous lecture.



In practice, deep learning models are trained using stochastic gradient descent, which is a variant of gradient descent.

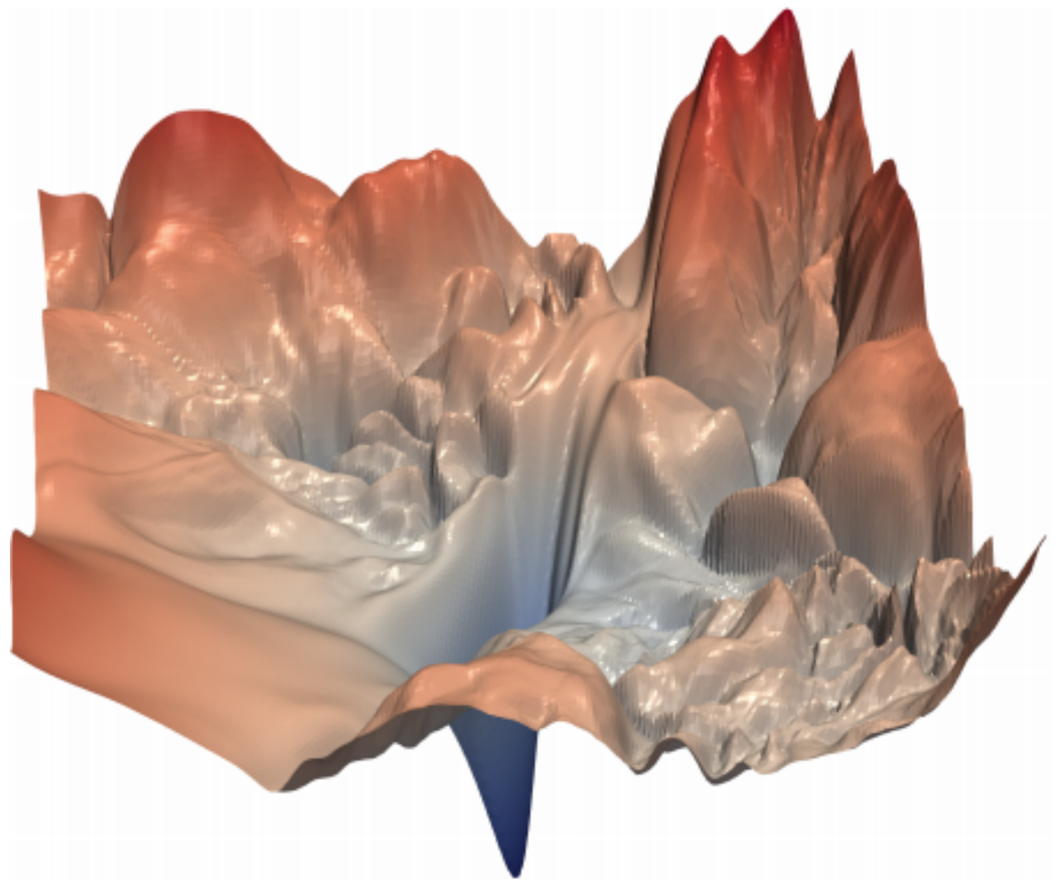
Training deep learning models

Training a neural network amounts to solving our usual optimization problem:

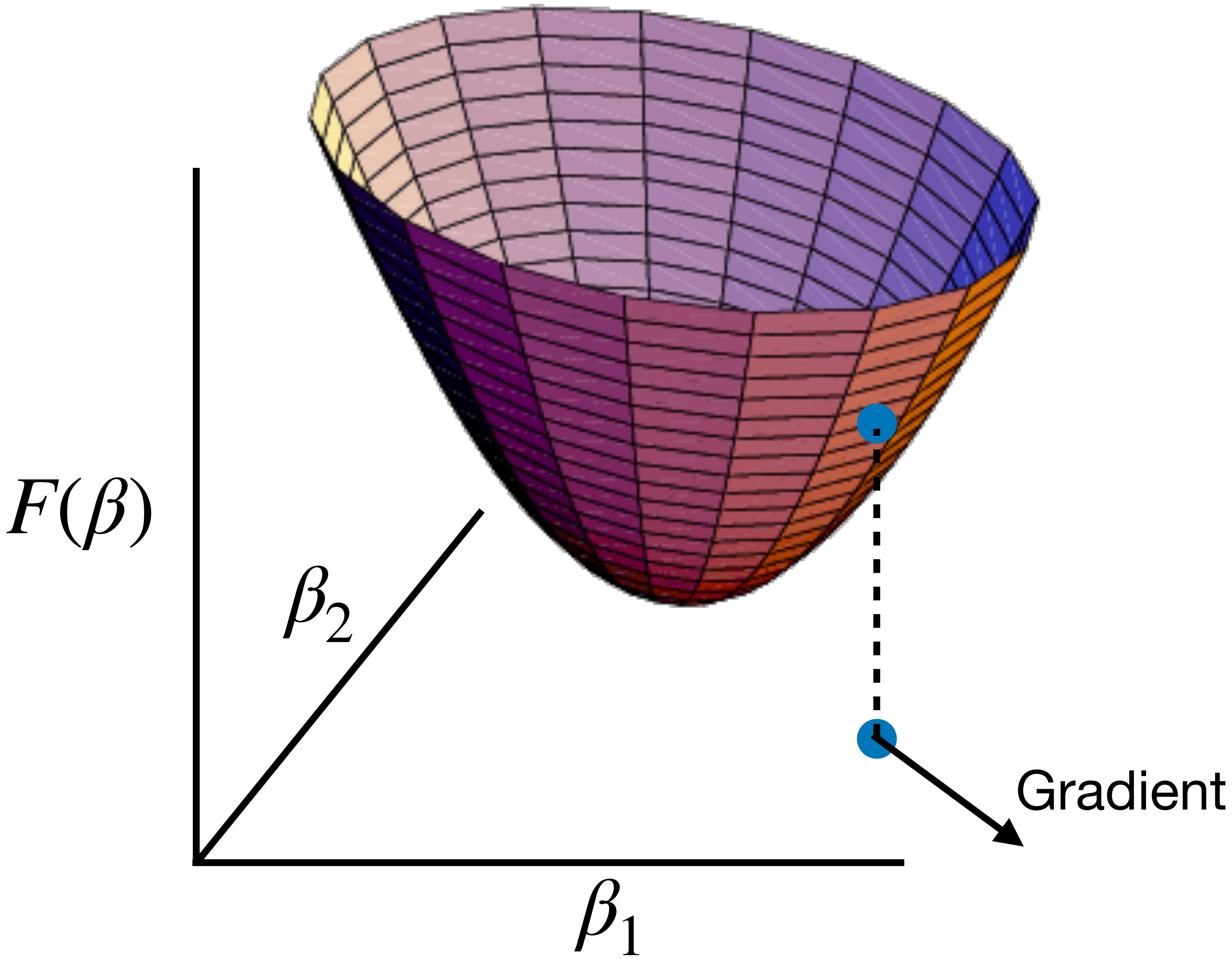
$$\widehat{\beta} = \arg\min_{\beta} \frac{1}{n}\sum_{i=1}^{n} L(Y_i, f_\beta(X_i)) + \lambda \cdot \text{penalty}(\beta) = \arg\min_{\beta} F(\beta)\,.$$

Unfortunately, neural networks have non-convex objective functions, causing issues discussed in the previous lecture.

In practice, deep learning models are trained using stochastic gradient descent, which is a variant of gradient descent.



https://arxiv.org/abs/1712.09913



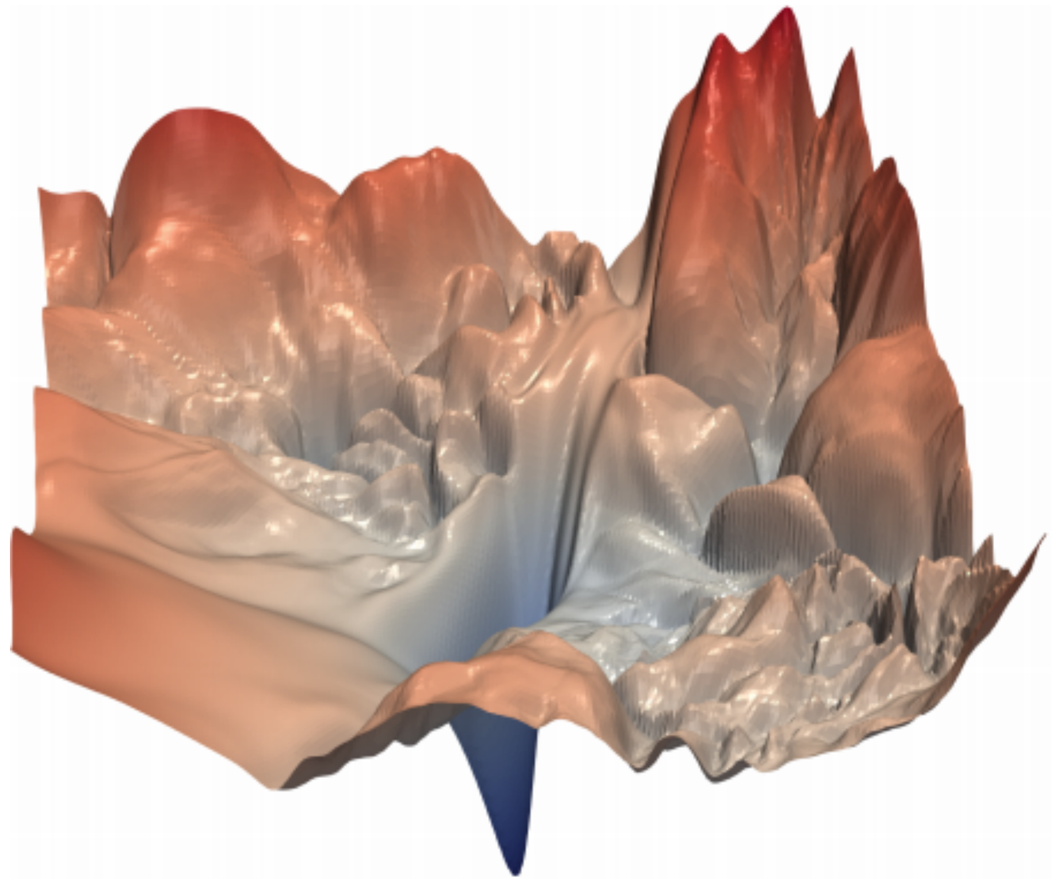
Training deep learning models

Training a neural network amounts to solving our usual optimization problem:

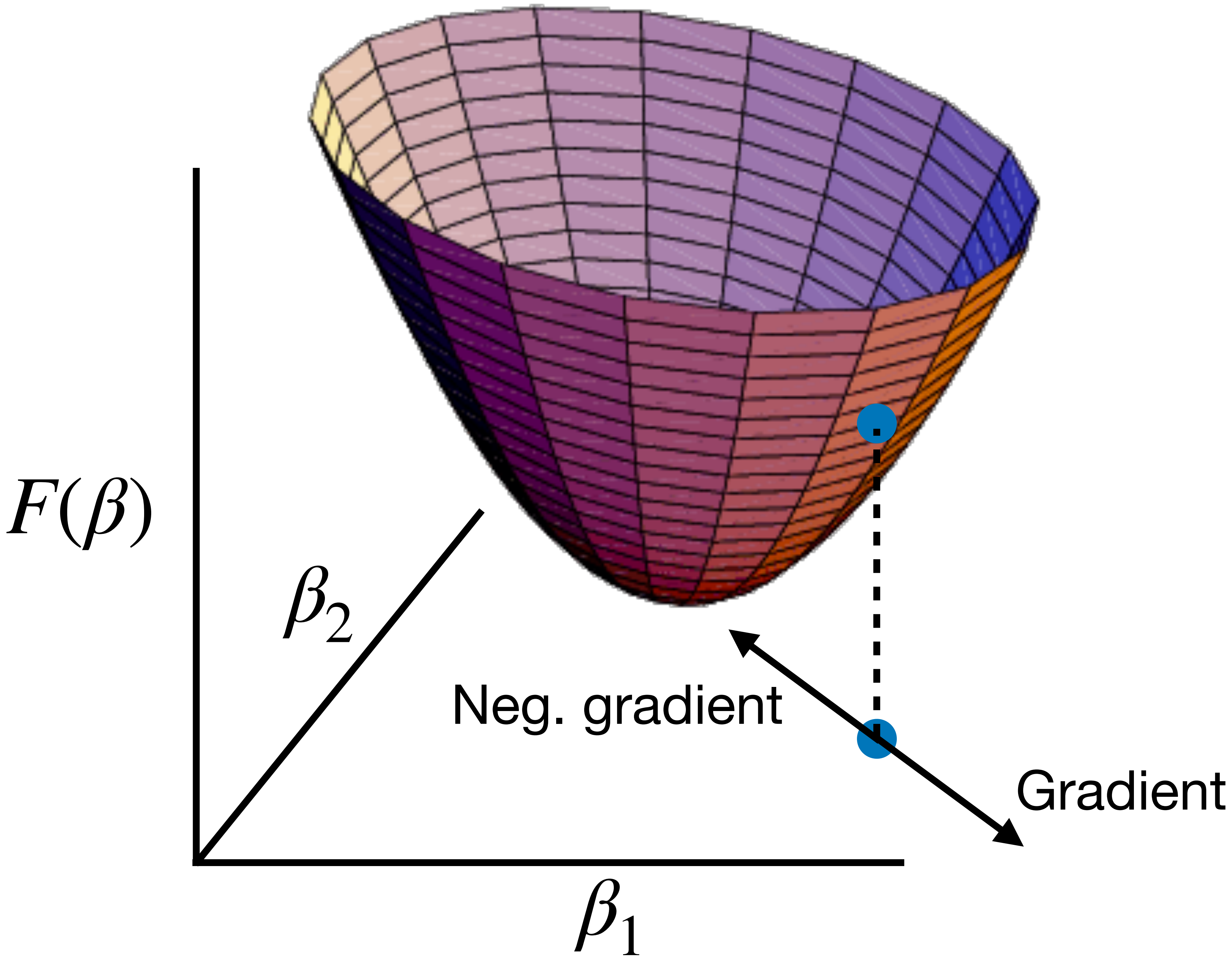
$$\widehat{\beta} = \arg\min_{\beta} \frac{1}{n}\sum_{i=1}^{n} L(Y_i, f_{\beta}(X_i)) + \lambda \cdot \text{penalty}(\beta) = \arg\min_{\beta} \; F(\beta)\,.$$

Unfortunately, neural networks have non-convex objective functions, causing issues discussed in the previous lecture.

In practice, deep learning models are trained using stochastic gradient descent, which is a variant of gradient descent.



https://arxiv.org/abs/1712.09913



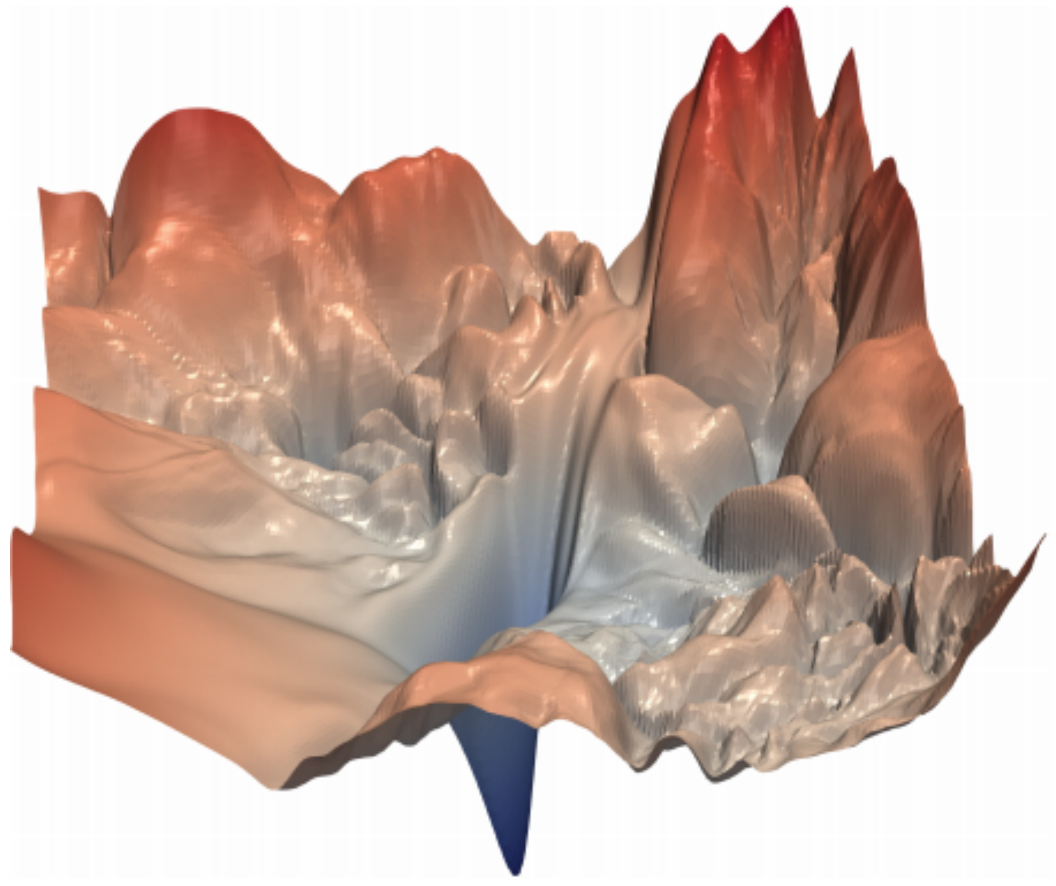
Training deep learning models

Training a neural network amounts to solving our usual optimization problem:

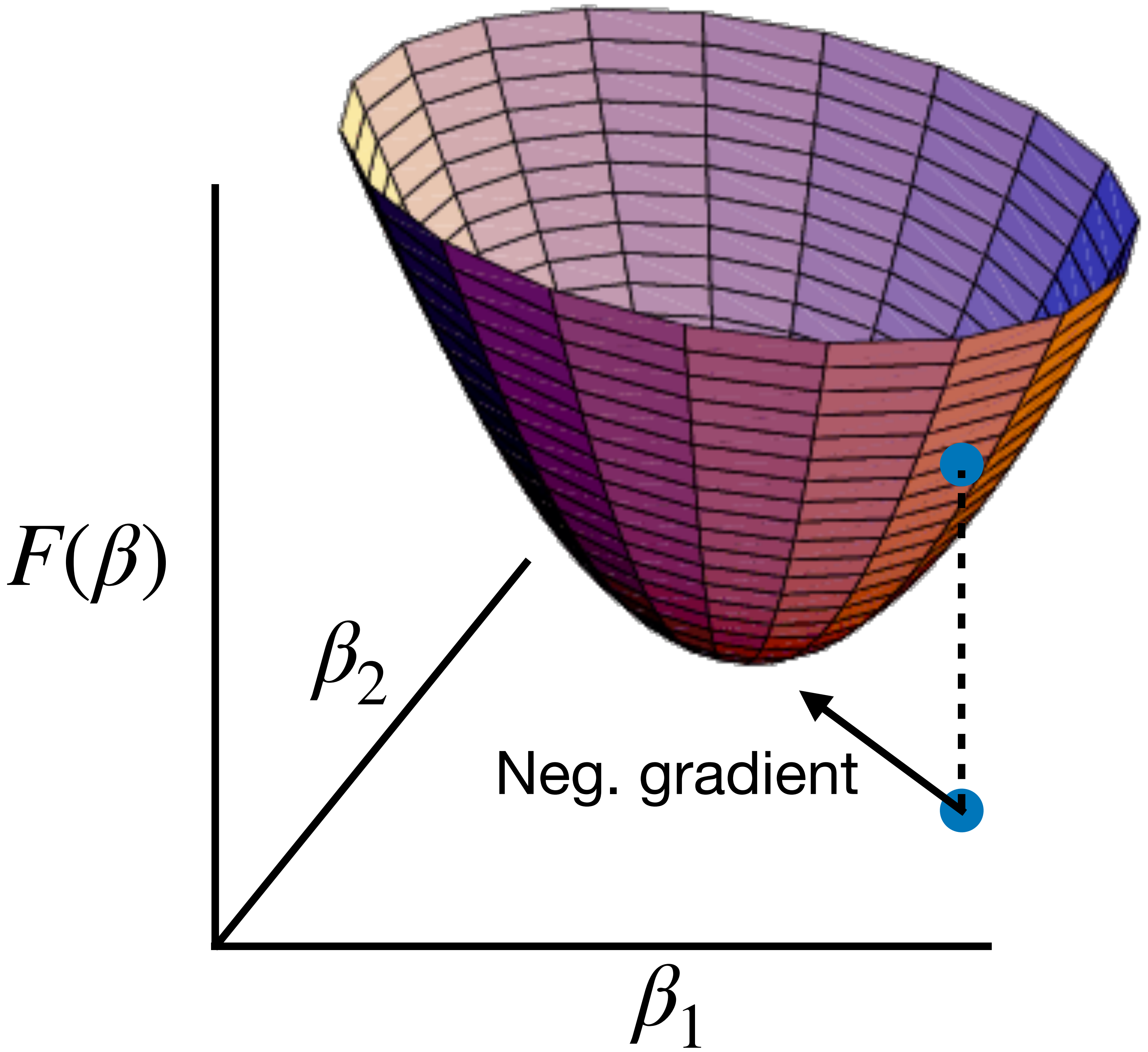
$$\widehat{\beta} = \arg\min_{\beta} \frac{1}{n}\sum_{i=1}^{n} L(Y_i, f_\beta(X_i)) + \lambda \cdot \text{penalty}(\beta) = \arg\min_{\beta} F(\beta)\,.$$

Unfortunately, neural networks have non-convex objective functions, causing issues discussed in the previous lecture.

In practice, deep learning models are trained using stochastic gradient descent, which is a variant of gradient descent.



https://arxiv.org/abs/1712.09913



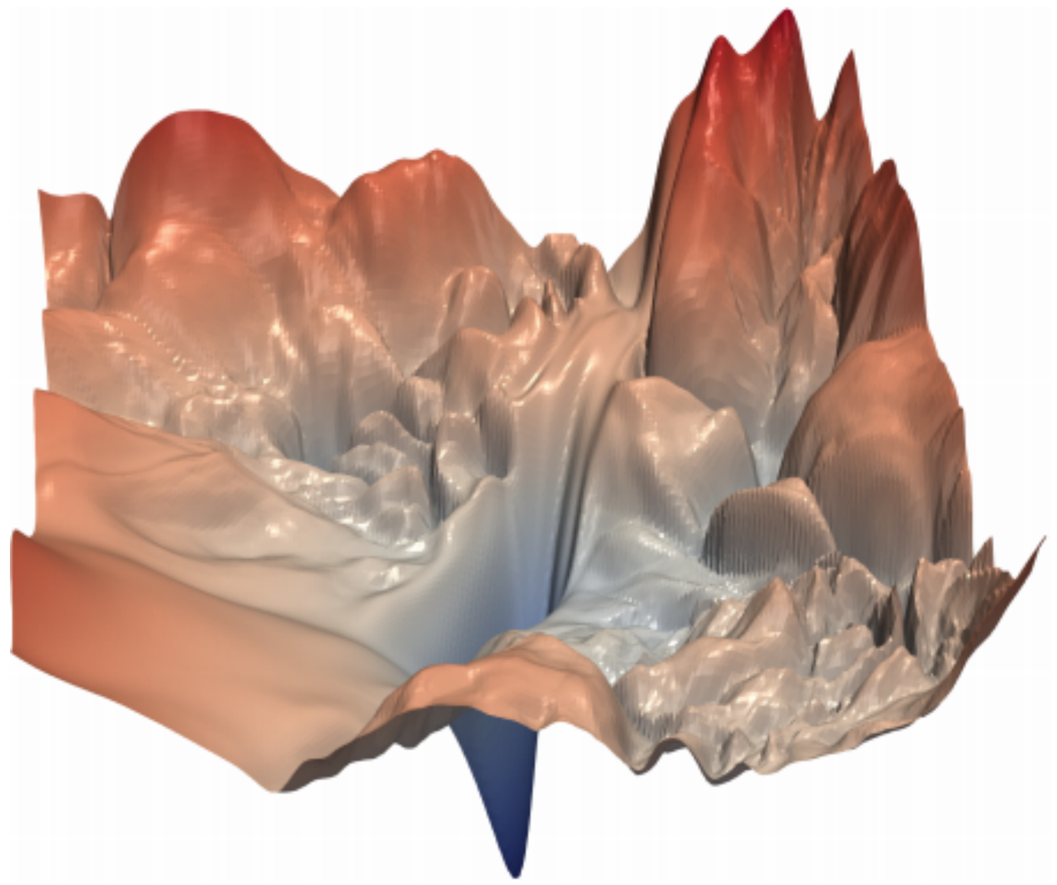
Training deep learning models

Training a neural network amounts to solving our usual optimization problem:

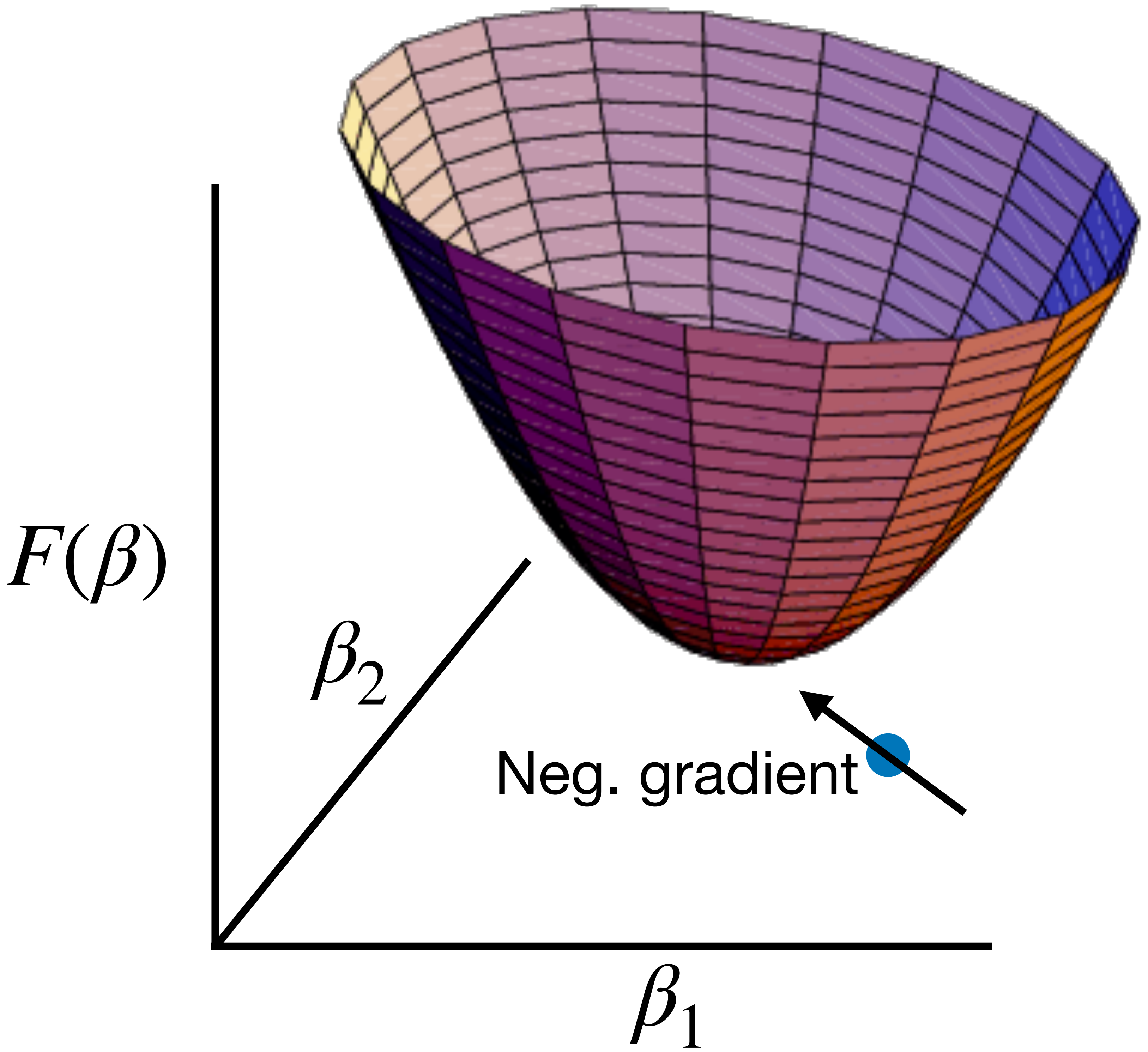
$$\widehat{\beta} = \arg\min_{\beta} \frac{1}{n}\sum_{i=1}^{n} L(Y_i, f_\beta(X_i)) + \lambda \cdot \text{penalty}(\beta) = \arg\min_{\beta} F(\beta)\,.$$

Unfortunately, neural networks have non-convex objective functions, causing issues discussed in the previous lecture.

In practice, deep learning models are trained using stochastic gradient descent, which is a variant of gradient descent.



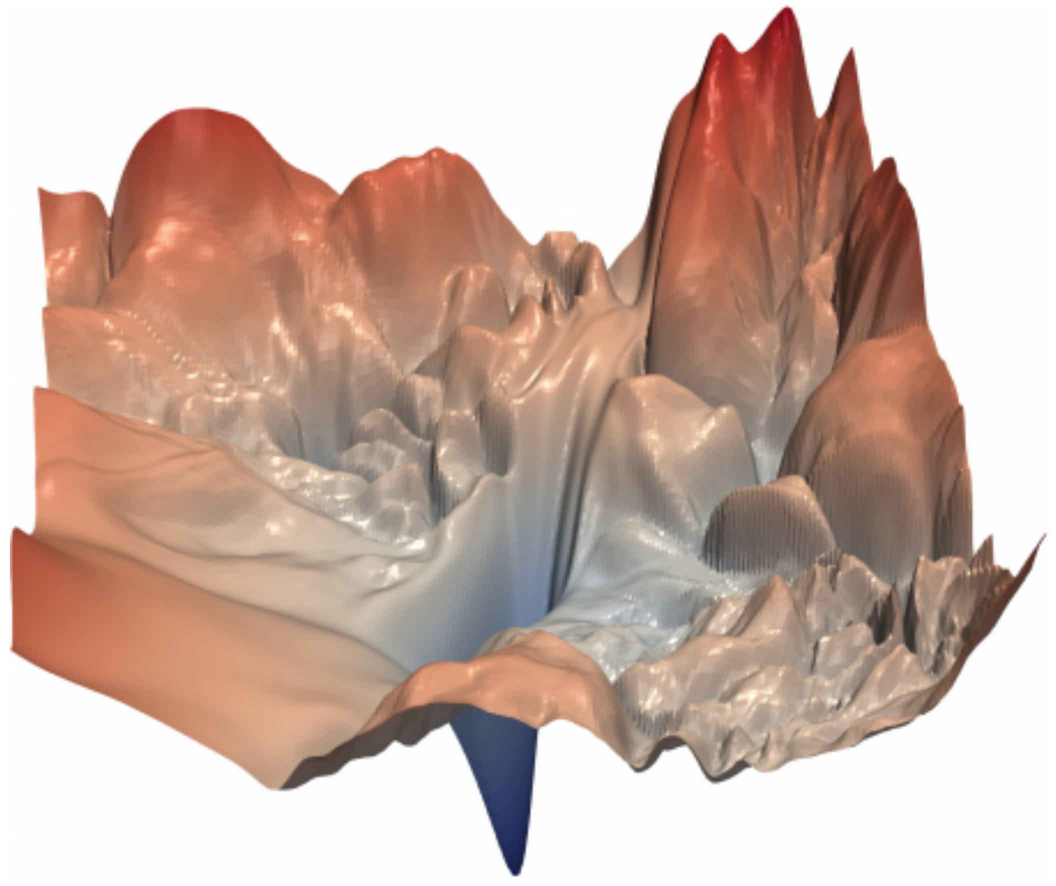
https://arxiv.org/abs/1712.09913



Training deep learning models

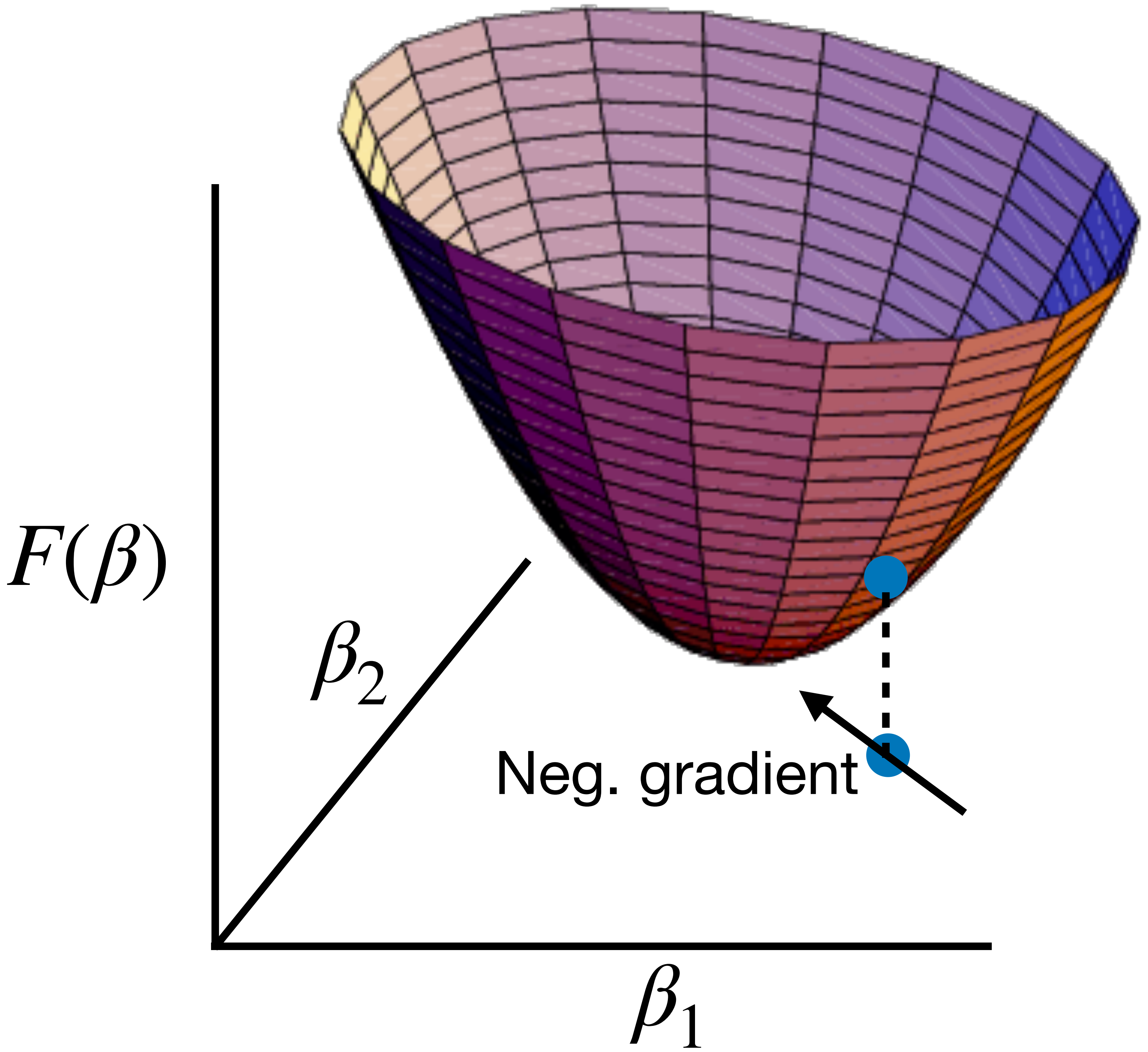
Training a neural network amounts to solving our usual optimization problem:

$$\widehat{\beta} = \arg\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} L(Y_i, f_\beta(X_i)) + \lambda \cdot \text{penalty}(\beta) = \arg\min_{\beta} F(\beta) \, .$$



https://arxiv.org/abs/1712.09913

Unfortunately, neural networks have non-convex objective functions, causing issues discussed in the previous lecture.

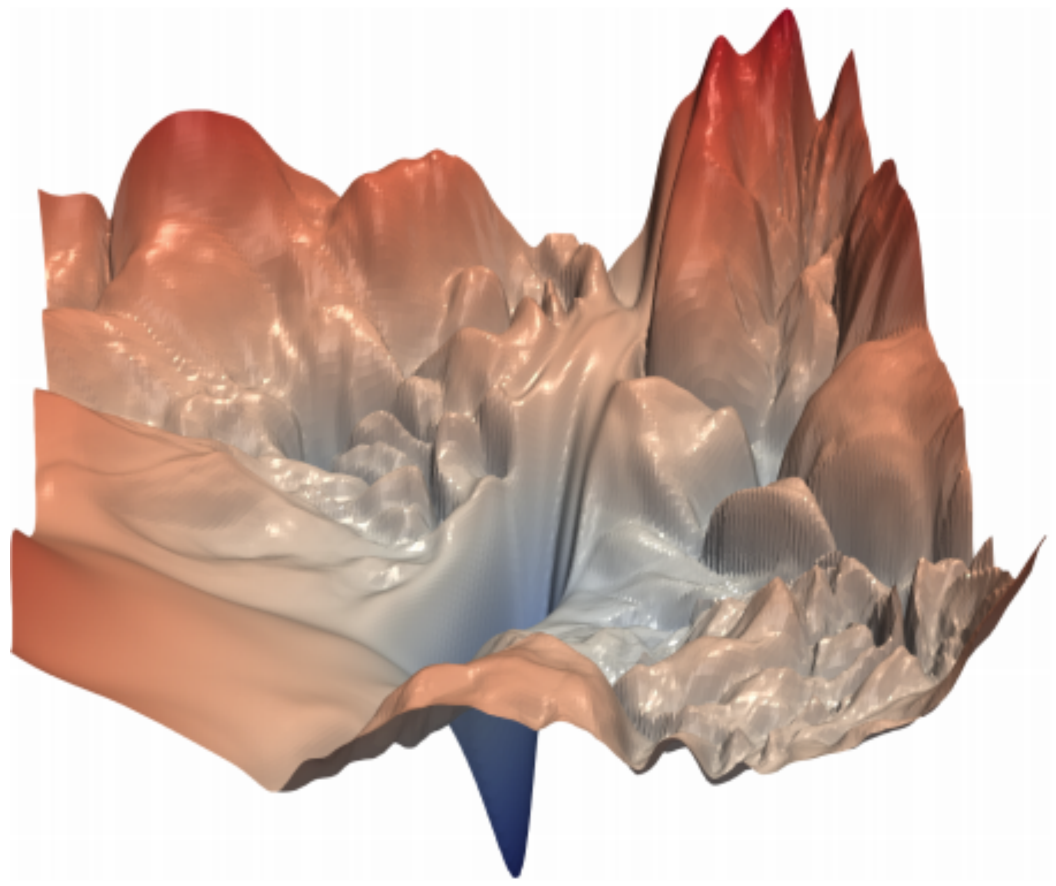


In practice, deep learning models are trained using stochastic gradient descent, which is a variant of gradient descent.

Training deep learning models

Training a neural network amounts to solving our usual optimization problem:

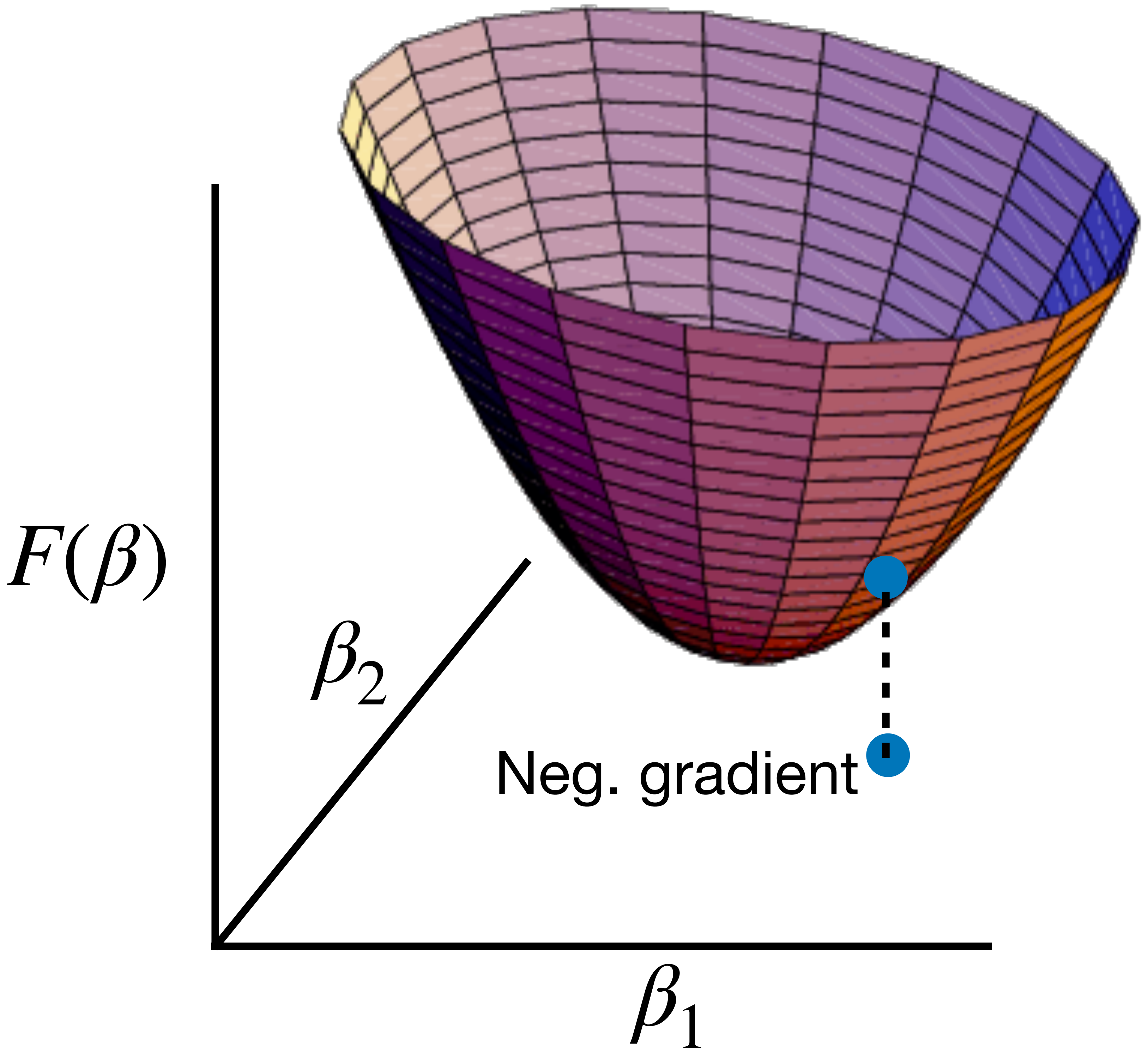
$$\widehat{\beta} = \arg\min_{\beta} \frac{1}{n}\sum_{i=1}^{n} L(Y_i, f_\beta(X_i)) + \lambda \cdot \text{penalty}(\beta) = \arg\min_{\beta} F(\beta)\,.$$



https://arxiv.org/abs/1712.09913

Unfortunately, neural networks have non-convex objective functions, causing issues discussed in the previous lecture.

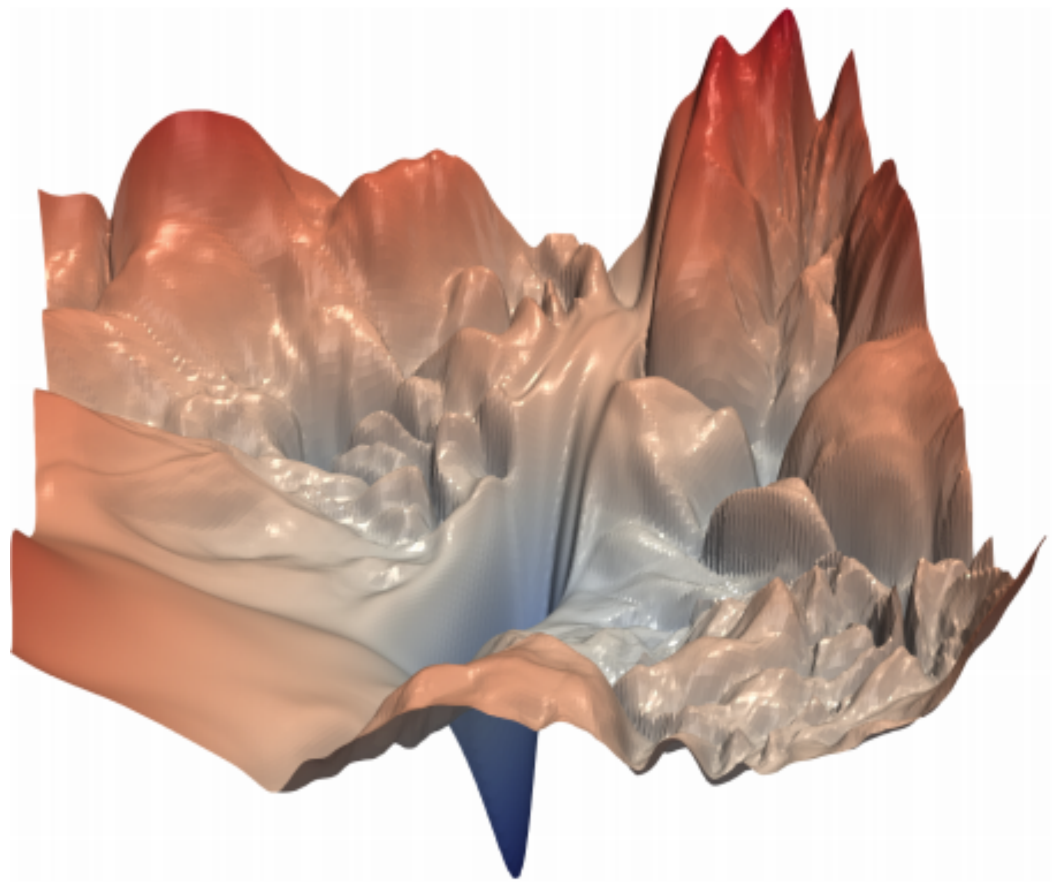
In practice, deep learning models are trained using stochastic gradient descent, which is a variant of gradient descent.



Training deep learning models

Training a neural network amounts to solving our usual optimization problem:

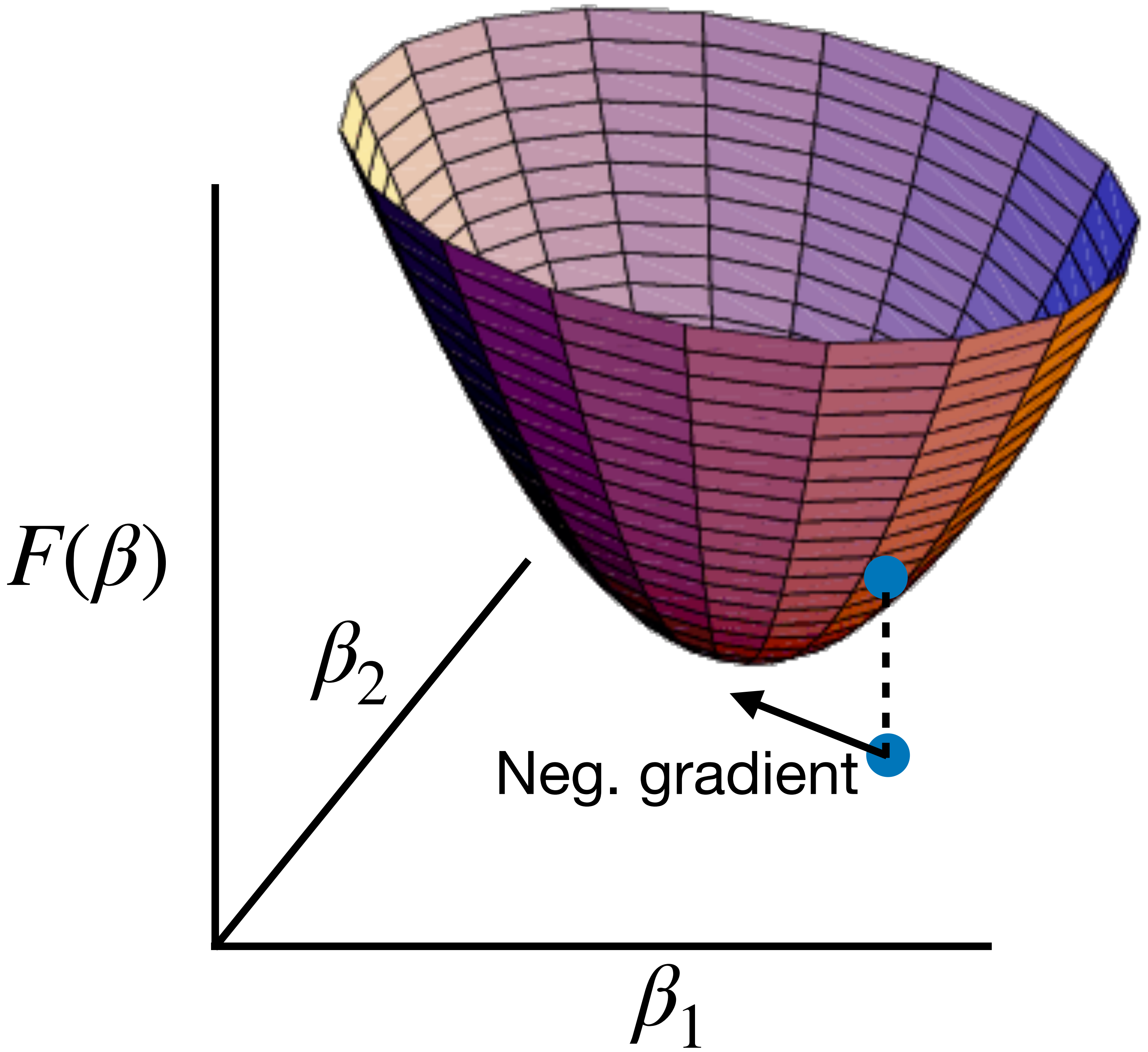
$$\widehat{\beta} = \arg\min_{\beta} \frac{1}{n}\sum_{i=1}^{n} L(Y_i, f_\beta(X_i)) + \lambda \cdot \text{penalty}(\beta) = \arg\min_{\beta} F(\beta)\,.$$



https://arxiv.org/abs/1712.09913

Unfortunately, neural networks have non-convex objective functions, causing issues discussed in the previous lecture.

In practice, deep learning models are trained using stochastic gradient descent, which is a variant of gradient descent.



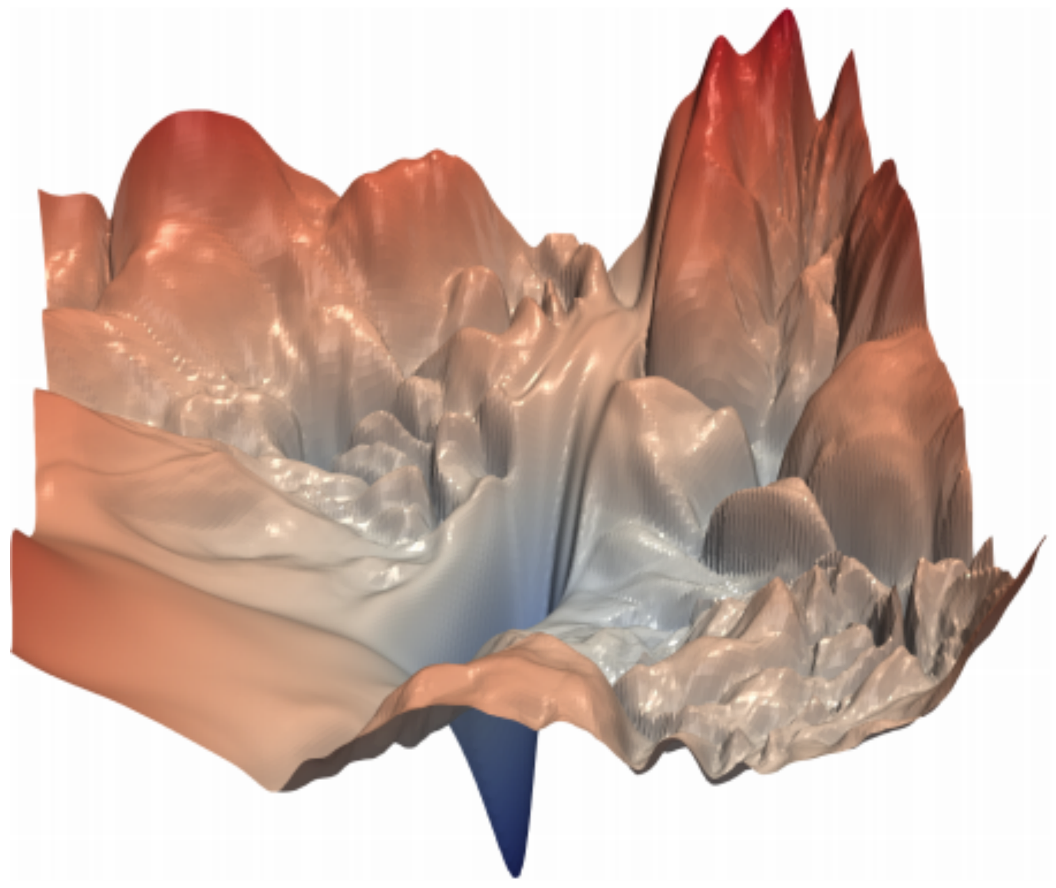
Training deep learning models

Training a neural network amounts to solving our usual optimization problem:

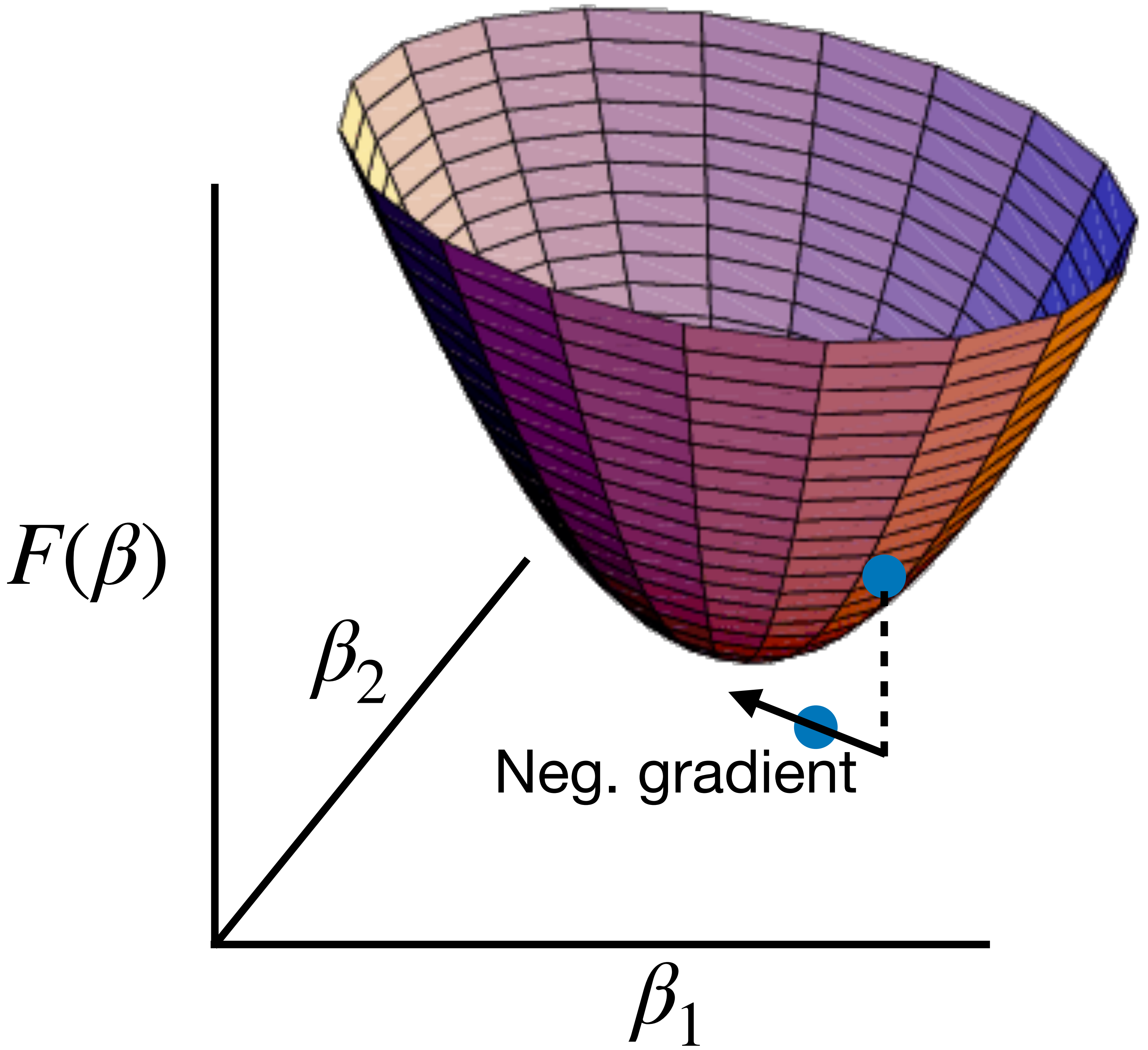
$$\widehat{\beta} = \arg\min_{\beta} \frac{1}{n}\sum_{i=1}^{n} L(Y_i, f_\beta(X_i)) + \lambda \cdot \text{penalty}(\beta) = \arg\min_{\beta} F(\beta)\,.$$

Unfortunately, neural networks have non-convex objective functions, causing issues discussed in the previous lecture.

In practice, deep learning models are trained using stochastic gradient descent, which is a variant of gradient descent.



https://arxiv.org/abs/1712.09913



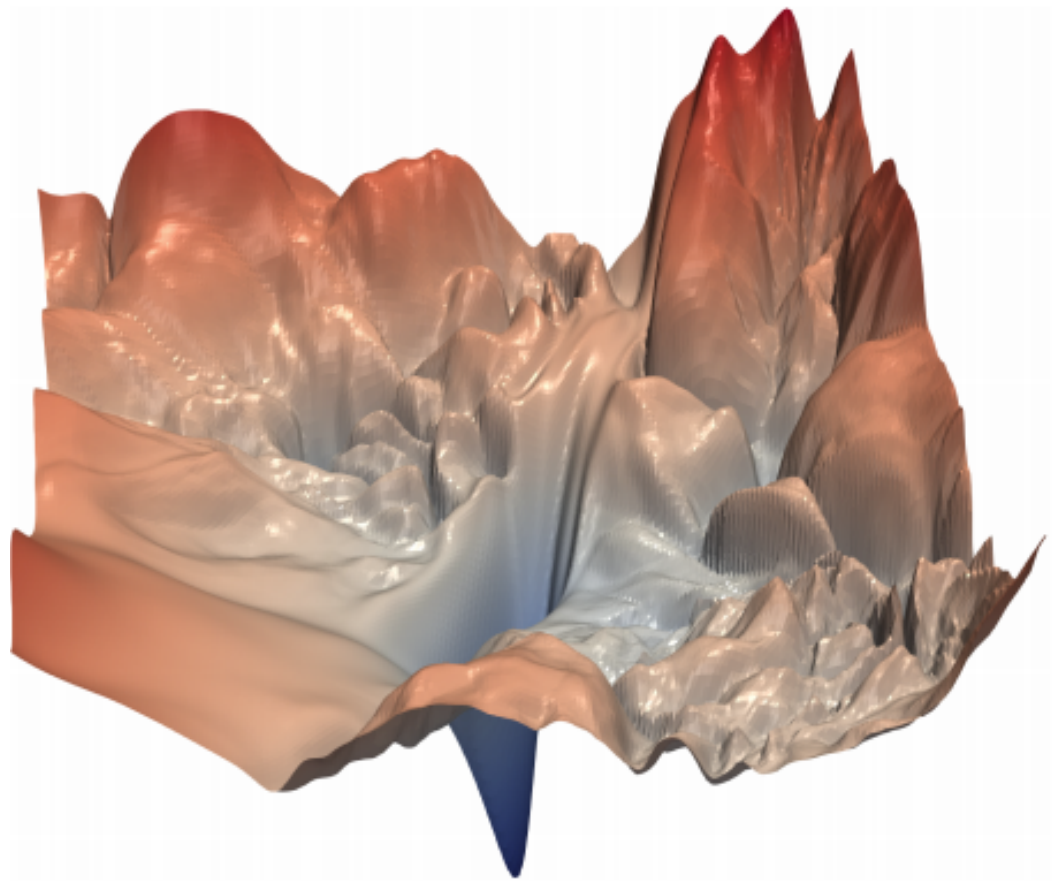
Training deep learning models

Training a neural network amounts to solving our usual optimization problem:

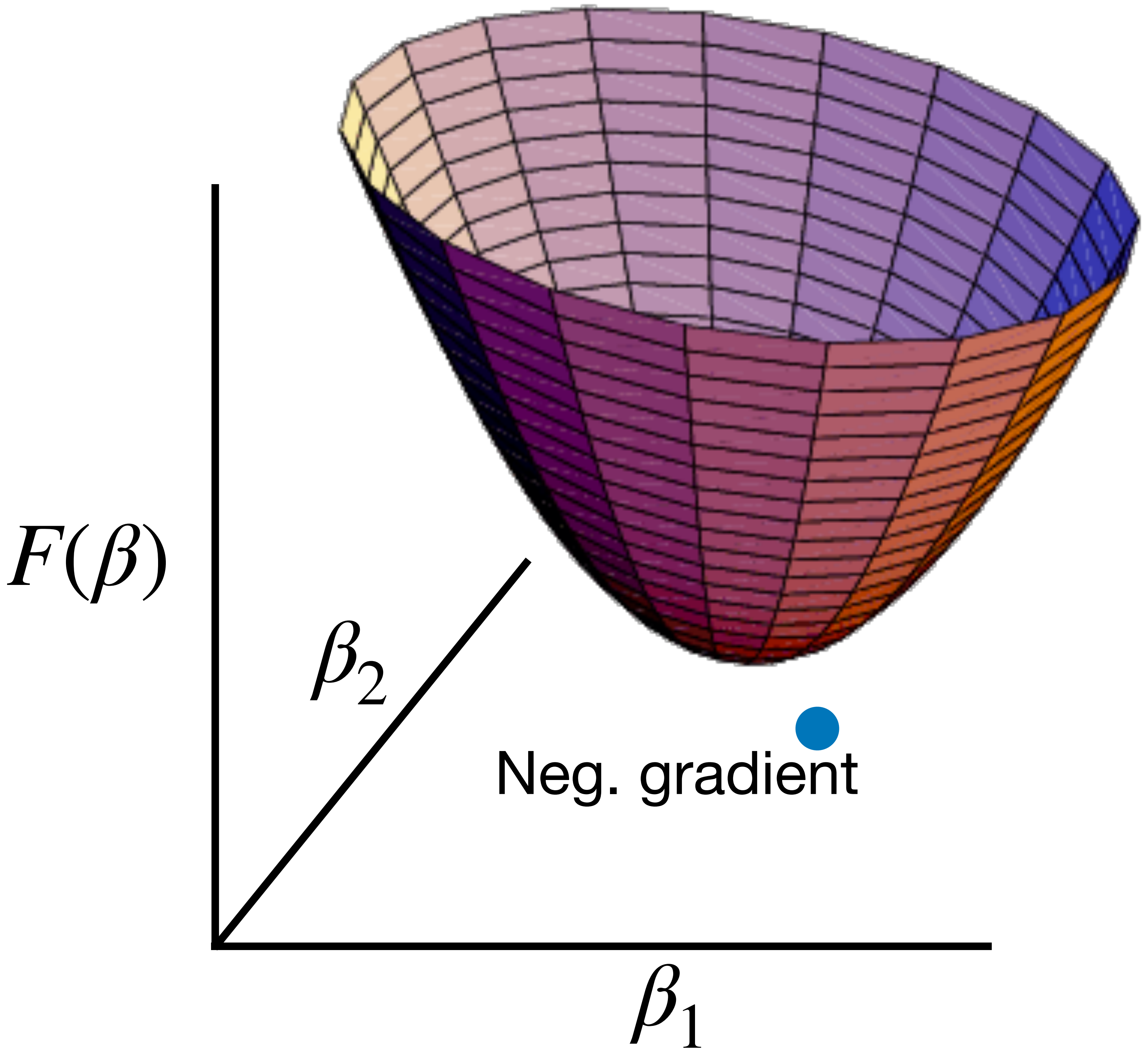
$$\widehat{\beta} = \arg\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} L(Y_i, f_\beta(X_i)) + \lambda \cdot \text{penalty}(\beta) = \arg\min_{\beta} F(\beta)\,.$$

Unfortunately, neural networks have non-convex objective functions, causing issues discussed in the previous lecture.

In practice, deep learning models are trained using stochastic gradient descent, which is a variant of gradient descent.



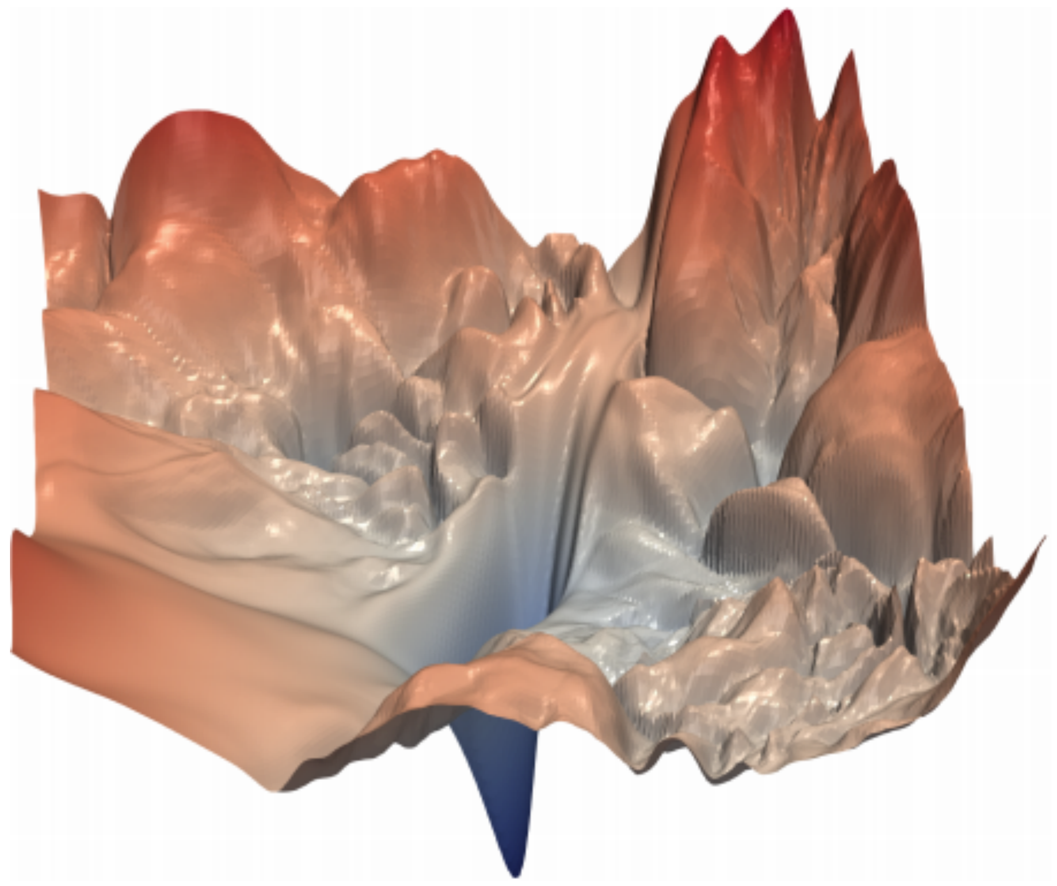
https://arxiv.org/abs/1712.09913



Training deep learning models

Training a neural network amounts to solving our usual optimization problem:

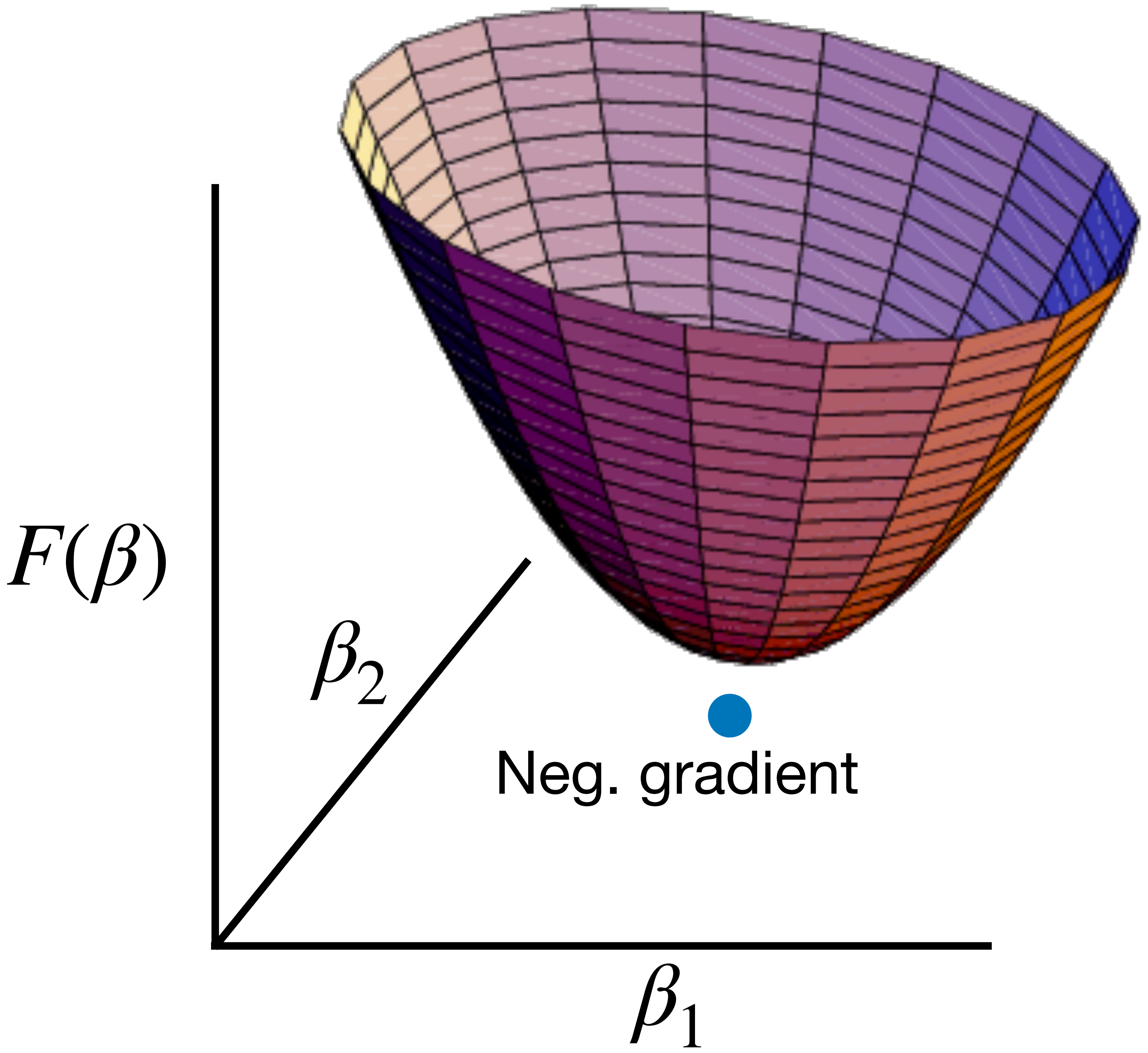
$$\widehat{\beta} = \arg\min_{\beta} \frac{1}{n}\sum_{i=1}^{n} L(Y_i, f_\beta(X_i)) + \lambda \cdot \text{penalty}(\beta) = \arg\min_{\beta}\ F(\beta)\,.$$



https://arxiv.org/abs/1712.09913

Unfortunately, neural networks have non-convex objective functions, causing issues discussed in the previous lecture.

In practice, deep learning models are trained using stochastic gradient descent, which is a variant of gradient descent.



The computational cost of gradient descent

The computational cost of gradient descent

$$\text{computational cost} = (\text{cost of computing the gradient}) \times (\text{number of iterations})$$

The computational cost of gradient descent

$$\text{computational cost} = (\text{cost of computing the gradient}) \times (\text{number of iterations})$$

Number of iterations has to do with the size of the learning rate (like boosting) and the shape of the objective function F.

The computational cost of gradient descent

computational cost $=$ (cost of computing the gradient) $\times$ (number of iterations)

Number of iterations has to do with the size of the learning rate (like boosting) and the shape of the objective function F.

Cost of computing the gradient has to do with the size of the training data:

The computational cost of gradient descent

computational cost = (cost of computing the gradient) × (number of iterations)

Number of iterations has to do with the size of the learning rate (like boosting) and the shape of the objective function F.

Cost of computing the gradient has to do with the size of the training data:

If $F(\beta) = \frac{1}{n}\sum_{i=1}^{n} L(Y_i, f_\beta(X_i))$, then $\nabla F(\beta) = \frac{1}{n}\sum_{i=1}^{n} \nabla L(Y_i, f_\beta(X_i))$.

The computational cost of gradient descent

computational cost $=$ (cost of computing the gradient) $\times$ (number of iterations)

Number of iterations has to do with the size of the learning rate (like boosting) and the shape of the objective function F.

Cost of computing the gradient has to do with the size of the training data:

If $F(\beta) = \frac{1}{n}\sum_{i=1}^{n} L(Y_i, f_\beta(X_i))$, then $\nabla F(\beta) = \frac{1}{n}\sum_{i=1}^{n} \nabla L(Y_i, f_\beta(X_i))$.

Each evaluation of the gradient requires a pass through the entire training data, and modern data sets can have millions of training observations.

The computational cost of gradient descent

computational cost = (cost of computing the gradient) × (number of iterations)

Number of iterations has to do with the size of the learning rate (like boosting) and the shape of the objective function F.

Cost of computing the gradient has to do with the size of the training data:

$$\text{If } F(\beta) = \frac{1}{n}\sum_{i=1}^{n} L(Y_i, f_\beta(X_i)), \text{ then } \nabla F(\beta) = \frac{1}{n}\sum_{i=1}^{n} \nabla L(Y_i, f_\beta(X_i)).$$

Each evaluation of the gradient requires a pass through the entire training data, and modern data sets can have millions of training observations.

This can make gradient descent prohibitively expensive.

Stochastic gradient descent

Stochastic gradient descent

Use subset $S \subseteq \{1, \ldots, n\}$ (mini-batch) of observations to approximate gradient:

$$\nabla F(\beta) = \frac{1}{n} \sum_{i=1}^{n} \nabla L(Y_i, f_\beta(X_i)) \approx \frac{1}{|S|} \sum_{i \in S} \nabla L(Y_i, f_\beta(X_i)).$$

Stochastic gradient descent

Use subset $S \subseteq \{1,\ldots,n\}$ (mini-batch) of observations to approximate gradient:

$$\nabla F(\beta) = \frac{1}{n}\sum_{i=1}^{n} \nabla L(Y_i, f_\beta(X_i)) \approx \frac{1}{|S|}\sum_{i \in S} \nabla L(Y_i, f_\beta(X_i)).$$

1. Choose some initial value of β.

Stochastic gradient descent

Use subset $S \subseteq \{1, \ldots, n\}$ (mini-batch) of observations to approximate gradient:

$$\nabla F(\beta) = \frac{1}{n} \sum_{i=1}^{n} \nabla L(Y_i, f_\beta(X_i)) \approx \frac{1}{|S|} \sum_{i \in S} \nabla L(Y_i, f_\beta(X_i)).$$

1. Choose some initial value of β.

2. Randomly assign observations to mini-batches $S_1, \ldots, S_M$ of a certain size

Stochastic gradient descent

Use subset $S \subseteq \{1,\ldots,n\}$ (mini-batch) of observations to approximate gradient:

$$\nabla F(\beta) = \frac{1}{n}\sum_{i=1}^{n} \nabla L(Y_i, f_\beta(X_i)) \approx \frac{1}{|S|}\sum_{i \in S} \nabla L(Y_i, f_\beta(X_i)).$$

1. Choose some initial value of β.

2. Randomly assign observations to mini-batches $S_1, \ldots, S_M$ of a certain size

3. Repeat until convergence:

Stochastic gradient descent

Use subset $S \subseteq \{1,\ldots,n\}$ (mini-batch) of observations to approximate gradient:

$$\nabla F(\beta) = \frac{1}{n}\sum_{i=1}^{n} \nabla L(Y_i, f_\beta(X_i)) \approx \frac{1}{|S|}\sum_{i\in S} \nabla L(Y_i, f_\beta(X_i)).$$

1. Choose some initial value of β.

2. Randomly assign observations to mini-batches $S_1, \ldots, S_M$ of a certain size

3. Repeat until convergence:

    - For $m = 1,\ldots,M$, update $\beta \leftarrow \beta - \gamma \cdot \frac{1}{|S_m|}\sum_{i\in S_m} \nabla L(Y_i, f_\beta(X_i))$.

Stochastic gradient descent

Use subset $S \subseteq \{1,\ldots,n\}$ (mini-batch) of observations to approximate gradient:

$$\nabla F(\beta) = \frac{1}{n}\sum_{i=1}^{n} \nabla L(Y_i, f_\beta(X_i)) \approx \frac{1}{|S|}\sum_{i\in S} \nabla L(Y_i, f_\beta(X_i)).$$

1. Choose some initial value of β.

2. Randomly assign observations to mini-batches $S_1, \ldots, S_M$ of a certain size

3. Repeat until convergence:

    - $\underbrace{\text{For } m = 1,\ldots,M,}_{\text{One “epoch”}}$ update $\beta \leftarrow \beta - \gamma \cdot \dfrac{1}{|S_m|}\displaystyle\sum_{i\in S_m} \nabla L(Y_i, f_\beta(X_i))$.

Stochastic gradient descent

Use subset $S \subseteq \{1,\ldots,n\}$ (mini-batch) of observations to approximate gradient:

$$\nabla F(\beta) = \frac{1}{n}\sum_{i=1}^{n} \nabla L(Y_i, f_\beta(X_i)) \approx \frac{1}{|S|}\sum_{i\in S} \nabla L(Y_i, f_\beta(X_i)).$$

1. Choose some initial value of β.

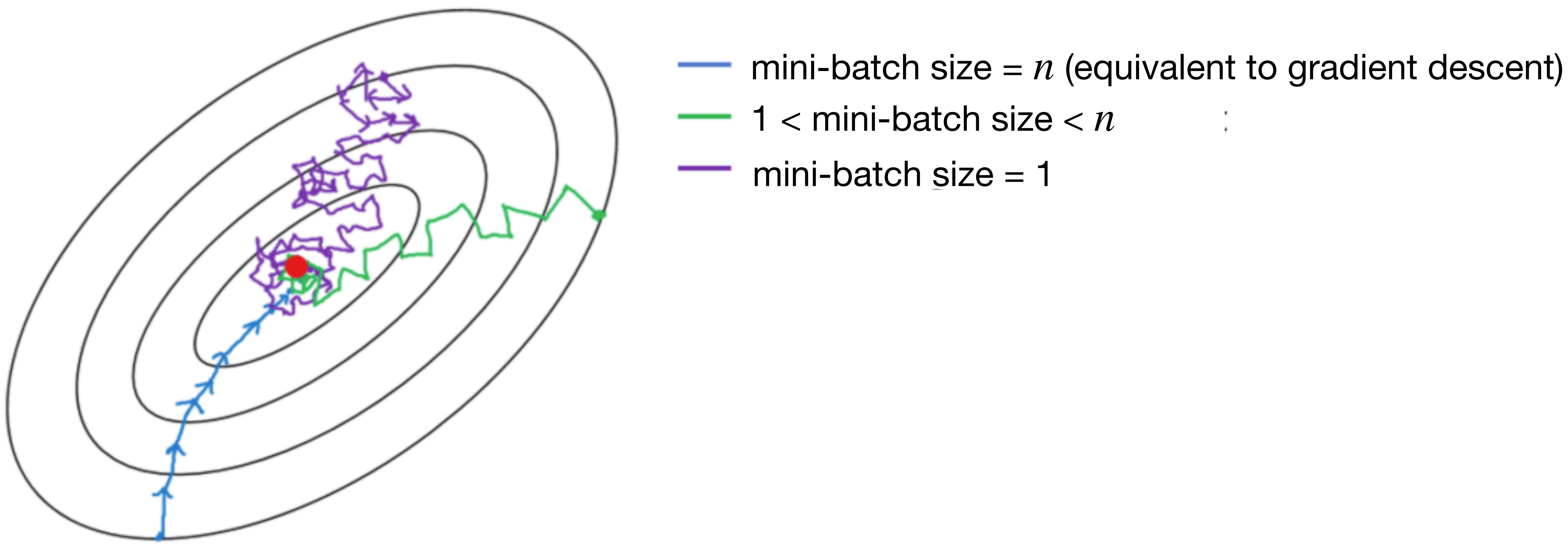
2. Randomly assign observations to mini-batches $S_1, \ldots, S_M$ of a certain size

3. Repeat until convergence:

- $\underbrace{\text{For } m = 1,\ldots,M}_{\text{One “epoch”}}$, update $\beta \leftarrow \beta - \gamma \cdot \dfrac{1}{|S_m|}\displaystyle\sum_{i\in S_m} \nabla L(Y_i, f_\beta(X_i)).$

Backpropagation: An efficient algorithm to compute $\nabla L(Y_i, f_\beta(X_i))$

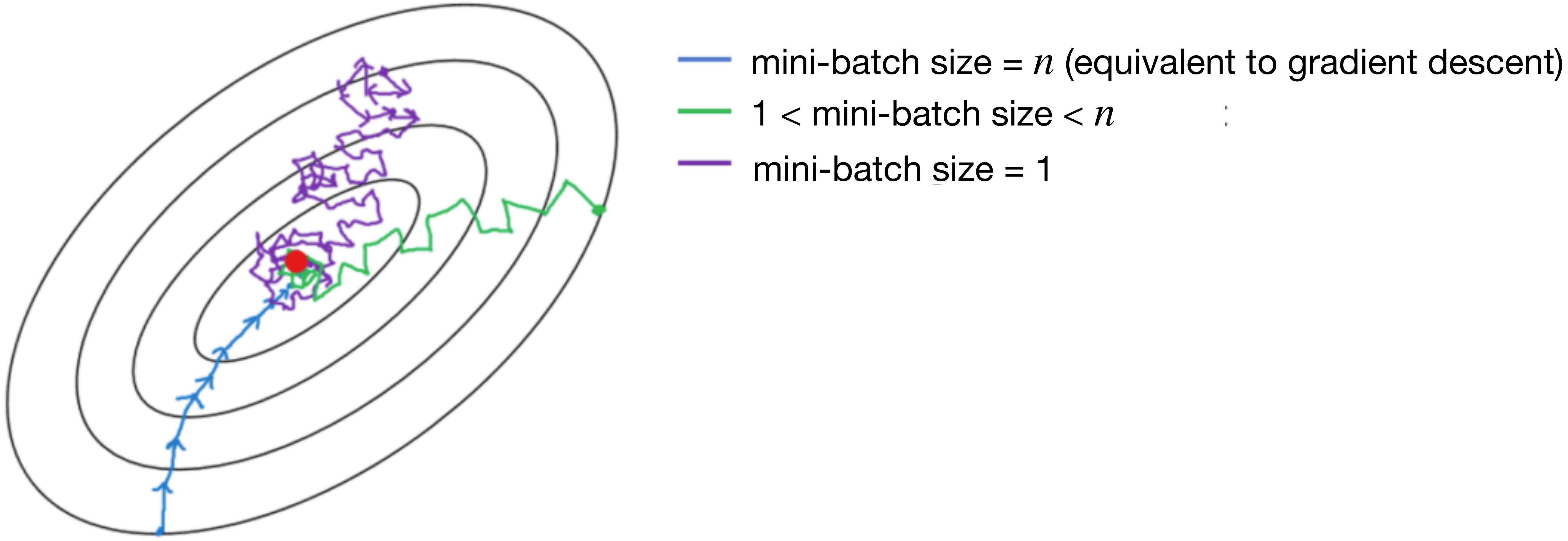
Behavior of stochastic gradient descent


mini-batch size = n (equivalent to gradient descent)
1 < mini-batch size < n
mini-batch size = 1

Source: https://medium.com/analytics-vidhya/gradient-descent-vs-stochastic-gd-vs-mini-batch-sgd-fbd3a2cb4ba4

Behavior of stochastic gradient descent

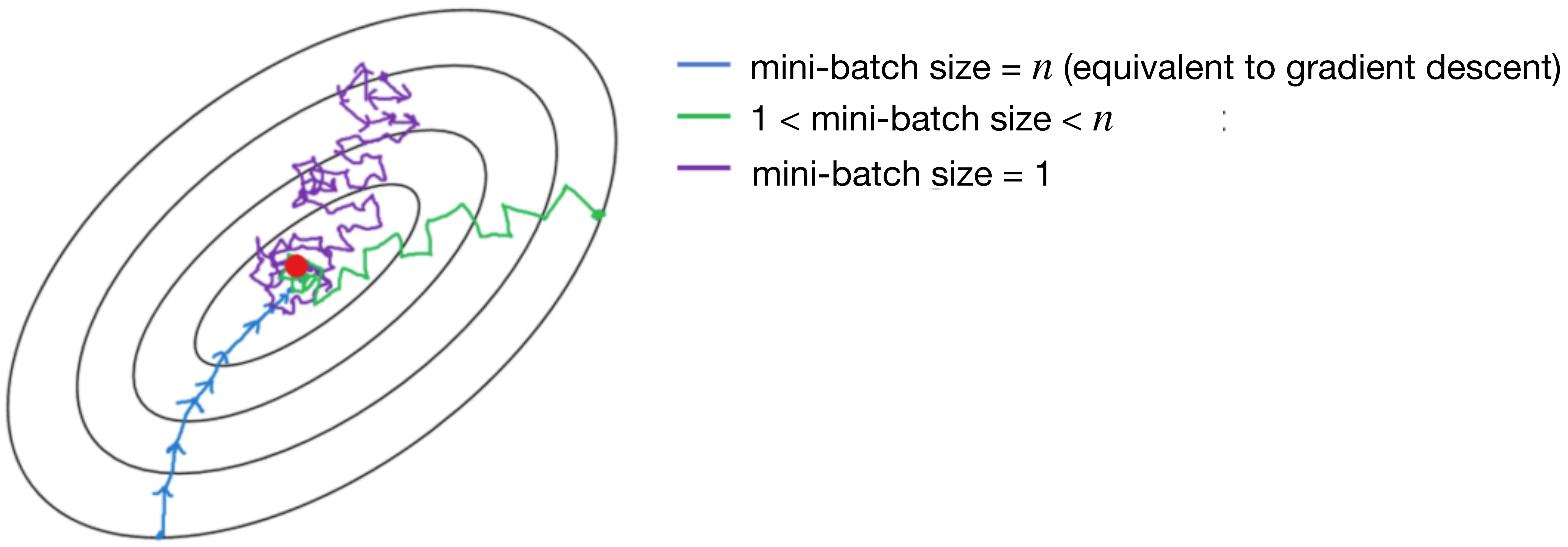
Stochastic gradient descent wobbles toward decreasing values of the objective:


mini-batch size = n (equivalent to gradient descent)
1 < mini-batch size < n
mini-batch size = 1

Source: https://medium.com/analytics-vidhya/gradient-descent-vs-stochastic-gd-vs-mini-batch-sgd-fbd3a2cb4ba4

Behavior of stochastic gradient descent

Stochastic gradient descent wobbles toward decreasing values of the objective:

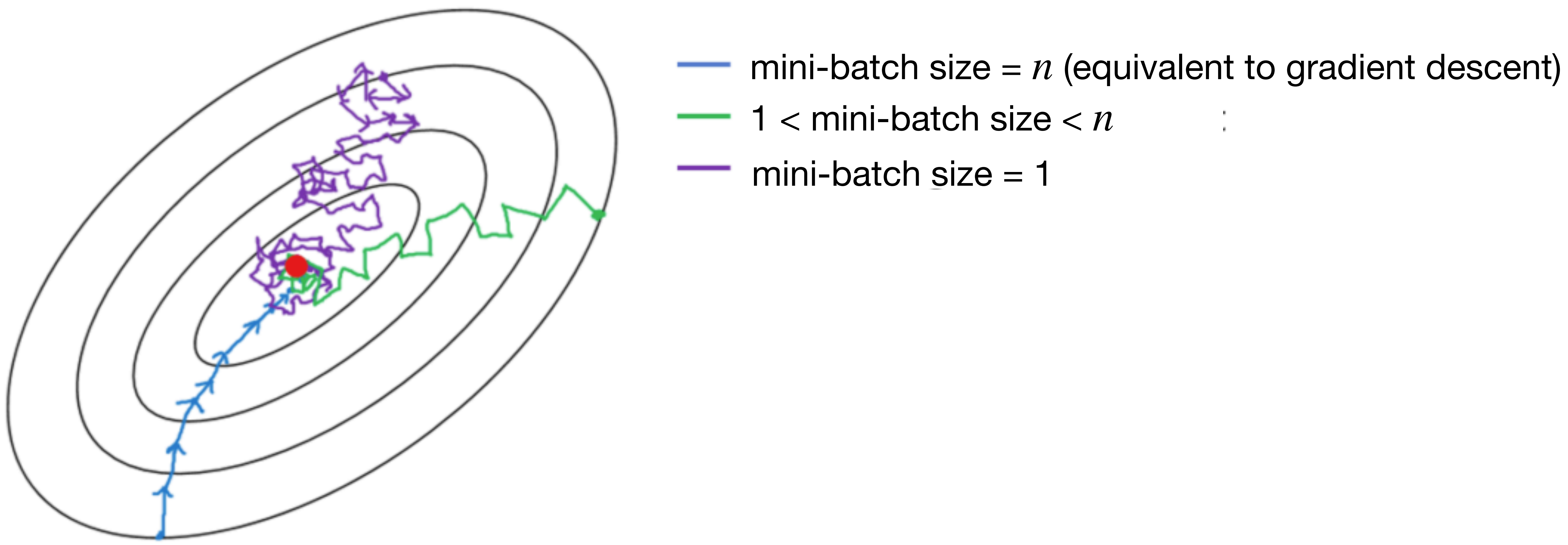

mini-batch size = n (equivalent to gradient descent)
1 < mini-batch size < n
mini-batch size = 1

Source: https://medium.com/analytics-vidhya/gradient-descent-vs-stochastic-gd-vs-mini-batch-sgd-fbd3a2cb4ba4

The smaller the mini-batch, the cheaper and more wobbly each step is;
Intermediate mini-batch sizes tend to work well, e.g. mini-batch size = 32.

Behavior of stochastic gradient descent

Stochastic gradient descent wobbles toward decreasing values of the objective:

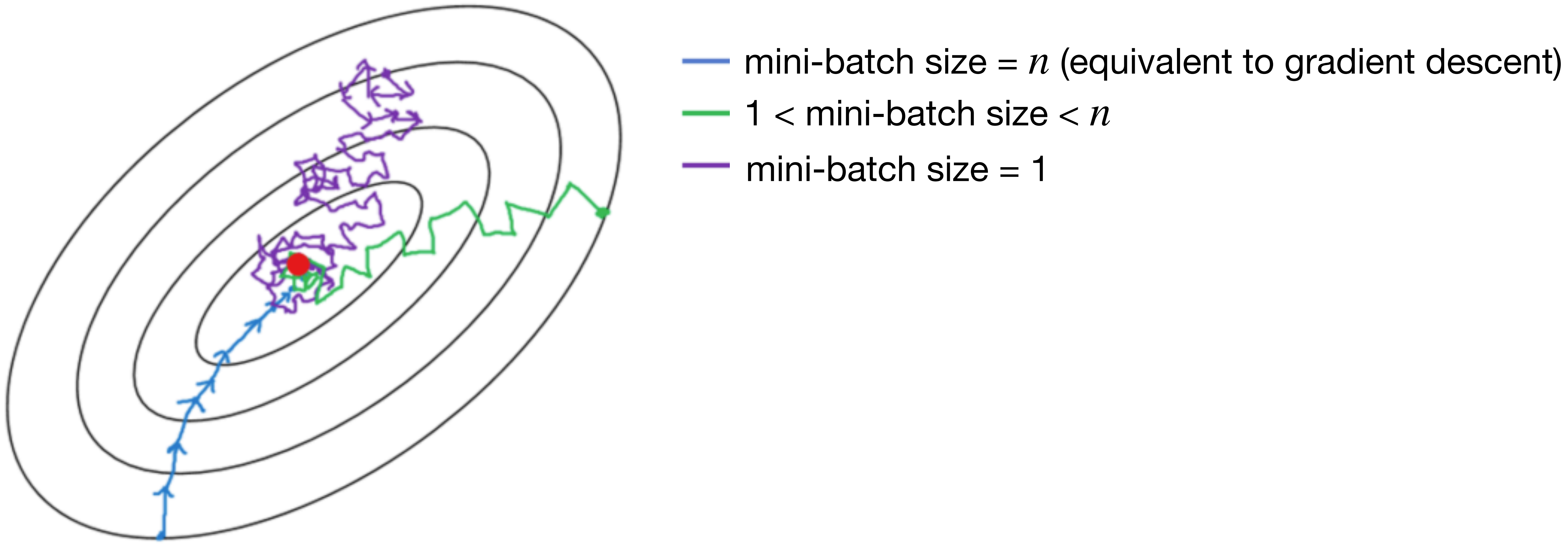

mini-batch size = n (equivalent to gradient descent)
1 < mini-batch size < n
mini-batch size = 1

Source: https://medium.com/analytics-vidhya/gradient-descent-vs-stochastic-gd-vs-mini-batch-sgd-fbd3a2cb4ba4

The smaller the mini-batch, the cheaper and more wobbly each step is;
Intermediate mini-batch sizes tend to work well, e.g. mini-batch size = 32.

Bonus: The extra randomness sometimes allows SGD to wobble past local minima.

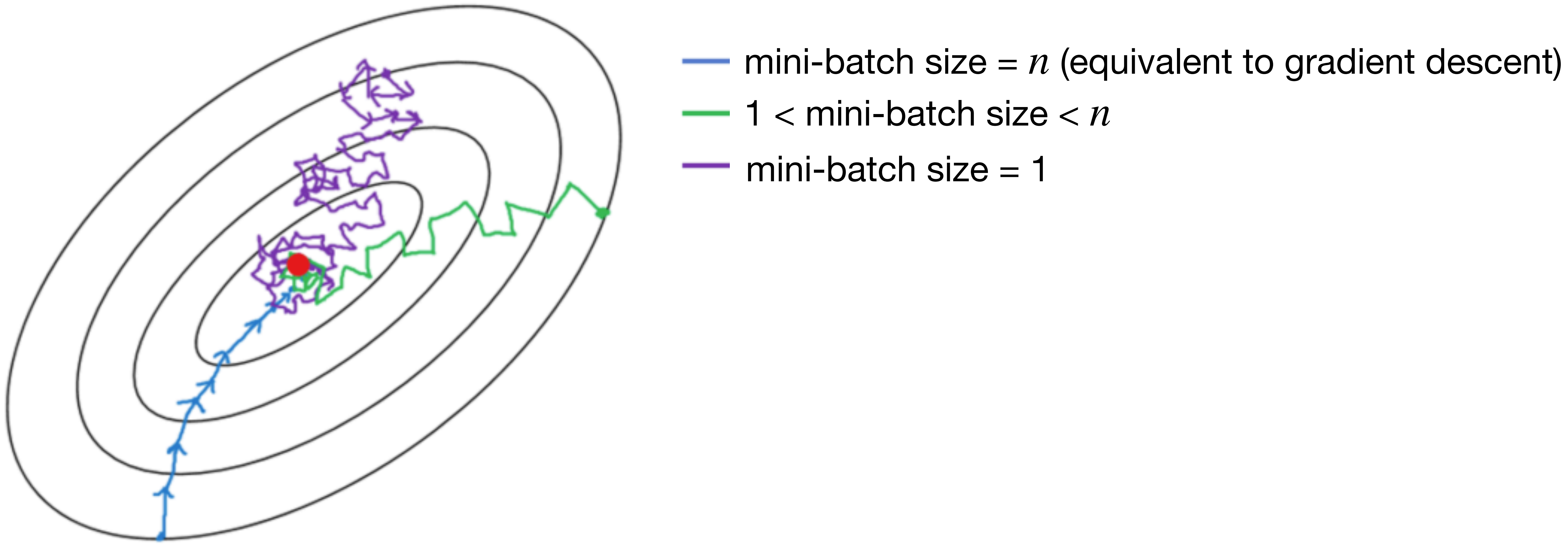
Stochastic gradient descent step versus epoch


mini-batch size = n (equivalent to gradient descent)
1 < mini-batch size < n
mini-batch size = 1

Source: https://medium.com/analytics-vidhya/gradient-descent-vs-stochastic-gd-vs-mini-batch-sgd-fbd3a2cb4ba4

Stochastic gradient descent step versus epoch

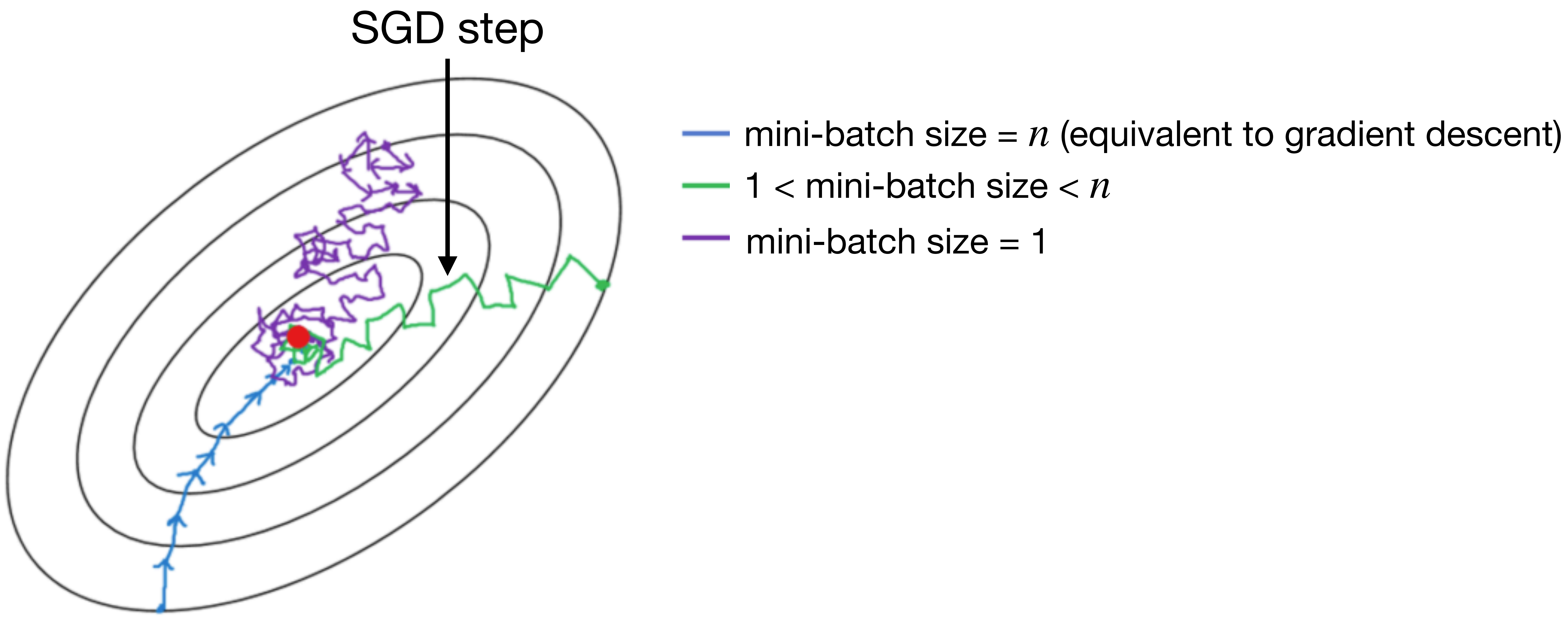
An epoch consists of SGD steps that cycle through all of the observations once.


mini-batch size = n (equivalent to gradient descent)
1 < mini-batch size < n
mini-batch size = 1

Source: https://medium.com/analytics-vidhya/gradient-descent-vs-stochastic-gd-vs-mini-batch-sgd-fbd3a2cb4ba4

Stochastic gradient descent step versus epoch

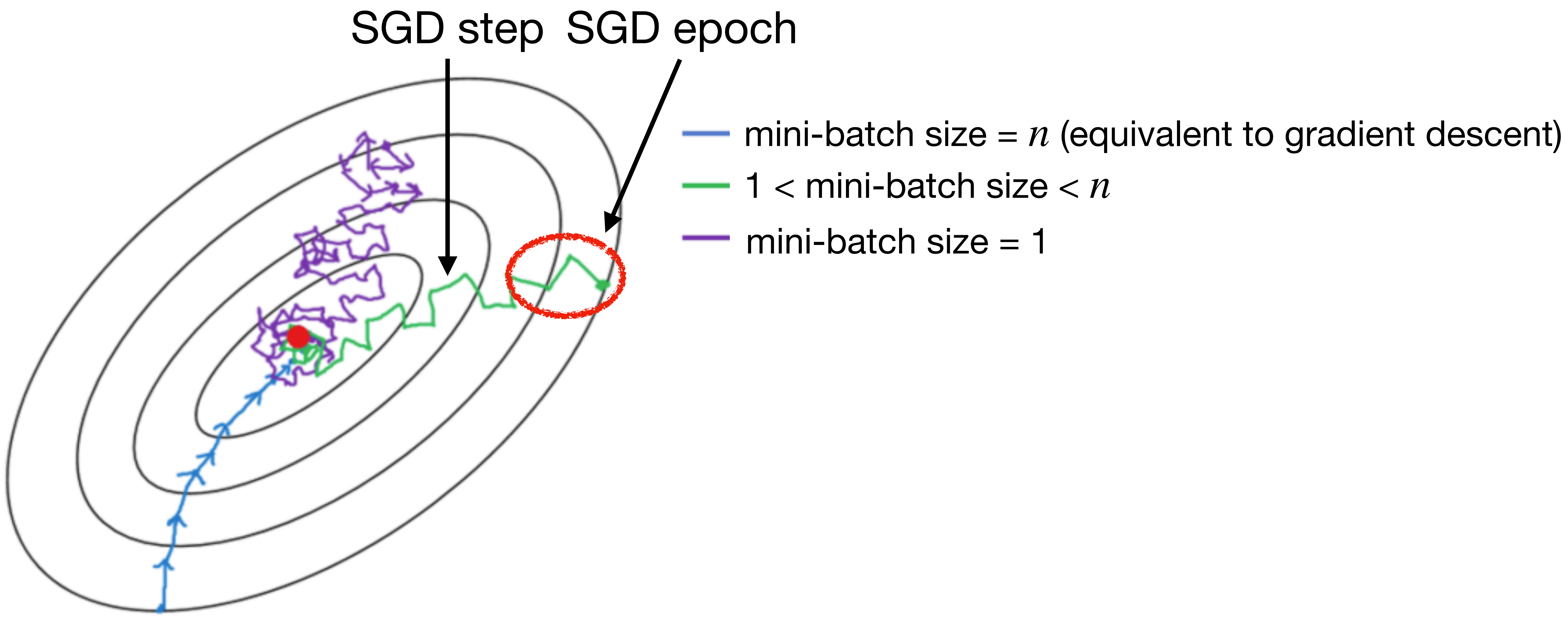
An epoch consists of SGD steps that cycle through all of the observations once.


SGD step
mini-batch size = n (equivalent to gradient descent)
1 < mini-batch size < n
mini-batch size = 1

Source: https://medium.com/analytics-vidhya/gradient-descent-vs-stochastic-gd-vs-mini-batch-sgd-fbd3a2cb4ba4

Stochastic gradient descent step versus epoch

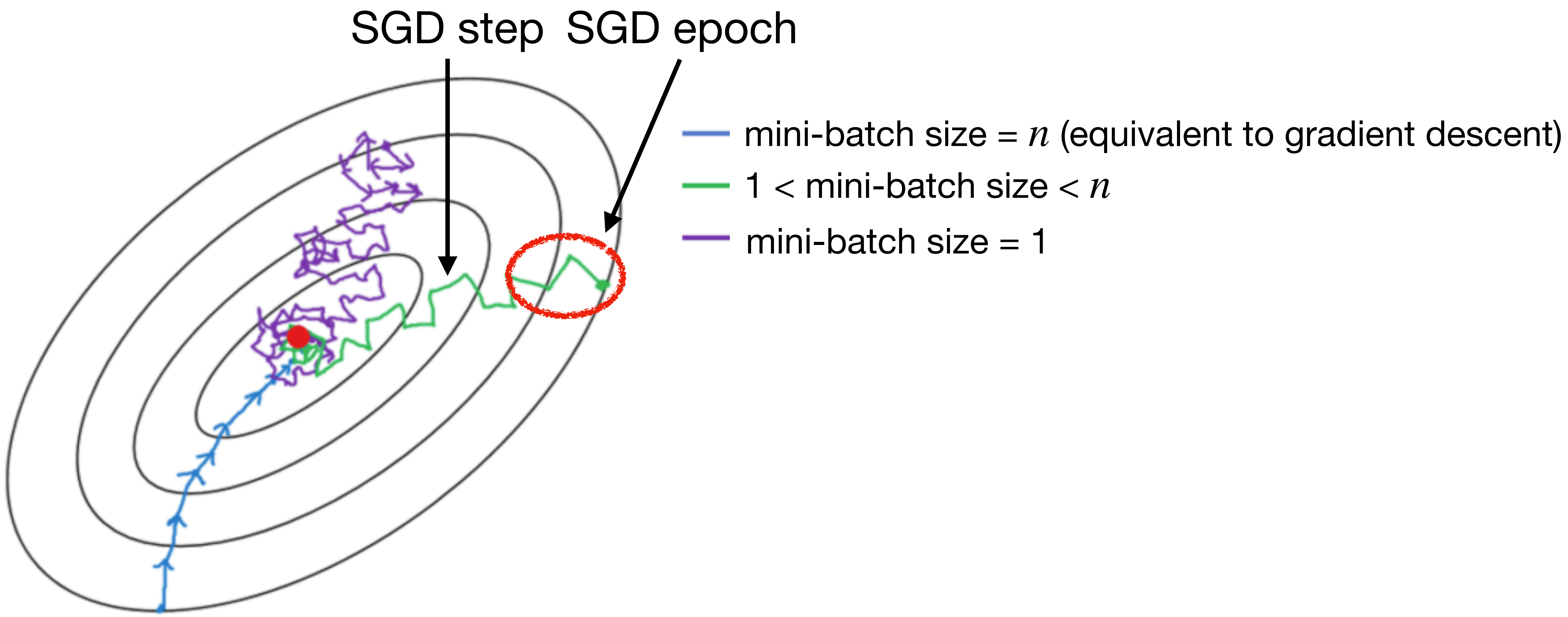
An epoch consists of SGD steps that cycle through all of the observations once.


SGD step
SGD epoch
mini-batch size = n (equivalent to gradient descent)
1 < mini-batch size < n
mini-batch size = 1

Source: https://medium.com/analytics-vidhya/gradient-descent-vs-stochastic-gd-vs-mini-batch-sgd-fbd3a2cb4ba4

Stochastic gradient descent step versus epoch

An epoch consists of SGD steps that cycle through all of the observations once.



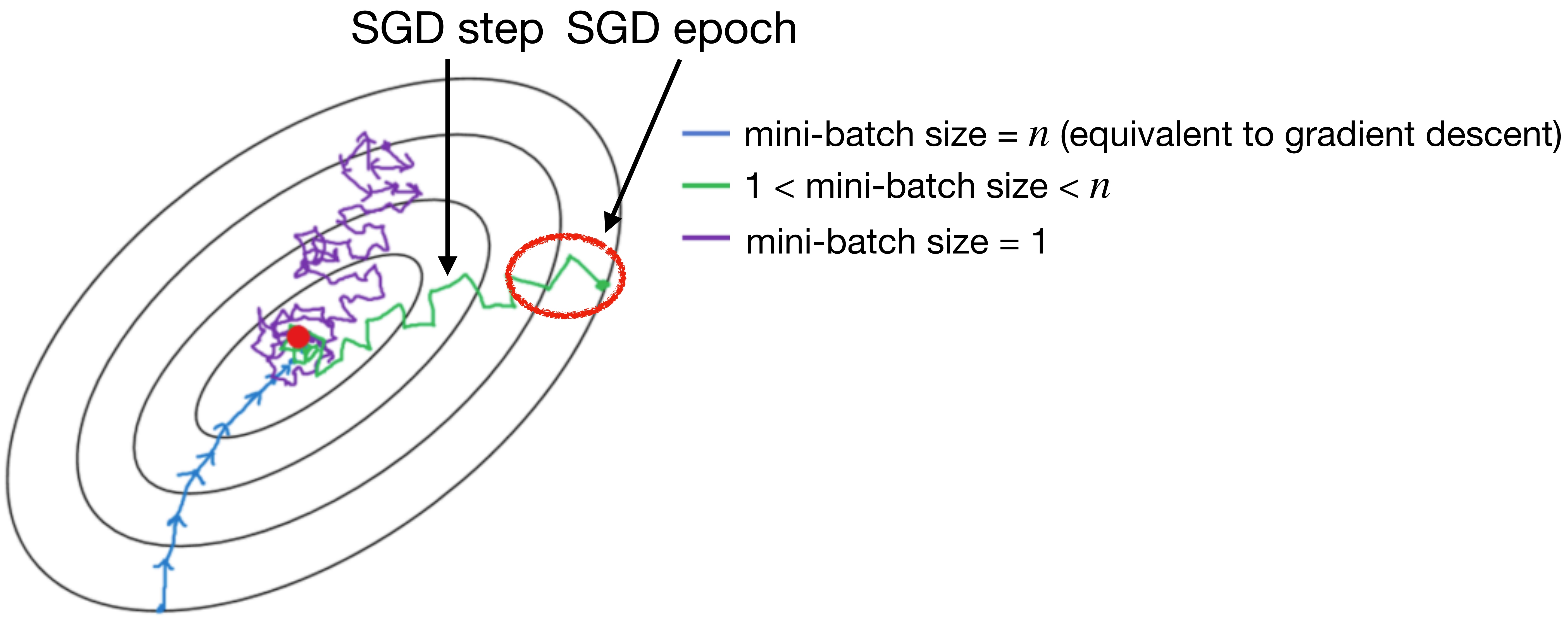
SGD step
SGD epoch
mini-batch size = n (equivalent to gradient descent)
1 < mini-batch size < n
mini-batch size = 1

Source: https://medium.com/analytics-vidhya/gradient-descent-vs-stochastic-gd-vs-mini-batch-sgd-fbd3a2cb4ba4

Observation	1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9
Epoch	Epoch 1									Epoch 2								

Stochastic gradient descent step versus epoch

An epoch consists of SGD steps that cycle through all of the observations once.

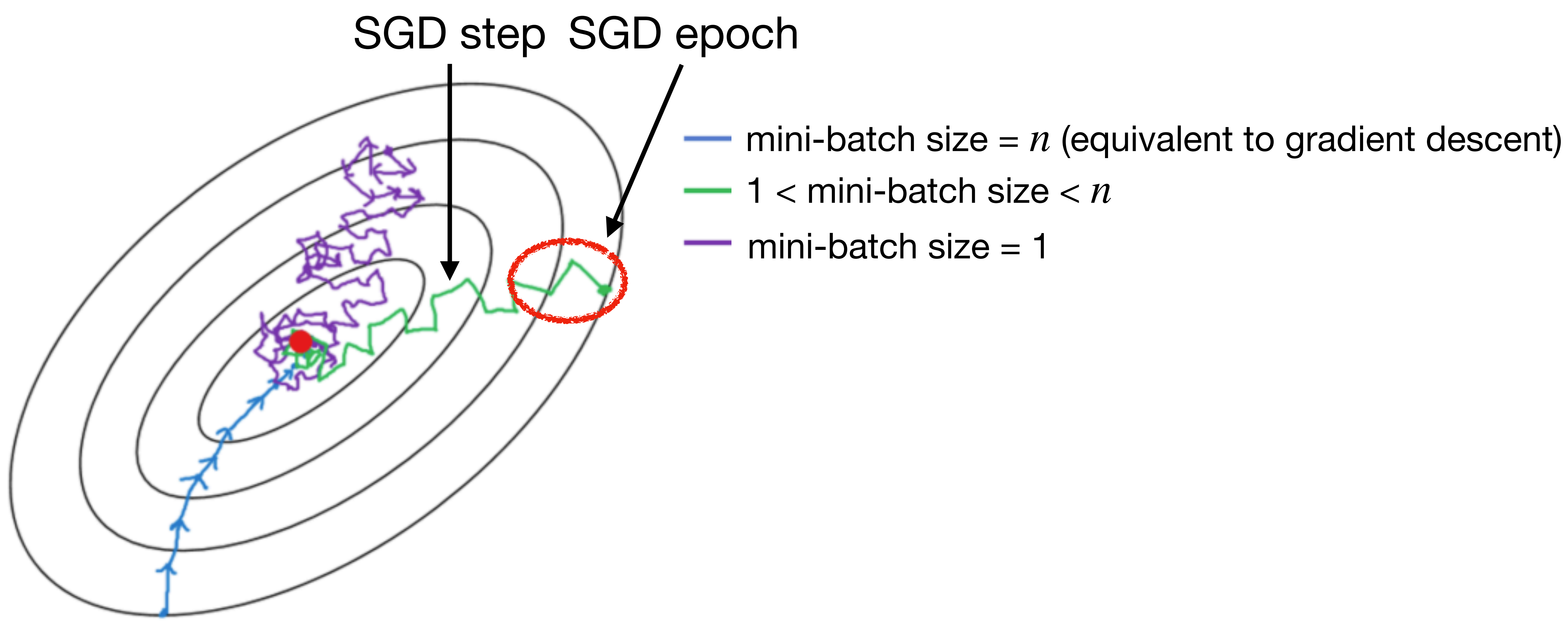

SGD step
SGD epoch
mini-batch size = n (equivalent to gradient descent)
1 < mini-batch size < n
mini-batch size = 1

Source: https://medium.com/analytics-vidhya/gradient-descent-vs-stochastic-gd-vs-mini-batch-sgd-fbd3a2cb4ba4

Observation	1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9
Epoch	Epoch 1									Epoch 2								
SGD step (mini-batch size 1)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

Stochastic gradient descent step versus epoch

An epoch consists of SGD steps that cycle through all of the observations once.

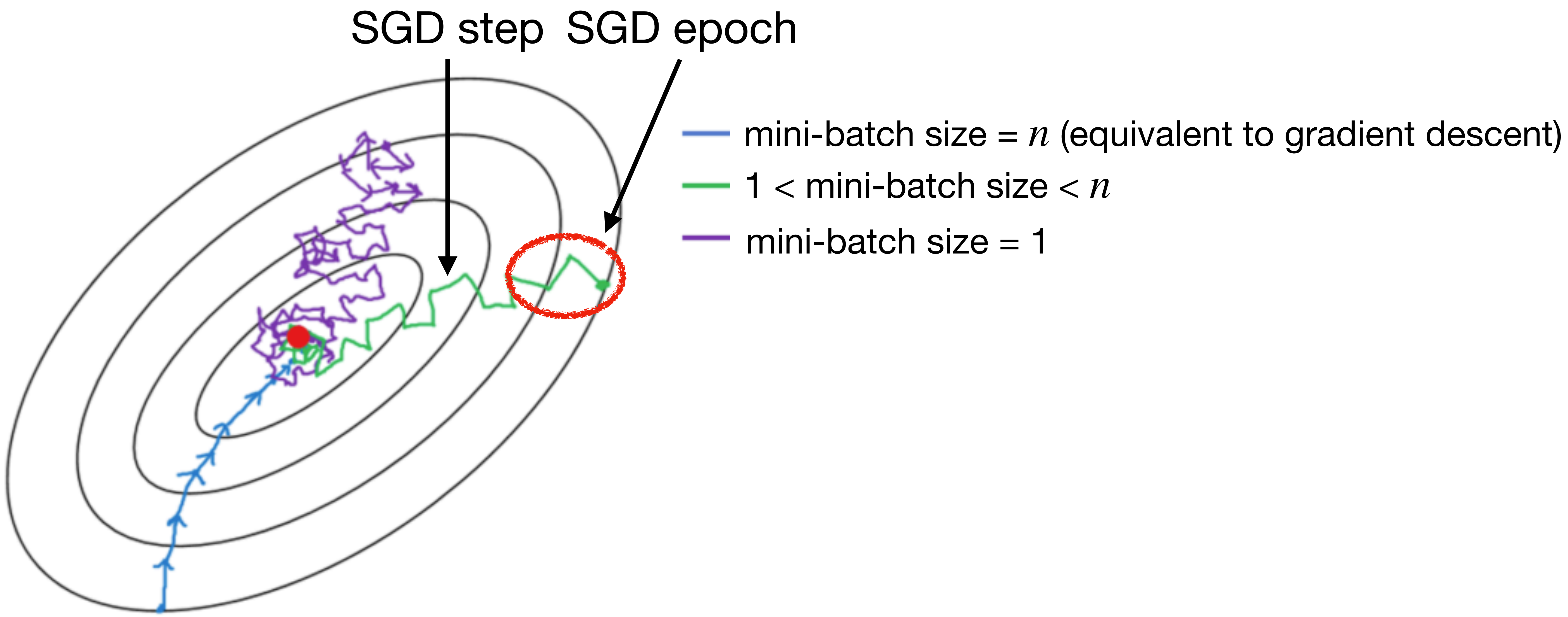

SGD step
SGD epoch
mini-batch size = n (equivalent to gradient descent)
1 < mini-batch size < n
mini-batch size = 1

Source: https://medium.com/analytics-vidhya/gradient-descent-vs-stochastic-gd-vs-mini-batch-sgd-fbd3a2cb4ba4

Observation	1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9
Epoch	Epoch 1									Epoch 2								
SGD step (mini-batch size 1)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
SGD step (mini-batch size 3)	Step 1			Step 2			Step 3			Step 4			Step 5			Step 6		

Stochastic gradient descent step versus epoch

An epoch consists of SGD steps that cycle through all of the observations once.


SGD step
SGD epoch
mini-batch size = n (equivalent to gradient descent)
1 < mini-batch size < n
mini-batch size = 1

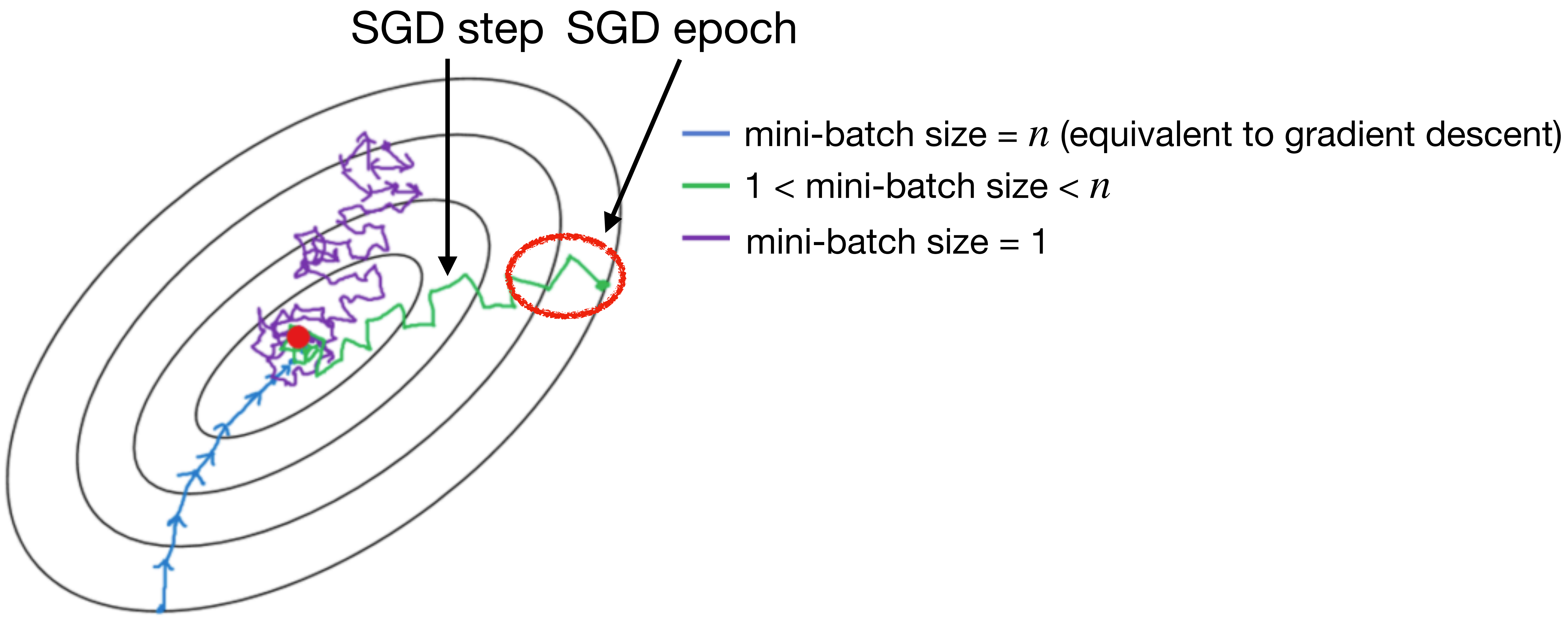
Source: https://medium.com/analytics-vidhya/gradient-descent-vs-stochastic-gd-vs-mini-batch-sgd-fbd3a2cb4ba4

E.g. $n = 9$, mini-batch size 3. Then each epoch consists of three SGD steps.

Observation	1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9
Epoch	Epoch 1									Epoch 2								
SGD step (mini-batch size 1)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
SGD step (mini-batch size 3)	Step 1			Step 2			Step 3			Step 4			Step 5			Step 6		

Stochastic gradient descent step versus epoch

An epoch consists of SGD steps that cycle through all of the observations once.

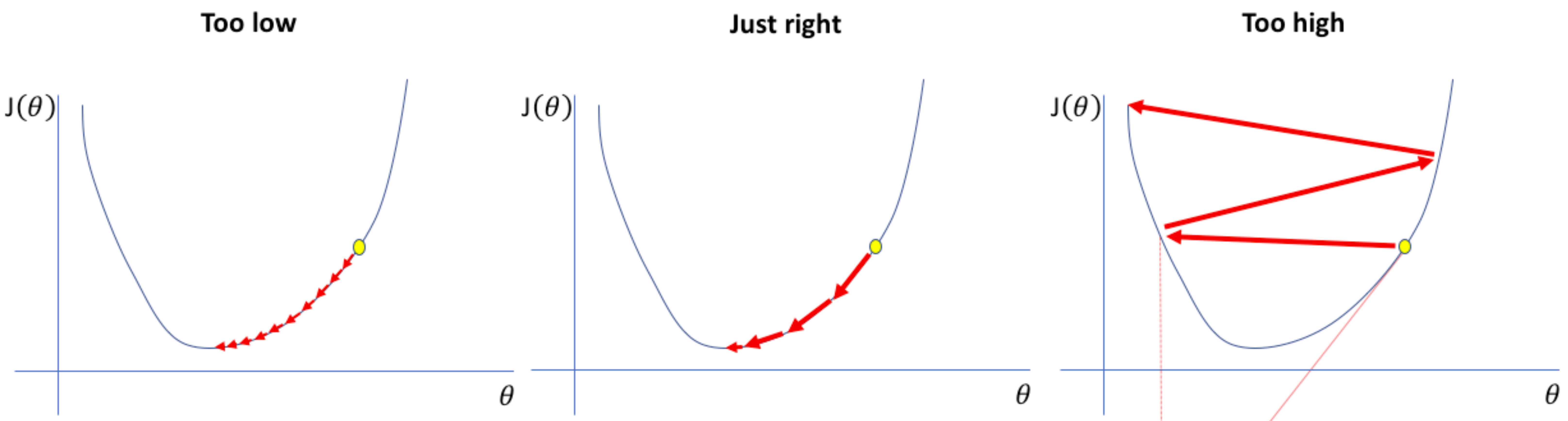

SGD step
SGD epoch
mini-batch size = n (equivalent to gradient descent)
1 < mini-batch size < n
mini-batch size = 1

Source: https://medium.com/analytics-vidhya/gradient-descent-vs-stochastic-gd-vs-mini-batch-sgd-fbd3a2cb4ba4

E.g. $n = 9$, mini-batch size 3. Then each epoch consists of three SGD steps.

Observation	1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9
Epoch	Epoch 1									Epoch 2								
SGD step (mini-batch size 1)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
SGD step (mini-batch size 3)	Step 1			Step 2			Step 3			Step 4			Step 5			Step 6		
Gradient step	Step 1									Step 2								

The learning rate for (stochastic) gradient descent



Too low

A small learning rate requires many updates before reaching the minimum point

Just right

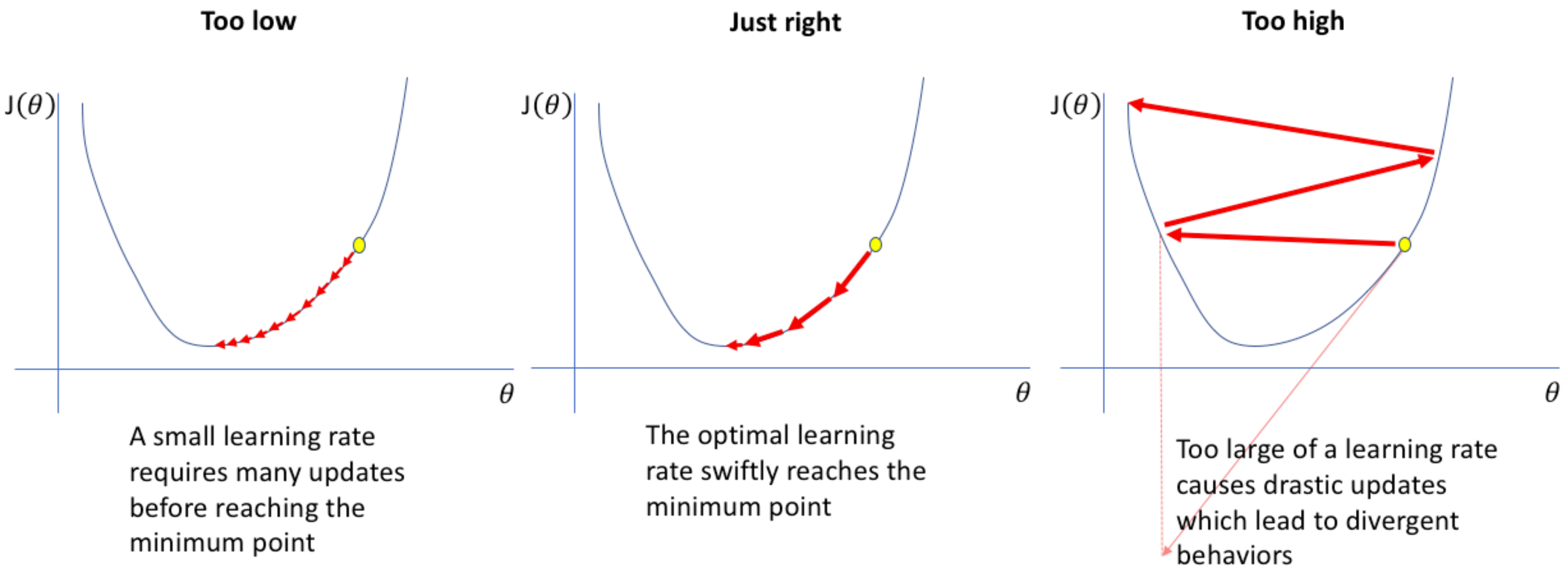
The optimal learning rate swiftly reaches the minimum point

Too high

Too large of a learning rate causes drastic updates which lead to divergent behaviors

Source: https://www.jeremyjordan.me/nn-learning-rate/

The learning rate for (stochastic) gradient descent



Too low

A small learning rate requires many updates before reaching the minimum point

Just right

The optimal learning rate swiftly reaches the minimum point

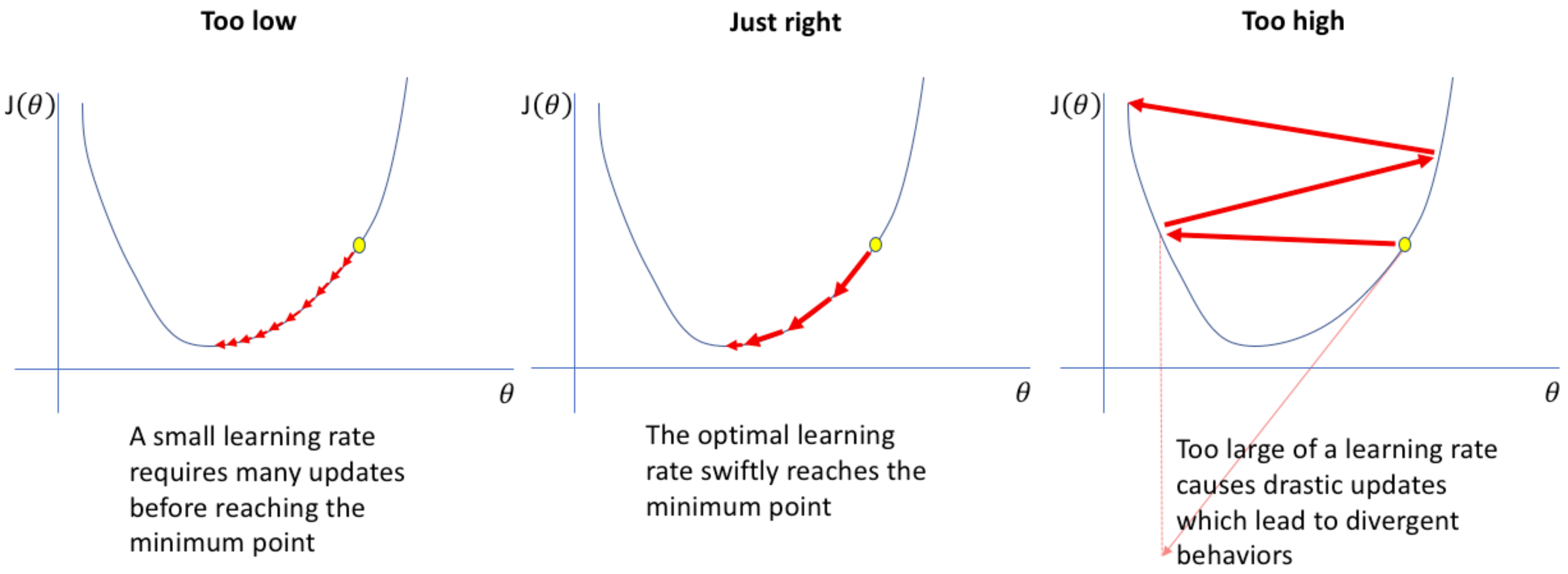
Too high

Too large of a learning rate causes drastic updates which lead to divergent behaviors

Source: https://www.jeremyjordan.me/nn-learning-rate/

- Setting the learning rate is more of an art than a science; might need to try a few values to get a good one.

The learning rate for (stochastic) gradient descent



A small learning rate requires many updates before reaching the minimum point

The optimal learning rate swiftly reaches the minimum point

Too large of a learning rate causes drastic updates which lead to divergent behaviors

Source: https://www.jeremyjordan.me/nn-learning-rate/

- Setting the learning rate is more of an art than a science; might need to try a few values to get a good one.
- Especially for non-convex optimization, people come up with clever strategies like shrinking learning rates, cycling learning rates, adaptive learning rates, etc. (RMSprop, Adam, AdaGrad, AdaDelta, …)

Regularization in deep learning models

Regularization in deep learning models

Explicit regularization via penalization. Ridge regression penalty is the most common; this kind of penalization is known as weight decay.

Regularization in deep learning models

Explicit regularization via penalization. Ridge regression penalty is the most common; this kind of penalization is known as weight decay.

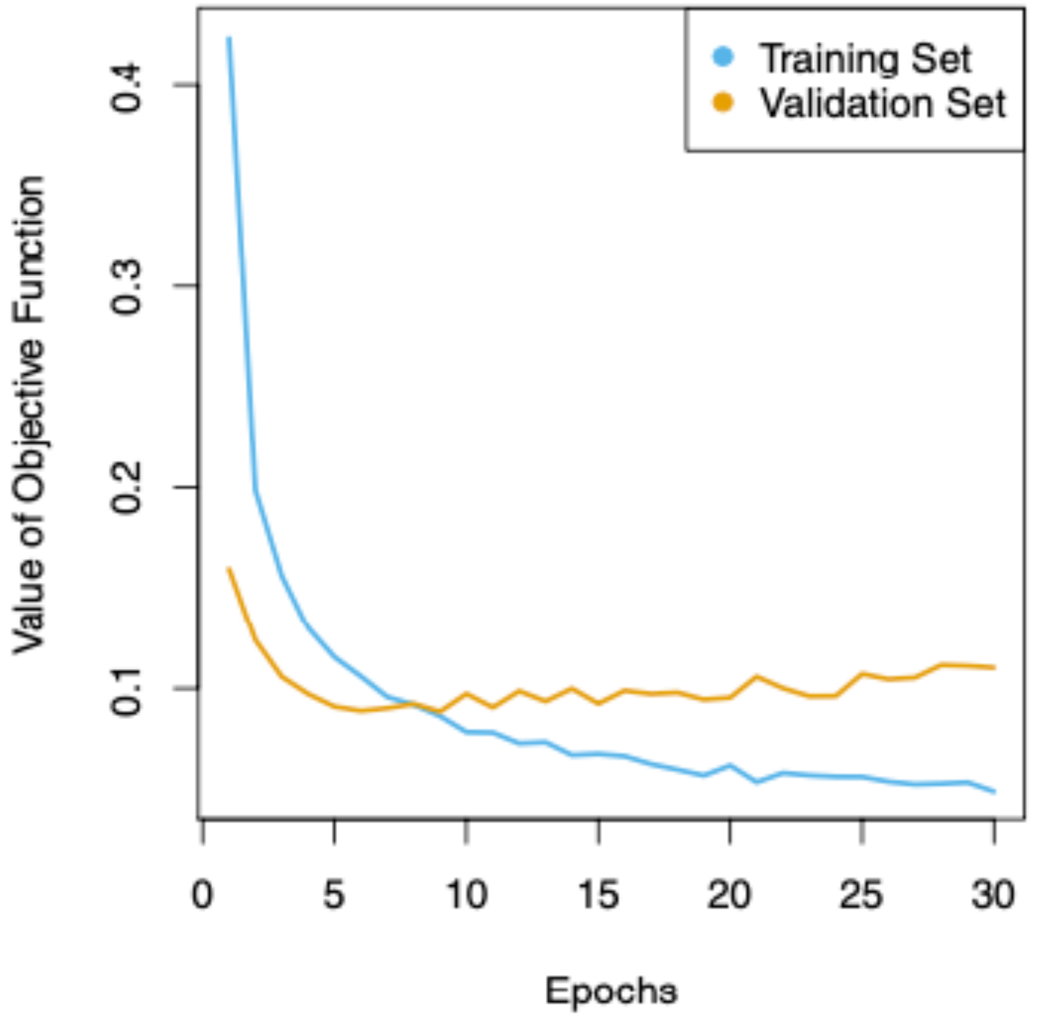
Implicit regularization: Other techniques to control complexity of the model.

Regularization in deep learning models

Explicit regularization via penalization. Ridge regression penalty is the most common; this kind of penalization is known as weight decay.

Implicit regularization: Other techniques to control complexity of the model.

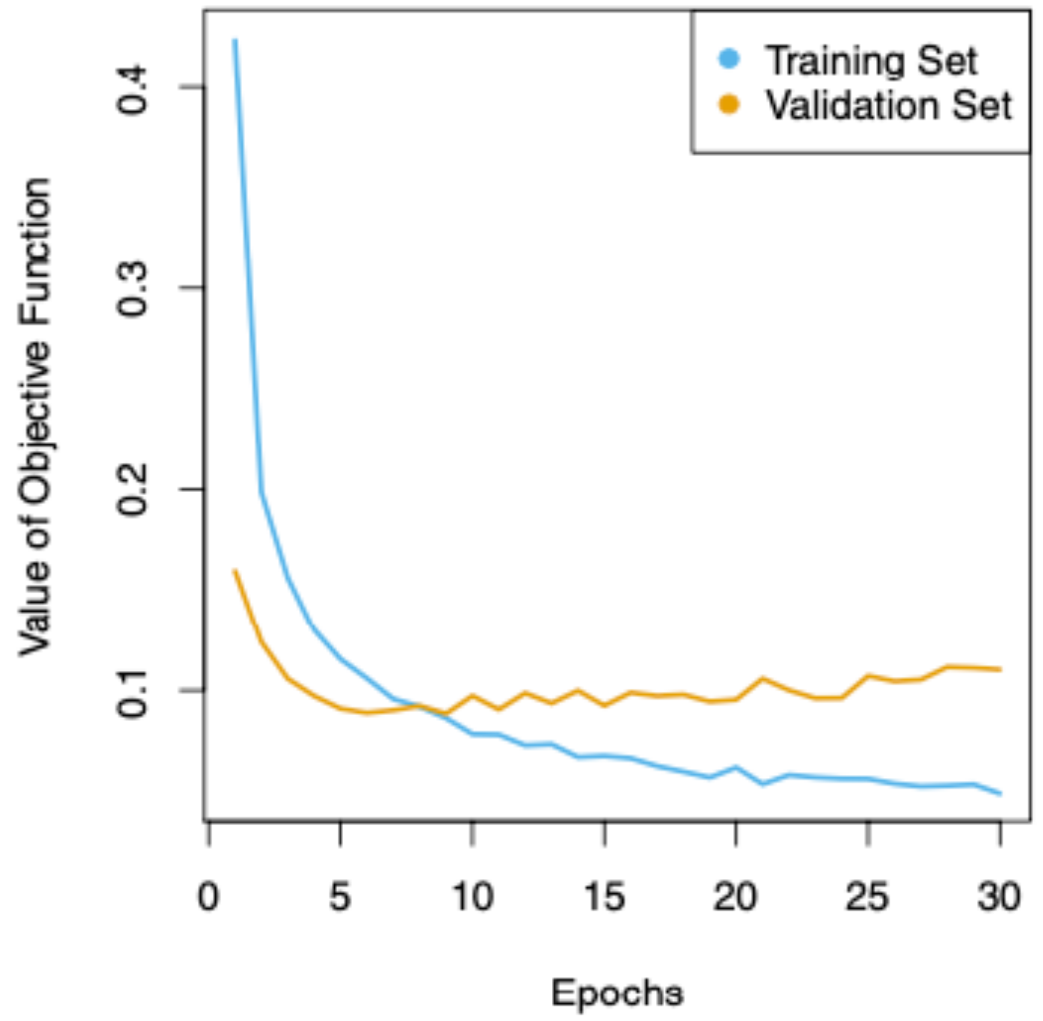
- Early stopping: Do not run SGD until convergence; rather, stop SGD when validation error starts to increase.



Regularization in deep learning models

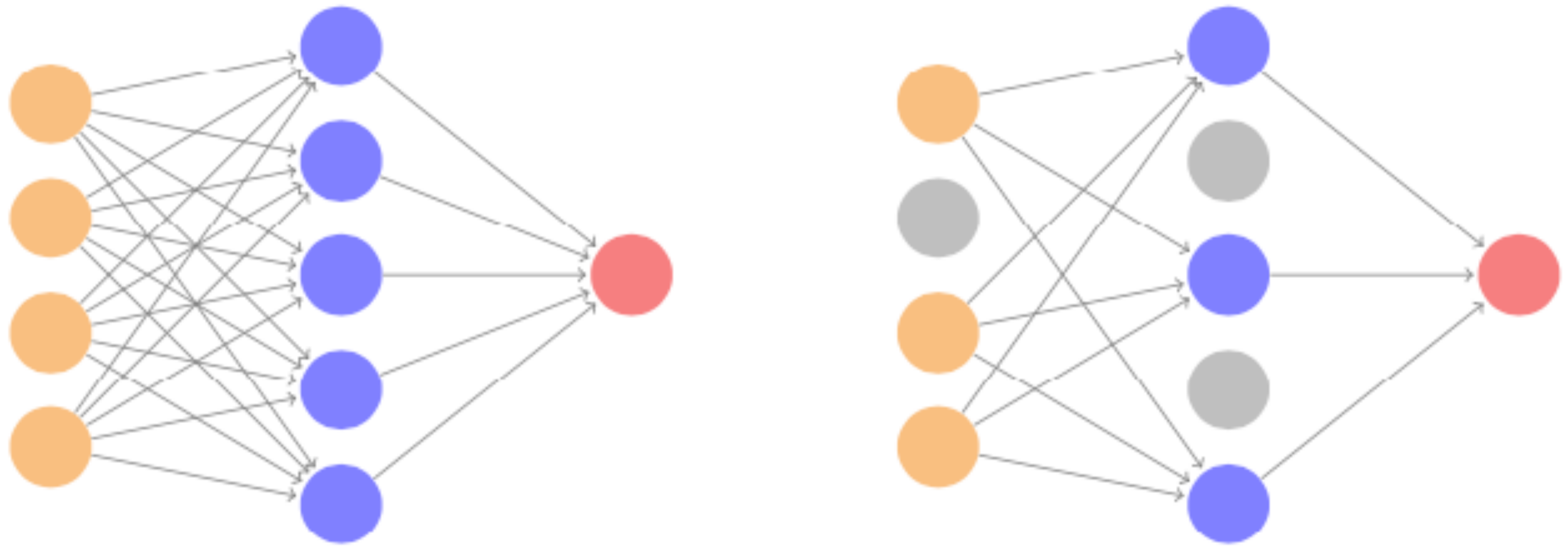
Explicit regularization via penalization. Ridge regression penalty is the most common; this kind of penalization is known as weight decay.

Implicit regularization: Other techniques to control complexity of the model.



- Early stopping: Do not run SGD until convergence; rather, stop SGD when validation error starts to increase.

- Dropout: At each SGD iteration, remove a randomly selected set of nodes from the network (analogous to sub-sampling features for random forests).



Model complexity and tuning in deep learning

Model complexity and tuning in deep learning

Deep learning has many tuning parameters:

Model complexity and tuning in deep learning

Deep learning has many tuning parameters:

- Network architecture (number of layers, units per layer)

Model complexity and tuning in deep learning

Deep learning has many tuning parameters:

- Network architecture (number of layers, units per layer)
- Optimization algorithm used (RMSprop, Adam, etc.)

Model complexity and tuning in deep learning

Deep learning has many tuning parameters:

- Network architecture (number of layers, units per layer)
- Optimization algorithm used (RMSprop, Adam, etc.)
- Weight decay parameter, dropout rate, number of SGD iterations

Model complexity and tuning in deep learning

Deep learning has many tuning parameters:

- Network architecture (number of layers, units per layer)
- Optimization algorithm used (RMSprop, Adam, etc.)
- Weight decay parameter, dropout rate, number of SGD iterations

More complex models arise from bigger networks, less weight decay, less dropout, more SGD iterations.

Model complexity and tuning in deep learning

Deep learning has many tuning parameters:

- Network architecture (number of layers, units per layer)
- Optimization algorithm used (RMSprop, Adam, etc.)
- Weight decay parameter, dropout rate, number of SGD iterations

More complex models arise from bigger networks, less weight decay, less dropout, more SGD iterations.

Deep learning performance can be sensitive to these parameters; there is no standard way to tune neural networks; tuning is computationally expensive.

Model complexity and tuning in deep learning

Deep learning has many tuning parameters:

- Network architecture (number of layers, units per layer)
- Optimization algorithm used (RMSprop, Adam, etc.)
- Weight decay parameter, dropout rate, number of SGD iterations

More complex models arise from bigger networks, less weight decay, less dropout, more SGD iterations.

Deep learning performance can be sensitive to these parameters; there is no standard way to tune neural networks; tuning is computationally expensive.

For computational savings, validation sets are typically employed instead of cross-validation to tune neural networks.

Summary

Summary

- Adding hidden layers to logistic regression gives multilayer neural networks.

Summary

- Adding hidden layers to logistic regression gives multilayer neural networks.
- Multilayer neural networks build increasingly complex representations of the input data (feature learning).

Summary

- Adding hidden layers to logistic regression gives multilayer neural networks.
- Multilayer neural networks build increasingly complex representations of the input data (feature learning).
- Stochastic gradient descent is used to train neural networks; it uses noisy approximations to the gradient but is much faster than gradient descent.

Summary

- Adding hidden layers to logistic regression gives multilayer neural networks.
- Multilayer neural networks build increasingly complex representations of the input data (feature learning).
- Stochastic gradient descent is used to train neural networks; it uses noisy approximations to the gradient but is much faster than gradient descent.
- Explicit and implicit regularization are used to control model complexity, the latter including early stopping and dropout.

Summary

- Adding hidden layers to logistic regression gives multilayer neural networks.
- Multilayer neural networks build increasingly complex representations of the input data (feature learning).
- Stochastic gradient descent is used to train neural networks; it uses noisy approximations to the gradient but is much faster than gradient descent.
- Explicit and implicit regularization are used to control model complexity, the latter including early stopping and dropout.
- Neural networks come with a smorgasbord of tuning parameters, which often require considerable effort to tune.