

Deep learning preliminaries

STAT 4710

November 15, 2022

Rolling into a new unit!

- ✓ **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

What is deep learning?

What is deep learning?

Deep learning is an enormously successful class of predictive models that has achieved state-of-the-art performance across a variety of domains:

What is deep learning?

Deep learning is an enormously successful class of predictive models that has achieved state-of-the-art performance across a variety of domains:

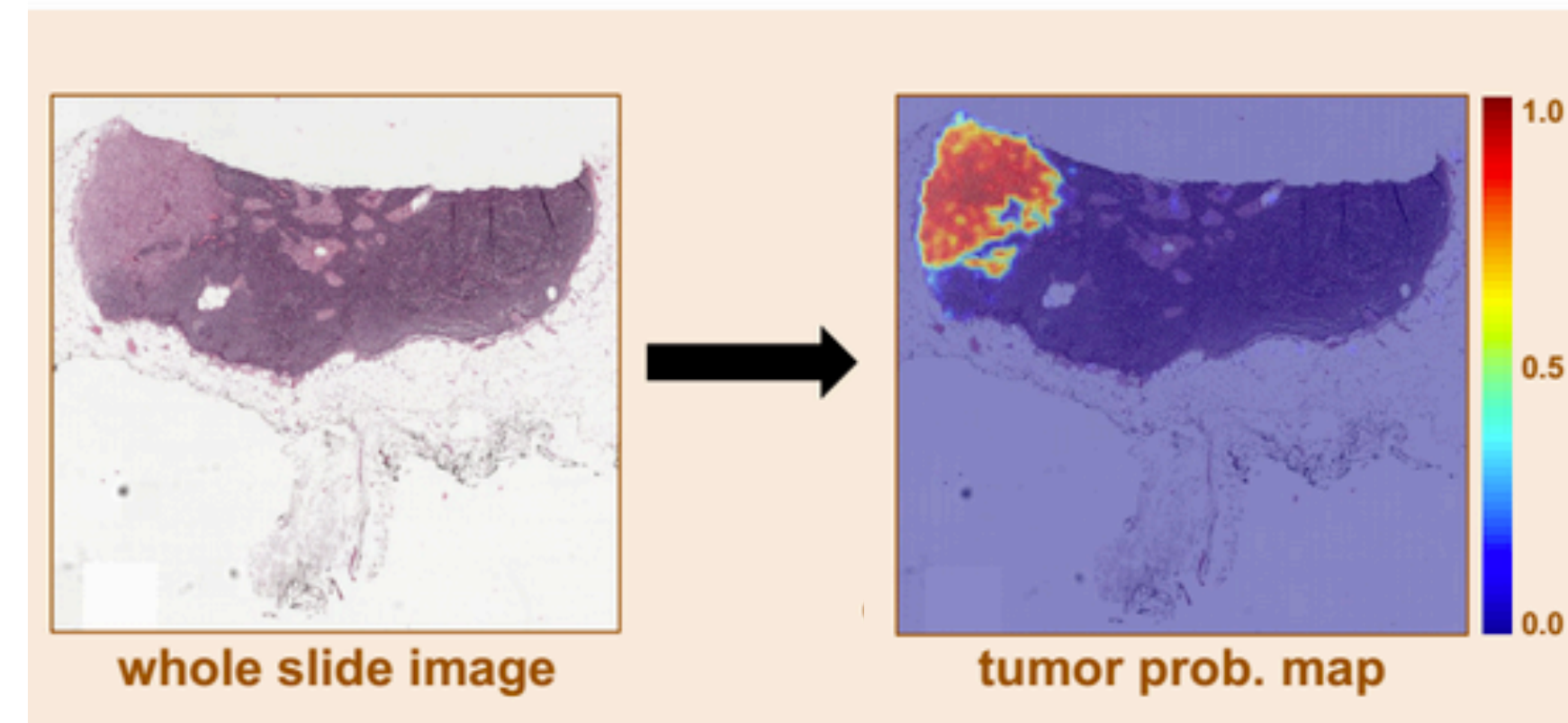
Image processing

What is deep learning?

Deep learning is an enormously successful class of predictive models that has achieved state-of-the-art performance across a variety of domains:

Image processing

- Medical image analysis



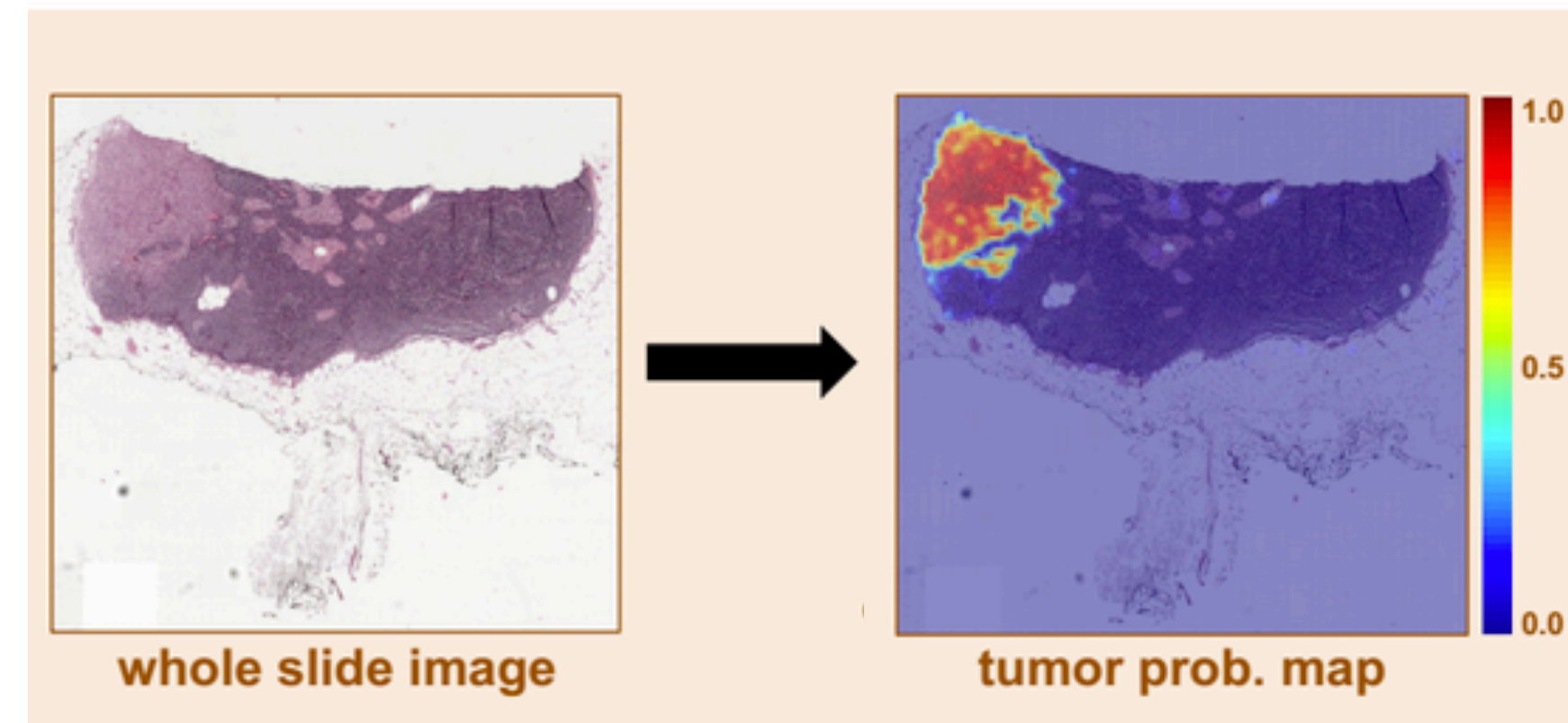
<https://towardsdatascience.com/understanding-cancer-using-machine-learning-84087258ee18>

What is deep learning?

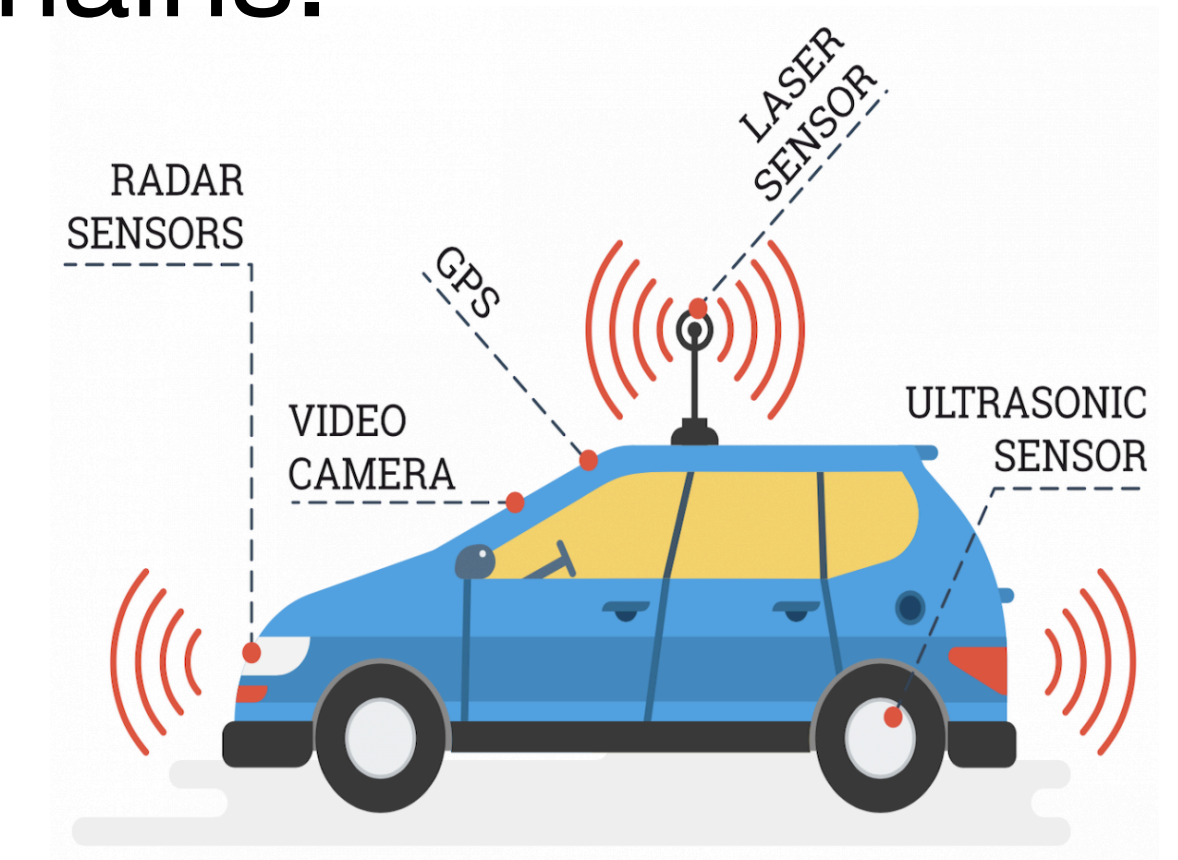
Deep learning is an enormously successful class of predictive models that has achieved state-of-the-art performance across a variety of domains:

Image processing

- Medical image analysis
- Self-driving cars



<https://towardsdatascience.com/understanding-cancer-using-machine-learning-84087258ee18>



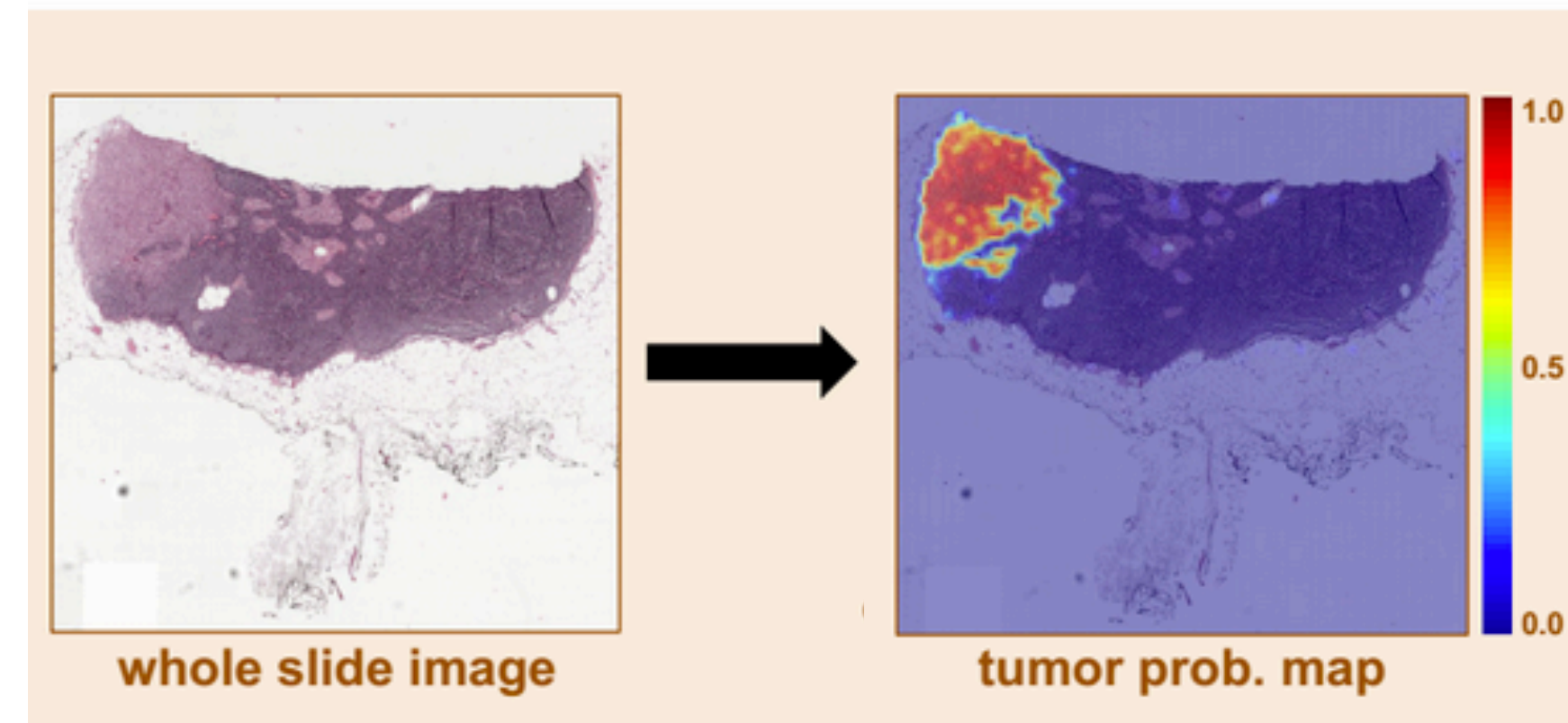
<https://www.eejournal.com/article/self-driving-cars-what-the-engineers-think/>

What is deep learning?

Deep learning is an enormously successful class of predictive models that has achieved state-of-the-art performance across a variety of domains:

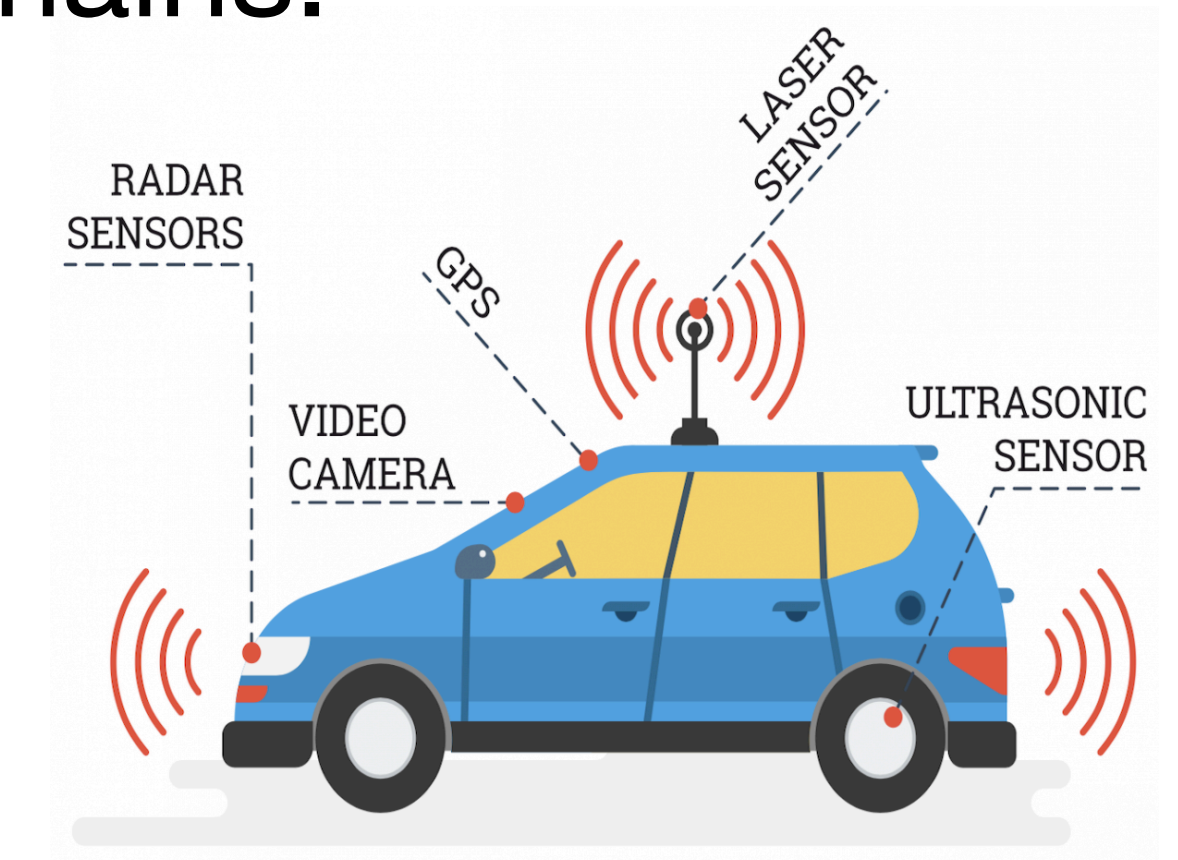
Image processing

- Medical image analysis
- Self-driving cars



<https://towardsdatascience.com/understanding-cancer-using-machine-learning-84087258ee18>

Natural language processing



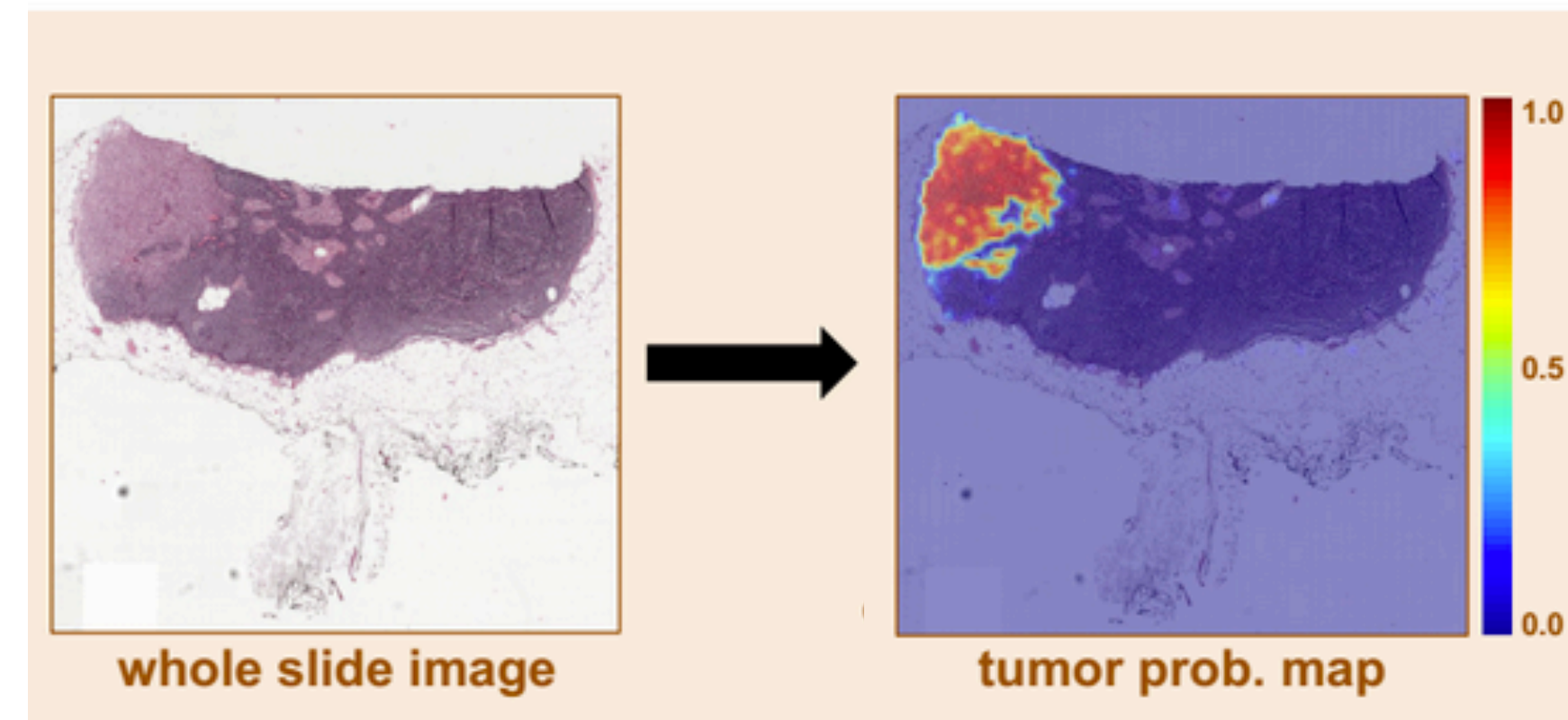
<https://www.eejournal.com/article/self-driving-cars-what-the-engineers-think/>

What is deep learning?

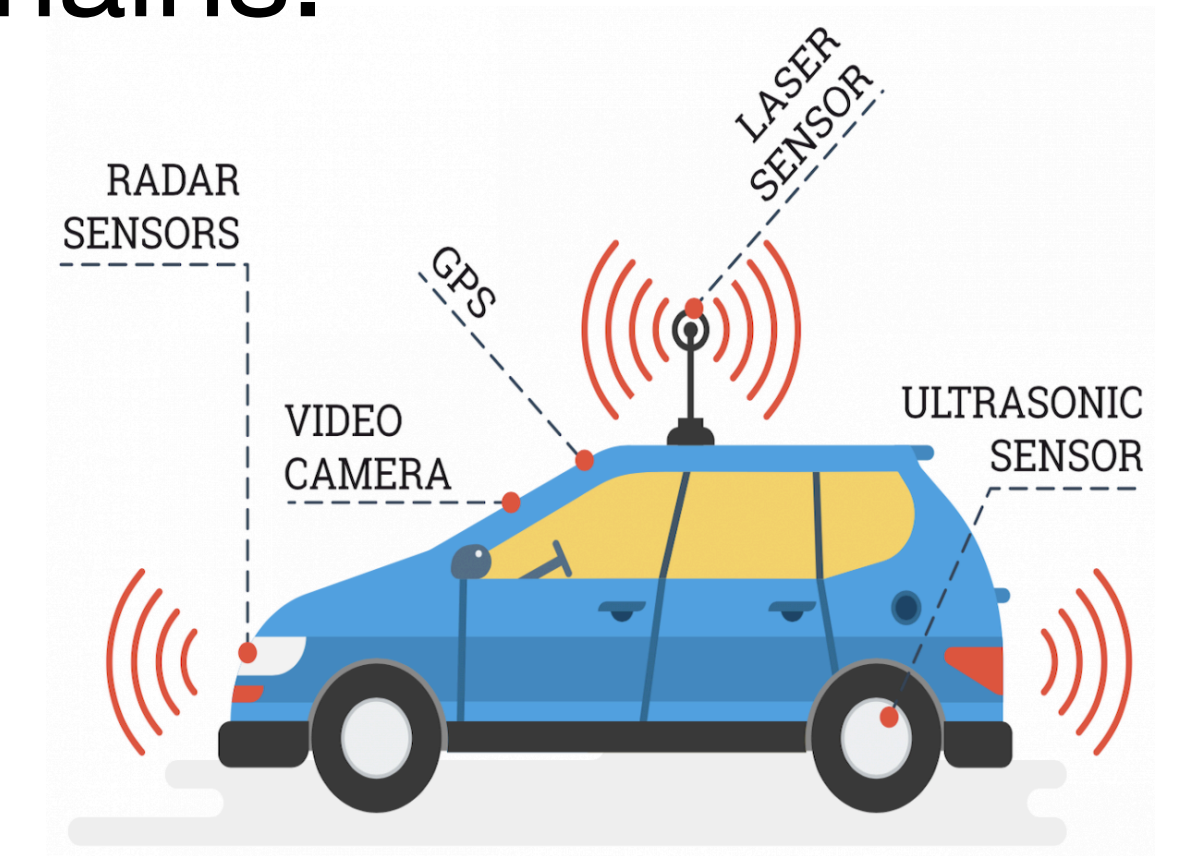
Deep learning is an enormously successful class of predictive models that has achieved state-of-the-art performance across a variety of domains:

Image processing

- Medical image analysis
- Self-driving cars



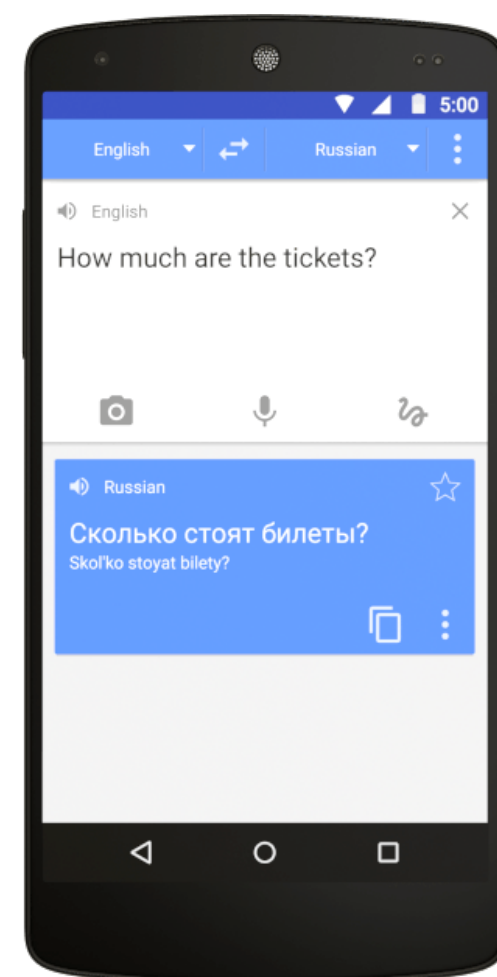
<https://towardsdatascience.com/understanding-cancer-using-machine-learning-84087258ee18>



<https://www.eejournal.com/article/self-driving-cars-what-the-engineers-think/>

Natural language processing

- Machine translation



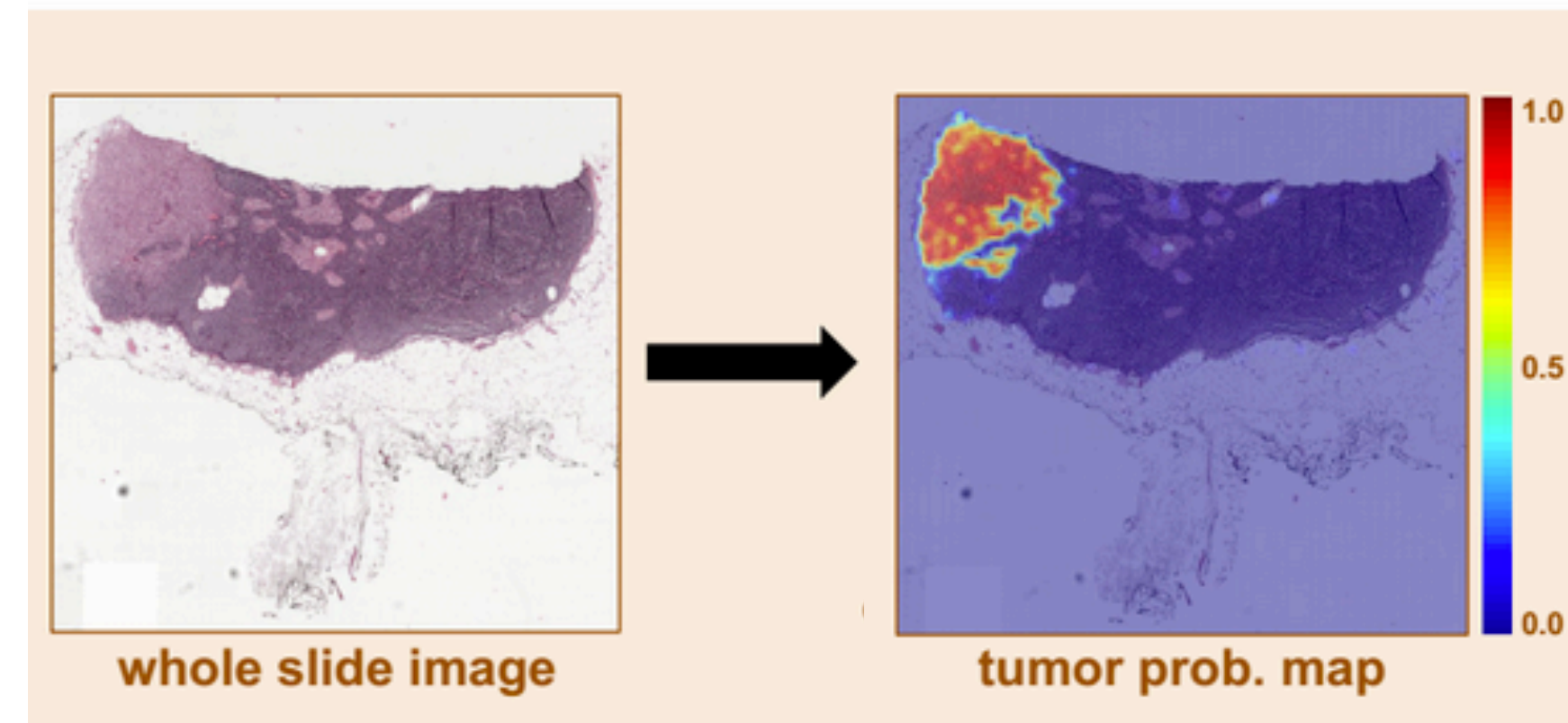
<https://translate.google.com/intl/en/about/>

What is deep learning?

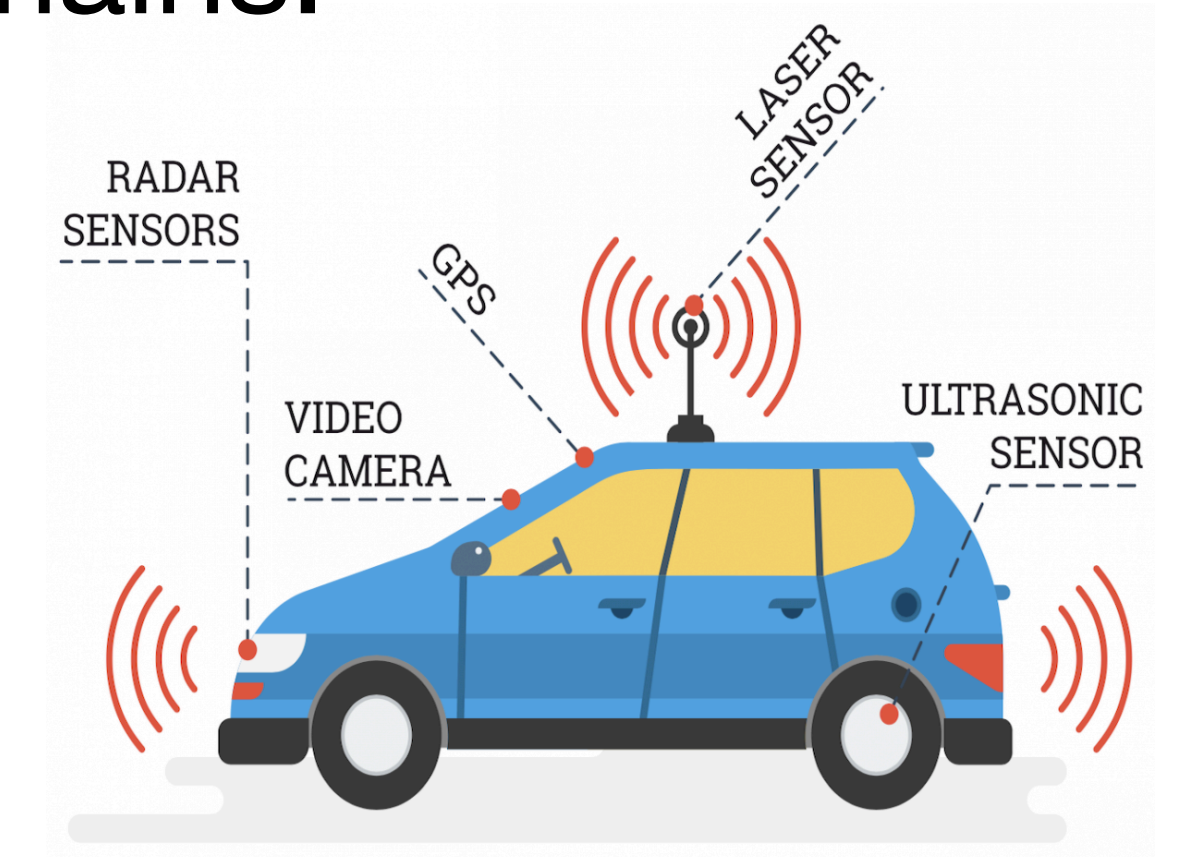
Deep learning is an enormously successful class of predictive models that has achieved state-of-the-art performance across a variety of domains:

Image processing

- Medical image analysis
- Self-driving cars



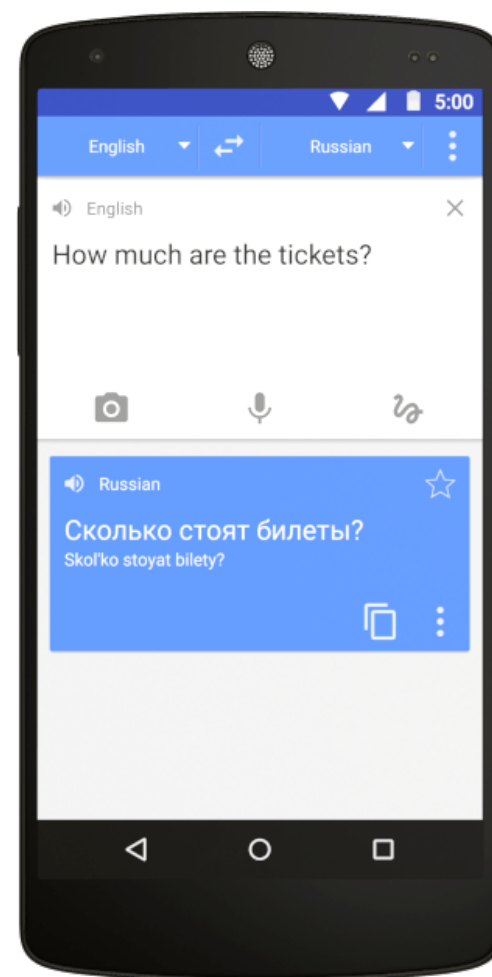
<https://towardsdatascience.com/understanding-cancer-using-machine-learning-84087258ee18>



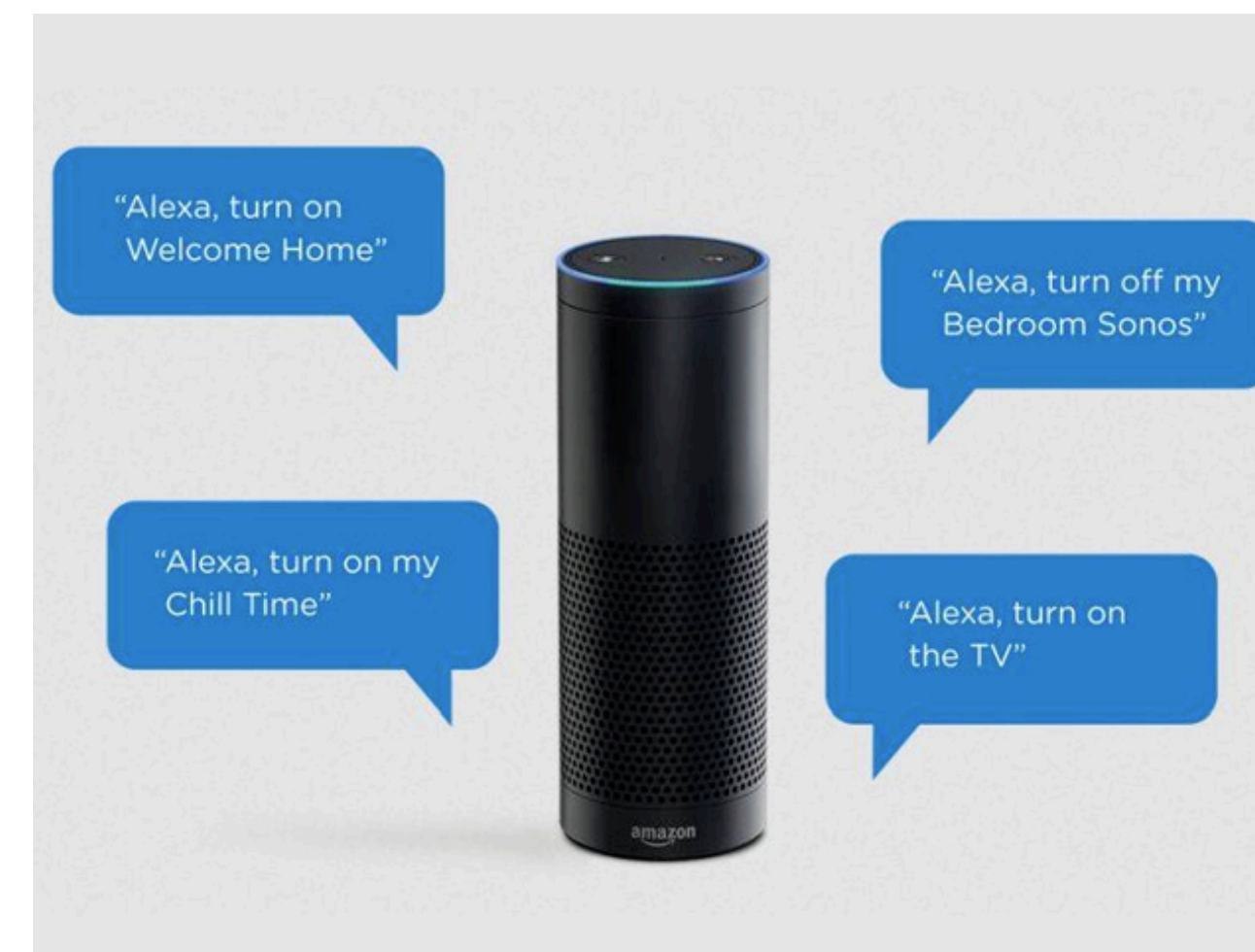
<https://www.eejournal.com/article/self-driving-cars-what-the-engineers-think/>

Natural language processing

- Machine translation
- Speech recognition



<https://translate.google.com/intl/en/about/>



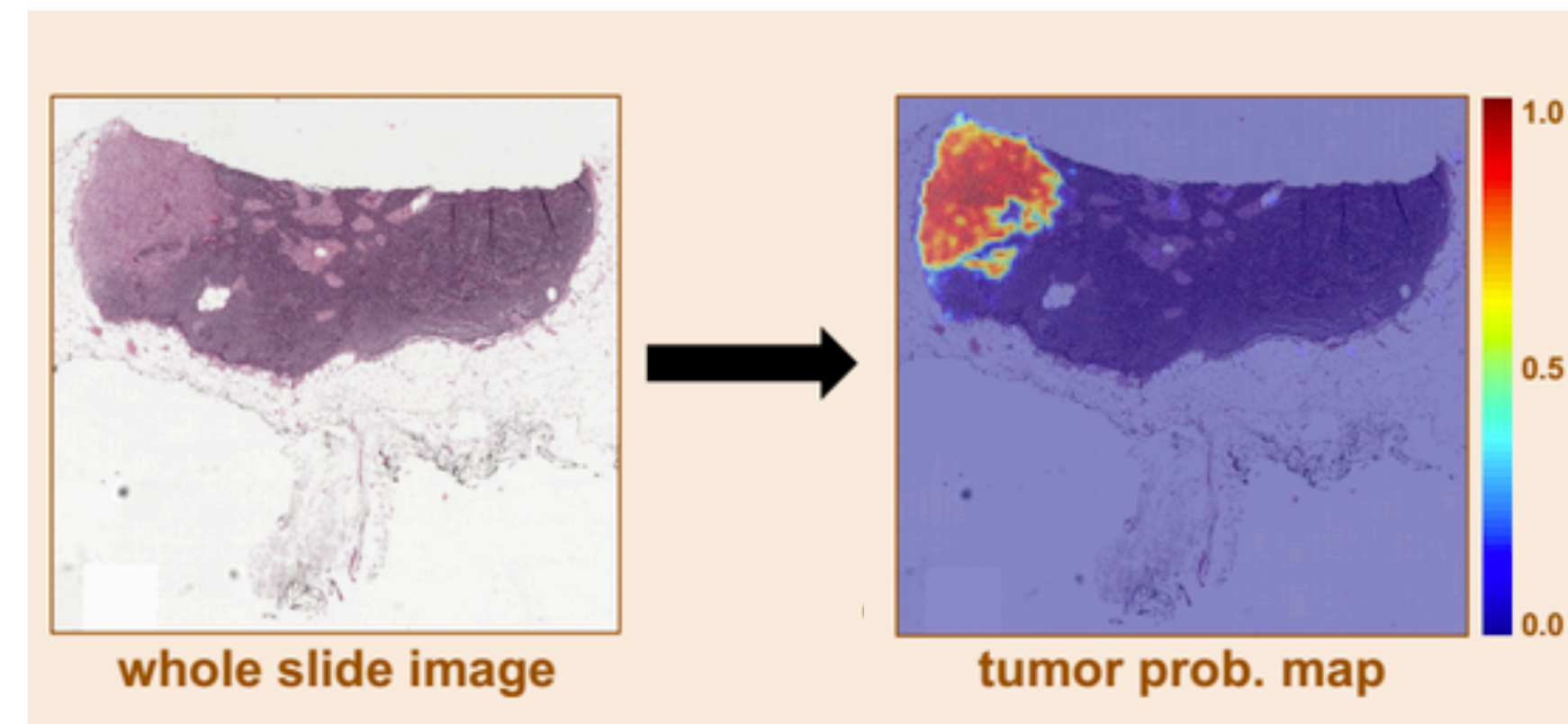
<https://brailleinstitute.org/event/online-introducing-amazon-alexa>

What is deep learning?

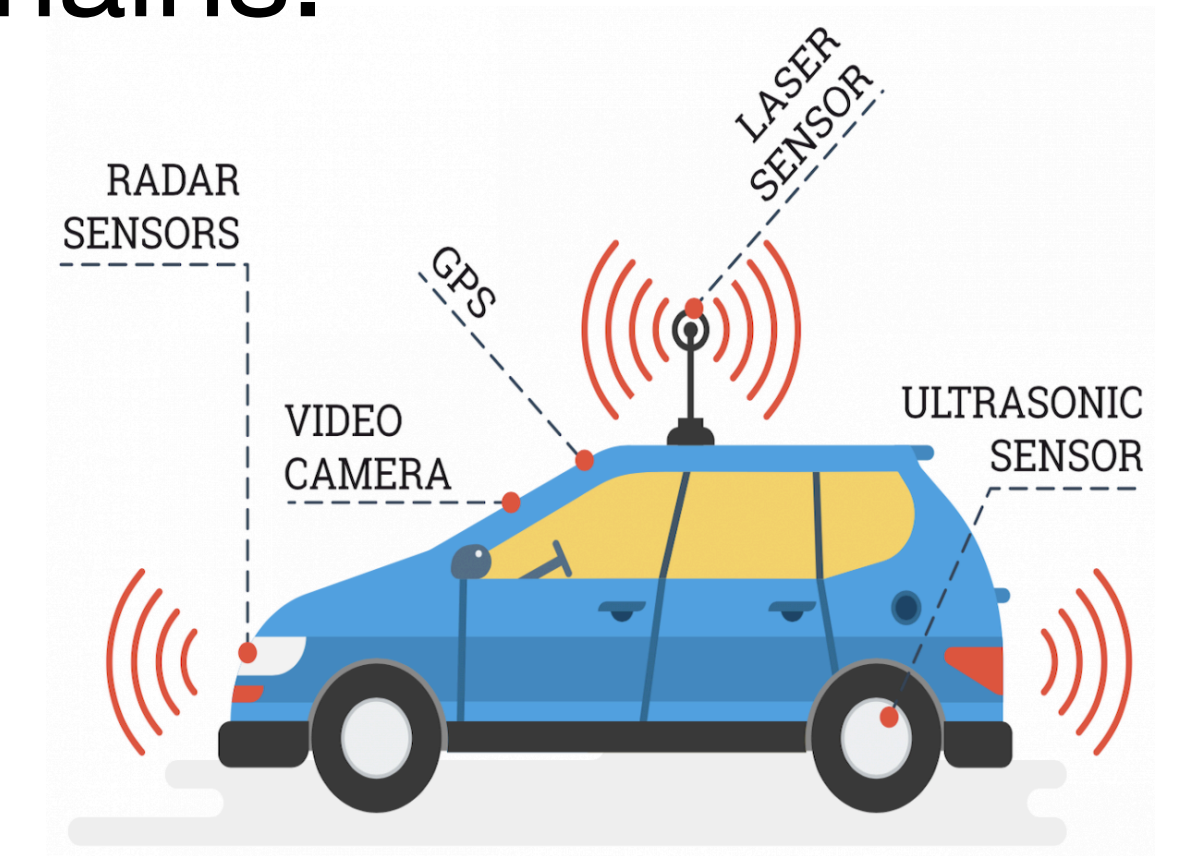
Deep learning is an enormously successful class of predictive models that has achieved state-of-the-art performance across a variety of domains:

Image processing

- Medical image analysis
- Self-driving cars



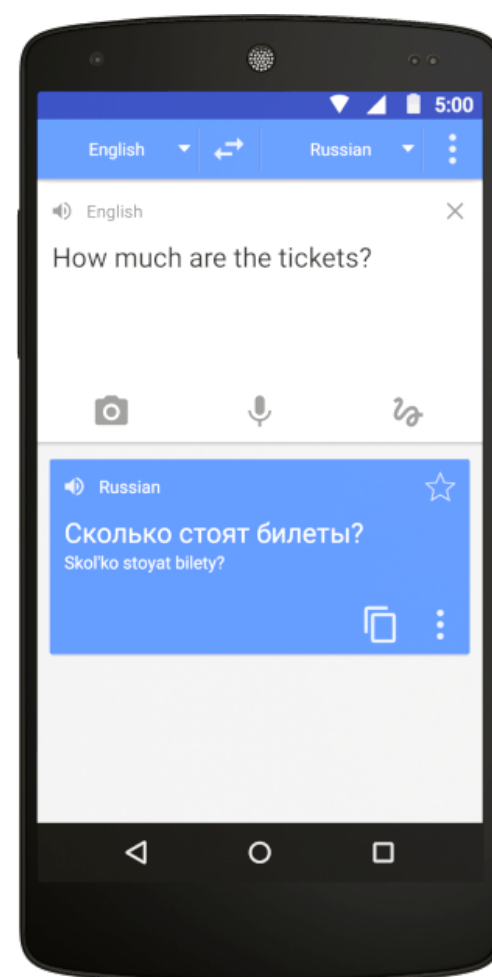
<https://towardsdatascience.com/understanding-cancer-using-machine-learning-84087258ee18>



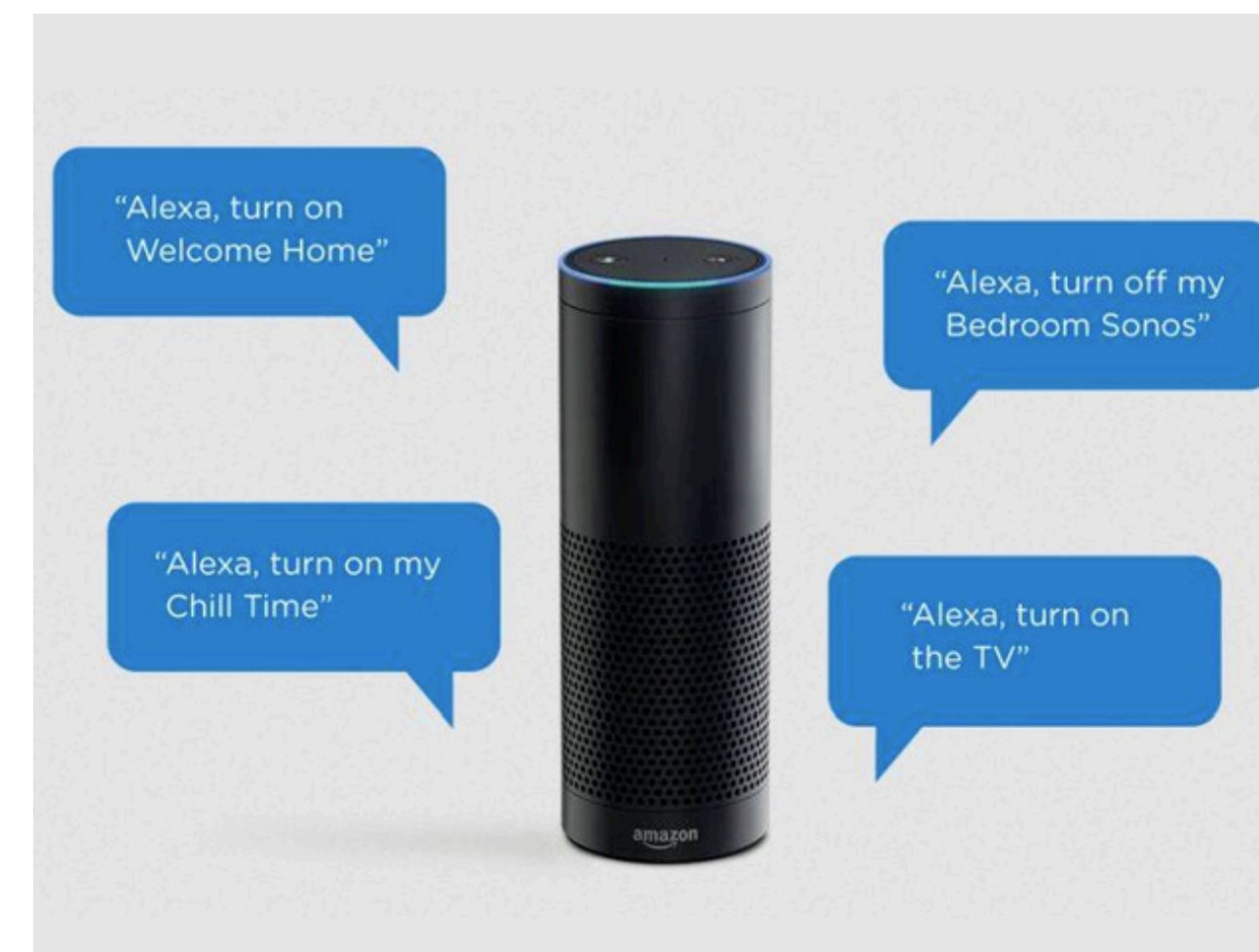
<https://www.eejournal.com/article/self-driving-cars-what-the-engineers-think/>

Natural language processing

- Machine translation
- Speech recognition
- Question answering



<https://translate.google.com/intl/en/about/>



<https://brailleinstitute.org/event/online-introducing-amazon-alexa>

Passage Sentence

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity.

Question

What causes precipitation to fall?

Answer Candidate

gravity

<https://paperswithcode.com/task/question-answering>

Game plan for Unit 5

Game plan for Unit 5

Lecture 1: Deep learning preliminaries

- Predictive models as graphs
- Training via optimization

Game plan for Unit 5

Lecture 1: Deep learning preliminaries

- Predictive models as graphs
- Training via optimization

Lecture 2: Neural networks

- Multi-layer predictive models
- Stochastic gradient descent

Game plan for Unit 5

Lecture 1: Deep learning preliminaries

- Predictive models as graphs
- Training via optimization

Lecture 2: Neural networks

- Multi-layer predictive models
- Stochastic gradient descent

Lecture 3: Deep learning for images

- Image classification
- Convolutional neural networks

Game plan for Unit 5

Lecture 1: Deep learning preliminaries

- Predictive models as graphs
- Training via optimization

Lecture 2: Neural networks

- Multi-layer predictive models
- Stochastic gradient descent

Lecture 3: Deep learning for images

- Image classification
- Convolutional neural networks

Lecture 4: Deep learning for text

- Document classification
- Recurrent neural networks

Game plan for Unit 5

Lecture 1: Deep learning preliminaries

- Predictive models as graphs
- Training via optimization

Lecture 2: Neural networks

- Multi-layer predictive models
- Stochastic gradient descent

Lecture 3: Deep learning for images

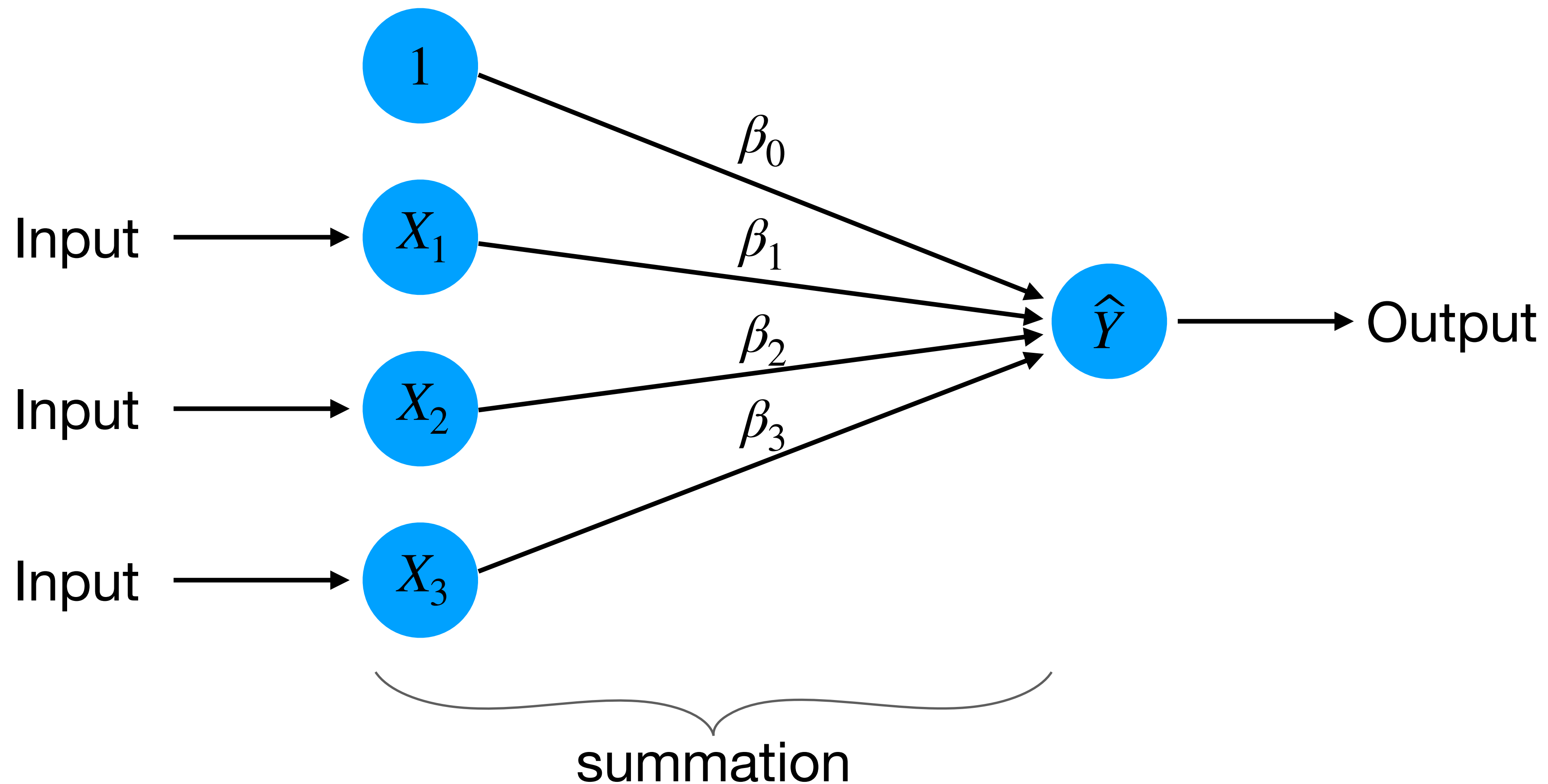
- Image classification
- Convolutional neural networks

Lecture 4: Deep learning for text

- Document classification
- Recurrent neural networks

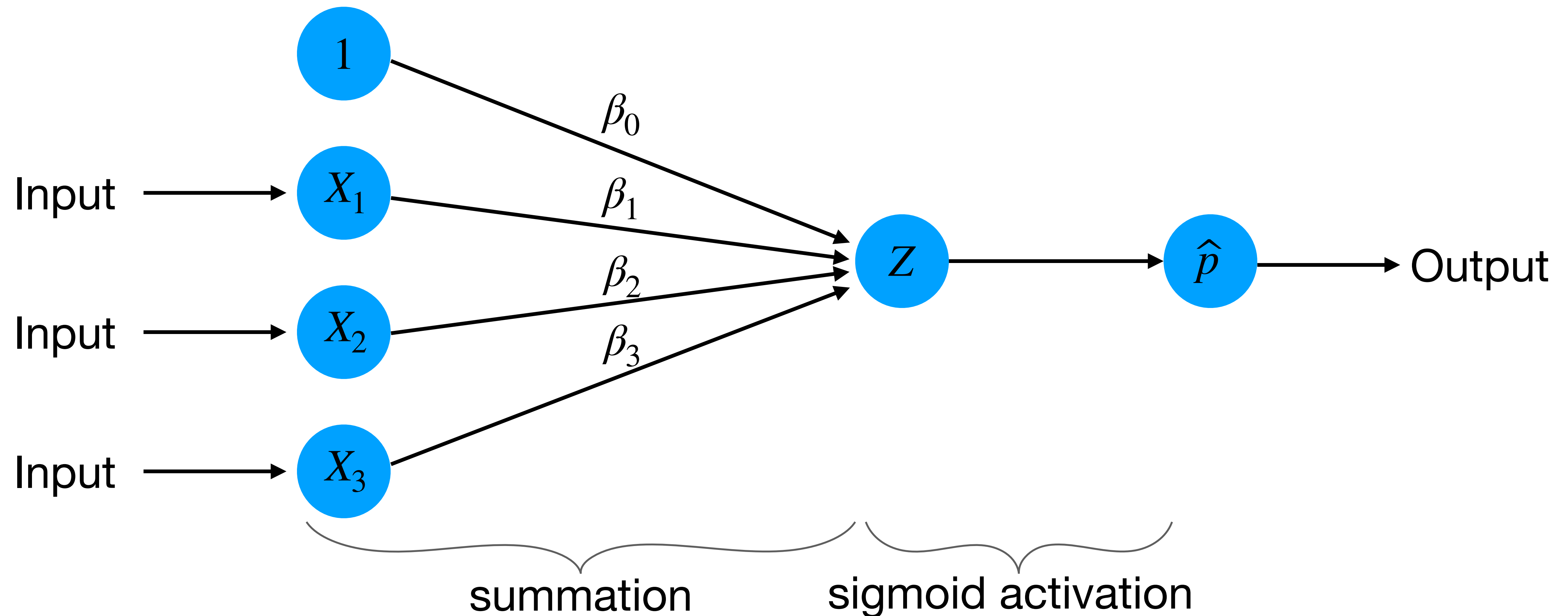
Models as graphs: Linear regression

$$\hat{Y} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3$$



Models as graphs: Logistic model

$$Z = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3; \quad \hat{p} = \text{logistic}(Z) = \frac{e^Z}{1 + e^Z}$$



Models as graphs: Multi-class logistic model

Suppose the response has more than two levels.



Models as graphs: Multi-class logistic model

Suppose the response has more than two levels.



Image is **flattened** to get
vector of input features

Models as graphs: Multi-class logistic model

Suppose the response has more than two levels.

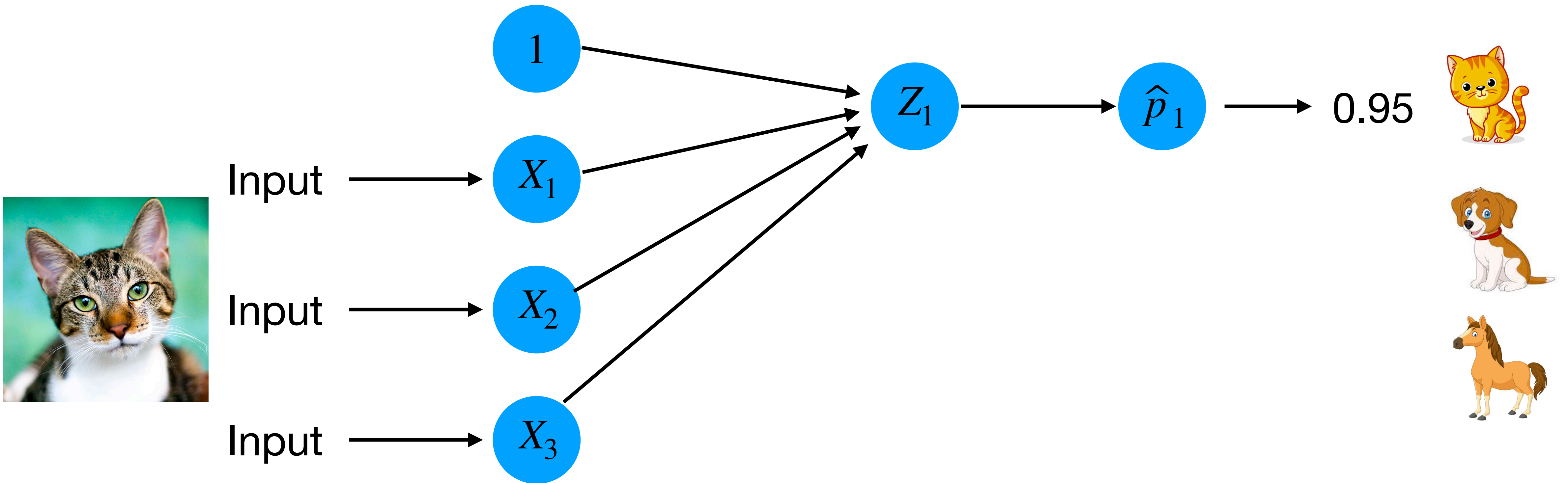


Image is flattened to get
vector of input features

Models as graphs: Multi-class logistic model

Suppose the response has more than two levels.

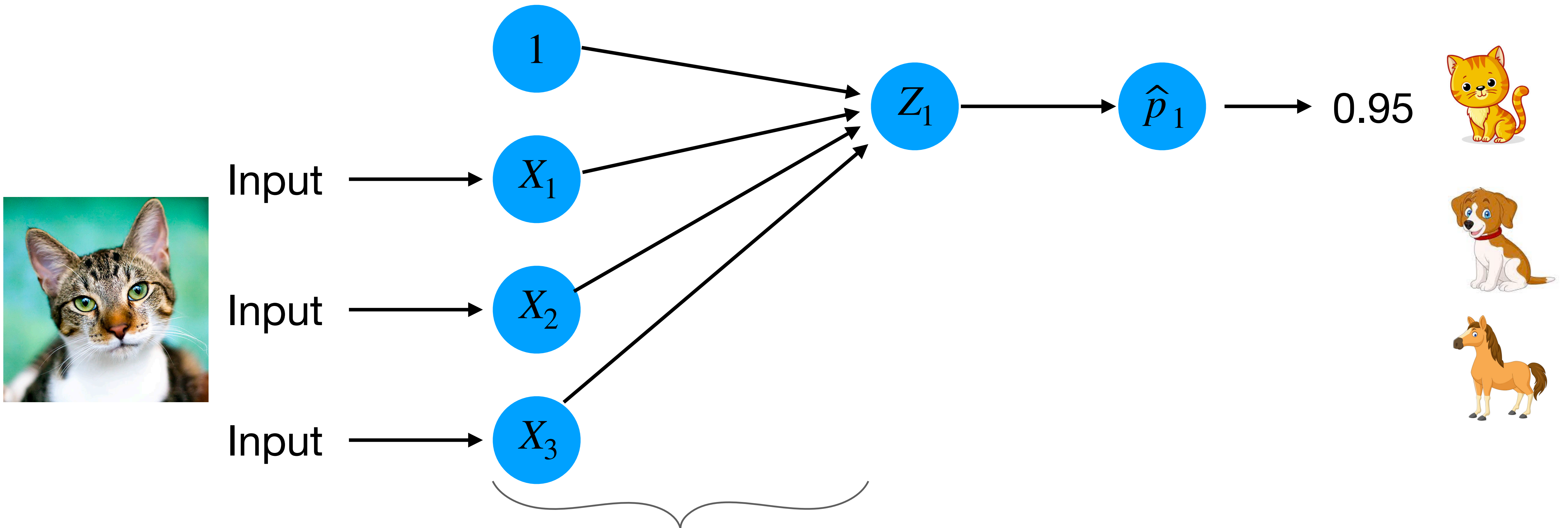


Image is flattened to get vector of input features

Models as graphs: Multi-class logistic model

Suppose the response has more than two levels.

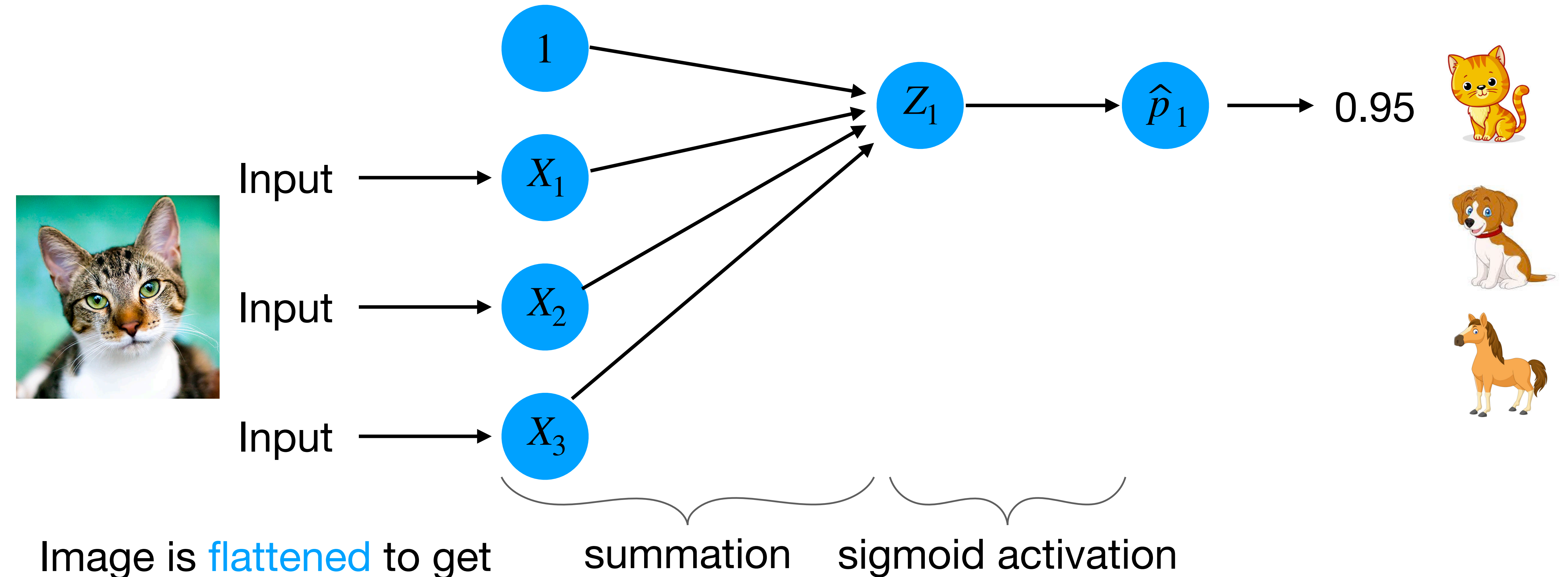


Image is **flattened** to get
vector of input features

Models as graphs: Multi-class logistic model

Suppose the response has more than two levels.

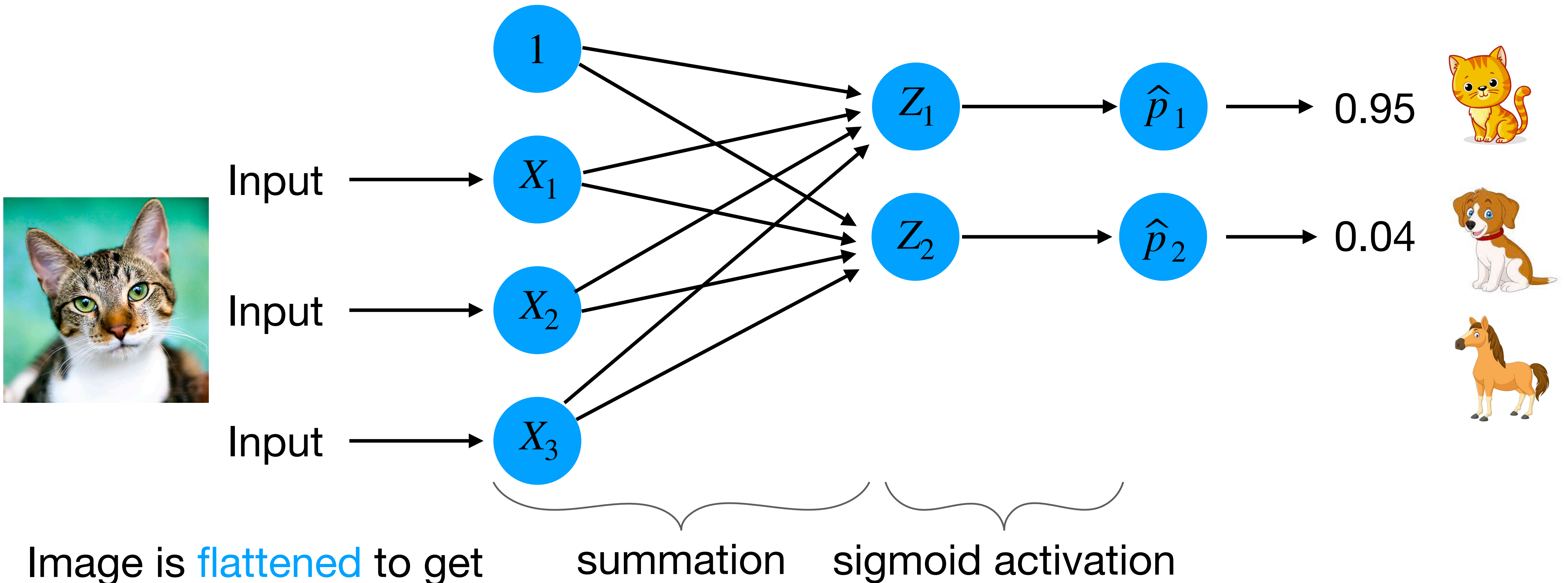


Image is **flattened** to get
vector of input features

Models as graphs: Multi-class logistic model

Suppose the response has more than two levels.

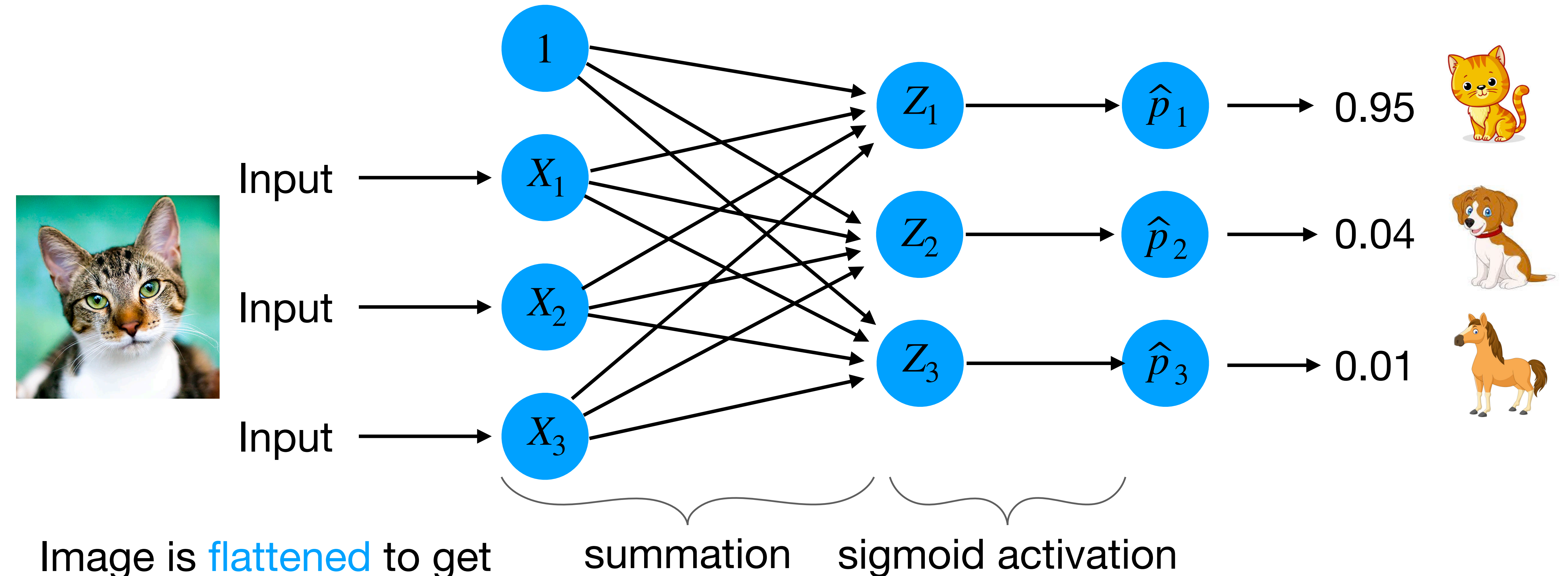


Image is flattened to get vector of input features

Models as graphs: Multi-class logistic model

Suppose the response has more than two levels.

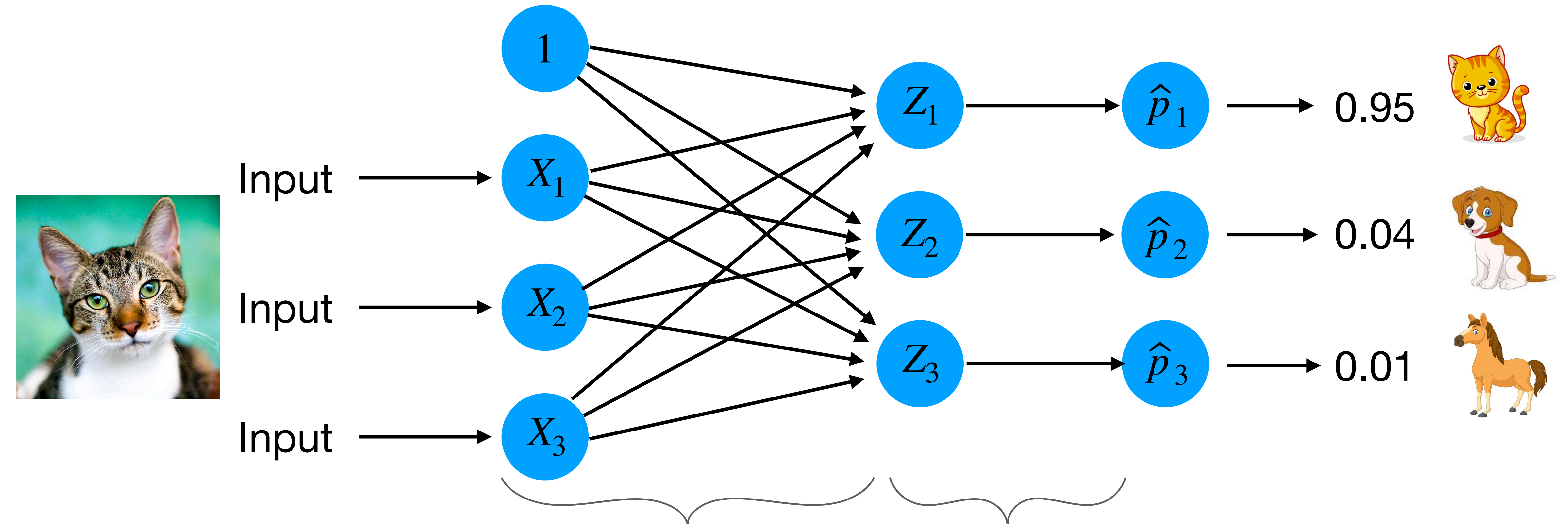
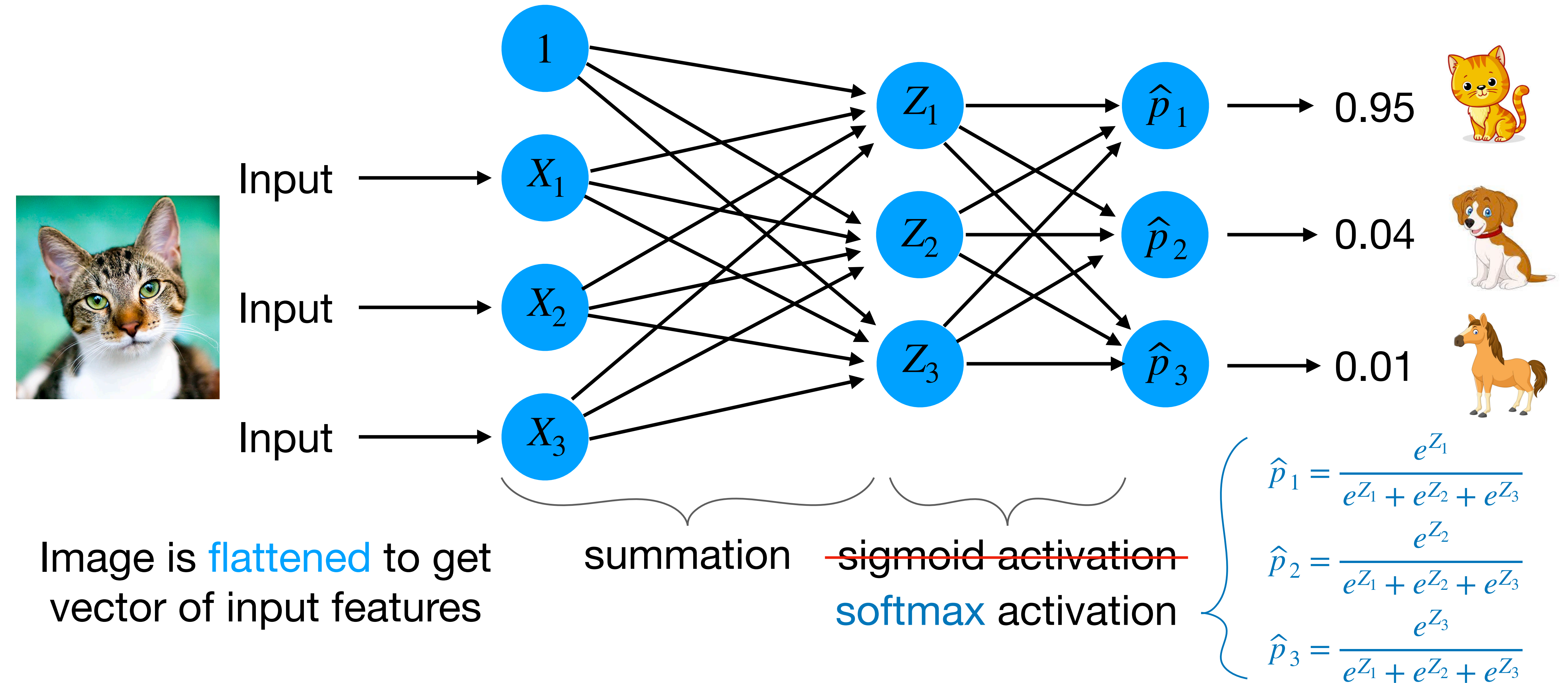


Image is flattened to get vector of input features

summation ~~sigmoid activation~~

Models as graphs: Multi-class logistic model

Suppose the response has more than two levels.



The cross-entropy loss function

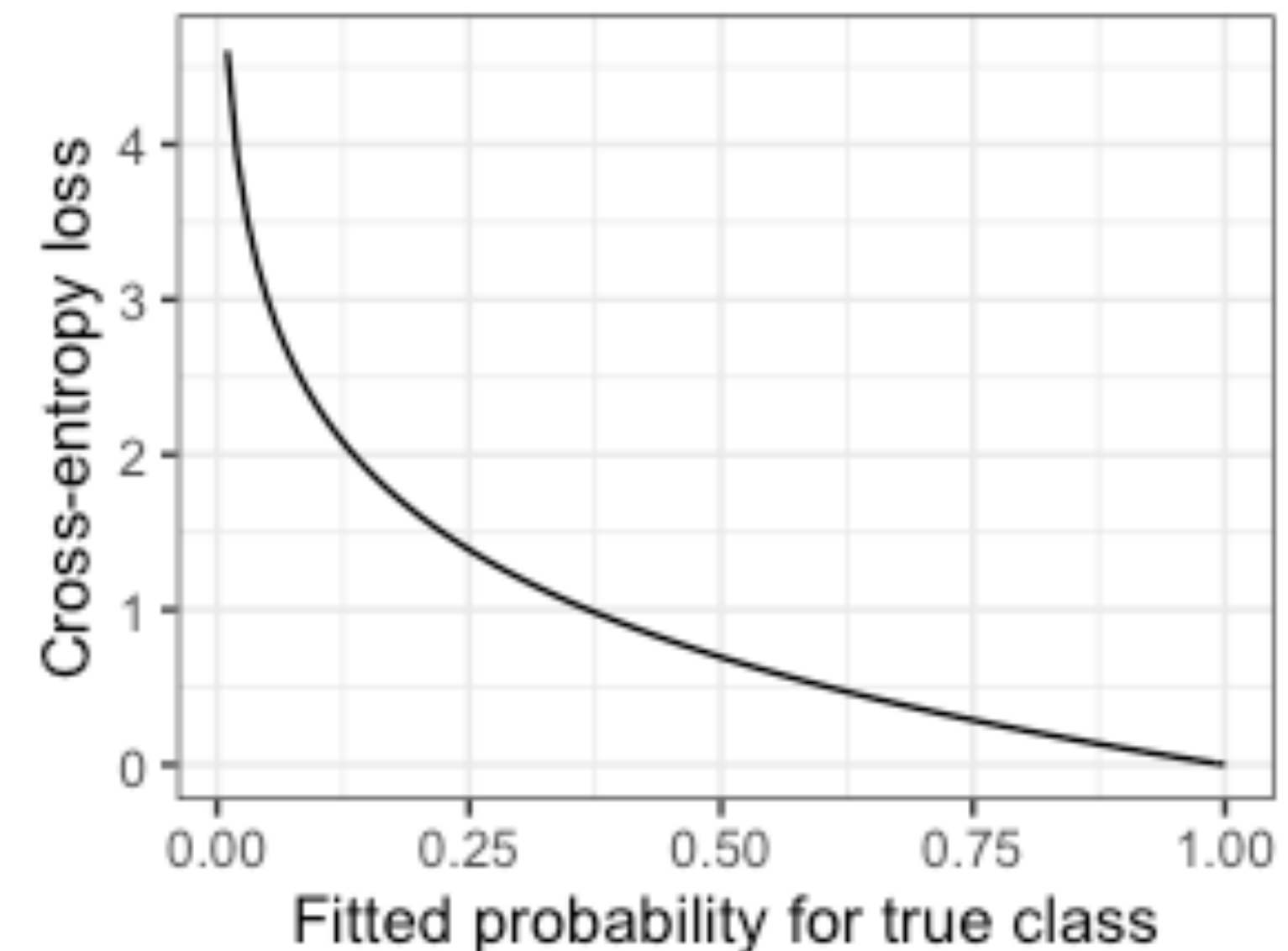
Suppose we have a true label Y and fitted probabilities $\hat{p}_1, \hat{p}_2, \hat{p}_3$. Define

$$\text{cross-entropy loss } L(Y, \hat{p}) = \begin{cases} -\log(\hat{p}_1) & \text{if } Y = 1; \\ -\log(\hat{p}_2) & \text{if } Y = 2; \\ -\log(\hat{p}_3) & \text{if } Y = 3. \end{cases}$$

The cross-entropy loss function

Suppose we have a true label Y and fitted probabilities $\hat{p}_1, \hat{p}_2, \hat{p}_3$. Define

$$\text{cross-entropy loss } L(Y, \hat{p}) = \begin{cases} -\log(\hat{p}_1) & \text{if } Y = 1; \\ -\log(\hat{p}_2) & \text{if } Y = 2; \\ -\log(\hat{p}_3) & \text{if } Y = 3. \end{cases}$$

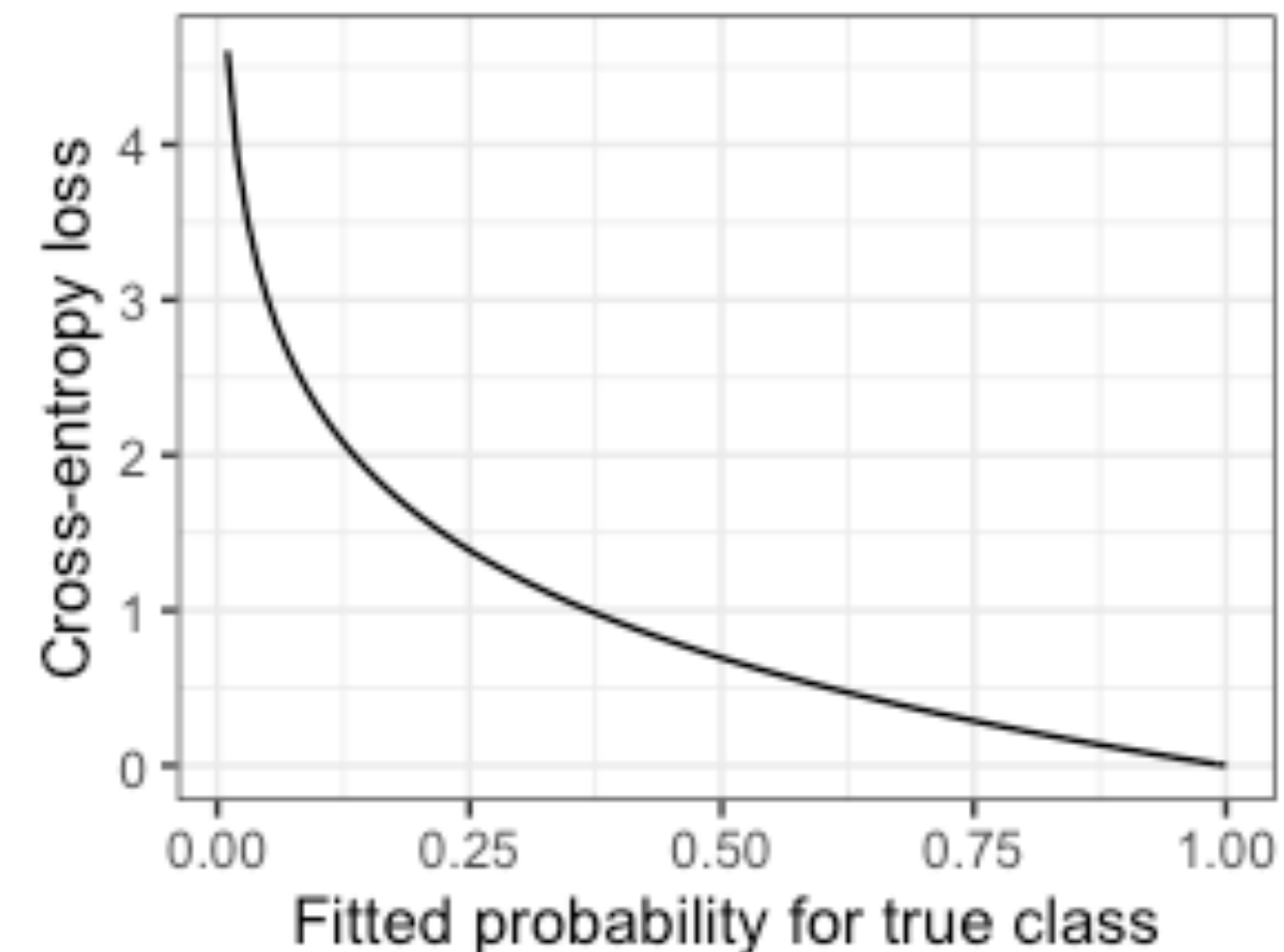


The cross-entropy loss function

Suppose we have a true label Y and fitted probabilities $\hat{p}_1, \hat{p}_2, \hat{p}_3$. Define

$$\text{cross-entropy loss } L(Y, \hat{p}) = \begin{cases} -\log(\hat{p}_1) & \text{if } Y = 1; \\ -\log(\hat{p}_2) & \text{if } Y = 2; \\ -\log(\hat{p}_3) & \text{if } Y = 3. \end{cases}$$

Greater probability attached to true class
→ smaller cross-entropy loss.



Training predictive models via optimization

Training predictive models via optimization

Define class of predictive models $f_{\beta}(X)$ indexed by some parameter vector β .

Training predictive models via optimization

Define class of predictive models $f_{\beta}(X)$ indexed by some parameter vector β .

Find member of this class that best fits the training data, as measured by the loss function L of predictions given true responses, possibly regularized:

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

Training predictive models via optimization

Define class of predictive models $f_{\beta}(X)$ indexed by some parameter vector β .

Find member of this class that best fits the training data, as measured by the loss function L of predictions given true responses, possibly regularized:

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

Training predictive models via optimization

Define **class of predictive models** $f_\beta(X)$ indexed by some parameter vector β .

Find member of this class that best fits the training data, as measured by the **loss function** L of predictions given true responses, possibly regularized:

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

For example, ridge regression has

$$L(Y_i, \hat{Y}_i) = (Y_i - \hat{Y}_i)^2; \quad f_\beta(X) = \beta_0 X_0 + \cdots + \beta_{p-1} X_{p-1}; \quad \text{penalty}(\beta) = \sum_{j=1}^{p-1} \beta_j^2.$$

Training predictive models via optimization

Define **class of predictive models** $f_{\beta}(X)$ indexed by some parameter vector β .

Find member of this class that best fits the training data, as measured by the **loss function** L of predictions given true responses, possibly regularized:

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

For example, ridge regression has

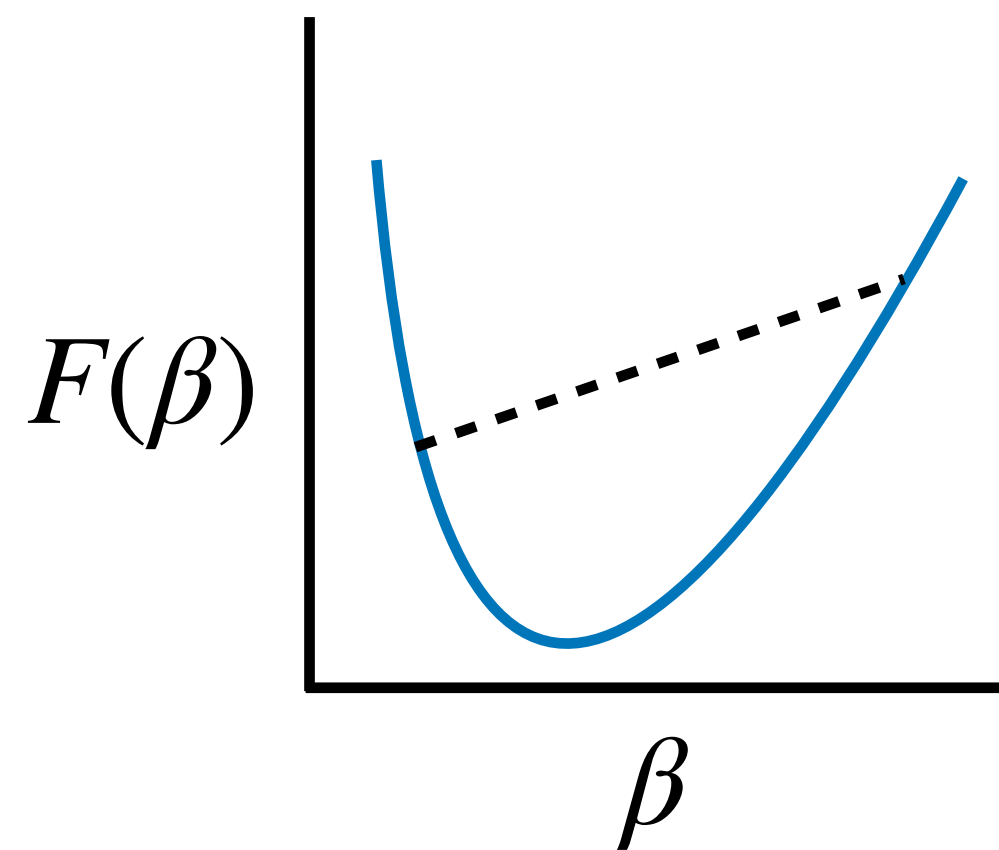
$$L(Y_i, \hat{Y}_i) = (Y_i - \hat{Y}_i)^2; \quad f_{\beta}(X) = \beta_0 X_0 + \cdots + \beta_{p-1} X_{p-1}; \quad \text{penalty}(\beta) = \sum_{j=1}^{p-1} \beta_j^2.$$

Training predictive models = solving optimization problems.

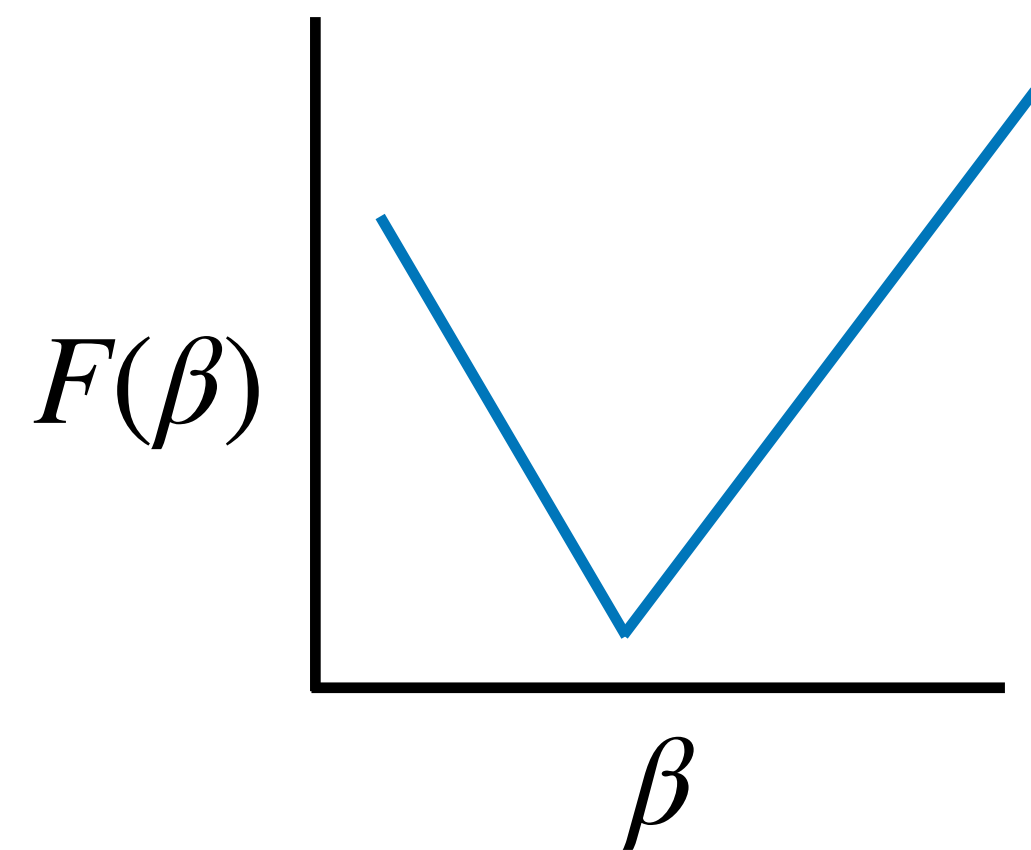
Convexity: A crucial property of F

The hardness of the optimization problem $\arg \min F(\beta)$ depends crucially on whether the objective function F is **convex**, or “bowl-shaped.”

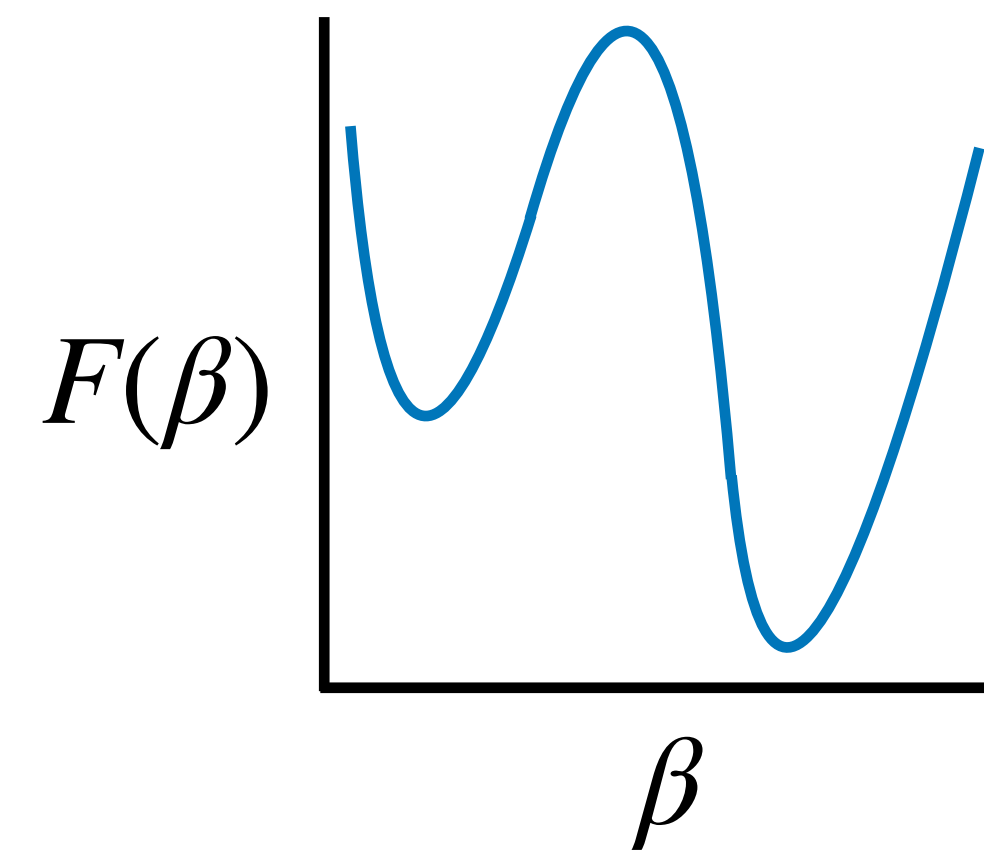
Convex



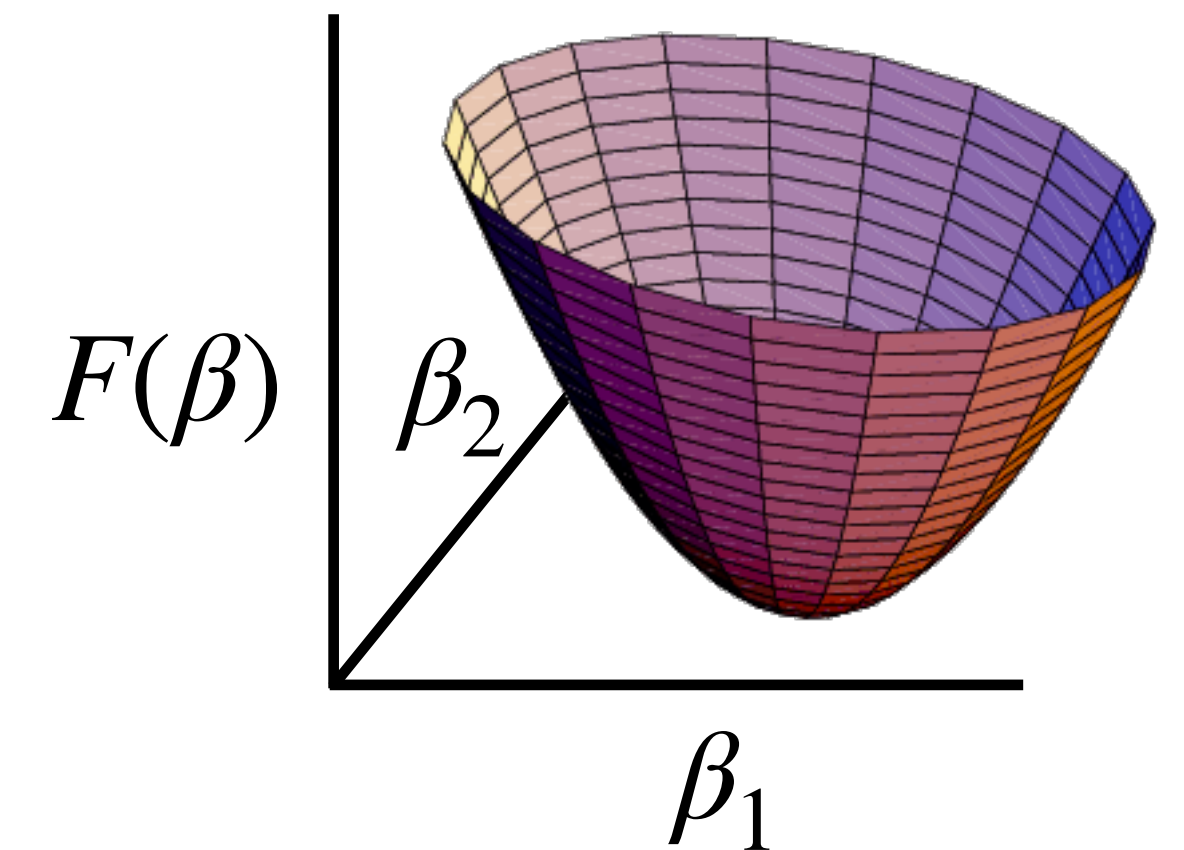
Convex



Non-convex

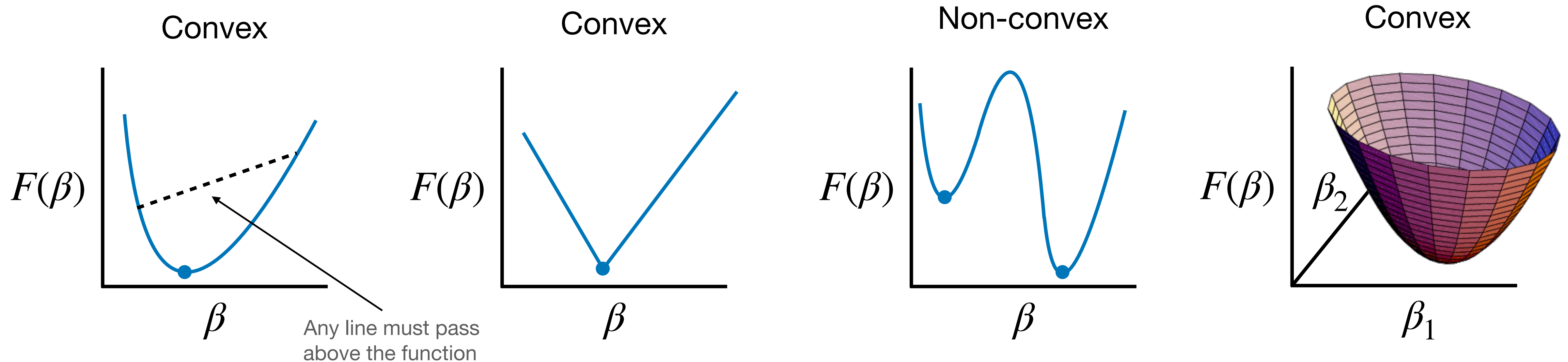


Convex



Convexity: A crucial property of F

The hardness of the optimization problem $\arg \min F(\beta)$ depends crucially on whether the objective function F is **convex**, or “bowl-shaped.”



For convex functions, any local minimum must also be a global minimum.

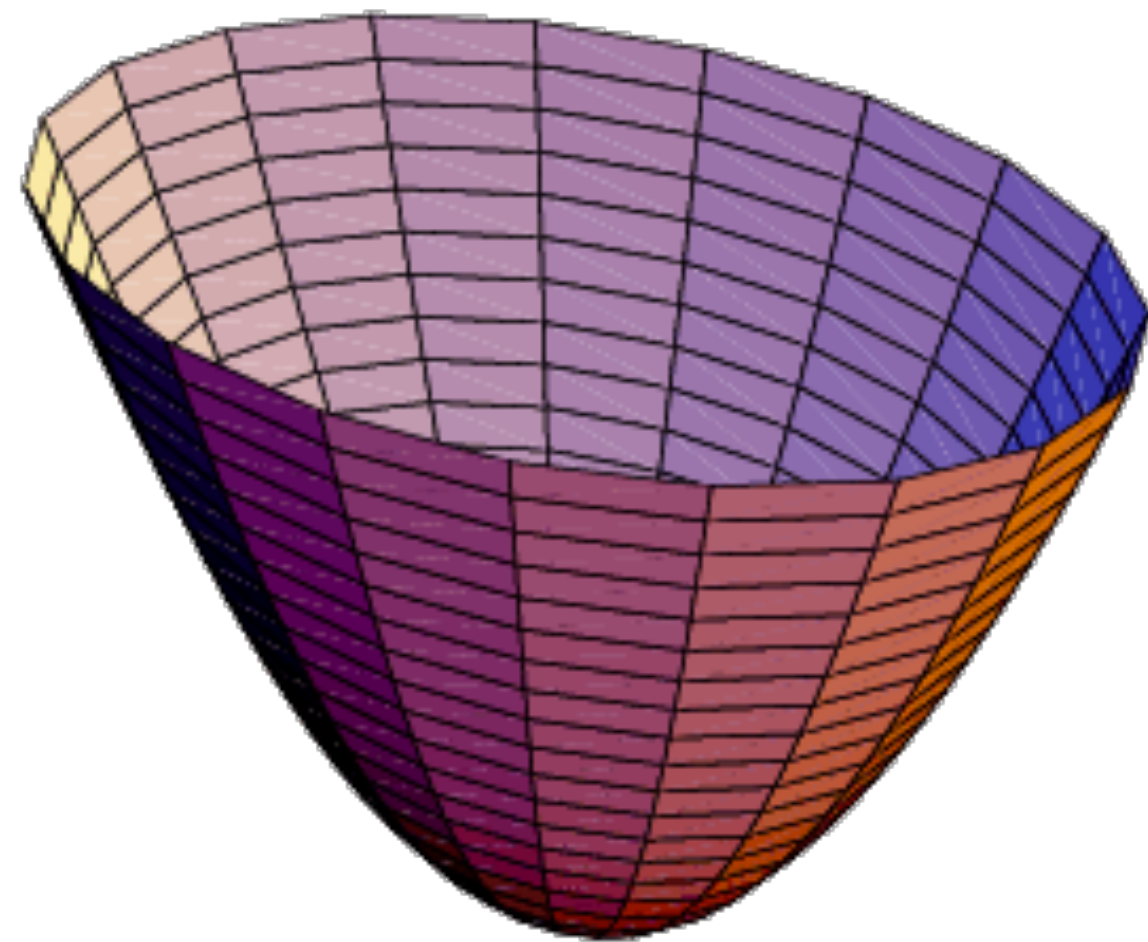
It is much easier to find local minima than global minima.

Which methods have convex objectives?

Which methods have convex objectives?

Convex

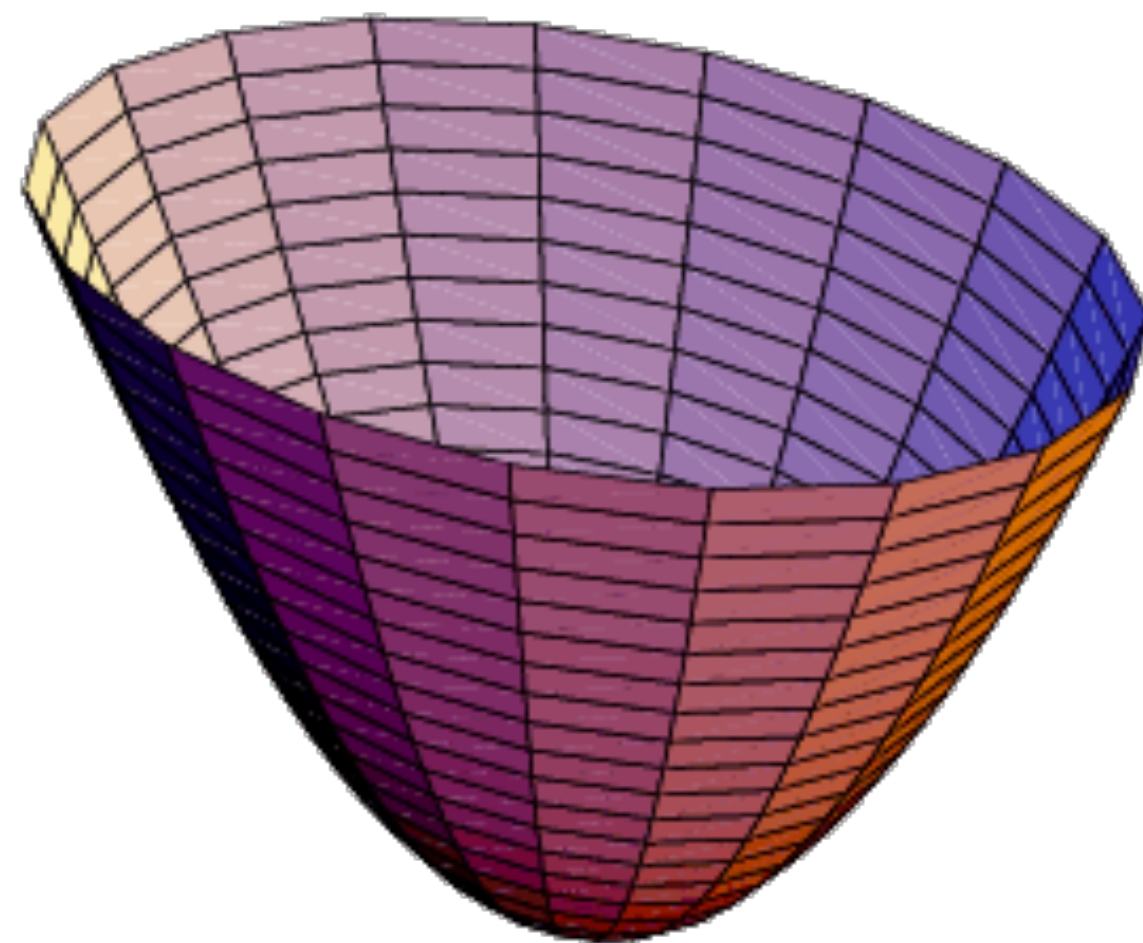
- Linear and logistic regression
- Linear and logistic regression with ridge or lasso penalties



Which methods have convex objectives?

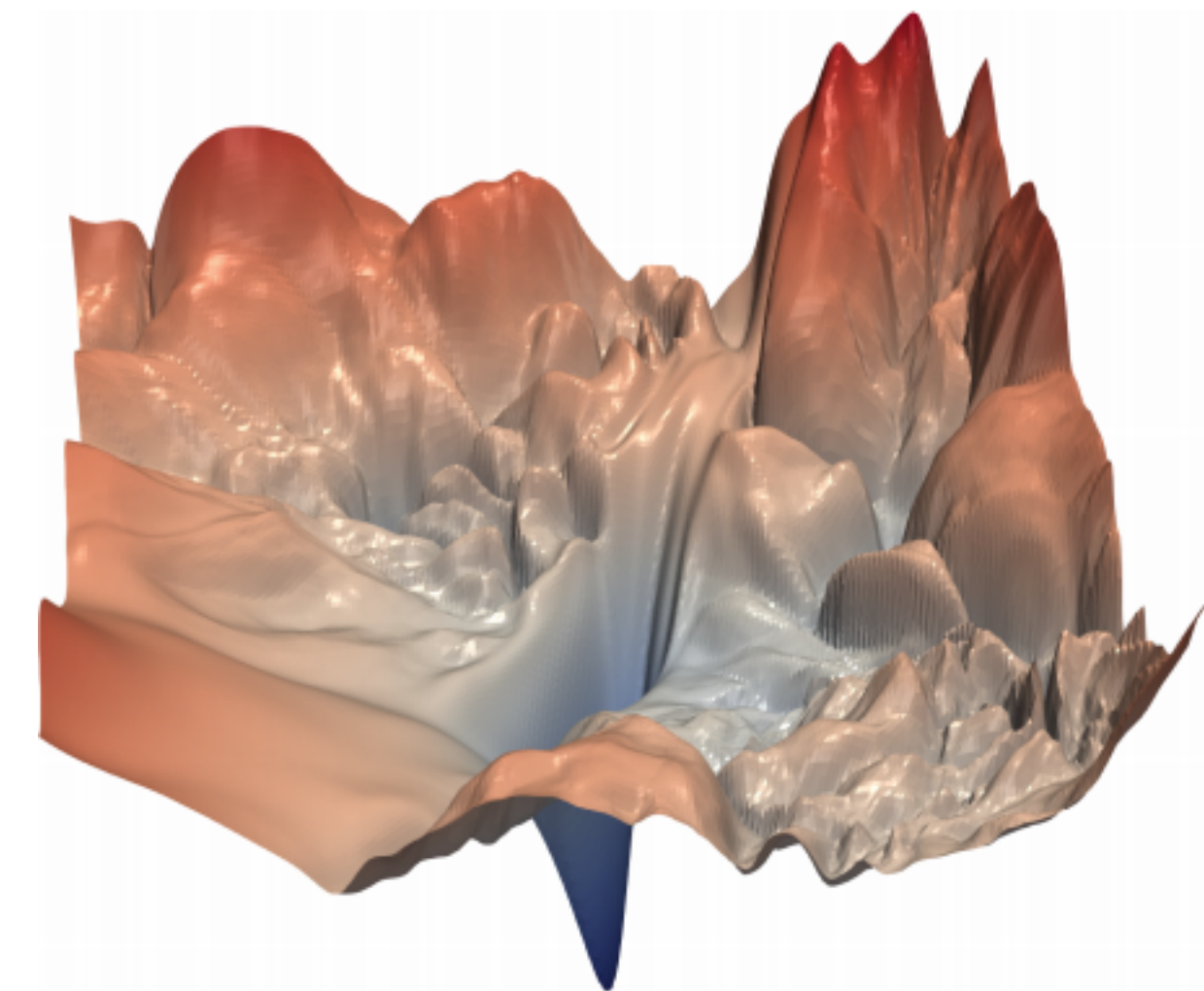
Convex

- Linear and logistic regression
- Linear and logistic regression with ridge or lasso penalties



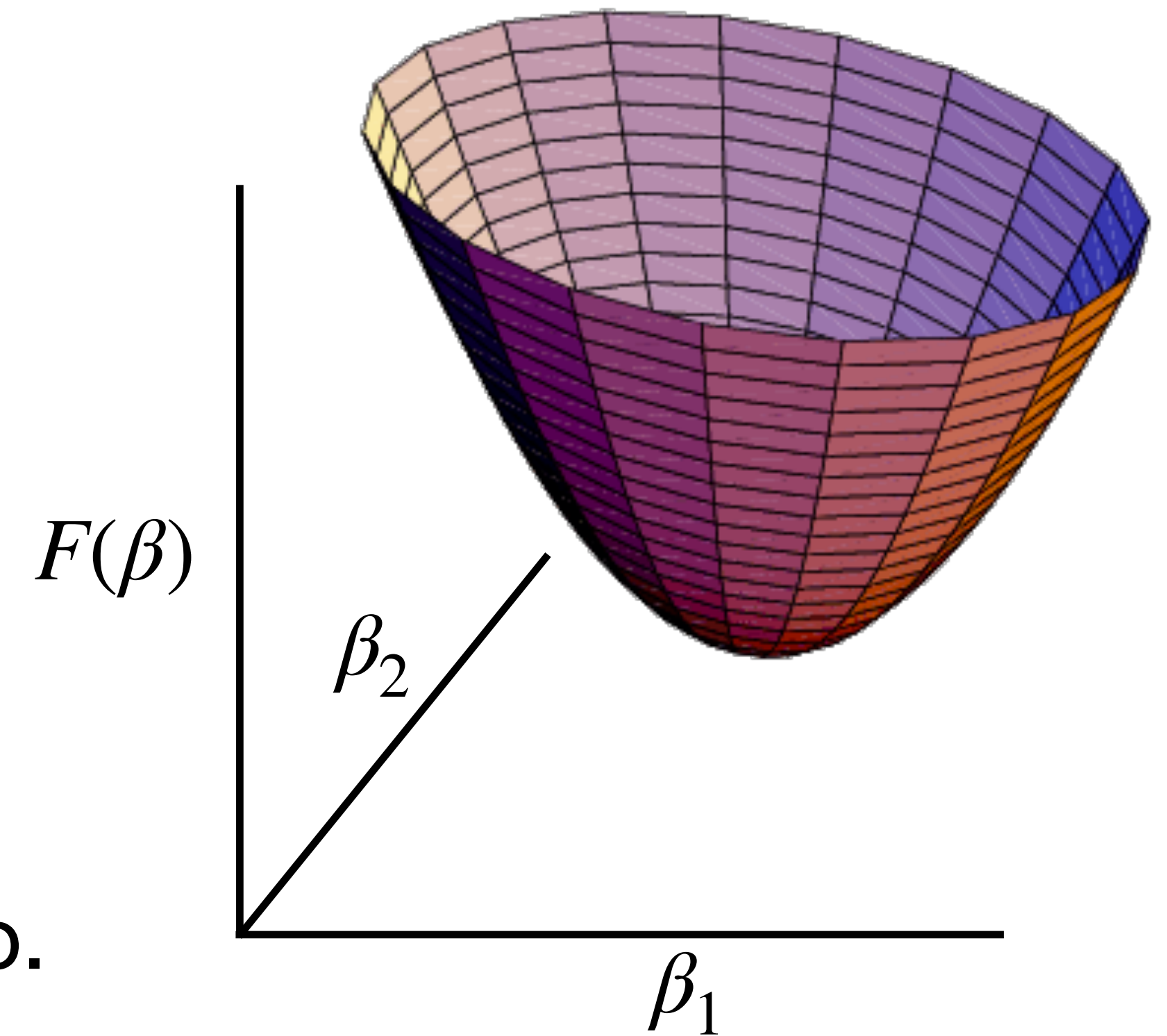
Not convex

- Tree-based methods
- Neural networks



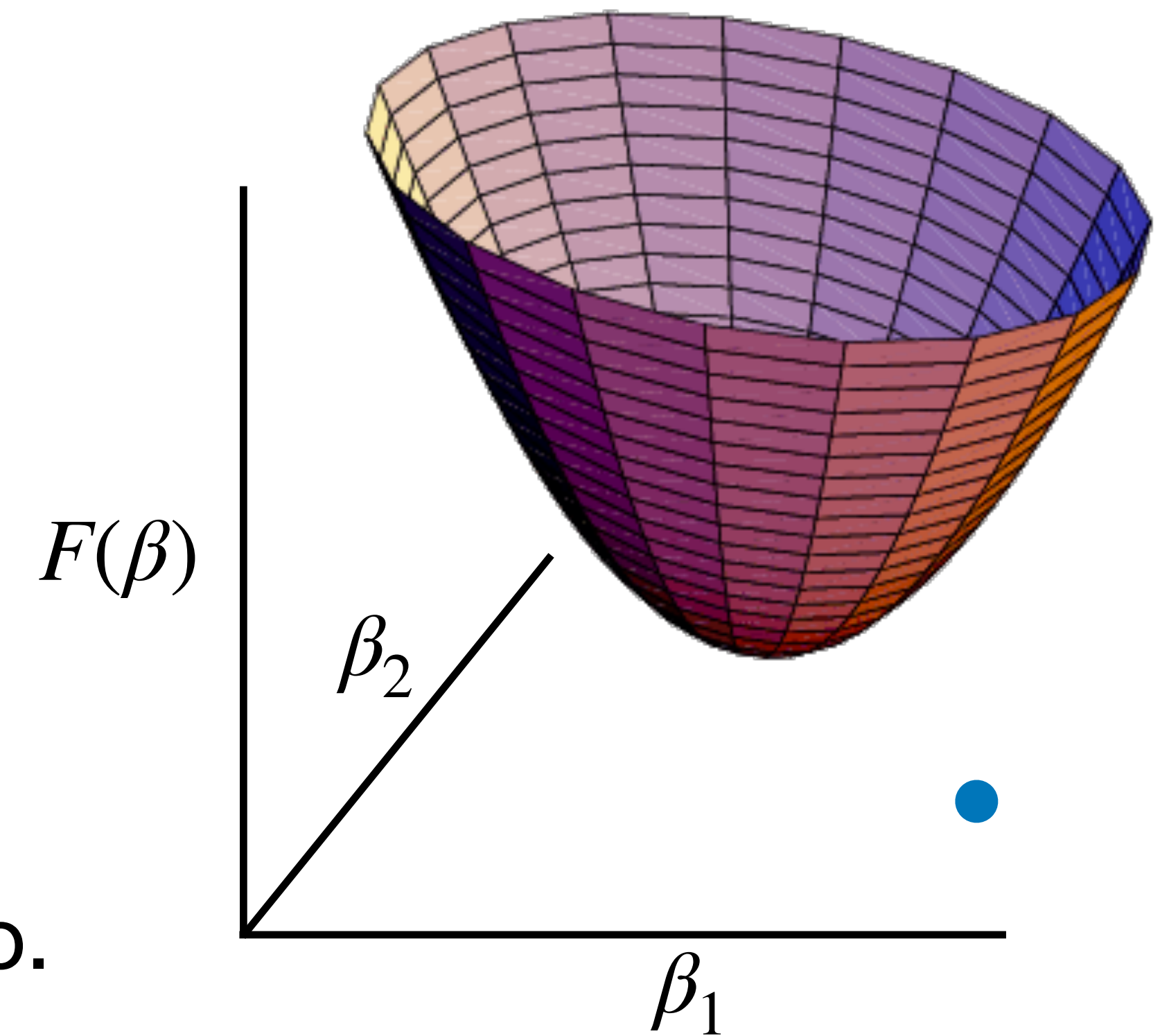
Gradient descent

1. Choose some initial value of β .
2. Evaluate the **gradient** $\nabla F(\beta)$ at that point; it is the direction in which F increases the fastest. The **negative gradient** is the direction in which F decreases the fastest.
3. Take small step in negative gradient direction:
 $\beta \leftarrow \beta - \gamma \nabla F(\beta)$; γ called the **learning rate**.
4. Repeat steps 2 and 3 until gradient is near zero.



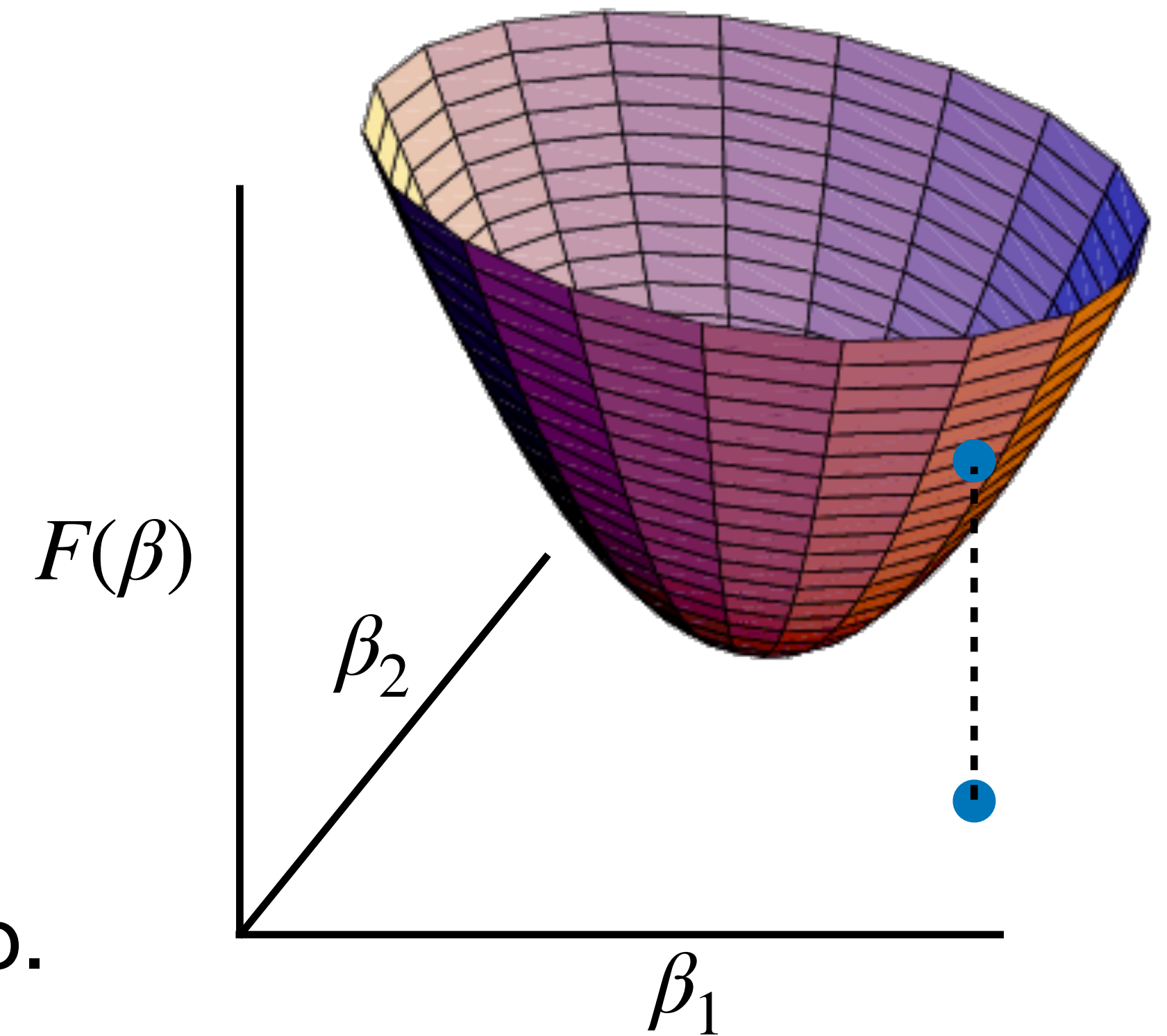
Gradient descent

1. Choose some initial value of β .
2. Evaluate the **gradient** $\nabla F(\beta)$ at that point; it is the direction in which F increases the fastest. The **negative gradient** is the direction in which F decreases the fastest.
3. Take small step in negative gradient direction:
 $\beta \leftarrow \beta - \gamma \nabla F(\beta)$; γ called the **learning rate**.
4. Repeat steps 2 and 3 until gradient is near zero.



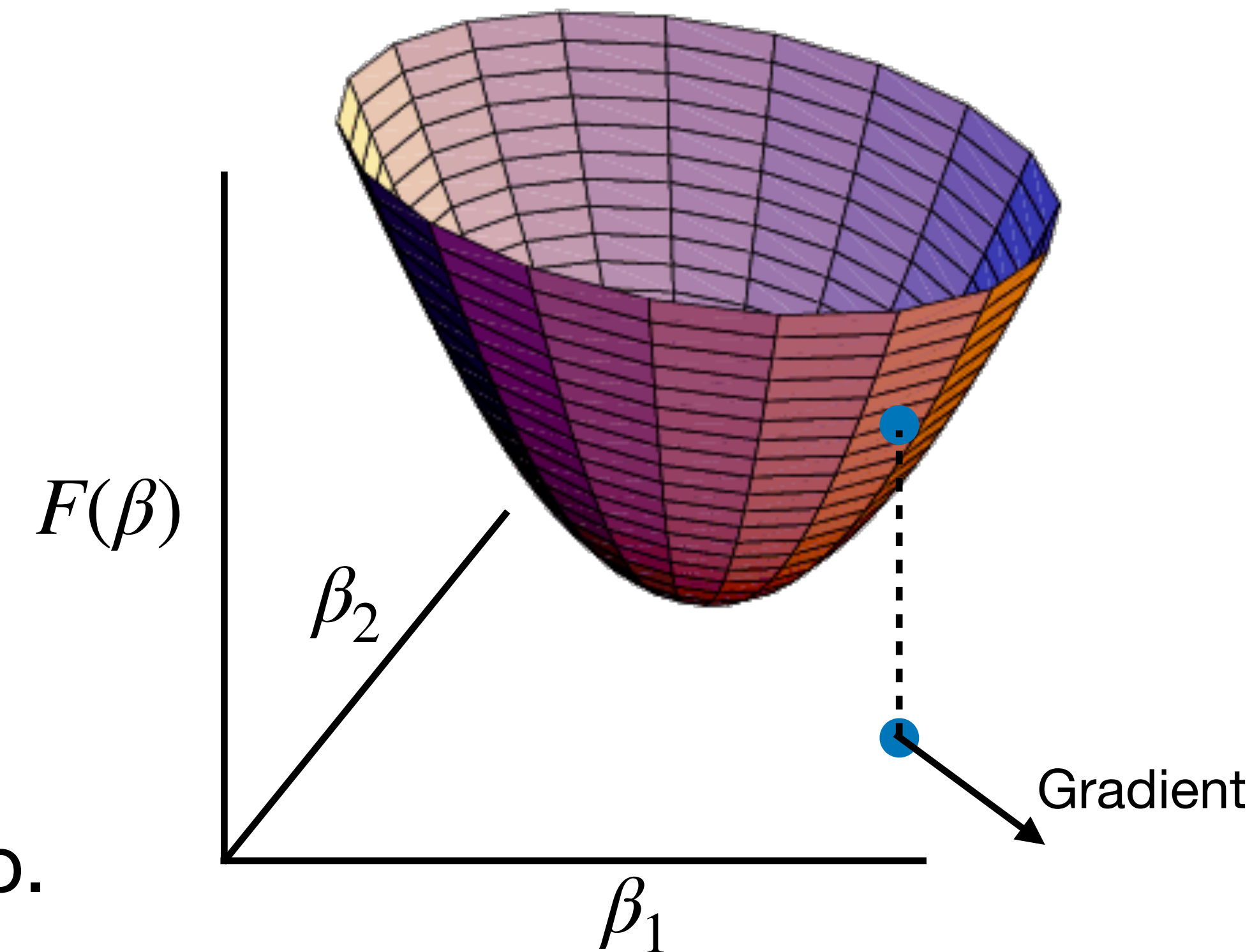
Gradient descent

1. Choose some initial value of β .
2. Evaluate the **gradient** $\nabla F(\beta)$ at that point; it is the direction in which F increases the fastest. The **negative gradient** is the direction in which F decreases the fastest.
3. Take small step in negative gradient direction:
 $\beta \leftarrow \beta - \gamma \nabla F(\beta)$; γ called the **learning rate**.
4. Repeat steps 2 and 3 until gradient is near zero.



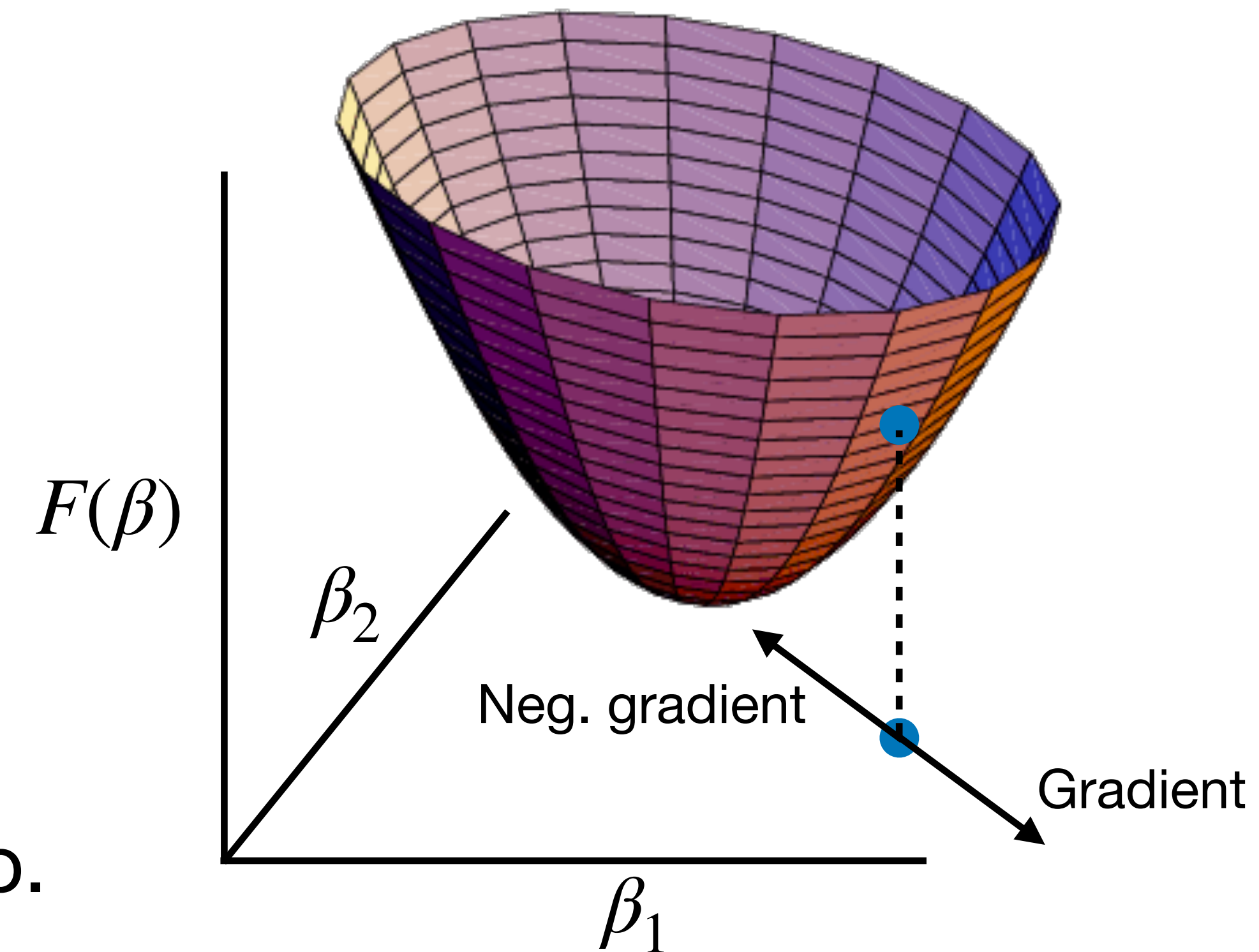
Gradient descent

1. Choose some initial value of β .
2. Evaluate the **gradient** $\nabla F(\beta)$ at that point; it is the direction in which F increases the fastest. The **negative gradient** is the direction in which F decreases the fastest.
3. Take small step in negative gradient direction:
 $\beta \leftarrow \beta - \gamma \nabla F(\beta)$; γ called the **learning rate**.
4. Repeat steps 2 and 3 until gradient is near zero.



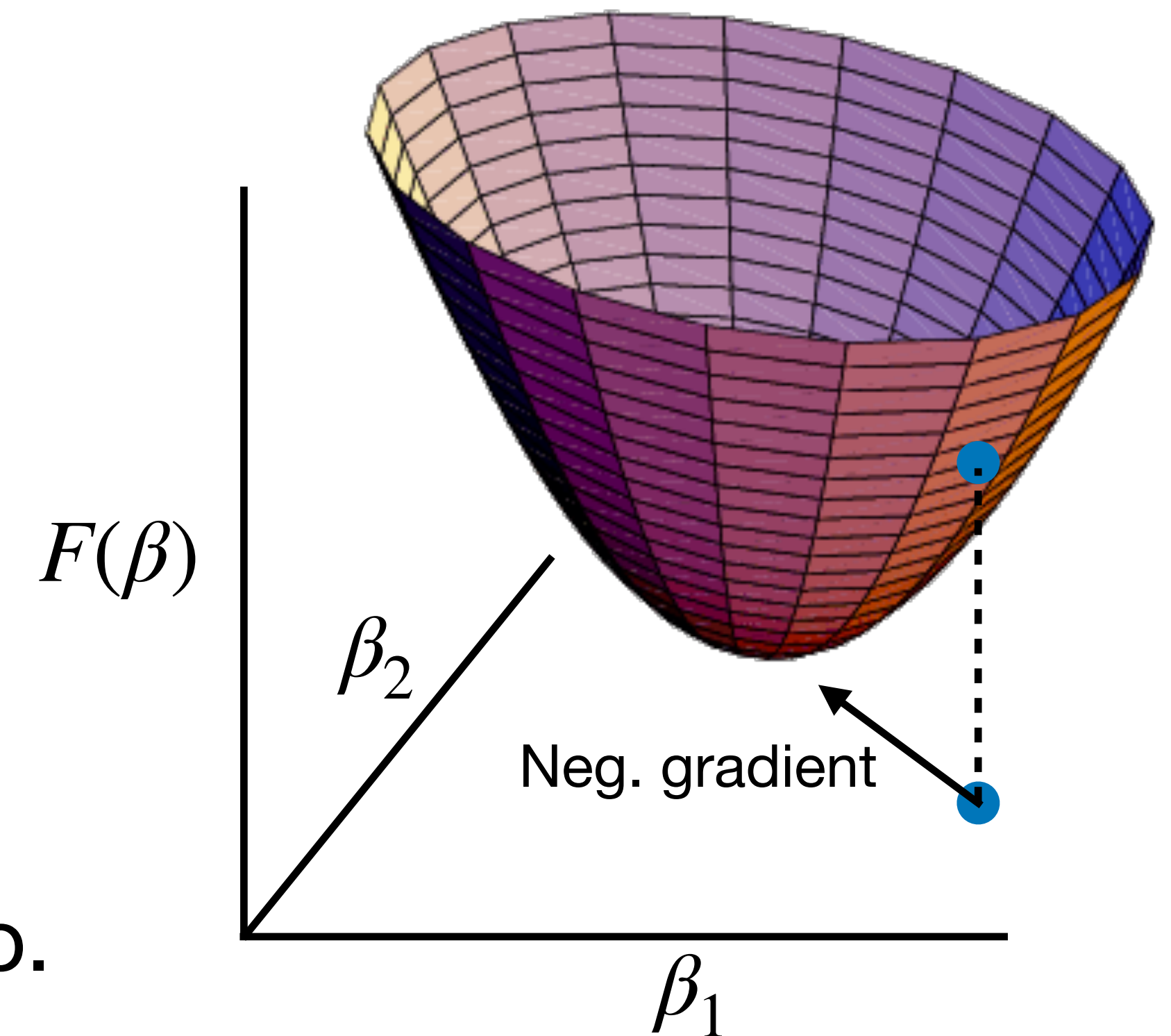
Gradient descent

1. Choose some initial value of β .
2. Evaluate the **gradient** $\nabla F(\beta)$ at that point; it is the direction in which F increases the fastest. The **negative gradient** is the direction in which F decreases the fastest.
3. Take small step in negative gradient direction:
 $\beta \leftarrow \beta - \gamma \nabla F(\beta)$; γ called the **learning rate**.
4. Repeat steps 2 and 3 until gradient is near zero.



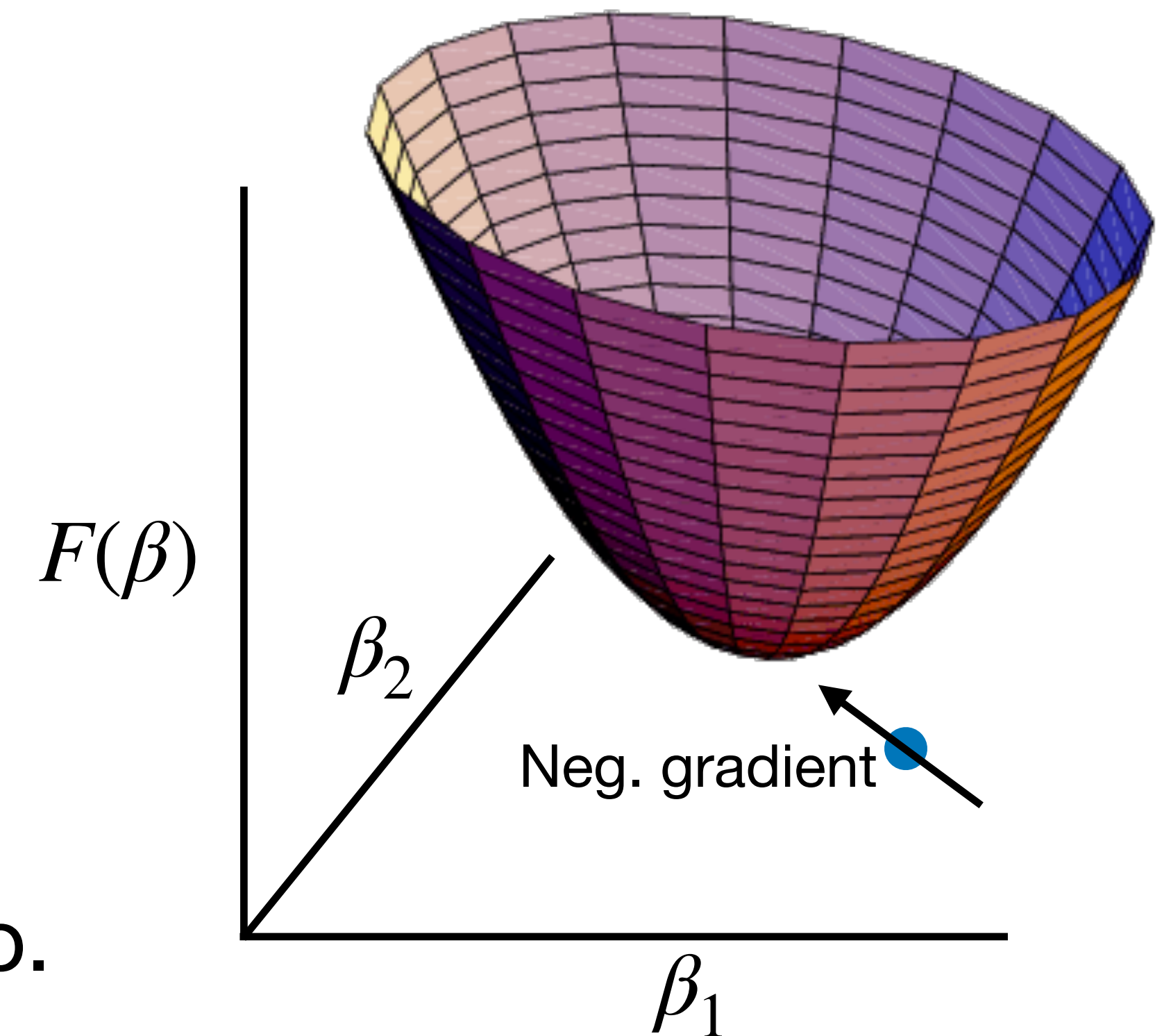
Gradient descent

1. Choose some initial value of β .
2. Evaluate the **gradient** $\nabla F(\beta)$ at that point; it is the direction in which F increases the fastest. The **negative gradient** is the direction in which F decreases the fastest.
3. Take small step in negative gradient direction:
 $\beta \leftarrow \beta - \gamma \nabla F(\beta)$; γ called the **learning rate**.
4. Repeat steps 2 and 3 until gradient is near zero.



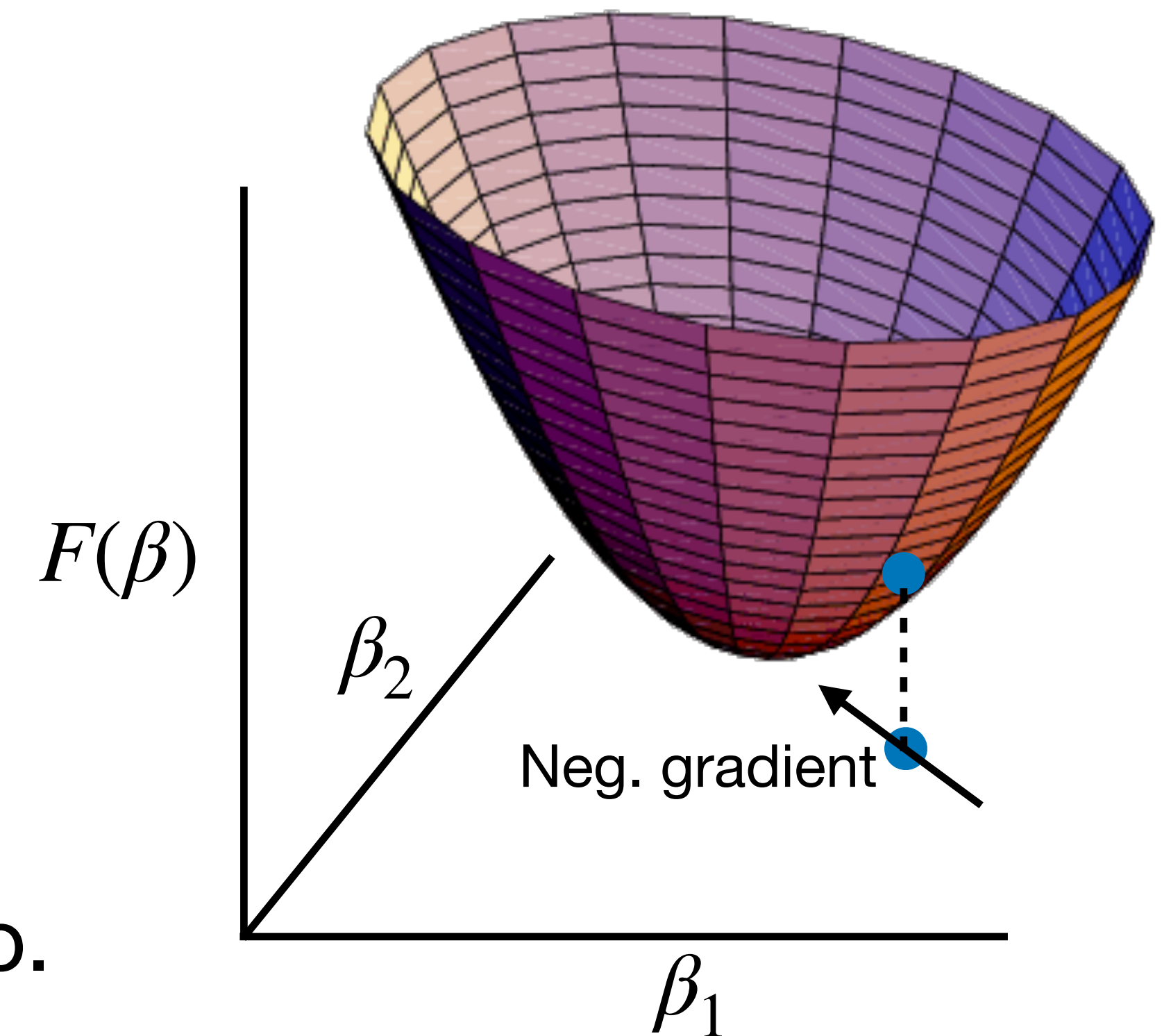
Gradient descent

1. Choose some initial value of β .
2. Evaluate the **gradient** $\nabla F(\beta)$ at that point; it is the direction in which F increases the fastest. The **negative gradient** is the direction in which F decreases the fastest.
3. Take small step in negative gradient direction:
 $\beta \leftarrow \beta - \gamma \nabla F(\beta)$; γ called the **learning rate**.
4. Repeat steps 2 and 3 until gradient is near zero.



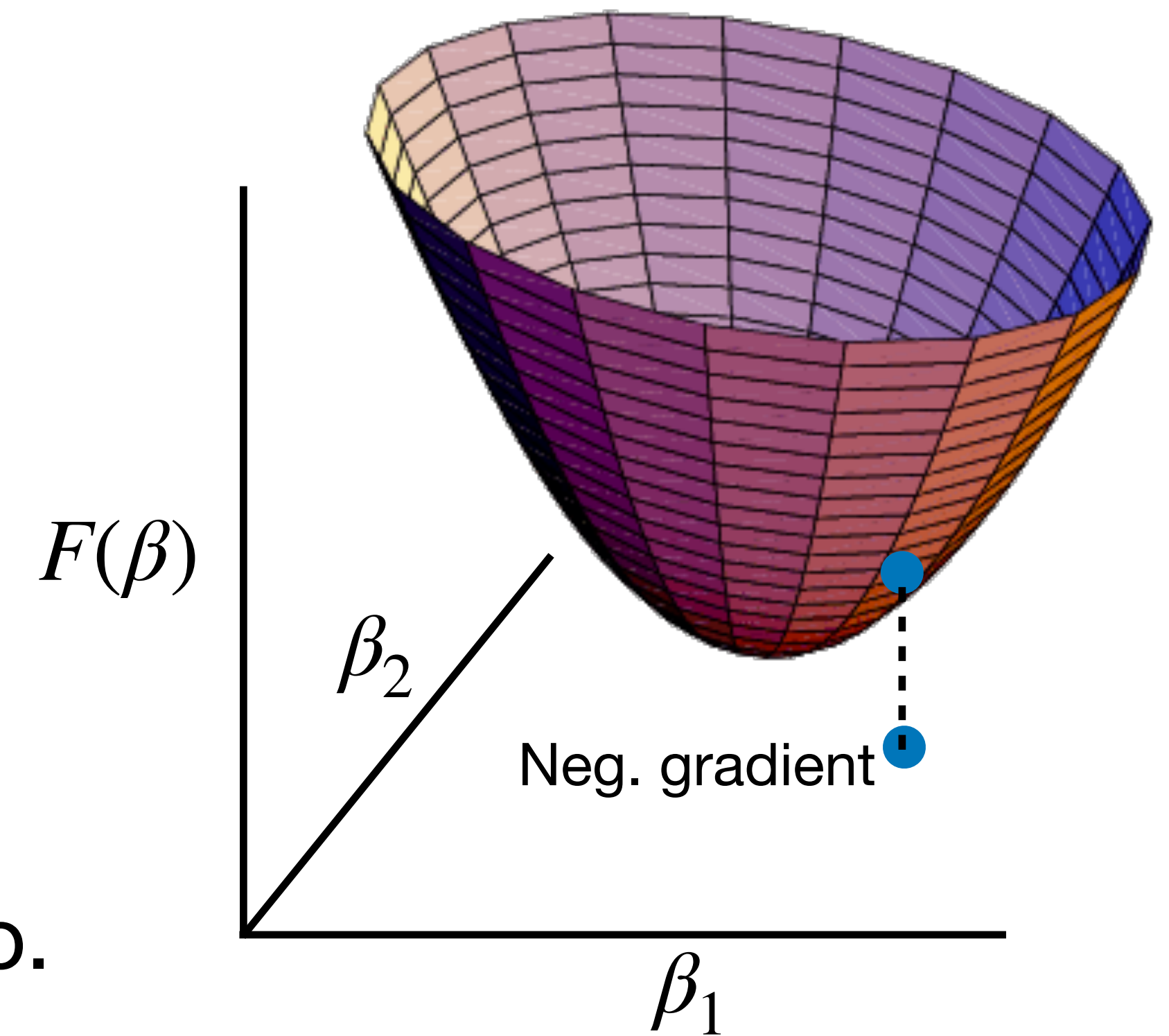
Gradient descent

1. Choose some initial value of β .
2. Evaluate the **gradient** $\nabla F(\beta)$ at that point; it is the direction in which F increases the fastest. The **negative gradient** is the direction in which F decreases the fastest.
3. Take small step in negative gradient direction:
 $\beta \leftarrow \beta - \gamma \nabla F(\beta)$; γ called the **learning rate**.
4. Repeat steps 2 and 3 until gradient is near zero.



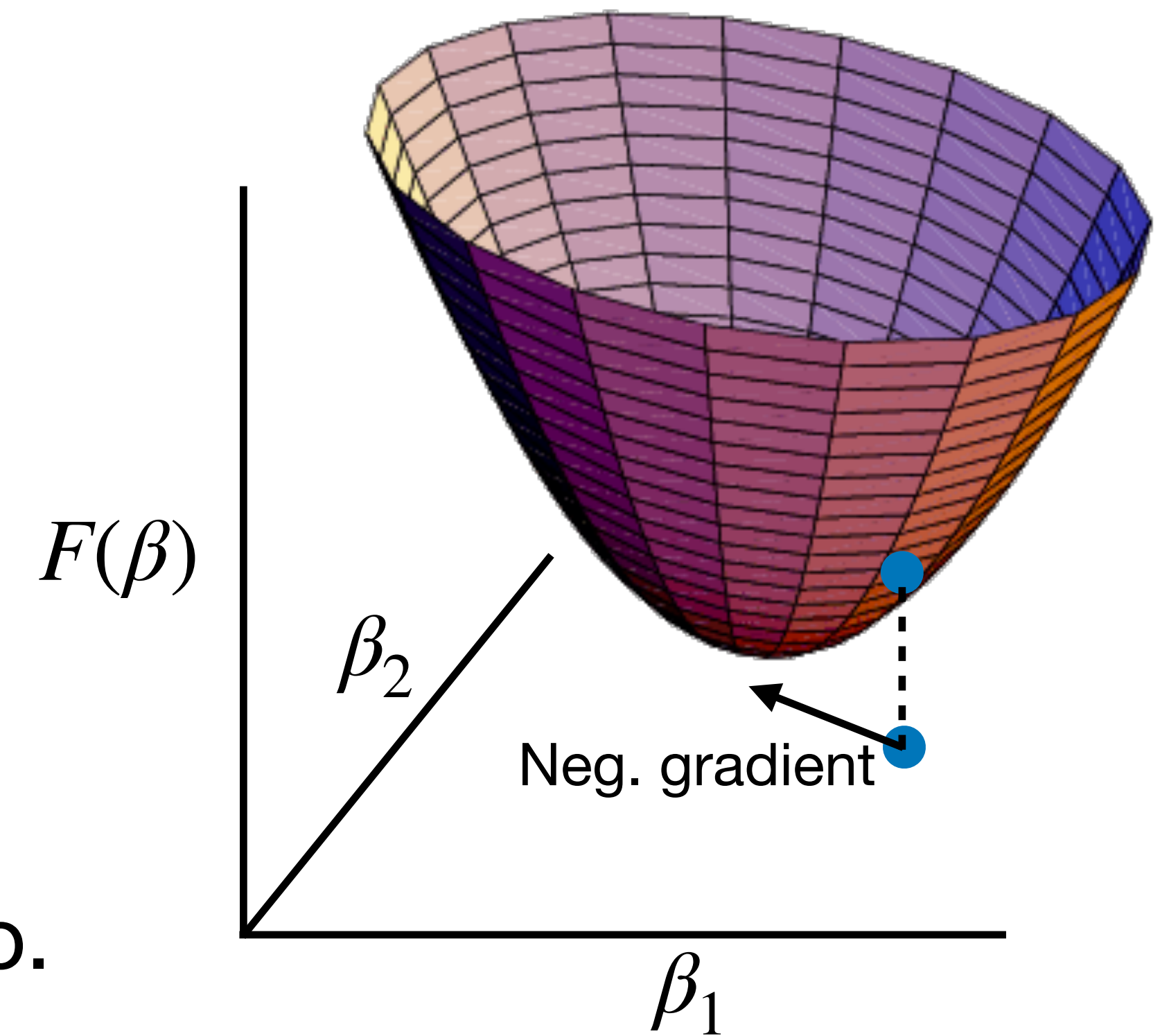
Gradient descent

1. Choose some initial value of β .
2. Evaluate the **gradient** $\nabla F(\beta)$ at that point; it is the direction in which F increases the fastest. The **negative gradient** is the direction in which F decreases the fastest.
3. Take small step in negative gradient direction:
 $\beta \leftarrow \beta - \gamma \nabla F(\beta)$; γ called the **learning rate**.
4. Repeat steps 2 and 3 until gradient is near zero.



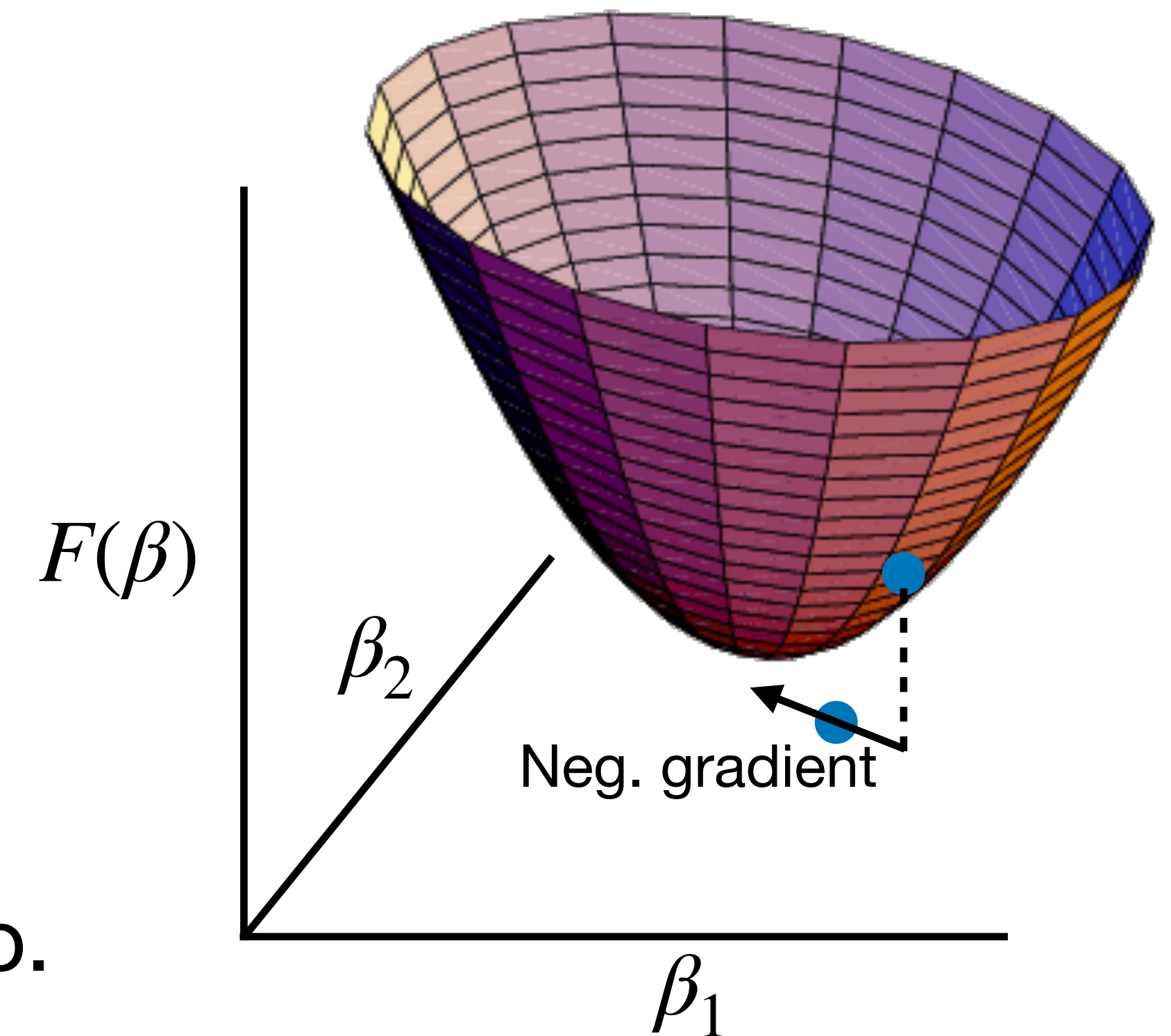
Gradient descent

1. Choose some initial value of β .
2. Evaluate the **gradient** $\nabla F(\beta)$ at that point; it is the direction in which F increases the fastest. The **negative gradient** is the direction in which F decreases the fastest.
3. Take small step in negative gradient direction:
 $\beta \leftarrow \beta - \gamma \nabla F(\beta)$; γ called the **learning rate**.
4. Repeat steps 2 and 3 until gradient is near zero.



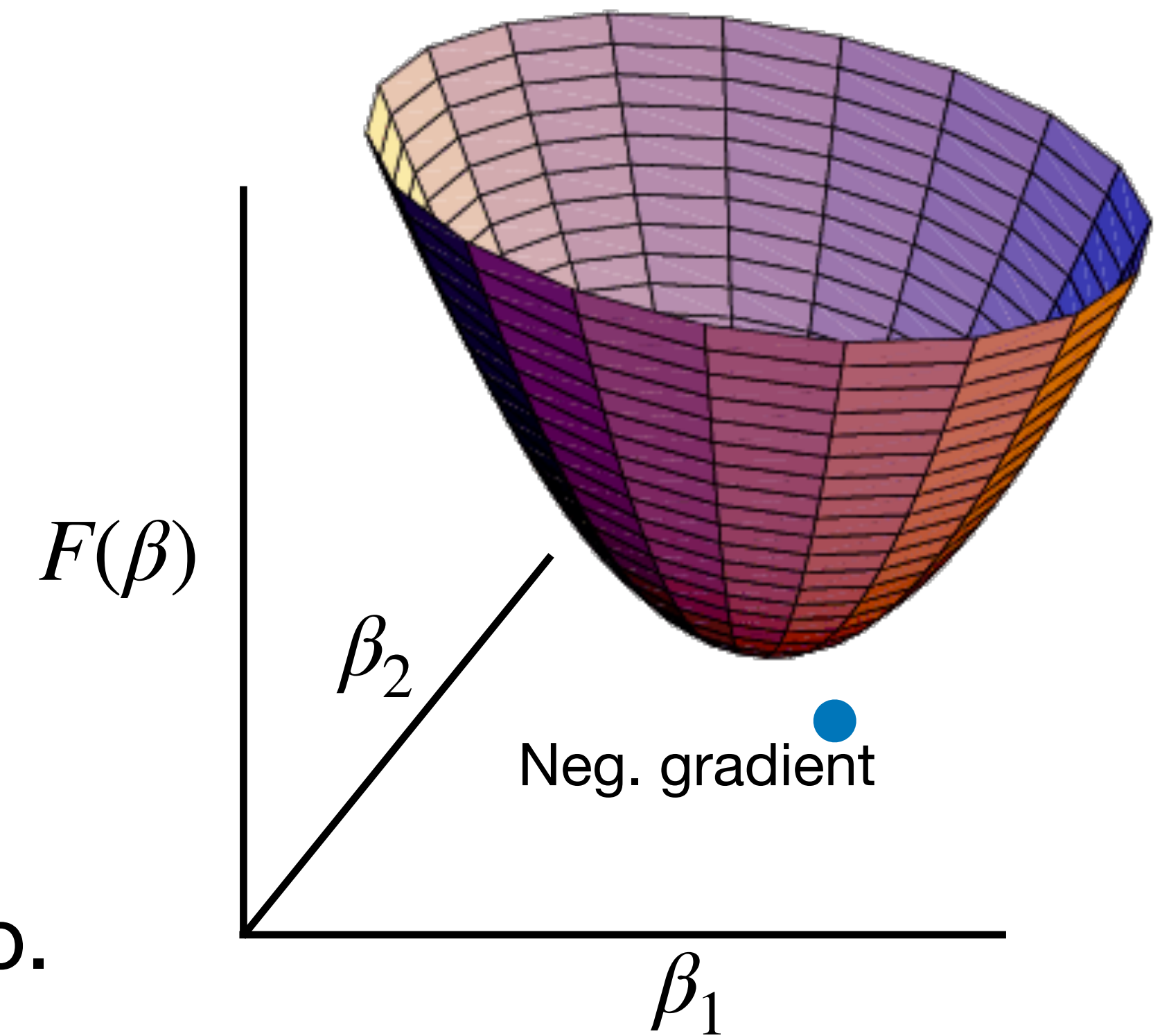
Gradient descent

1. Choose some initial value of β .
2. Evaluate the **gradient** $\nabla F(\beta)$ at that point; it is the direction in which F increases the fastest. The **negative gradient** is the direction in which F decreases the fastest.
3. Take small step in negative gradient direction:
 $\beta \leftarrow \beta - \gamma \nabla F(\beta)$; γ called the **learning rate**.
4. Repeat steps 2 and 3 until gradient is near zero.



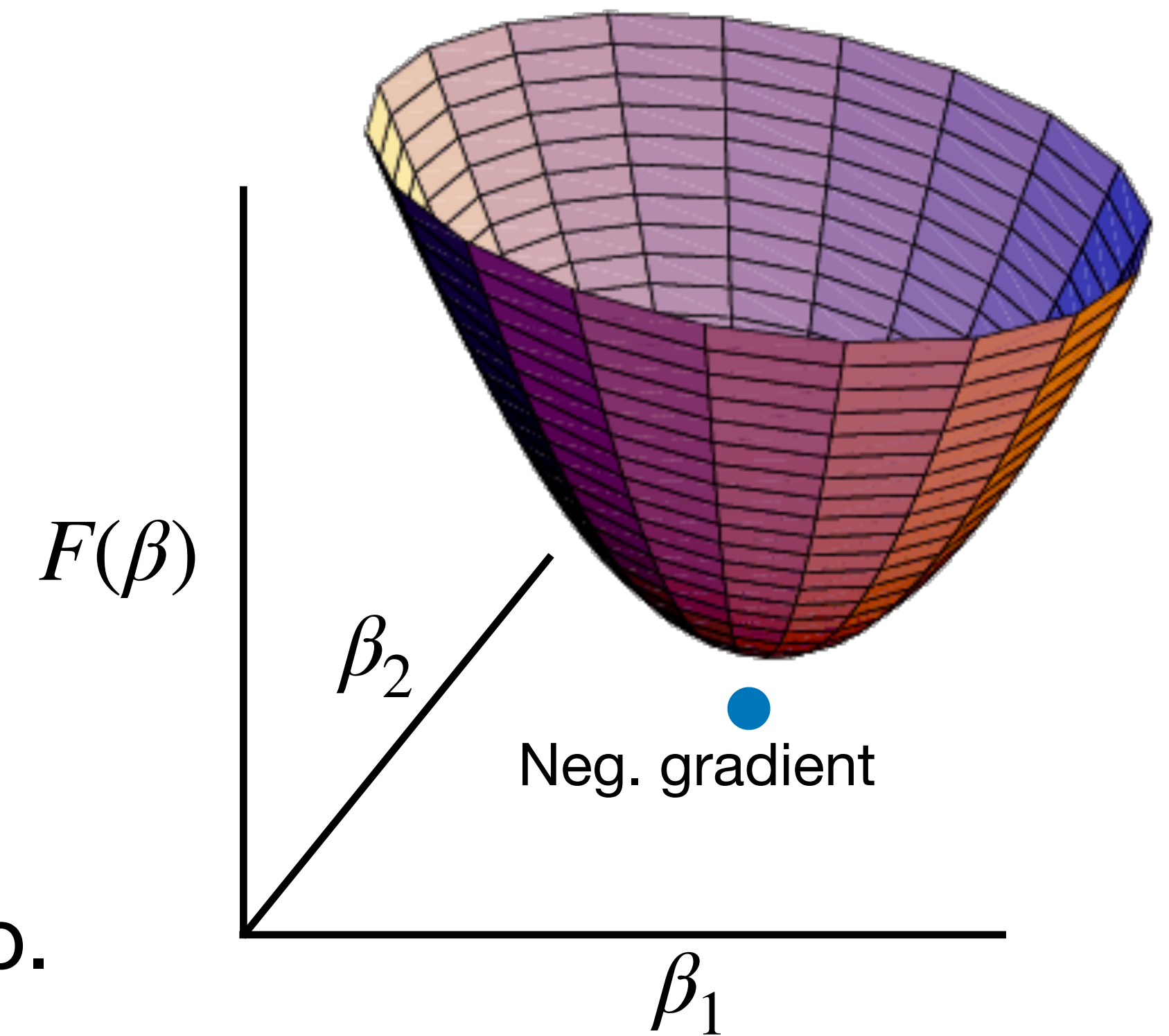
Gradient descent

1. Choose some initial value of β .
2. Evaluate the **gradient** $\nabla F(\beta)$ at that point; it is the direction in which F increases the fastest. The **negative gradient** is the direction in which F decreases the fastest.
3. Take small step in negative gradient direction:
 $\beta \leftarrow \beta - \gamma \nabla F(\beta)$; γ called the **learning rate**.
4. Repeat steps 2 and 3 until gradient is near zero.



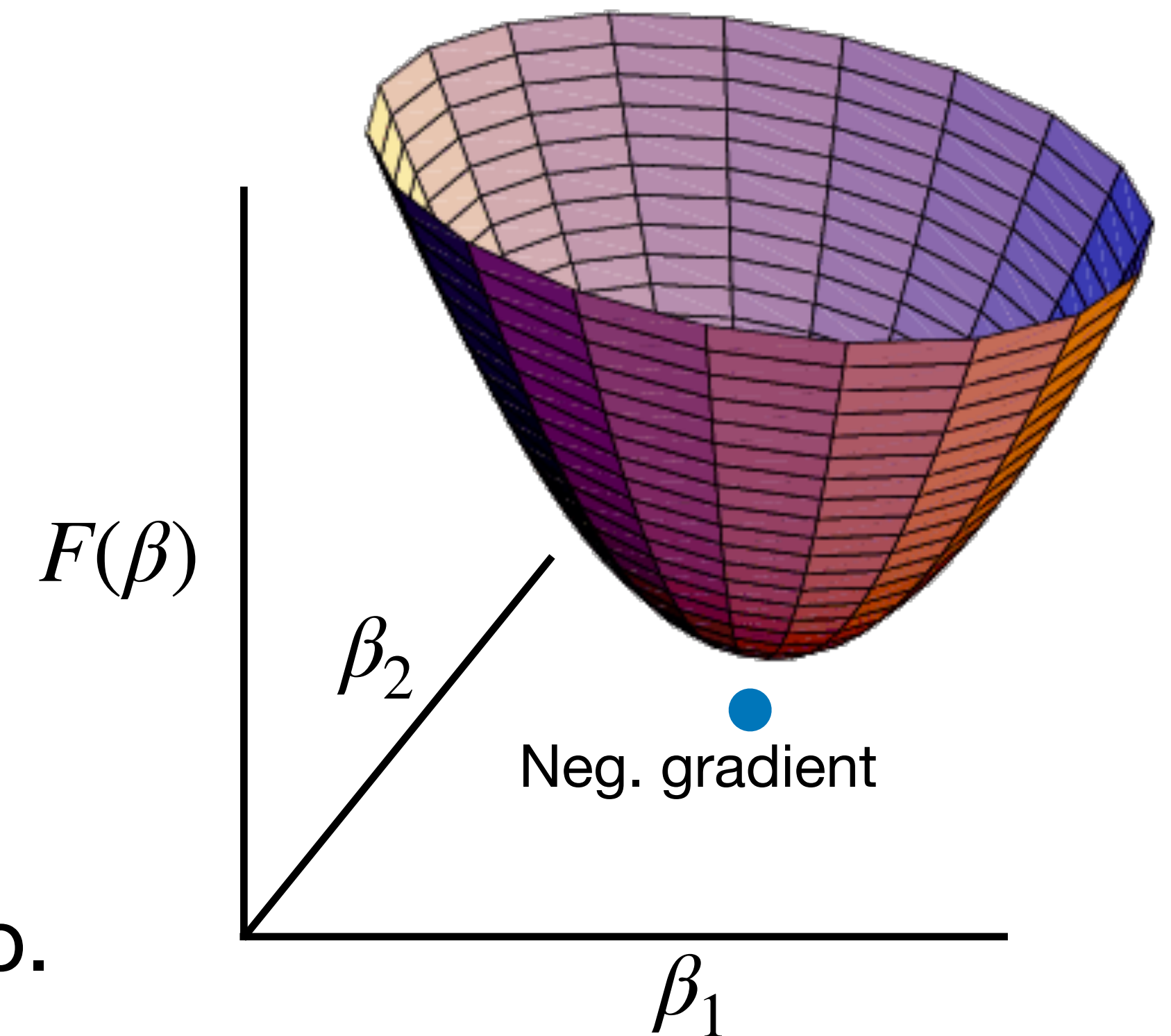
Gradient descent

1. Choose some initial value of β .
2. Evaluate the **gradient** $\nabla F(\beta)$ at that point; it is the direction in which F increases the fastest. The **negative gradient** is the direction in which F decreases the fastest.
3. Take small step in negative gradient direction:
 $\beta \leftarrow \beta - \gamma \nabla F(\beta)$; γ called the **learning rate**.
4. Repeat steps 2 and 3 until gradient is near zero.



Gradient descent

1. Choose some initial value of β .
2. Evaluate the **gradient** $\nabla F(\beta)$ at that point; it is the direction in which F increases the fastest. The **negative gradient** is the direction in which F decreases the fastest.
3. Take small step in negative gradient direction:
 $\beta \leftarrow \beta - \gamma \nabla F(\beta)$; γ called the **learning rate**.
4. Repeat steps 2 and 3 until gradient is near zero.



As long as the learning rate γ is not too large, gradient descent is guaranteed to converge to a global minimum regardless of initialization **if F is convex**.

Gradient descent for non-convex optimization

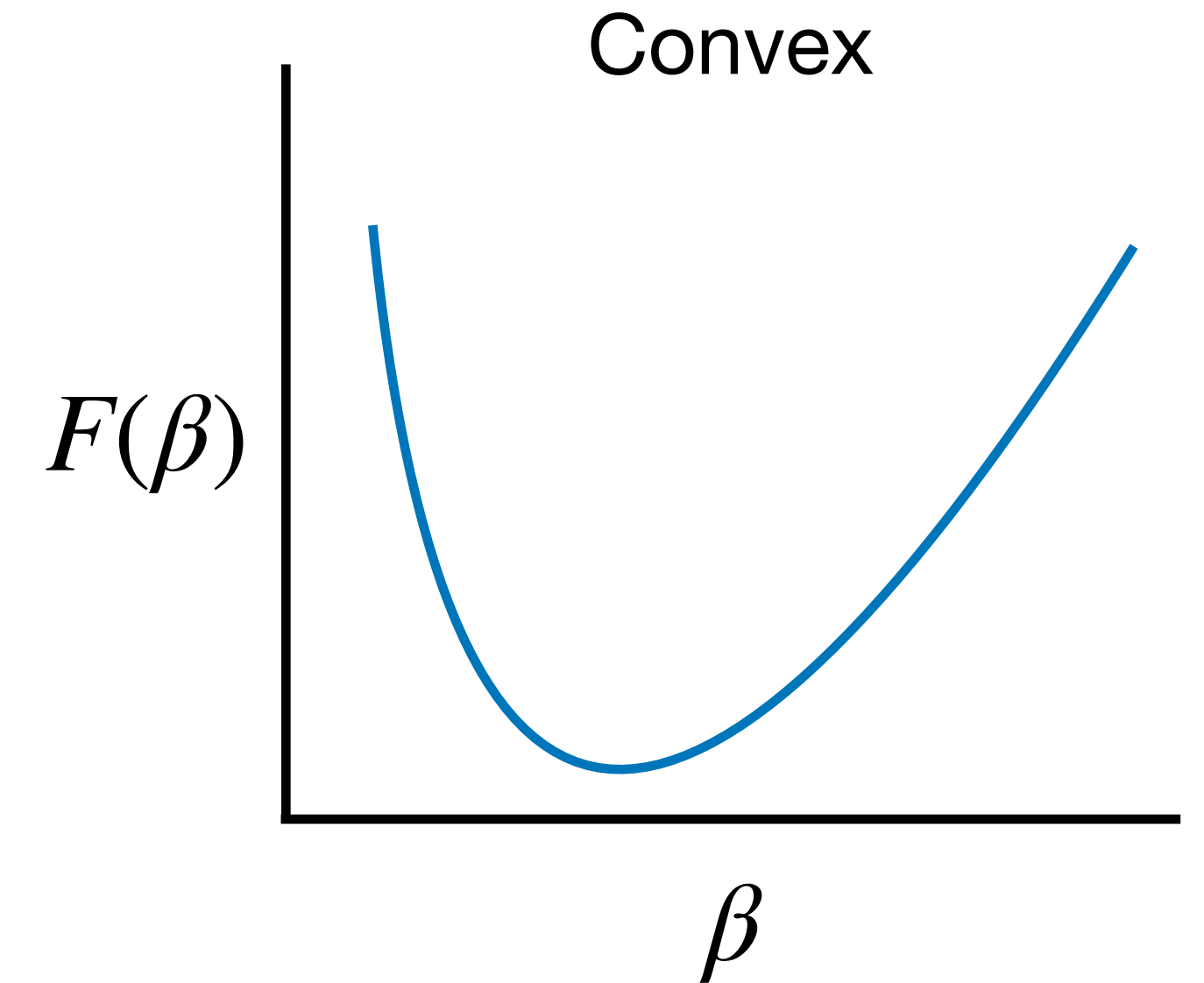
Gradient descent for non-convex optimization

Think about gradient descent as a ball rolling down a hill.

Gradient descent for non-convex optimization

Think about gradient descent as a ball rolling down a hill.

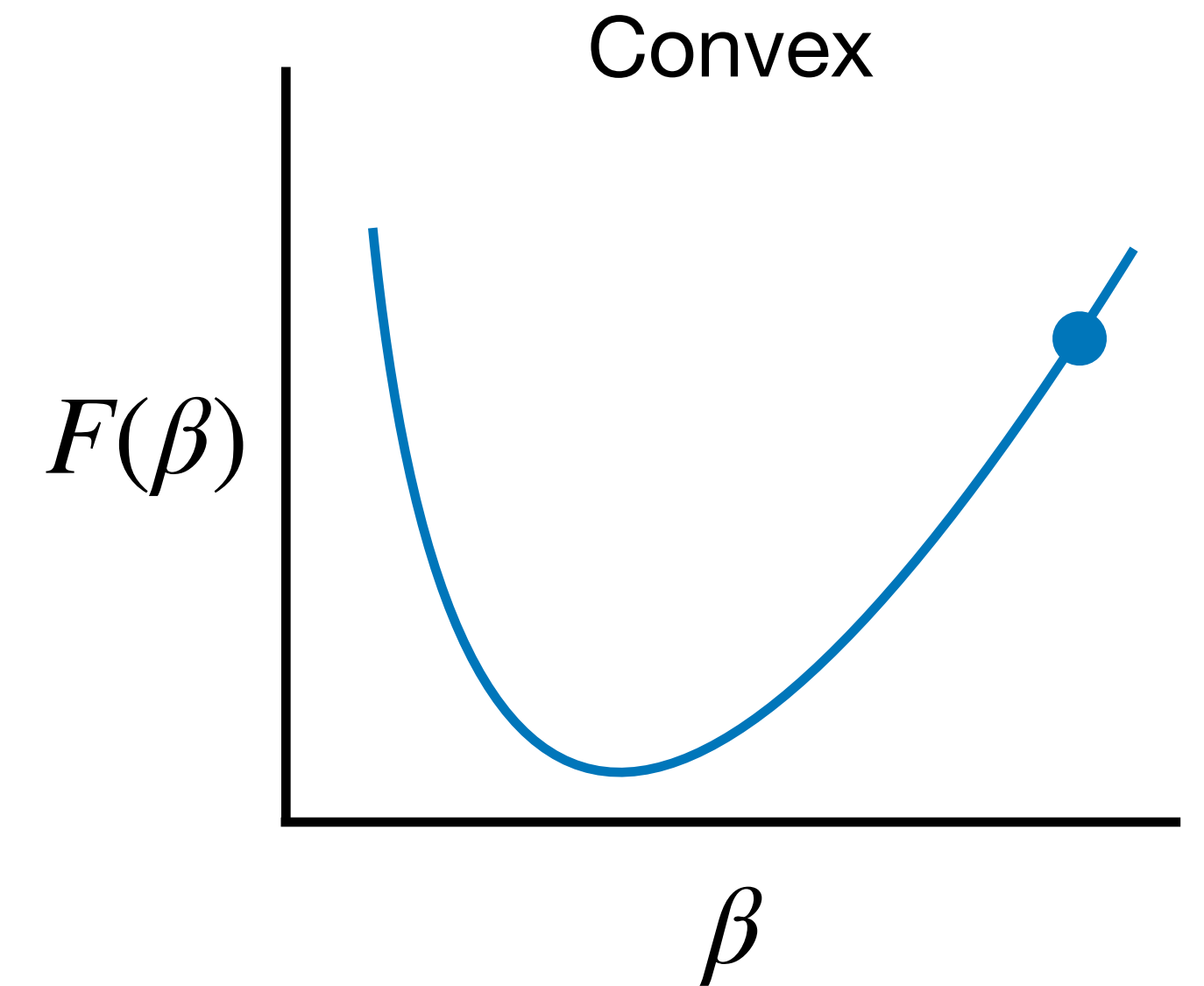
For convex functions, there is only one place (the global minimum) for the ball to roll, no matter where it starts.



Gradient descent for non-convex optimization

Think about gradient descent as a ball rolling down a hill.

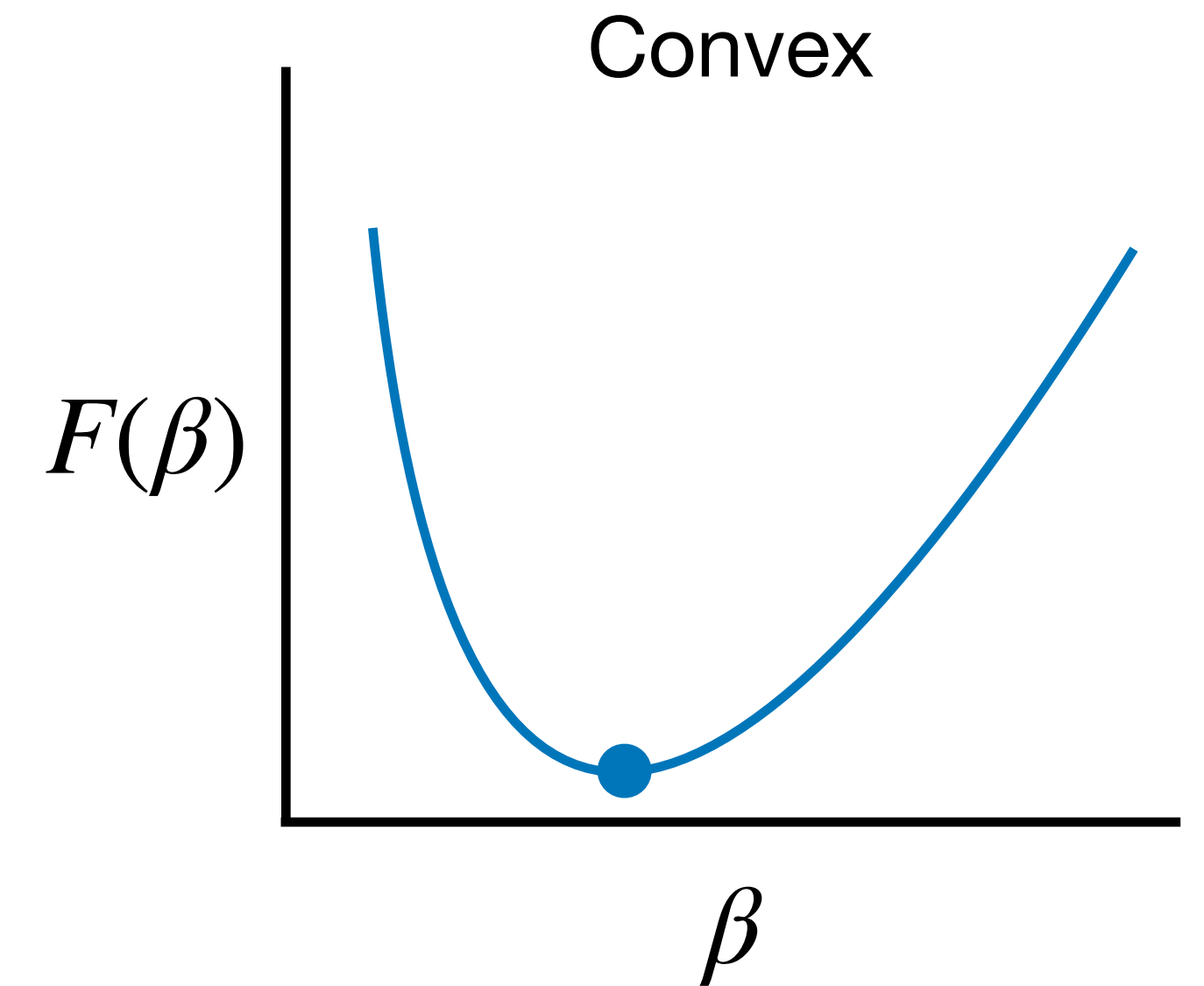
For convex functions, there is only one place (the global minimum) for the ball to roll, no matter where it starts.



Gradient descent for non-convex optimization

Think about gradient descent as a ball rolling down a hill.

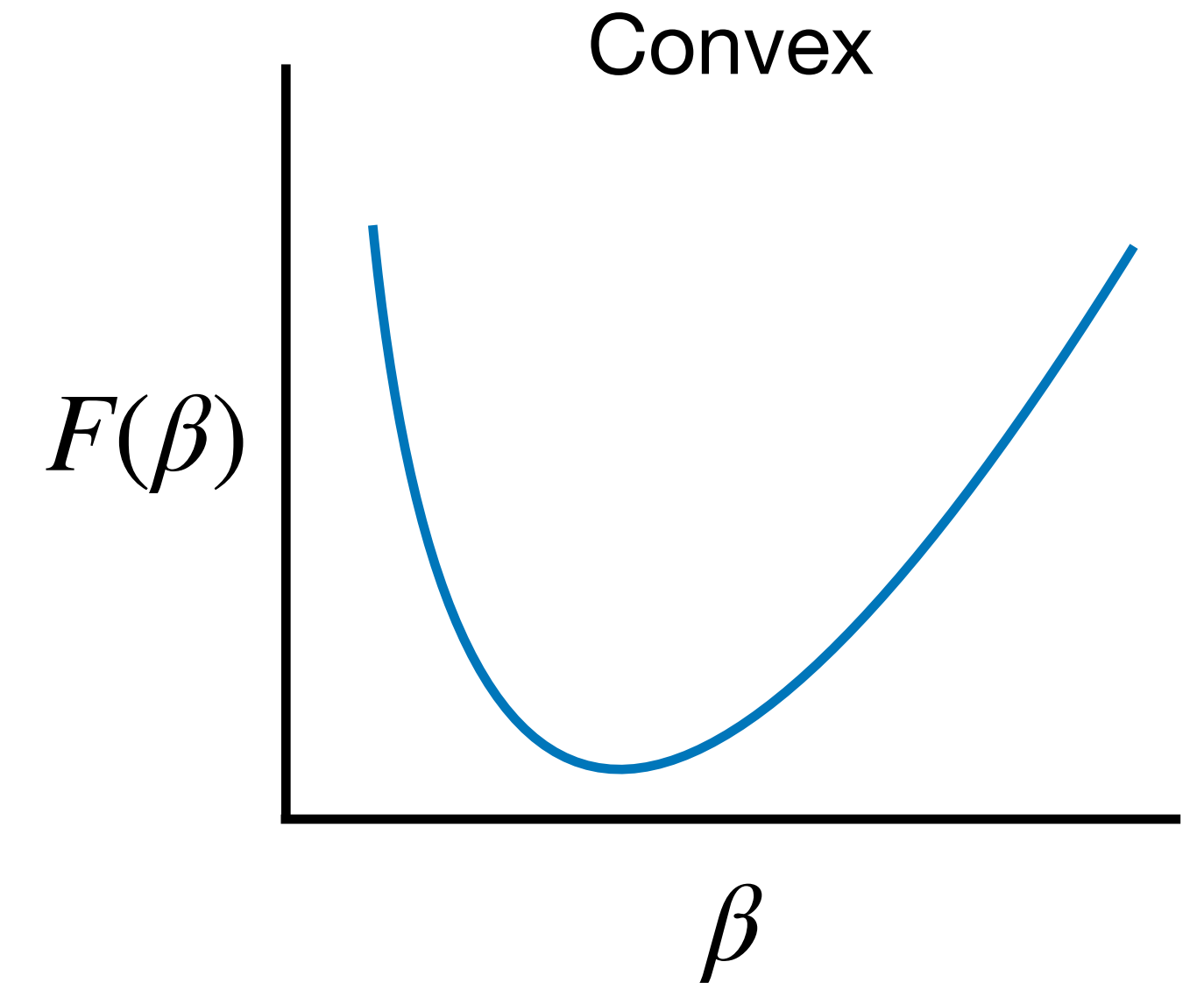
For convex functions, there is only one place (the global minimum) for the ball to roll, no matter where it starts.



Gradient descent for non-convex optimization

Think about gradient descent as a ball rolling down a hill.

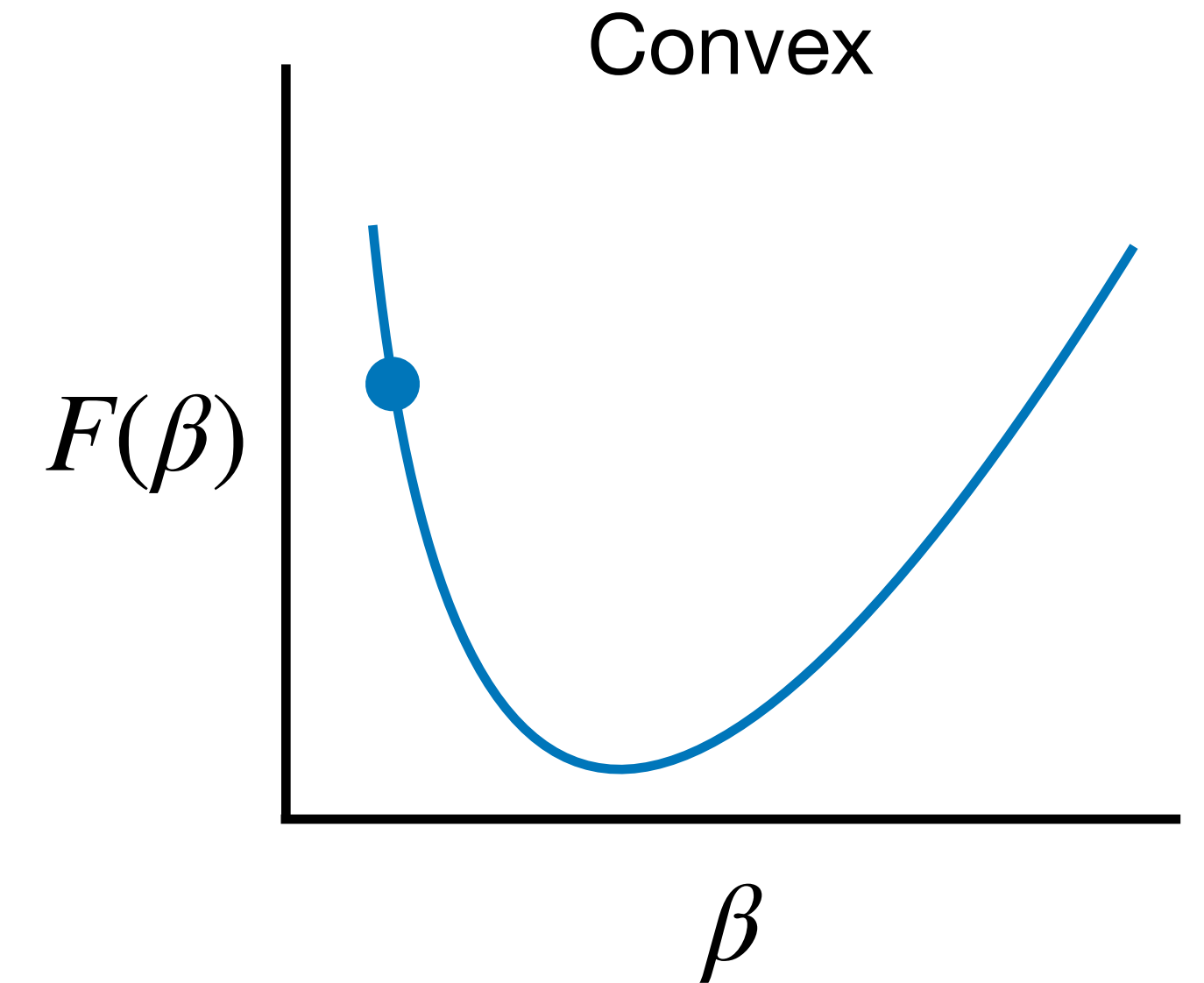
For convex functions, there is only one place (the global minimum) for the ball to roll, no matter where it starts.



Gradient descent for non-convex optimization

Think about gradient descent as a ball rolling down a hill.

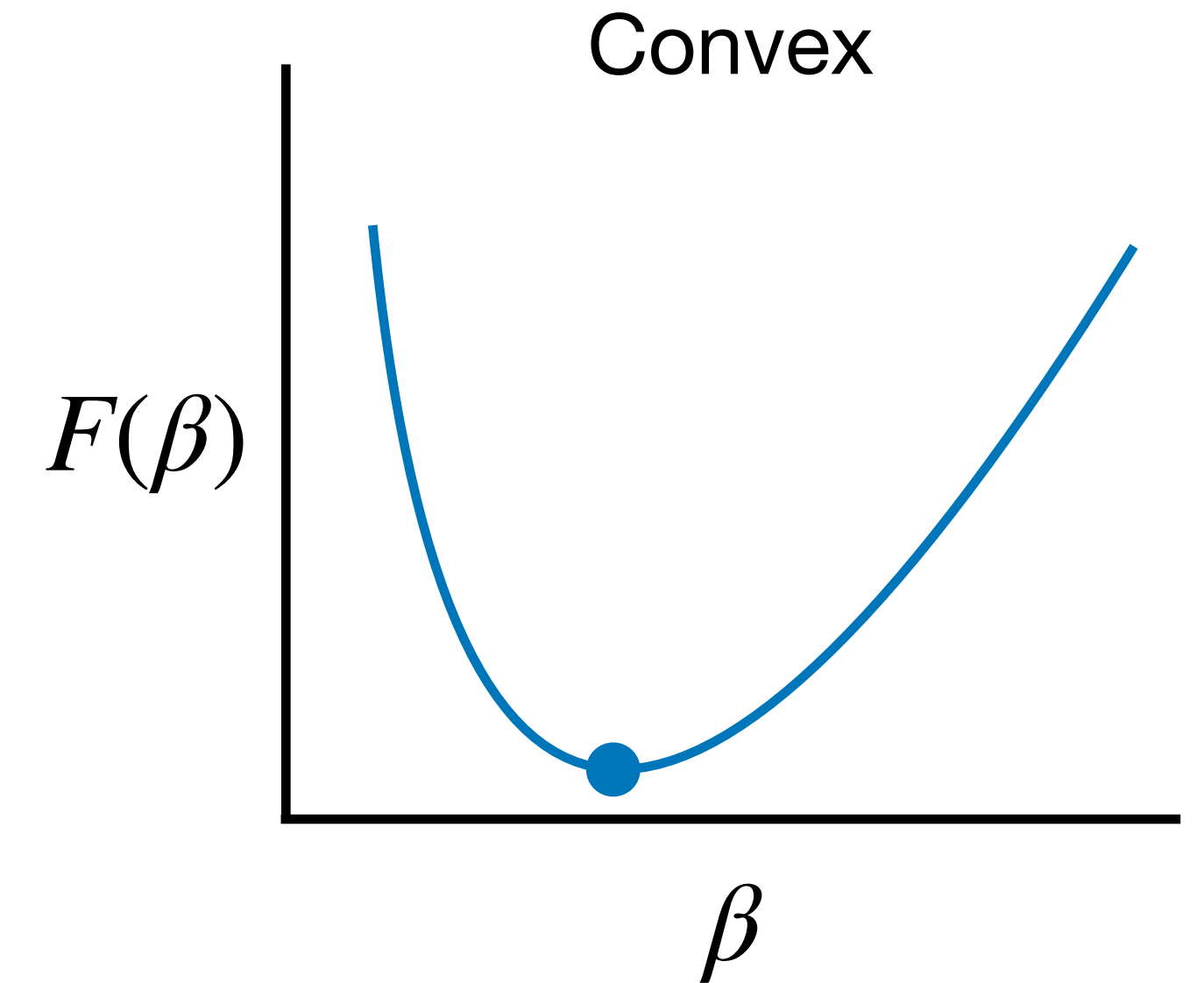
For convex functions, there is only one place (the global minimum) for the ball to roll, no matter where it starts.



Gradient descent for non-convex optimization

Think about gradient descent as a ball rolling down a hill.

For convex functions, there is only one place (the global minimum) for the ball to roll, no matter where it starts.

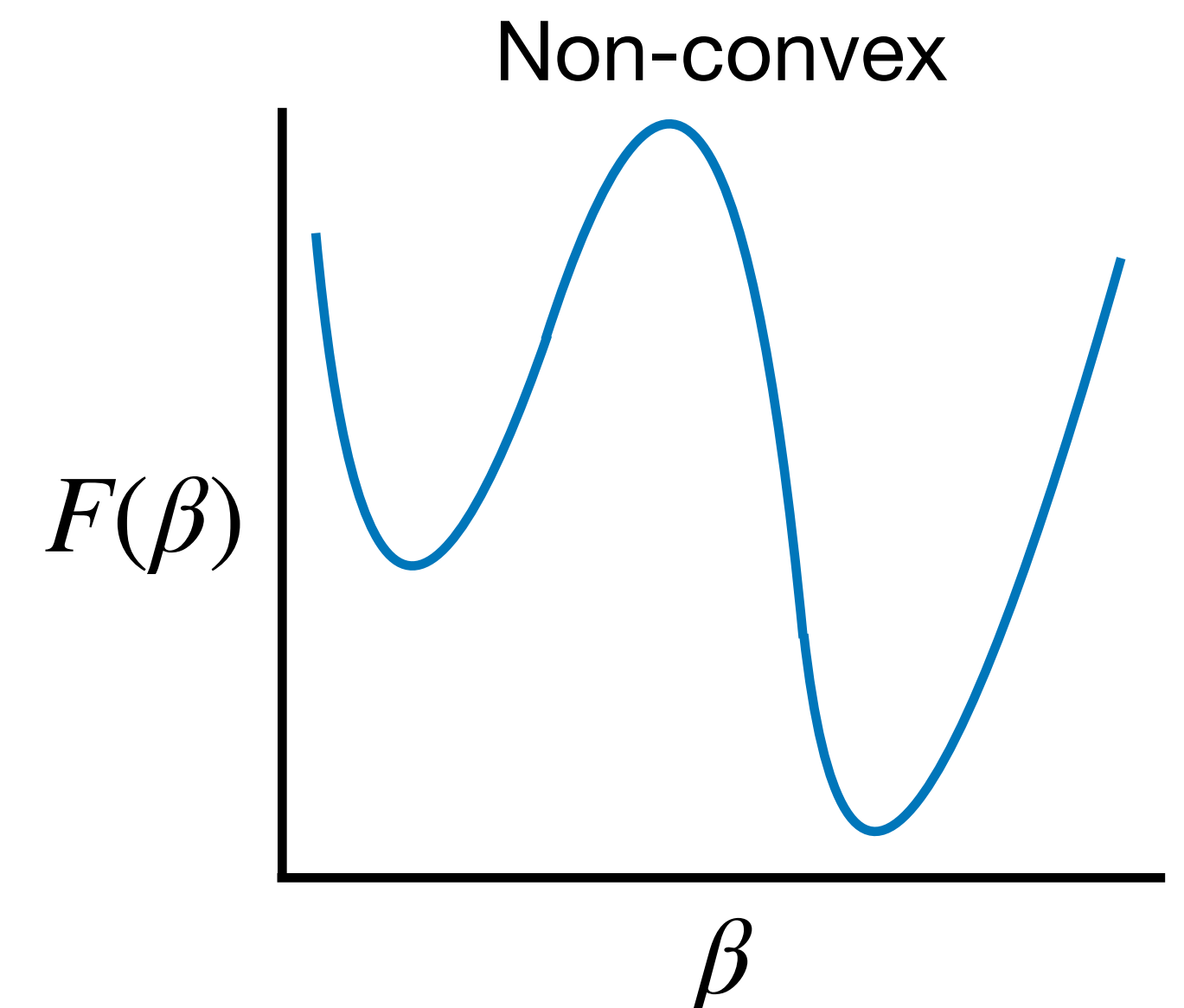
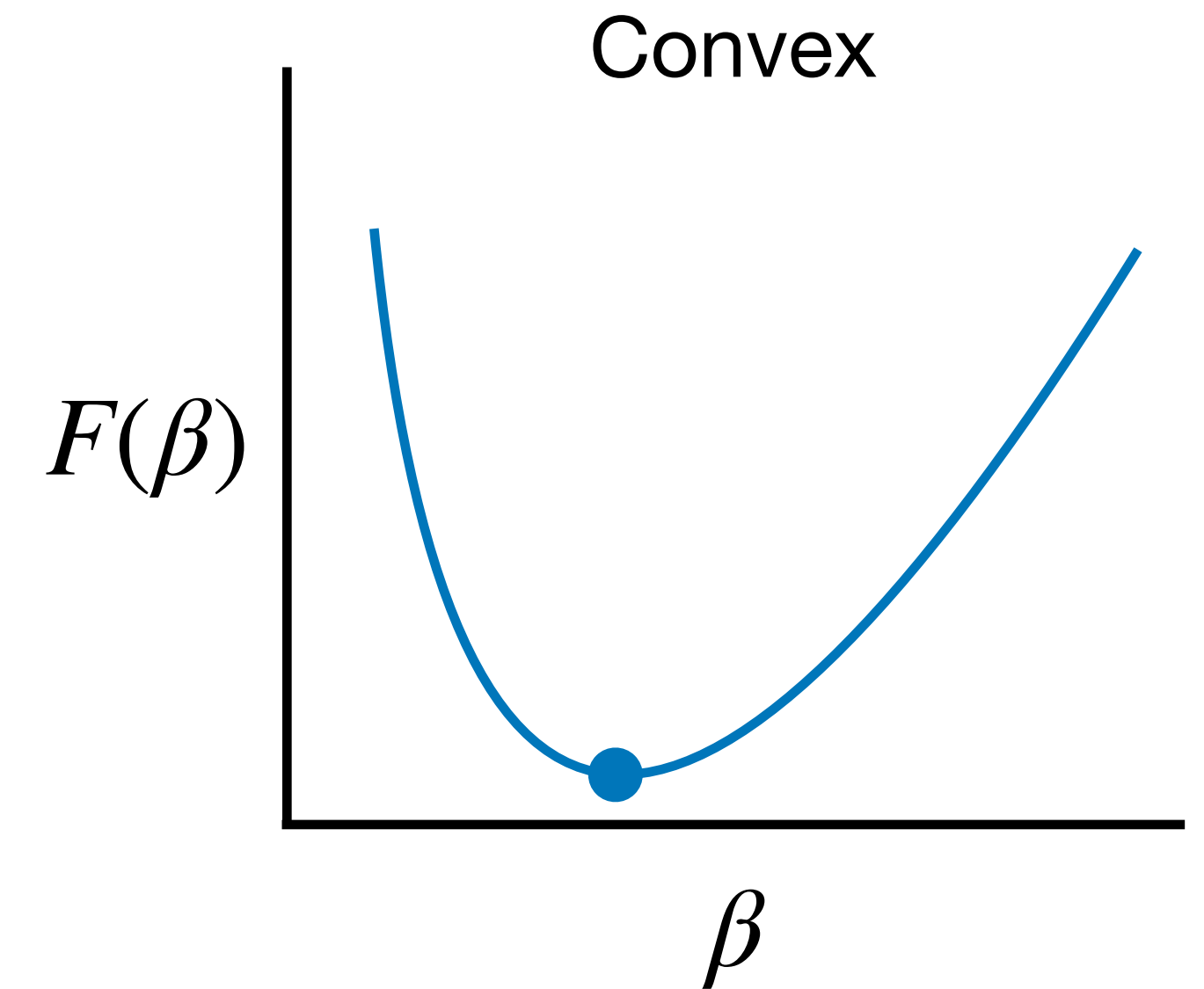


Gradient descent for non-convex optimization

Think about gradient descent as a ball rolling down a hill.

For convex functions, there is only one place (the global minimum) for the ball to roll, no matter where it starts.

For non-convex functions, the ball can roll into any of the local minima, most of which are not global minima.

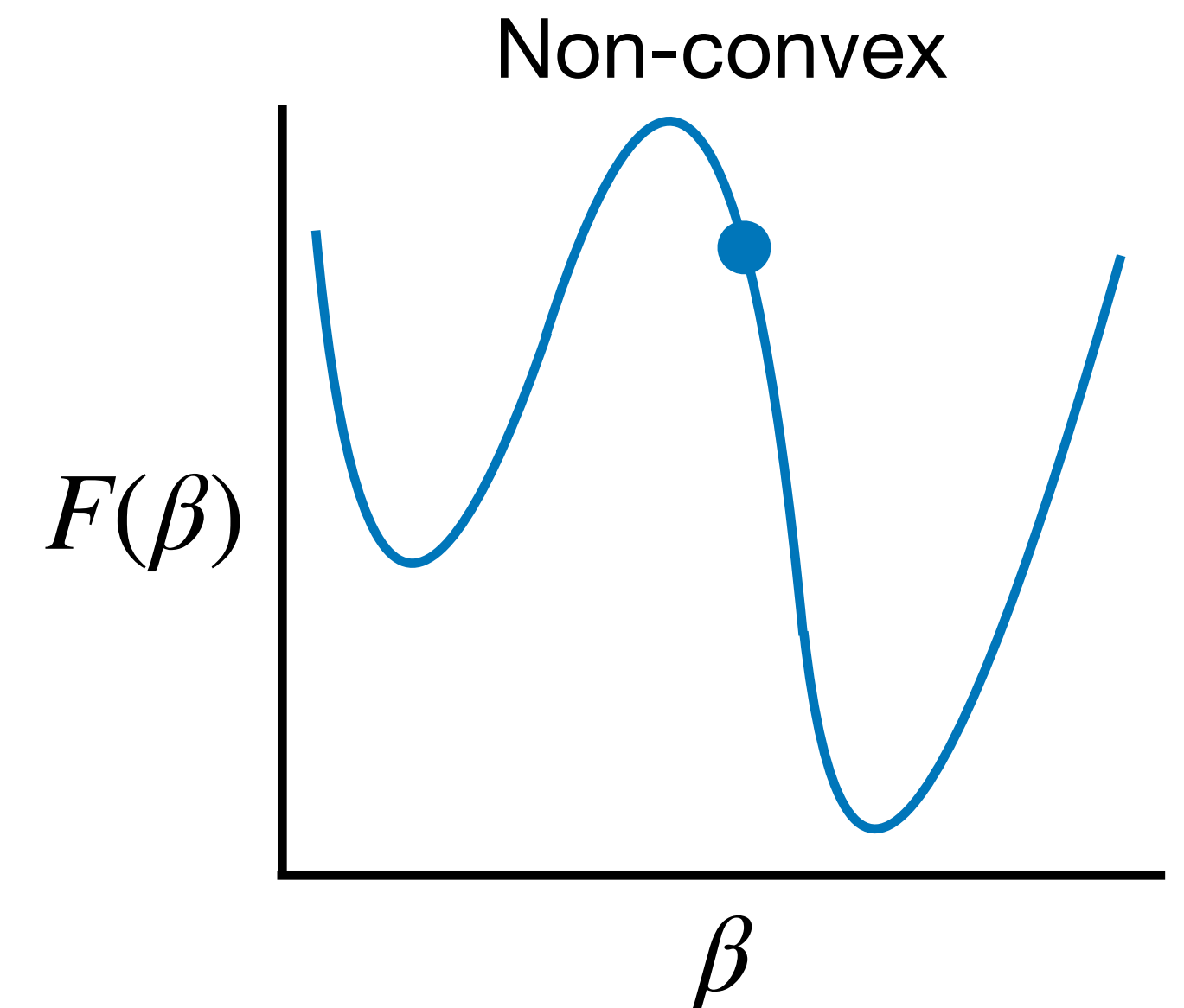
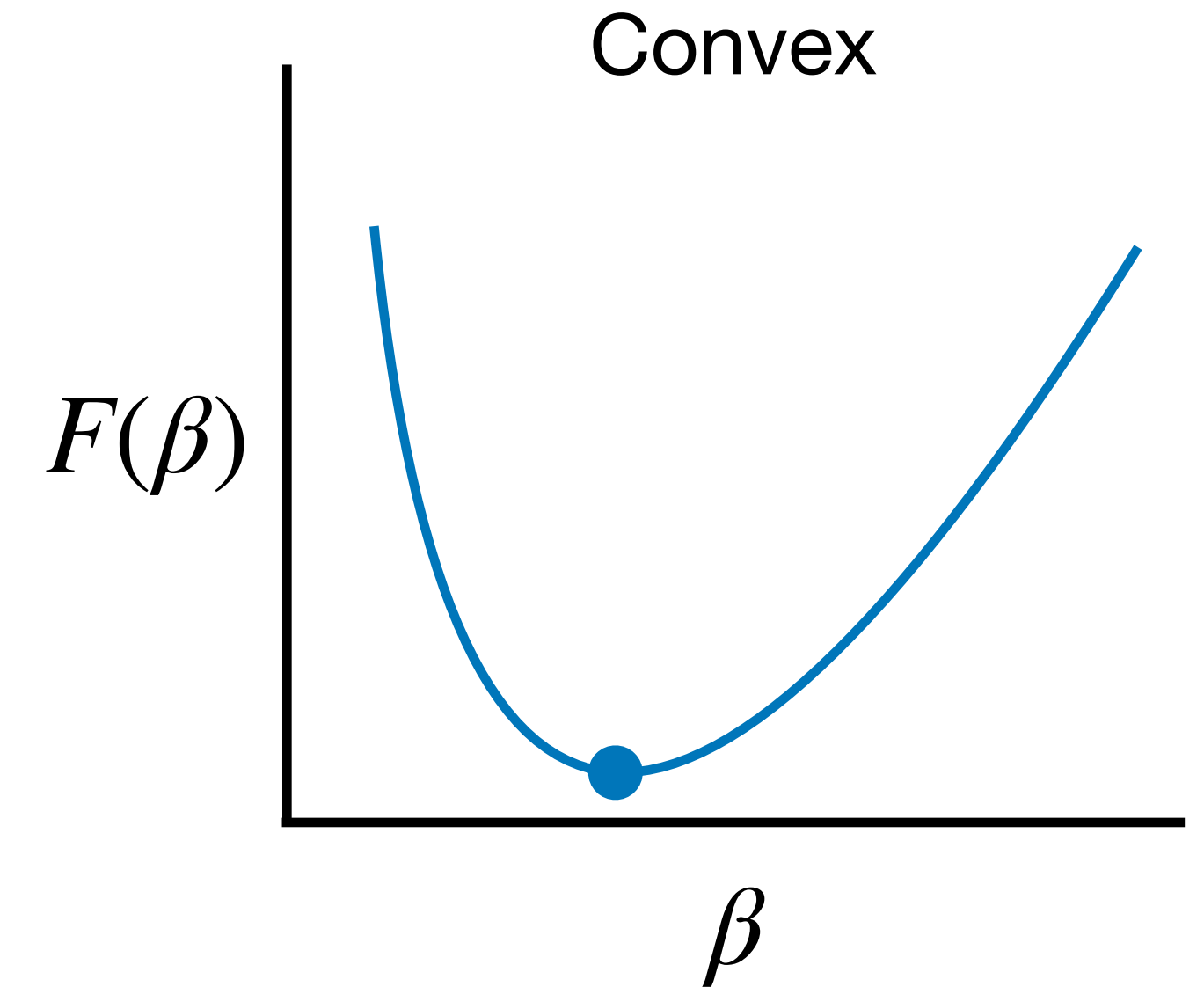


Gradient descent for non-convex optimization

Think about gradient descent as a ball rolling down a hill.

For convex functions, there is only one place (the global minimum) for the ball to roll, no matter where it starts.

For non-convex functions, the ball can roll into any of the local minima, most of which are not global minima.

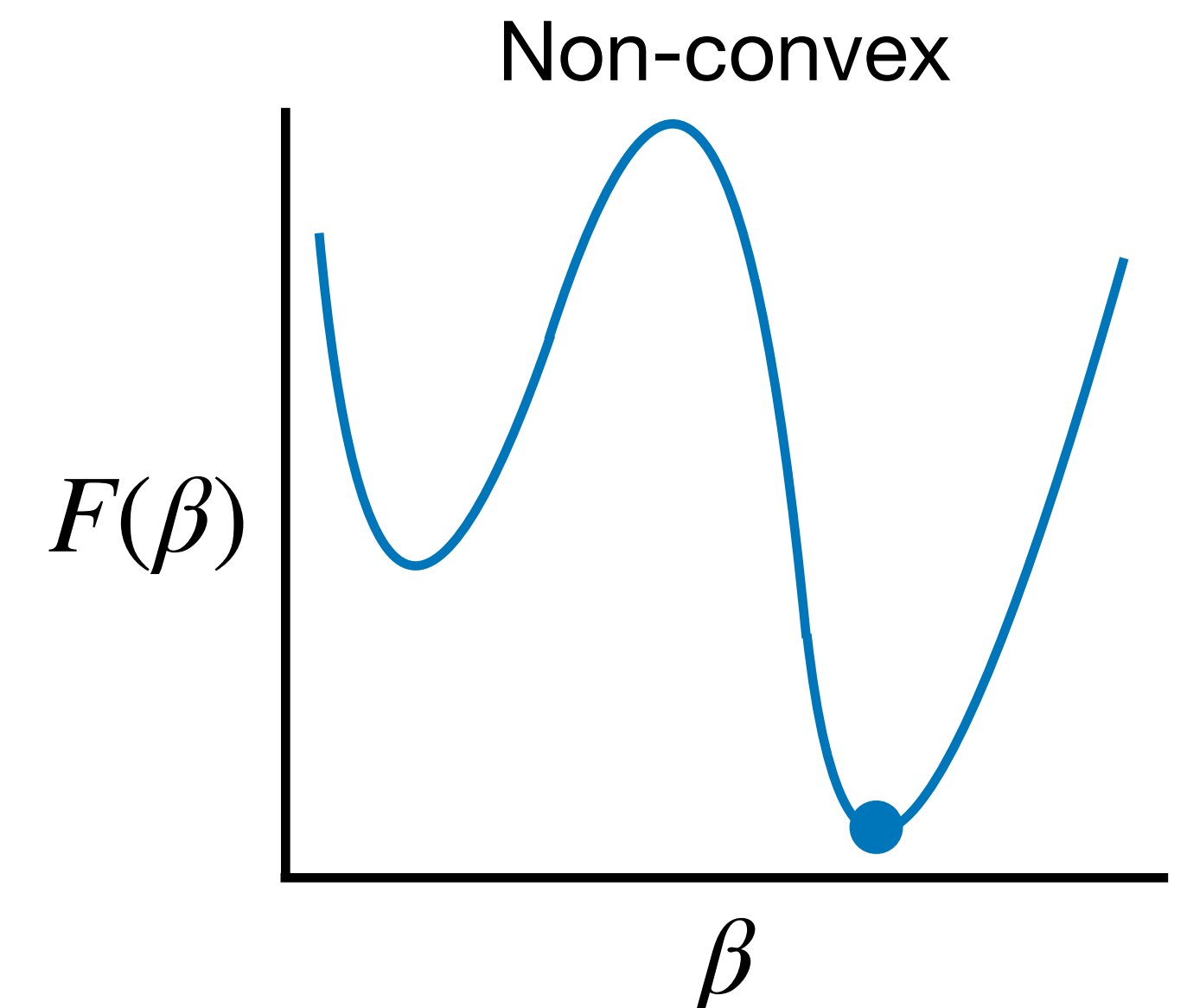
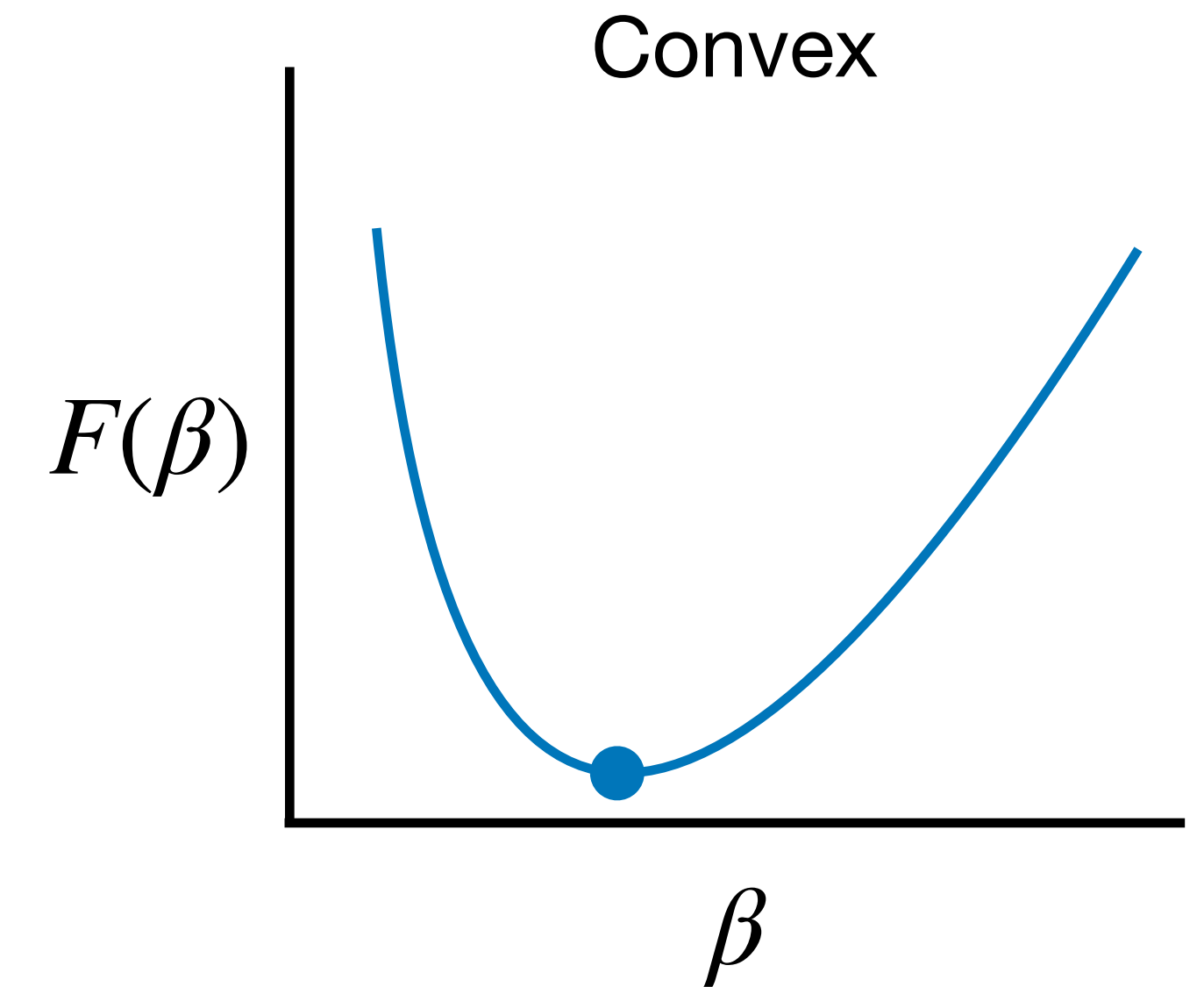


Gradient descent for non-convex optimization

Think about gradient descent as a ball rolling down a hill.

For convex functions, there is only one place (the global minimum) for the ball to roll, no matter where it starts.

For non-convex functions, the ball can roll into any of the local minima, most of which are not global minima.

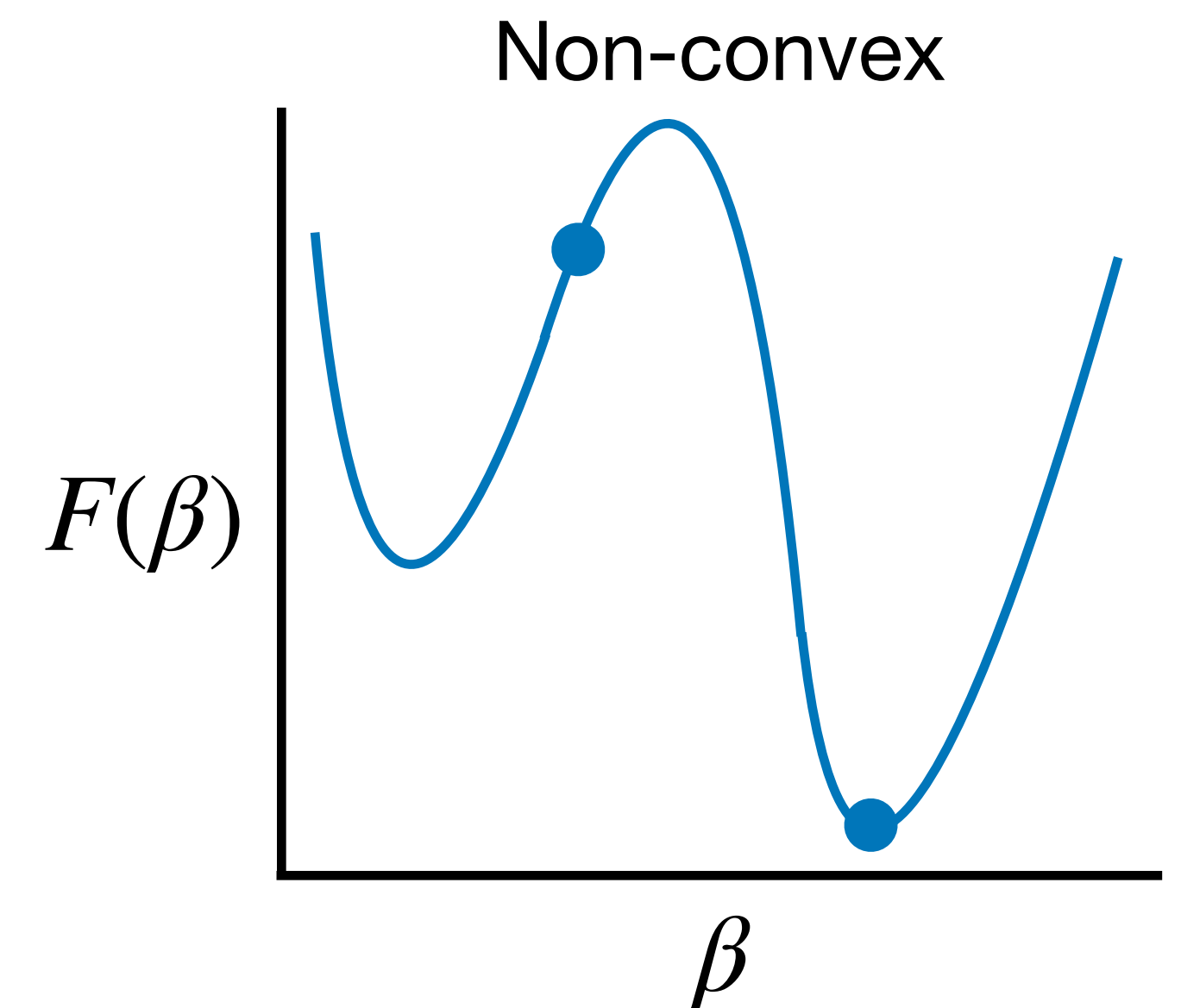
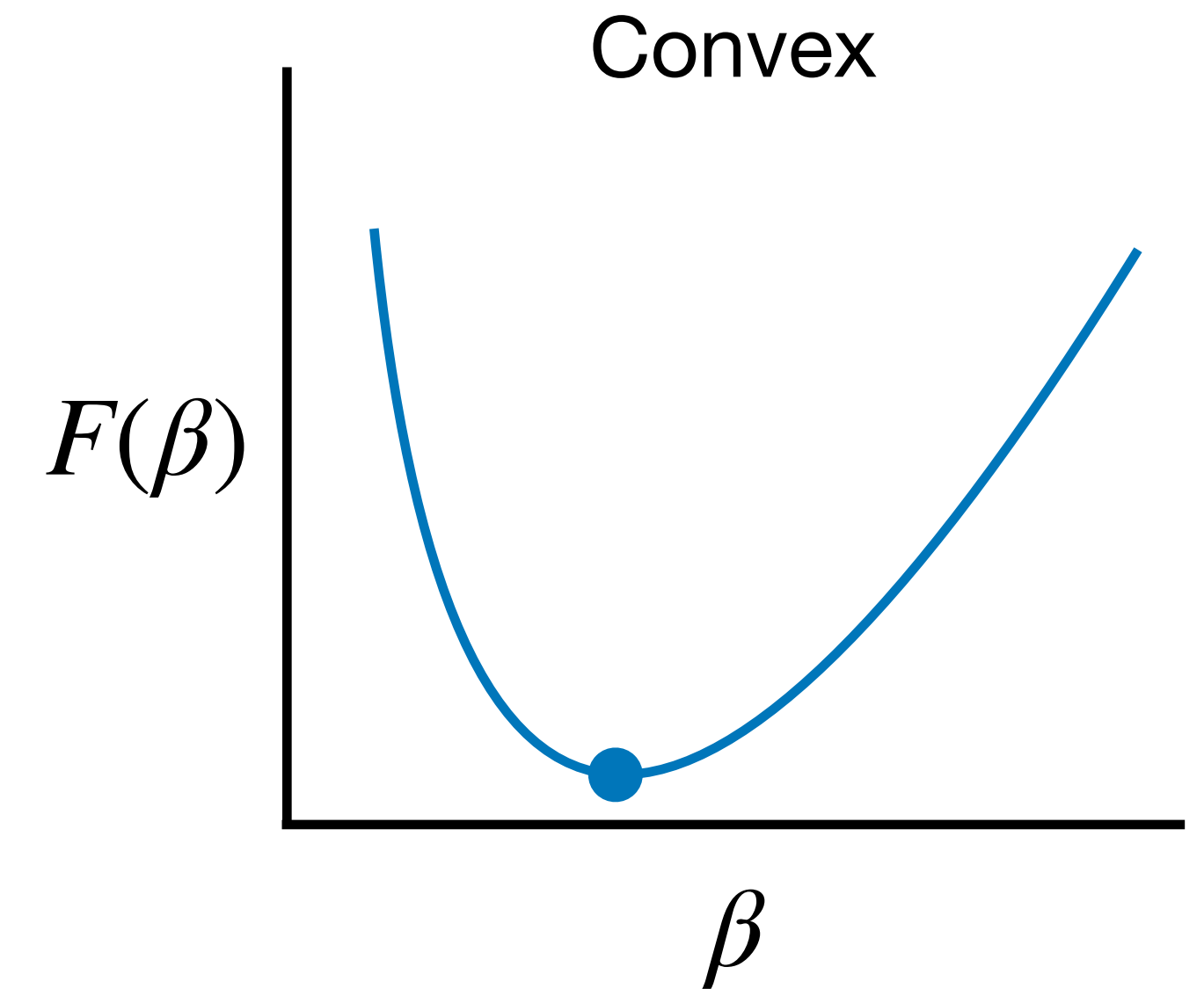


Gradient descent for non-convex optimization

Think about gradient descent as a ball rolling down a hill.

For convex functions, there is only one place (the global minimum) for the ball to roll, no matter where it starts.

For non-convex functions, the ball can roll into any of the local minima, most of which are not global minima.

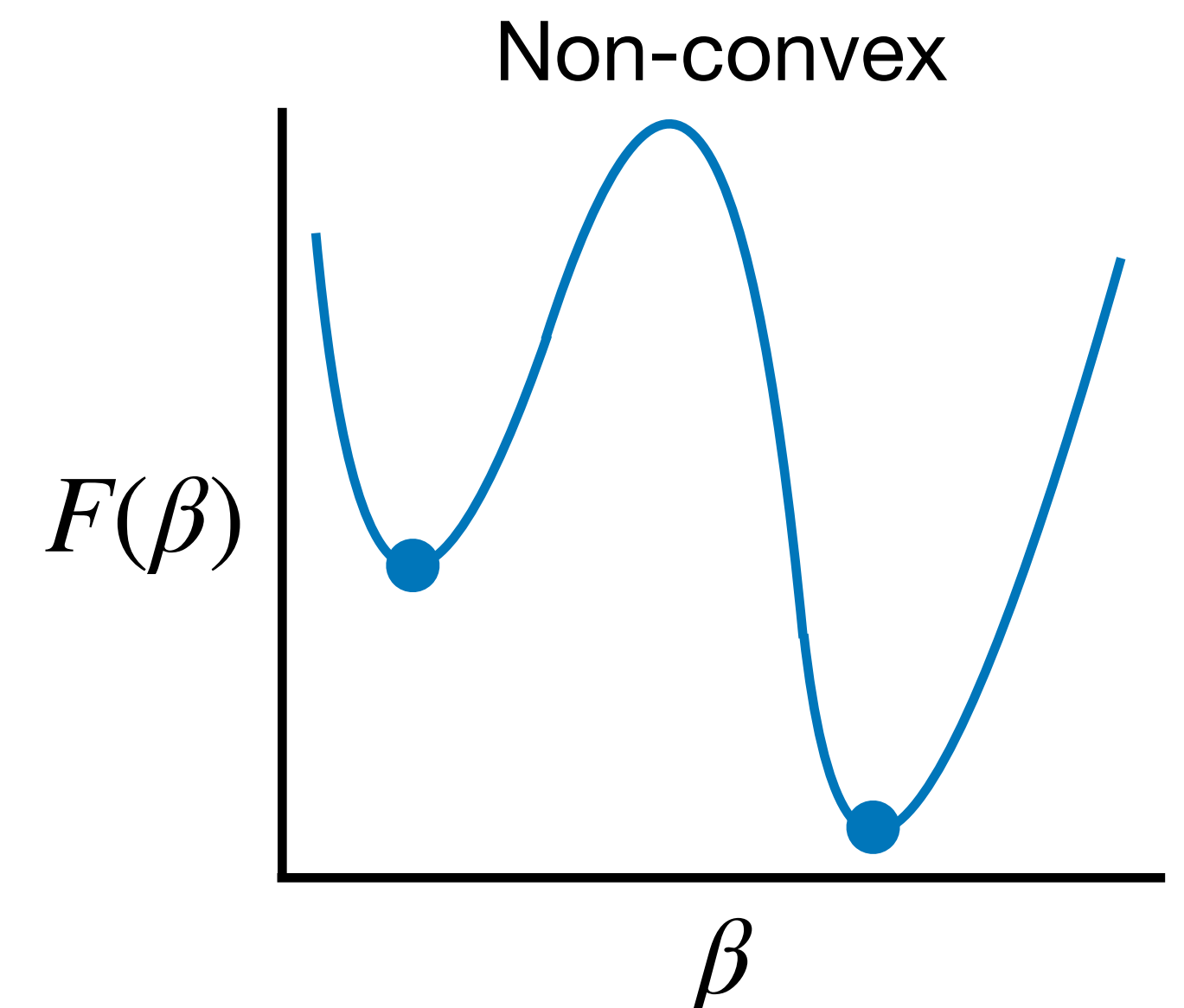
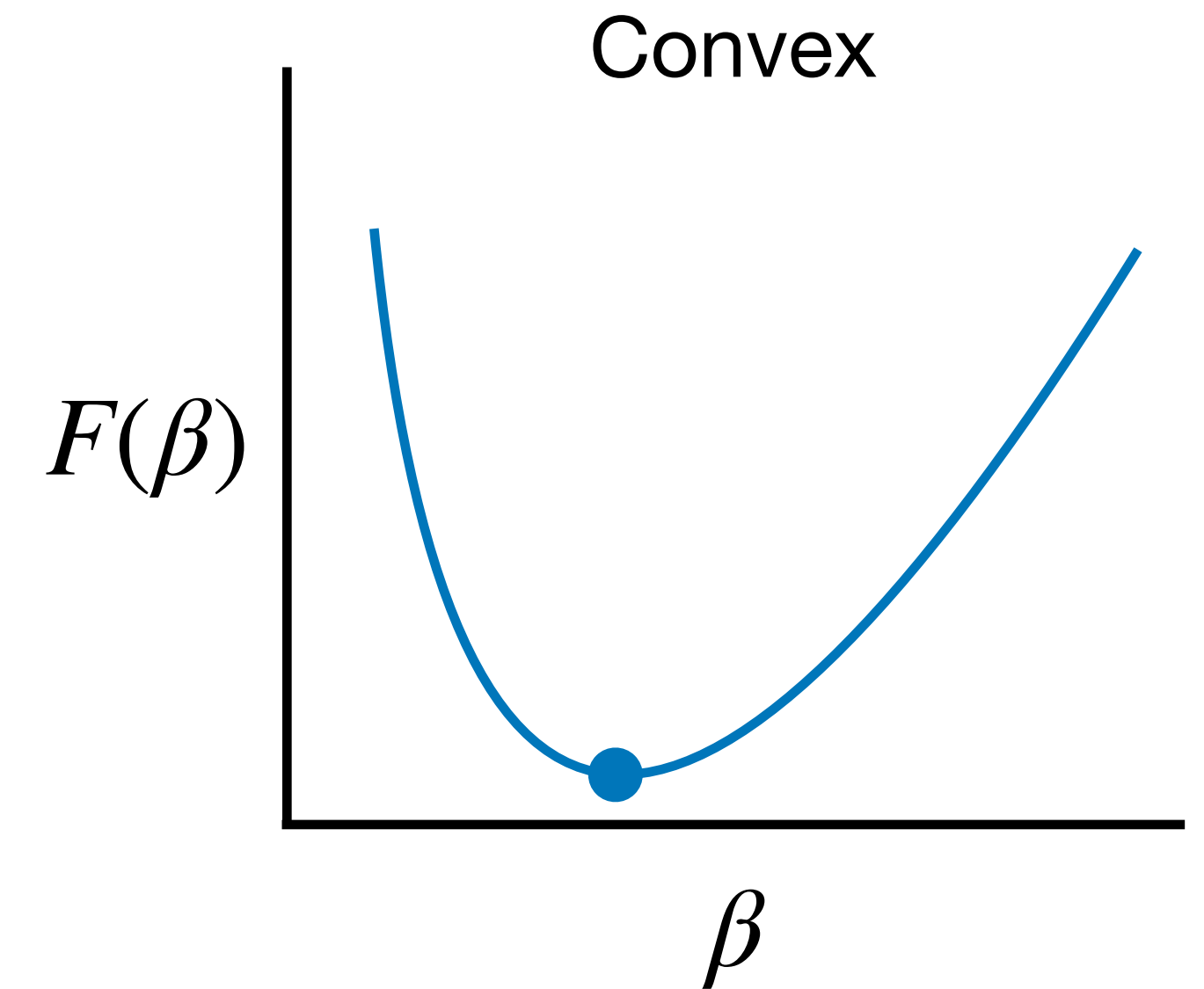


Gradient descent for non-convex optimization

Think about gradient descent as a ball rolling down a hill.

For convex functions, there is only one place (the global minimum) for the ball to roll, no matter where it starts.

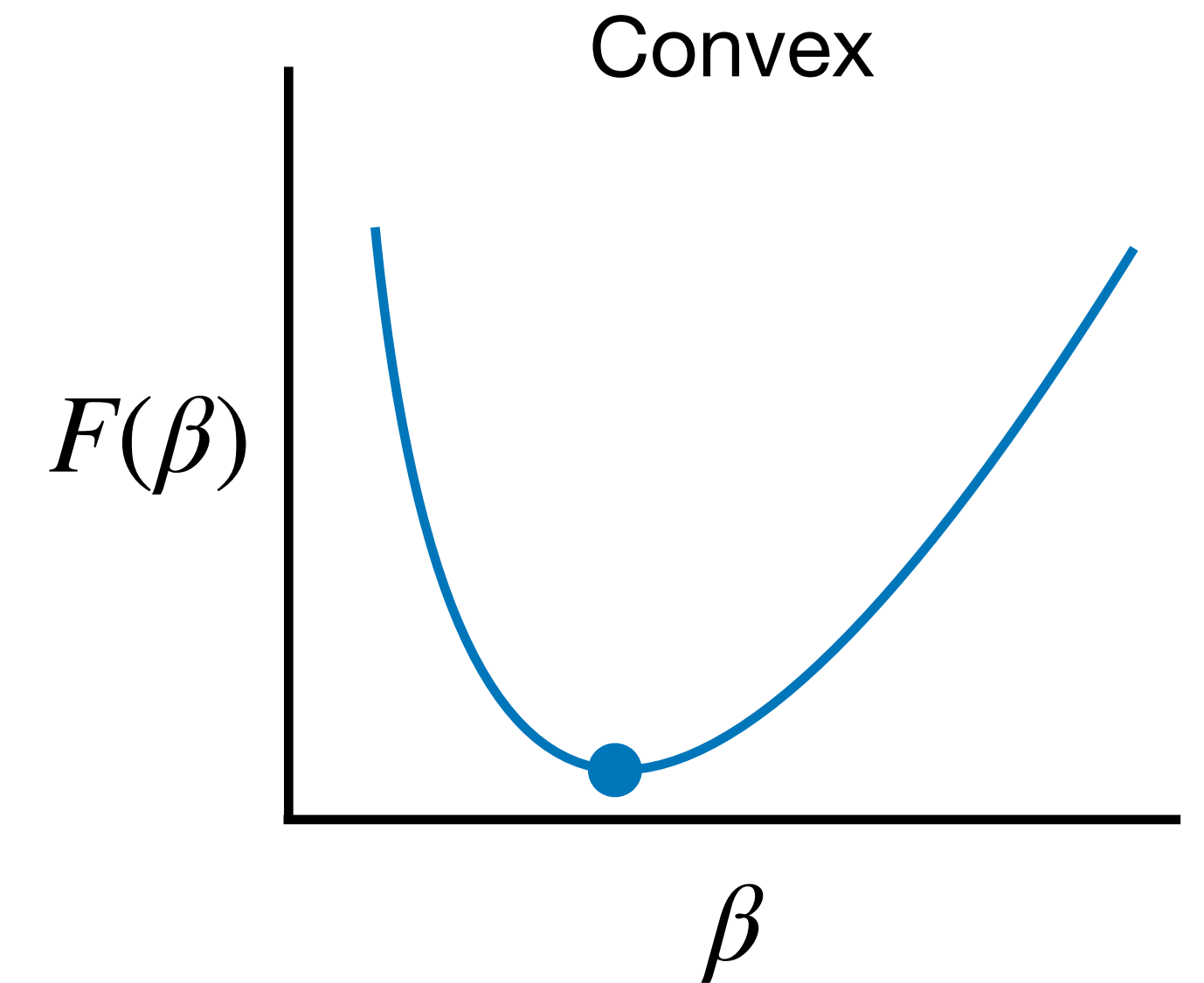
For non-convex functions, the ball can roll into any of the local minima, most of which are not global minima.



Gradient descent for non-convex optimization

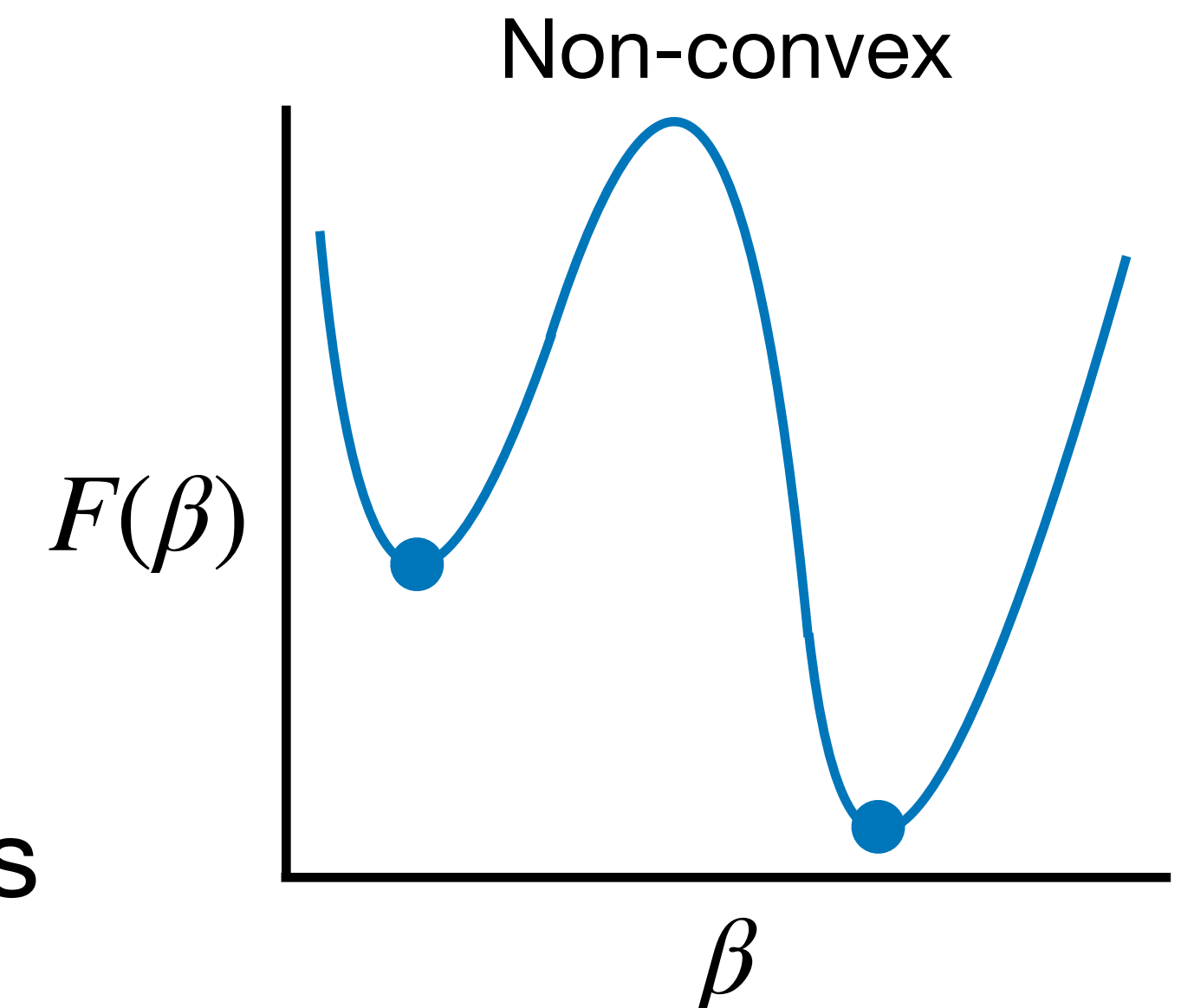
Think about gradient descent as a ball rolling down a hill.

For convex functions, there is only one place (the global minimum) for the ball to roll, no matter where it starts.



For non-convex functions, the ball can roll into any of the local minima, most of which are not global minima.

While it is computationally infeasible to find global minima for non-convex optimization,



- Local minima may still give reasonable models
- Other tricks, like multiple restarts, give better solutions

Summary

Summary

- We can think of certain predictive models as graphs.

Summary

- We can think of certain predictive models as graphs.
- We extended logistic regression to the case of more than two output classes, and defined the **cross-entropy loss** that is used for training such models.

Summary

- We can think of certain predictive models as graphs.
- We extended logistic regression to the case of more than two output classes, and defined the **cross-entropy loss** that is used for training such models.
- Solving optimization problems is a key part of training predictive models.

Summary

- We can think of certain predictive models as graphs.
- We extended logistic regression to the case of more than two output classes, and defined the **cross-entropy loss** that is used for training such models.
- Solving optimization problems is a key part of training predictive models.
- Hardness of optimization depends on whether objective function is **convex**; linear and logistic regression are convex but trees and neural networks are not.

Summary

- We can think of certain predictive models as graphs.
- We extended logistic regression to the case of more than two output classes, and defined the **cross-entropy loss** that is used for training such models.
- Solving optimization problems is a key part of training predictive models.
- Hardness of optimization depends on whether objective function is **convex**; linear and logistic regression are convex but trees and neural networks are not.
- **Gradient descent** is a common way to “go downhill” along an objective function, arriving at a local minimum (and for convex objectives, a global one).

Summary

- We can think of certain predictive models as graphs.
- We extended logistic regression to the case of more than two output classes, and defined the **cross-entropy loss** that is used for training such models.
- Solving optimization problems is a key part of training predictive models.
- Hardness of optimization depends on whether objective function is **convex**; linear and logistic regression are convex but trees and neural networks are not.
- **Gradient descent** is a common way to “go downhill” along an objective function, arriving at a local minimum (and for convex objectives, a global one).

[Quiz Practice](#)