INF721

2024/2

UFV

Deep Learning

L8: Regularization

Logistics

Announcements

- Midterm I is next week!
- FP1 - Project Proposal is out!

Last Lecture

- Dataset Splitting Techniques
- Regression evaluation metrics
  - MSE, MAE, RMSE, R-squared
- Classification evaluation metrics
  - Confusion matrix
  - Accuracy, precision, recall, f1-score

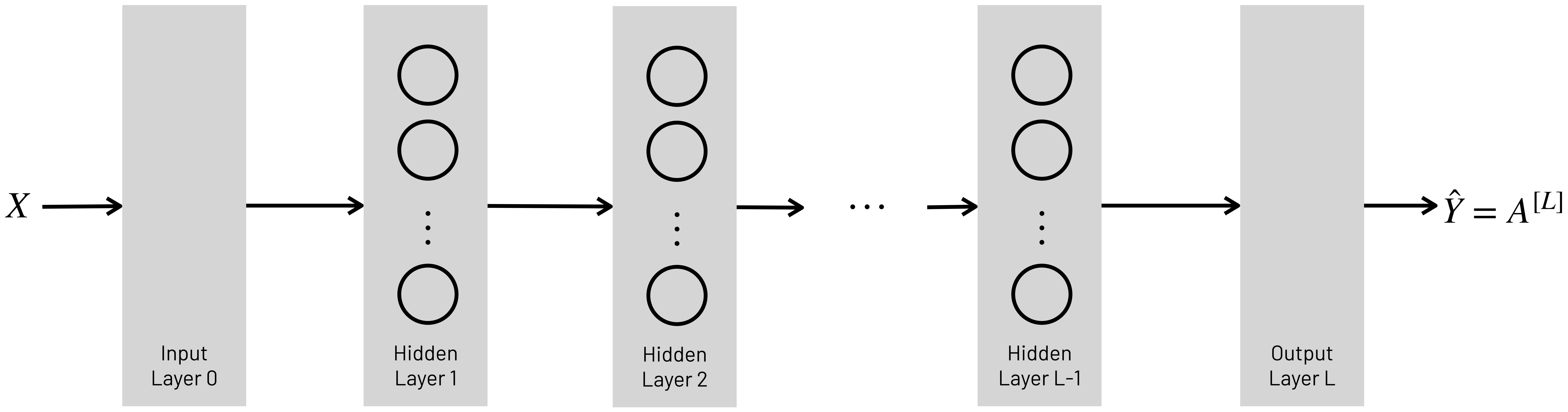
Lecture Outline

- Experiments with neural networks
- Dealing with underfiting
- Dealing with overfitting
  - Regularization
    - L1 Regularization
    - L2 Regularization
    - Dropout

Experimenting With Neural Networks

How de we choose number of hidden layers, number of hidden units, activation funtions, learning rate, ...?

Experiment with different configurations and pick the one with best performance on the validation set!



UFV

Experiments with Neural Networks

Different results can be obtained when experimenting with neural networks:

Training error	High	Low	Low
Validation error	High	High	Low
	Underfit (High bias)	**Overfit (High variance)**	**Good Fit**
			Our goal!

UFV

Experiments with Neural Networks

Image Classification of cats vs. dogs

Assume balanced dataset and a human baseline with prediction accuracy ~100%

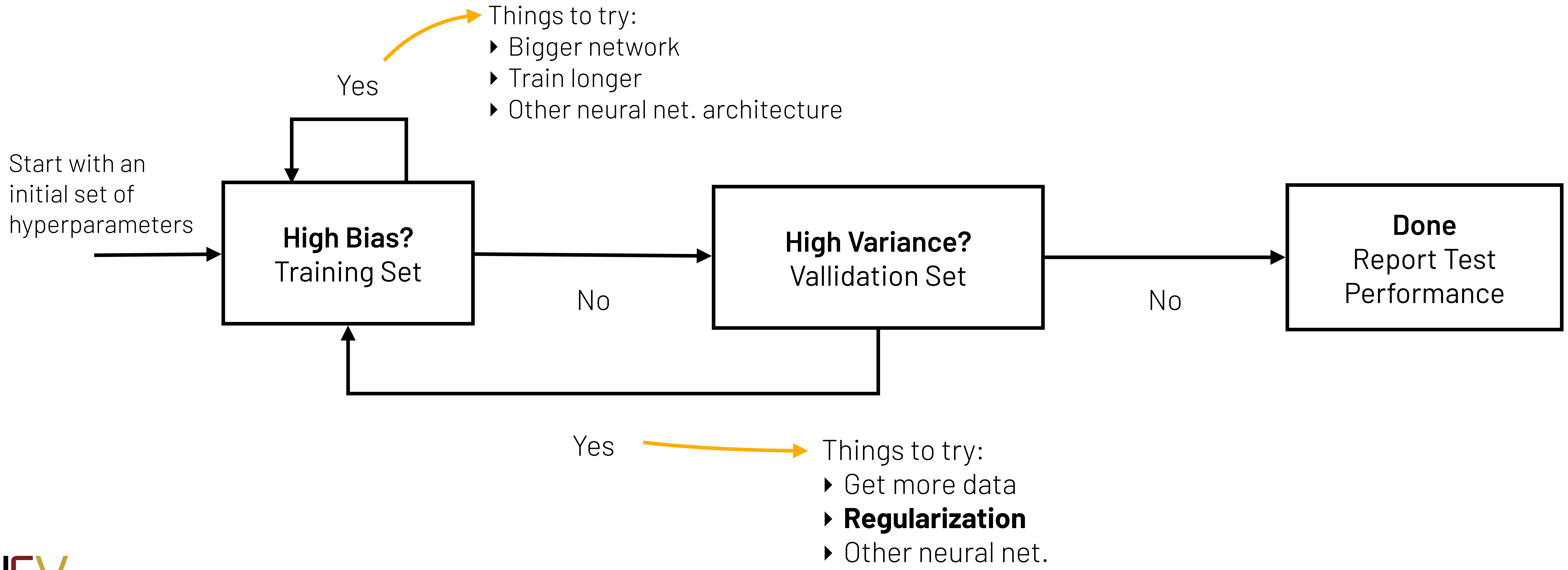
Accuracy	45%	99%	95%
Accuracy	42%	67%	94%
	Underfit (High bias)	**Overfit (High variance)**	**Good Fit**

Our goal!

UFV

Experimenting With Neural Networks

It is almost impossible to guess the write values for hyperparameters in your first attempt to building a neural network, so here is a basic experimental recipe:



Regularization

In Machine Learning, **regularization** consistst of simplifying models with the goal of reducing overfit:

- L1 regularization
- L2 regularization
- Dropout
- Early stopping (training for less time)
- Augmenting the dataset

UFV

Vector Norms

In Linear Algera, a **norm** is a function $||\cdot|| : X \rightarrow \mathbb{R}^+$ that maps a vector into a real non-negative number with the following properties:

For any vectors $\mathbf{x}, \mathbf{y} \in X$ e $\alpha \in \mathbb{R}$:

1. $||\cdot|| \geq 0$ and $||\mathbf{x}|| = 0$ if $\mathbf{x} = 0$
2. $||\mathbf{x} + \mathbf{y}|| \leq ||\mathbf{x}|| + ||\mathbf{y}||$
3. $||\alpha\mathbf{x}|| = |\alpha|||\mathbf{x}||$

Vector l^p-norms

Norms l^p are an especial type of norm, defined as follows:

$$l^p = \|\mathbf{x}\|_p = (\sum_{i=1}^{n} |x_i|^p)^{\frac{1}{p}}$$

Two l^p norms are very common:

- Norm $l^1 = \|\mathbf{x}\|_1 = (\sum_{i=1}^{n} |x_i|^1)^{\frac{1}{1}} = \sum_{i=1}^{n} |x_i|$
- Norm $l^2 = \|\mathbf{x}\|_2 = (\sum_{i=1}^{n} |x_i|^2)^{\frac{1}{2}} = \sqrt{(\sum_{i=1}^{n} |x_i|^2)}$ – Euclidian norm

Exercise: Vector Norms

Compute the norm l^1 and l^2 for the following weight vector:

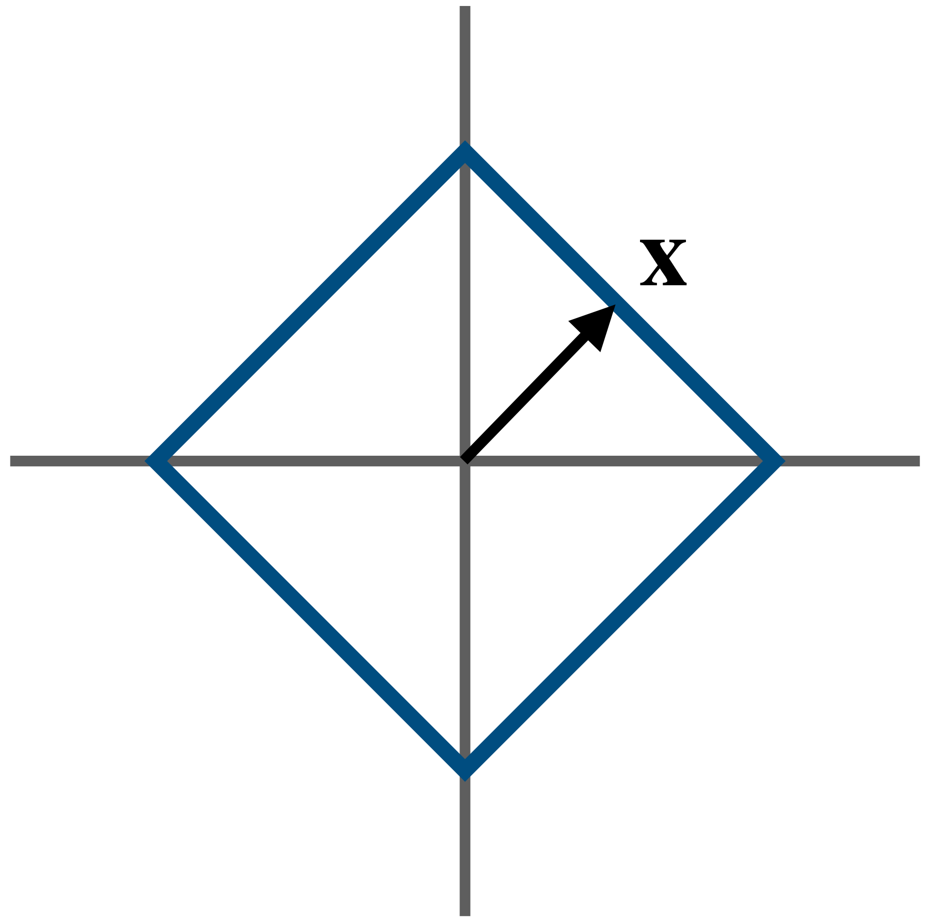
$\mathbf{w} = [-1,2]$

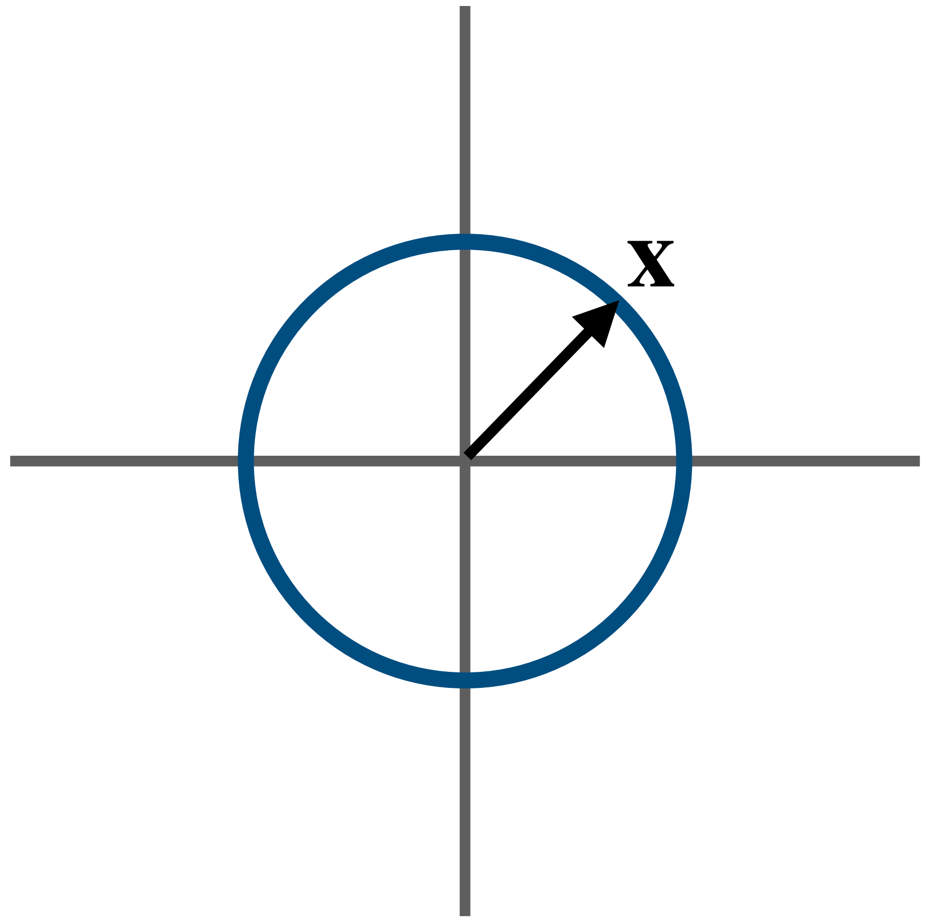
$$\|\mathbf{x}\|_1 = (\sum_{i=1}^{n} |x_i|)$$

$$\|\mathbf{x}\|_2 = \sqrt{(}\sum_{i=1}^{n} |x_i|^2)$$

Geometric Representation of Vector Norms l_p

Unit circle $(\mathbf{x} \in \mathbb{R}^2 : \|\mathbf{x}\| = 1)$ for norms l^1 and l^2:





$$l^1 = \|\mathbf{x}\|_1 = (\sum_{i=1}^{n} |x_i|) \qquad l^2 = \|\mathbf{x}\|_2 = \sqrt{(\sum_{i=1}^{n} |x_i|^2)}$$

UFV

Matrix Norms

Matrix norms are functions that map a matrxi into a real non-negative number with the same properties of the vector norms. The matrix norms $\|\cdot\|_p$ treat a matrix $m \times n$ as a vector with mn dimensions:

$$\|A\|_p = (\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^p)^{\frac{1}{p}}$$

Two very popular matrix norms $\|\cdot\|_p$ are:

- Norm L1 $\|A\|_1 = \sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|$
- Norm L2 (Frobenius) $\|A\|_2 = \sqrt{(\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2)}$

UFV

Exercise: Matrix Norms

$$\|A\|_1 = \sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|$$

$$\|A\|_2 = \sqrt{(} \sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2)$$

Calculate the norm 1 and 2 for the following weight matrices:

$$W = \begin{bmatrix} 0.1 & -0.05 \\ 0.02 & 0.15 \end{bmatrix}$$

UFV

L1 Regularization

L1 regularization sums **the norm** $\|\cdot\|_1$ to the loss function to penalize neural networks with weights with high values:

$$L(h) = -\frac{1}{m}\sum_{i=1}^{m} L(y^{(i)}, \hat{y}^{(i)}) + \frac{\lambda}{2m}\sum_{l}\|W^{[l]}\|_1$$

In linear/logistic regression, we use the vector norm instead of the matrix one!

where λ is a hyperparameter controlling the penalization.

$$\|W^{[l]}\|_1 = \sum_{i=1}^{n^{[l-1]}}\sum_{j=1}^{n^{[l]}} |a_{ij}|$$

L1 regularization makes the weight matrix W sparse!

L2 Regularization

L2 regulariztion sums **the square of the norm** $||\cdot||_2$ to the loss function to penalize neural networks with weights with high values:

$$L(h) = -\frac{1}{m}\sum_{i=1}^{m} L(y^{(i)}, \hat{y}^{(i)}) + \frac{\lambda}{2m}\sum_{l} ||W^{[l]}||_2^2$$

In linear/logistic regression, we use the vector norm instead of the matrix one!

where λ is a hyperparameter controlling the penalization.

$$||W^{[l]}||_2^2 = (\sqrt{(\sum_{i=1}^{n^{[l-1]}}\sum_{j=1}^{n^{[l]}} |a_{ij}|^2}))^2 = \sum_{i=1}^{n^{[l-1]}}\sum_{j=1}^{n^{[l]}} |a_{ij}|^2$$

L2 regularization decays the weight matrix W over time, but doesn't tend to make weights exactly zero!

UFV

Exercise: Regularization

Considering a weight metrix $W = \begin{bmatrix} 0.1 & -0.05 \\ 0.02 & 0.15 \end{bmatrix}$, gradients $dW = \begin{bmatrix} 0.3 & 0.2 \\ 0.1 & -0.4 \end{bmatrix}$ and a learning rate of $\alpha = 0.1$, show how the weights would be updated after one step of gradient descent.

a) Gradient Descent with L1 regularization: $W = W - \alpha(dW + \frac{\lambda}{m} sign(W))$

b) Gradient Descent with L2 regularization: $W = W - \alpha(dW + \frac{\lambda}{m} W)$

UFV

The Effect of L2 Regularization

Weight update without regularization:

$$W^{[l]} = W^{[l]} - \alpha \boxed{dW^{[l]}}$$ Partial derivative of the loss function with respect to W^l

Weight update with regularization:

$$W^{[l]} = W^{[l]} - \alpha\boxed{(dW^{[l]} + \frac{\lambda}{m}W^{[l]})}$$ Partial derivative of the regularized loss function with respect to W^l

$$W^{[l]} = W^{[l]} - \frac{\alpha\lambda}{m}W^{[l]} - \alpha dW$$

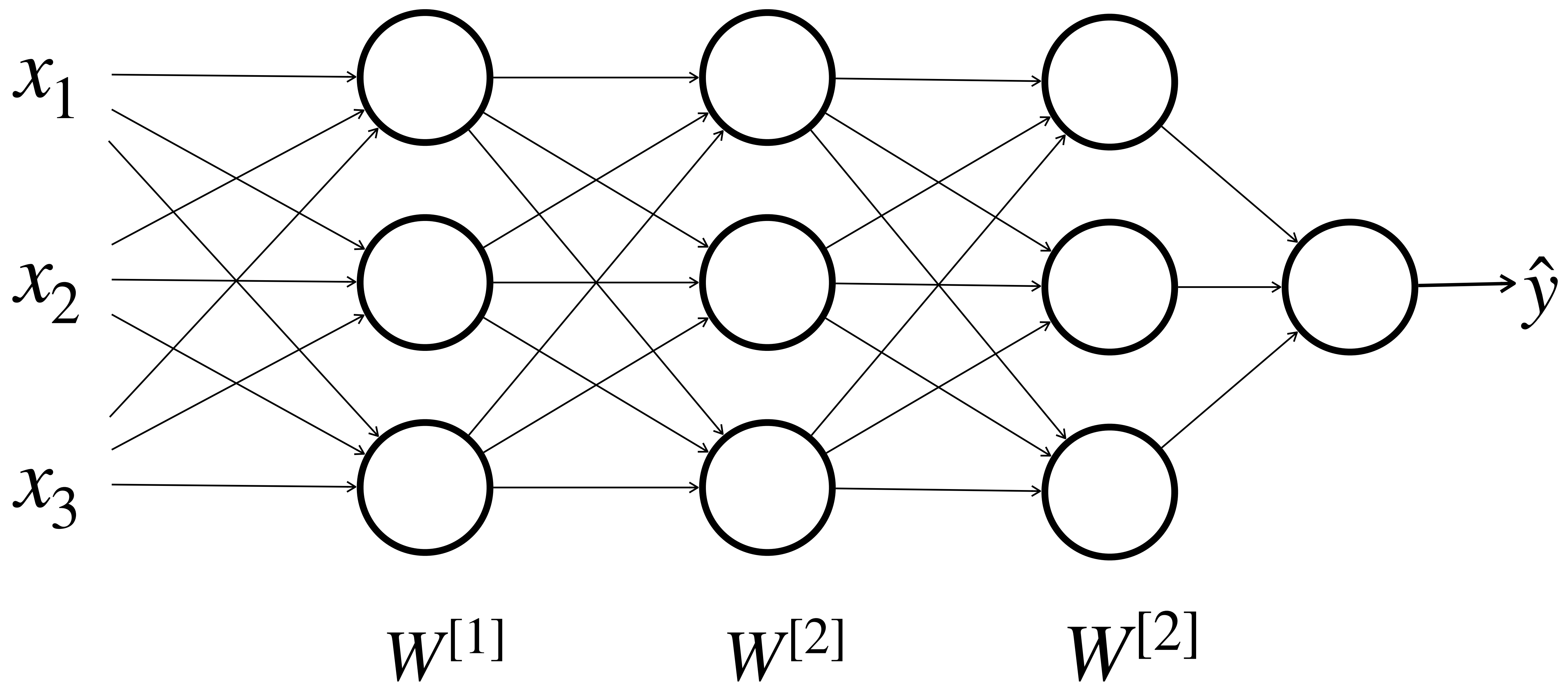
$$W^{[l]} = \underbrace{(1 - \frac{\alpha\lambda}{m})}_{<1}W^{[l]} - \alpha dW$$

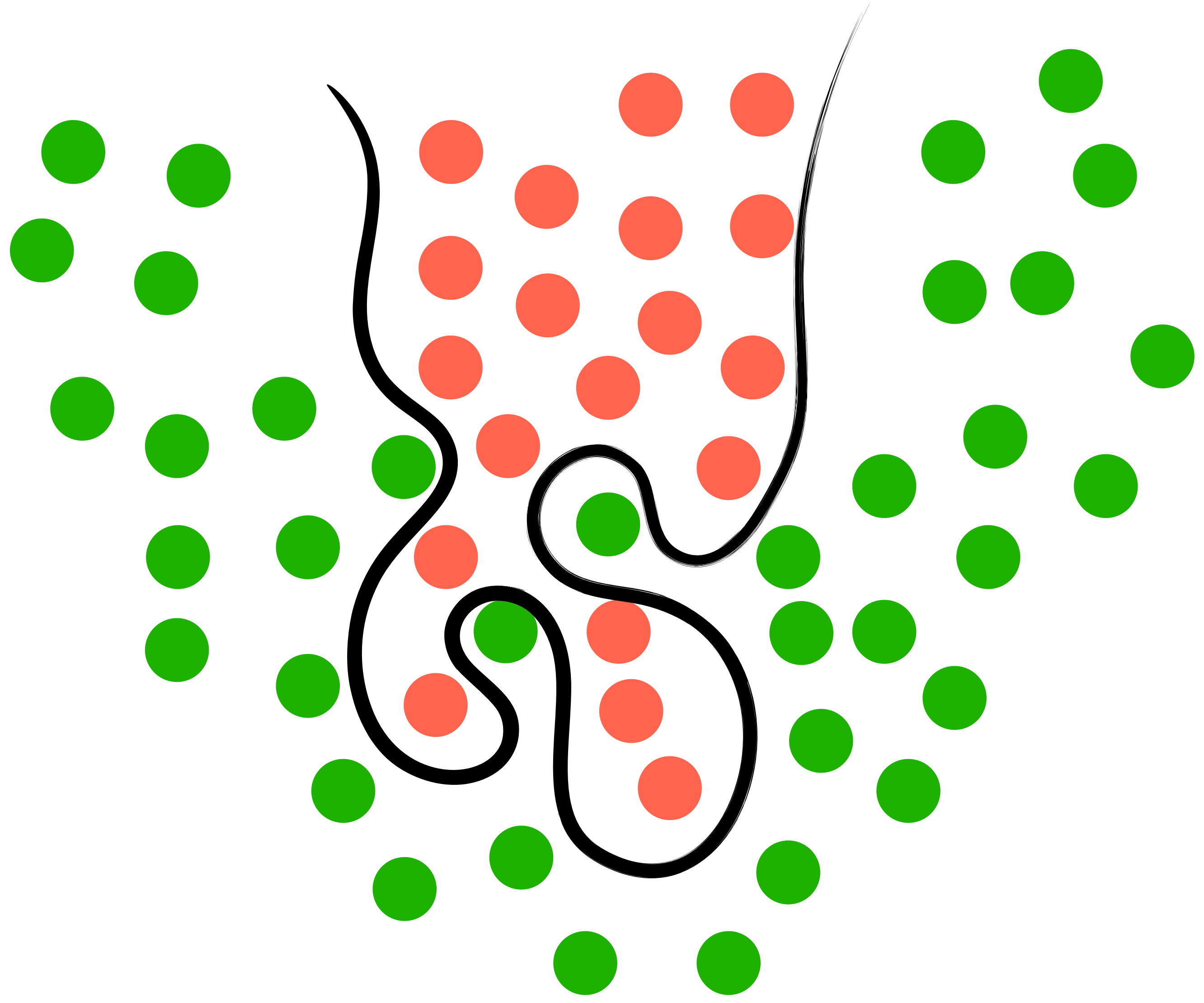
< 1 → L2 regularization decreases the values of weights $W^{[l]}$ and because of that it's also called *Weight Decay.*

UFV

Why regularization prevents overfitting?

$$L(h) = -\frac{1}{m}\sum_{i=1}^{m} L(y^{(i)}, \hat{y}^{(i)})$$

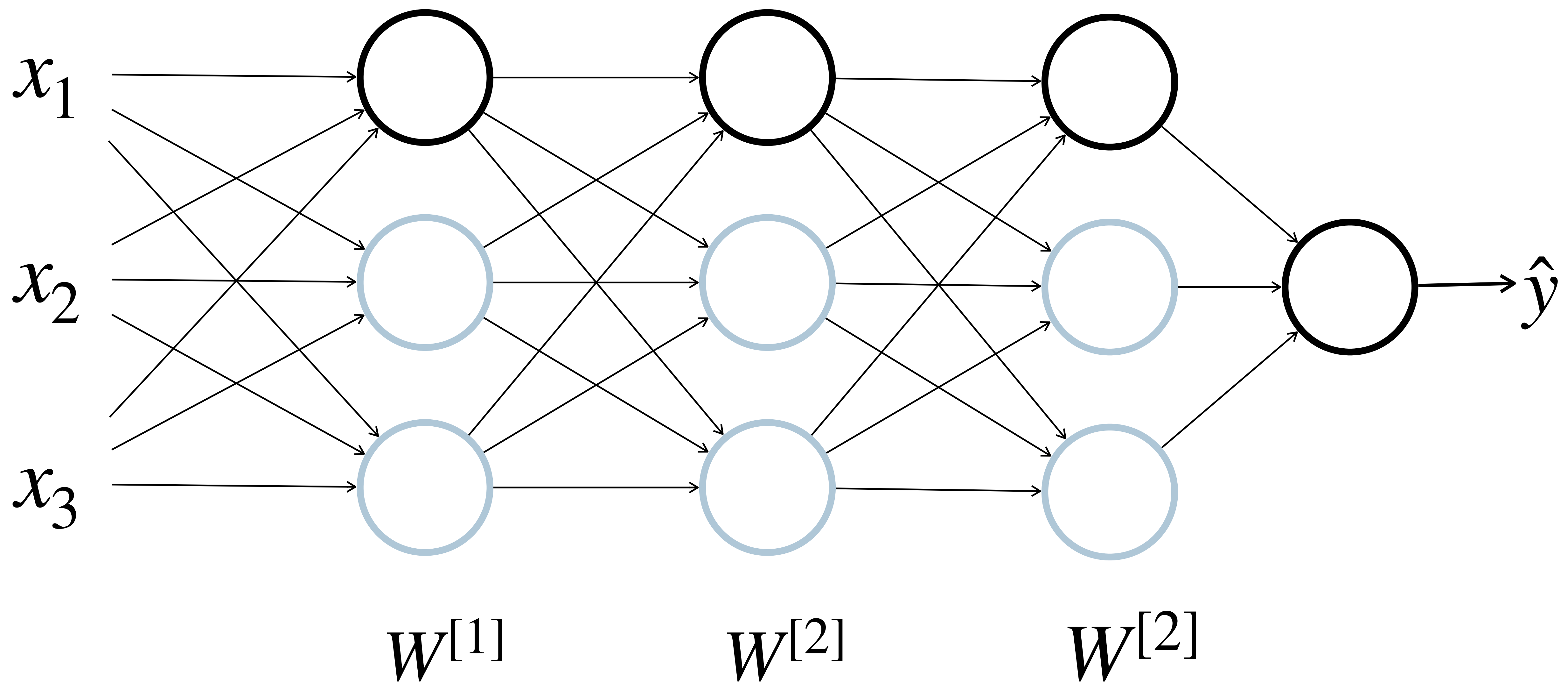


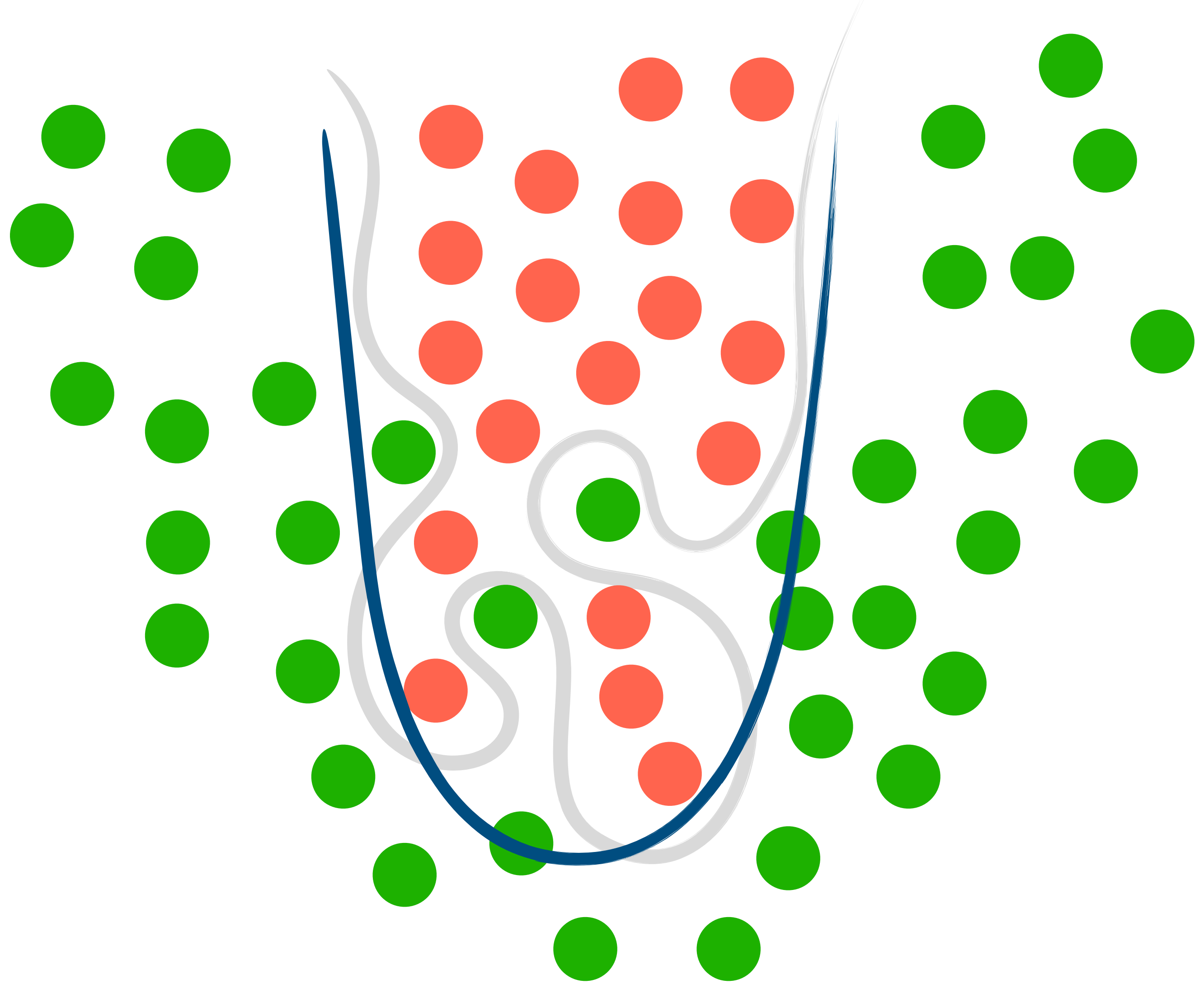


Consider a neural network with 4 layers that is overfitting when trained with loss function L.
Notice how the decision boundary is capturing the details of the training data.

Why regularization prevents overfitting?

$$L(h) = -\frac{1}{m}\sum_{i=1}^{m} L(y^{(i)}, \hat{y}^{(i)}) + \frac{\lambda}{2m}\sum_{l} \|W^{[l]}\|_2^2 \longrightarrow W^{[l]} \approx 0$$

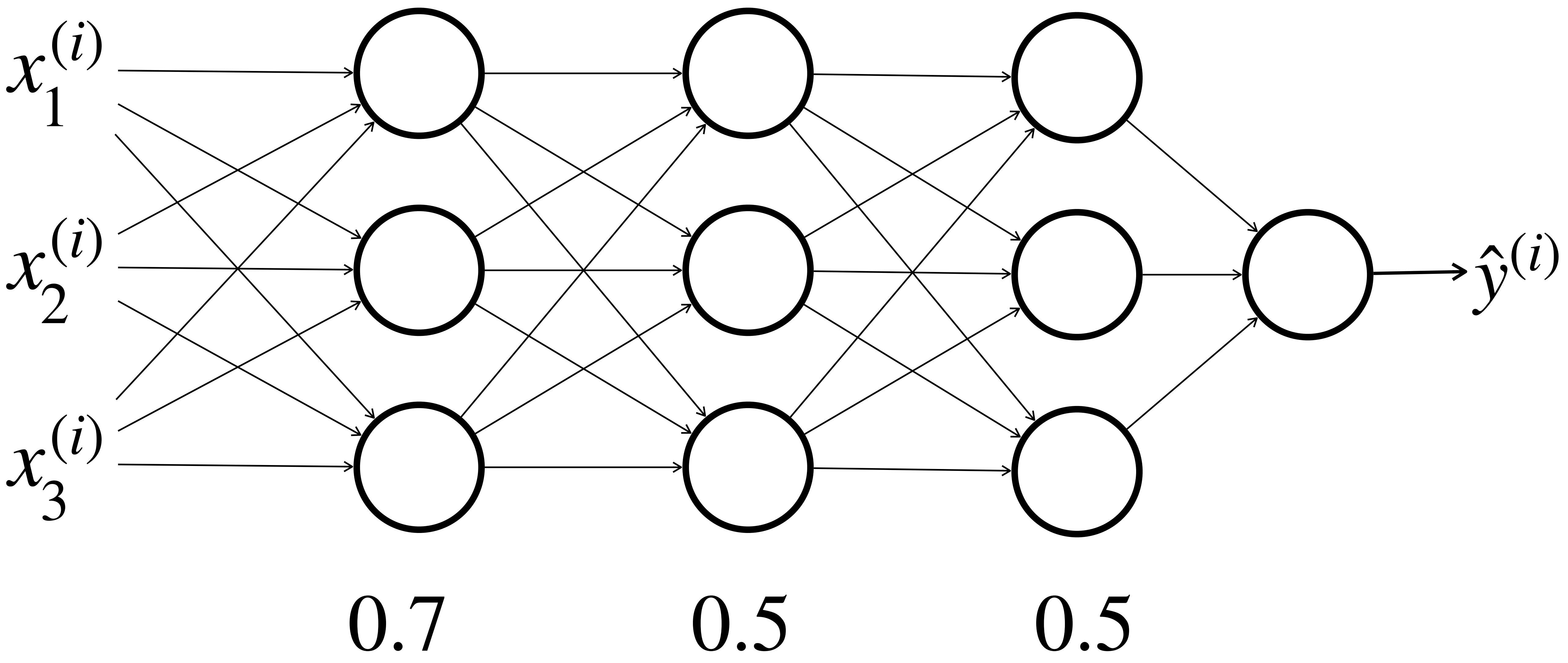




By reducing the weights of some neurons, regularization simplifies the assumption of a neural networks at training time, making the decision boundary simpler as well.

Dropout

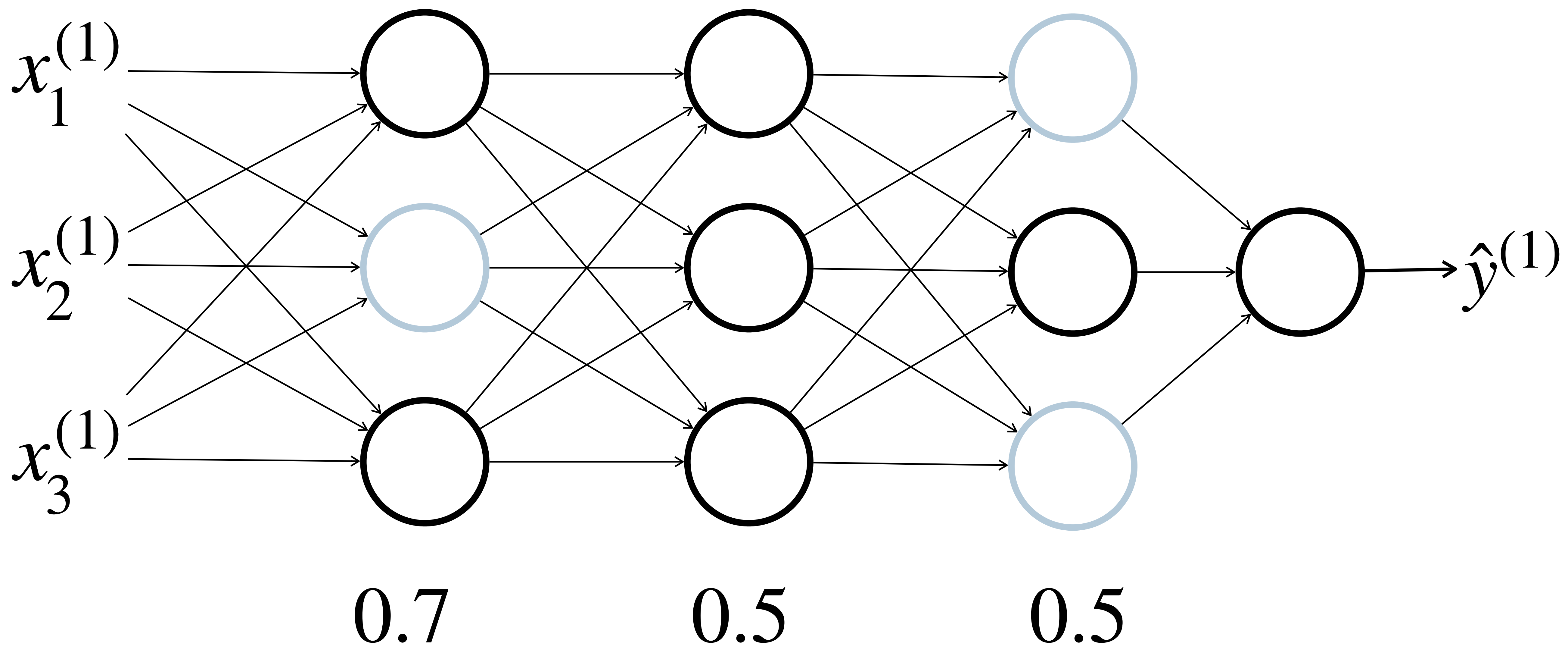
Dropout is a regularization technique that disables random neurons before calculating the error for each example in the training set.



Each layer is given a probability to keep the neurons in that layer active before calculating the error for each example (i).

Dropout

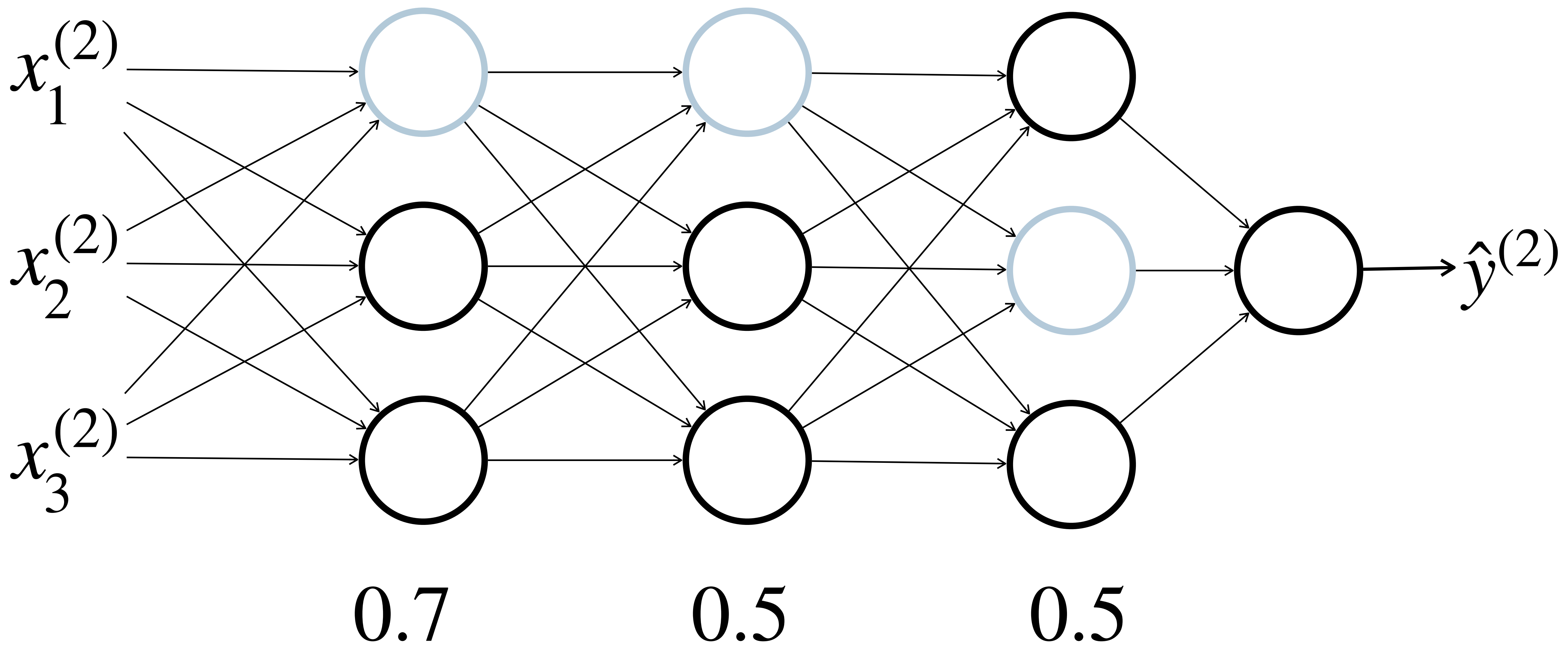
Dropout is a regularization technique that disables random neurons before calculating the error for each example in the training set.



UFV

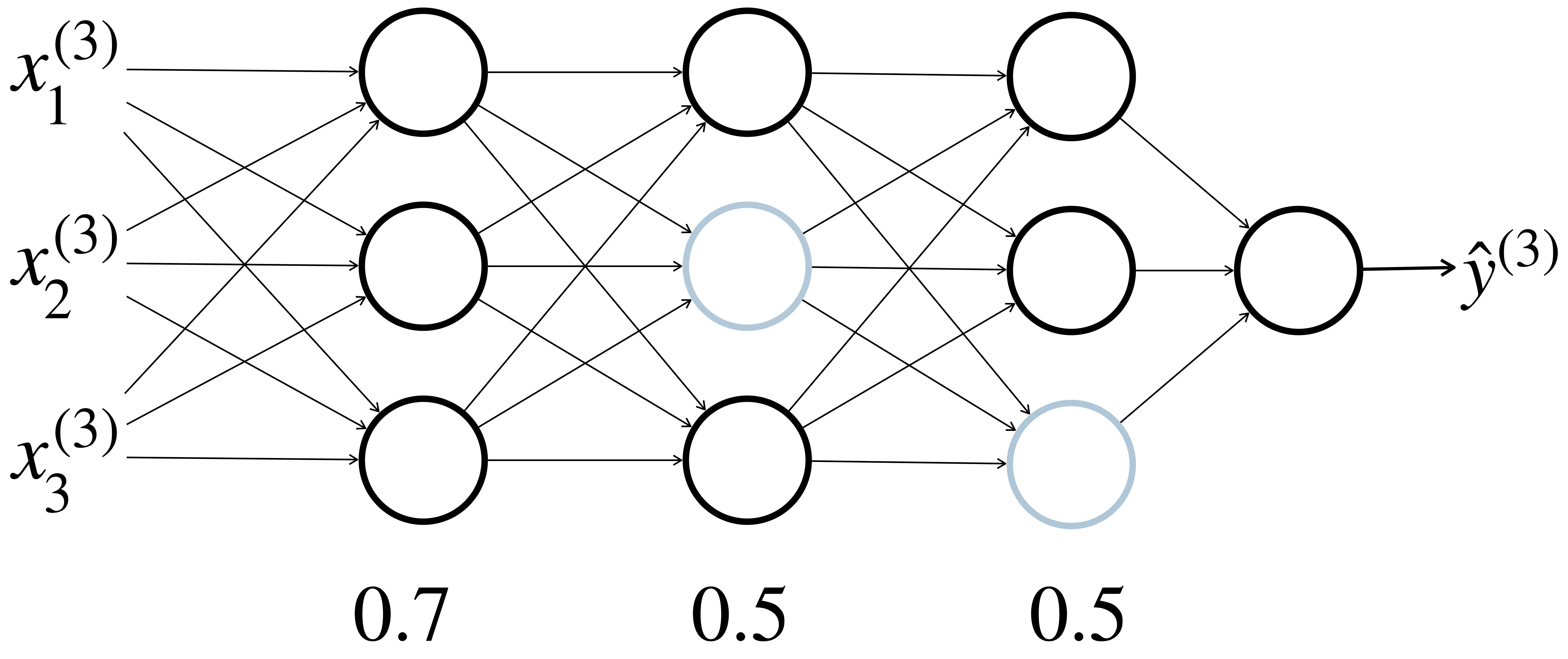
Dropout

Dropout is a regularization technique that disables random neurons before calculating the error for each example in the training set.



Dropout

Dropout is a regularization technique that disables random neurons before calculating the error for each example in the training set.



A different neural network configuration is trained for each example (i), forcing a distribution of weights among the neurons of a layer in a more uniform way, not on just one or a few inputs.

Data Augmentation

Data Augmentation consists of generating new examples to your dataset by applying transformations the original examples of your dataset:

Original

Flip

Rotate

Original

Distort

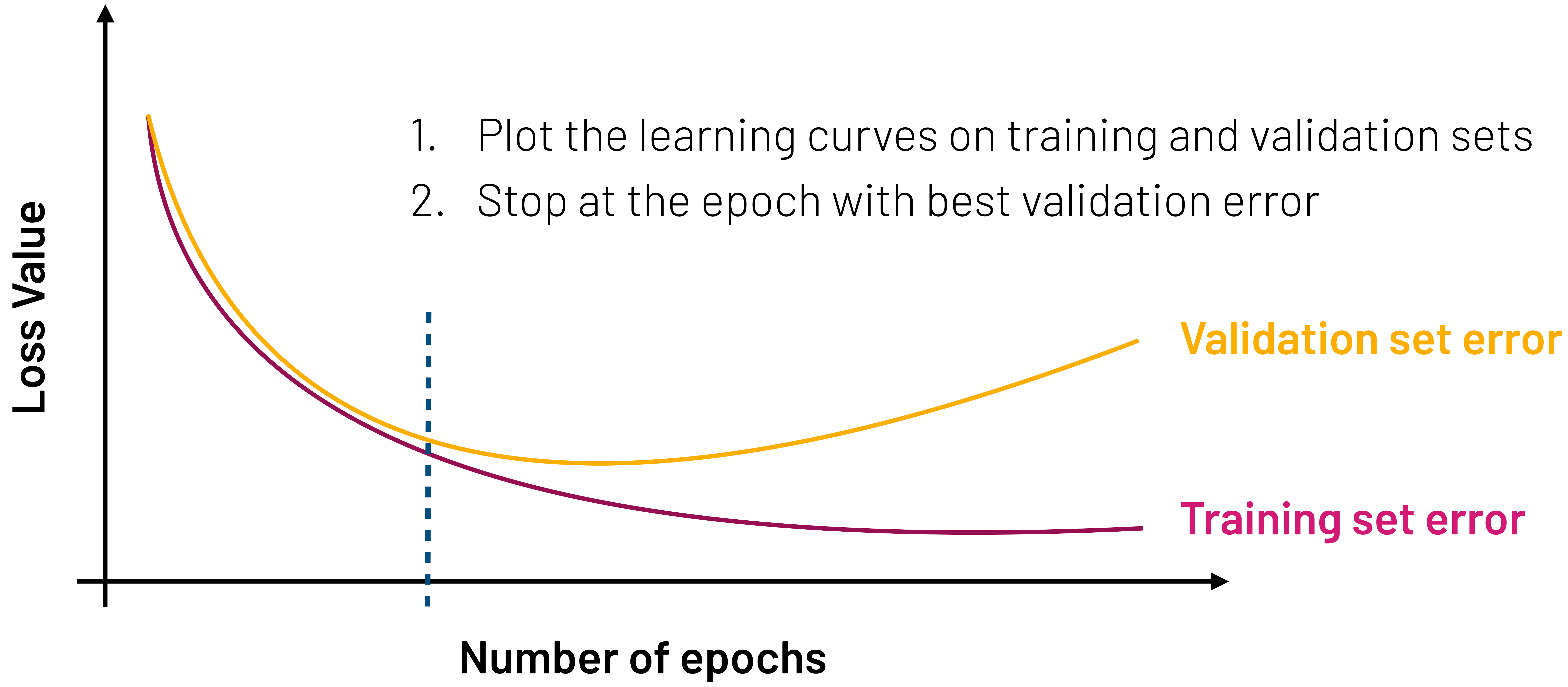
Distort + Rotate

UFV

Early Stopping

Early stopping consists of running gradient descent for less epochs.

1. Plot the learning curves on training and validation sets
2. Stop at the epoch with best validation error



UFV

Next Lecture

L9: Advanced Optimization Algorithms

Mini-batch Gradient Descent, RMSProp, Adam