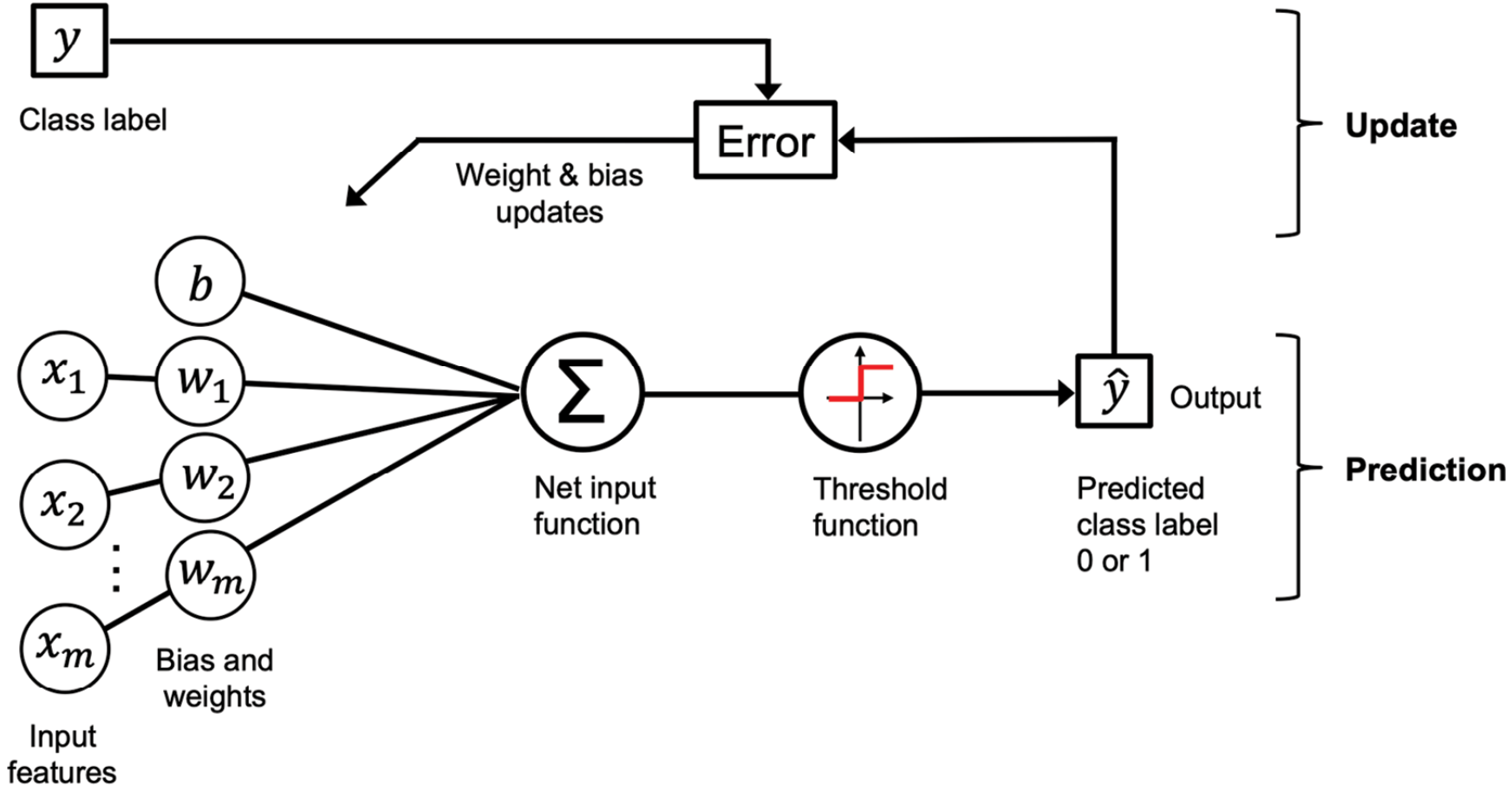
THE UNIVERSITY of TENNESSEE KNOXVILLE

Lecture 31: Multilayer Perceptron

Sergei V. Kalinin

Training Linear Neuron



From S. Raschka, Machine Learning with PyTorch and Scikit-Learn

Training Linear Neuron

- Initialize the weights and bias unit to 0 or small random numbers
- For each training example, ***x(i)***:
- Compute the output value, $\boldsymbol{y(i) = w^T x(i) + b}$
- Update the weights and bias unit: $w_j := w_j + \Delta w_j$ and $b := b + \Delta b$
- Where $\Delta w_j = \eta\left(y^{(i)} - \hat{y}^{(i)}\right)x_j^{(i)}$ and $\Delta b = \eta\left(y^{(i)} - \hat{y}^{(i)}\right)$

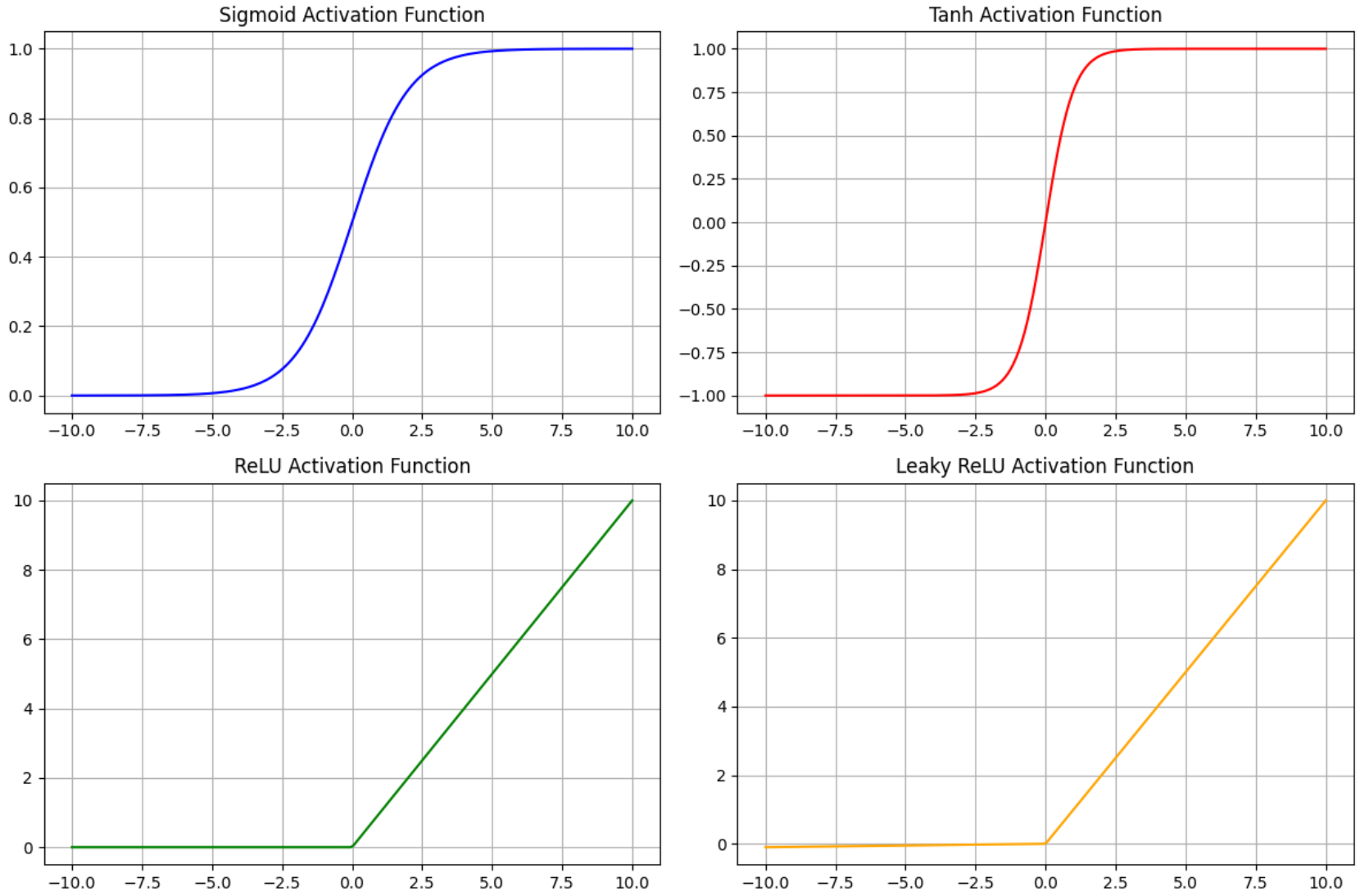
Each weight, w_j, corresponds to a feature, x_j, in the dataset,

η is the **learning rate** (typically a constant between 0.0 and 1.0),

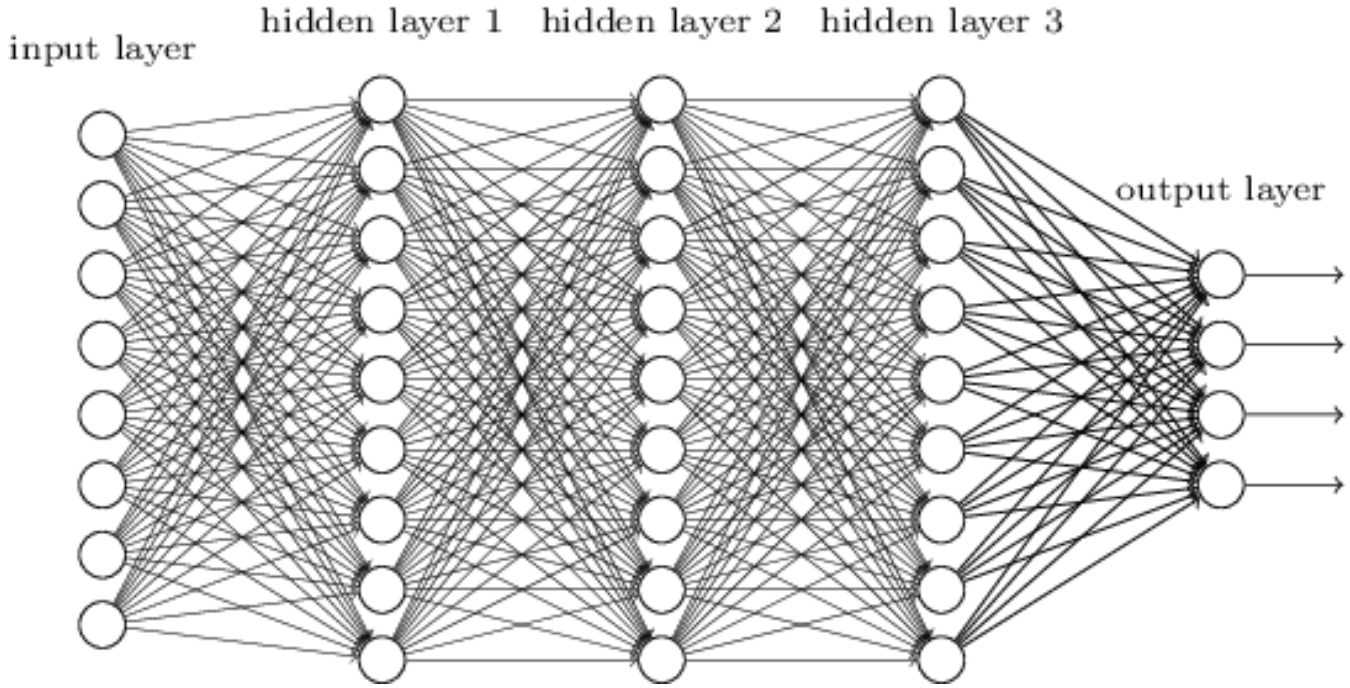
$y^{(i)}$ is the **true class label** of the *i*th training example,

$\hat{y}^{(i)}$ is the **predicted class label**

Activation functions



Putting Neurons Together



- Composed of multiple layers of artificial neurons.
- Each layer processes inputs received, applies a transformation (weights, biases, activation function), and passes the output to the next layer.
- Training a DNN involves adjusting weights and biases using backpropagation and a chosen optimization algorithm.
- The deep architecture enable the network to learn complex and abstract patterns in data.

Loss functions for supervised ML

A loss function, also known as a cost function, quantifies the difference between the predicted values and the actual target values. It guides the training of neural networks by providing a measure to minimize during optimization

Mean Squared Error (MSE): Used for regression problems.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Measures the average squared difference between actual and predicted values

Cross-Entropy Loss: Used for classification problems.

$$CE = -\sum_{i=1}^{N} y_i \log(\hat{y}_i)$$

Measures the performance of a classification model whose output is a probability value between 0 and 1

- Loss functions provide the primary feedback signal for learning.
- The choice of loss function can significantly affect the model's performance and convergence

Backpropagation

Backpropagation is a mechanism used to update the weights in a neural network efficiently, based on the error rate obtained in the previous epoch (i.e., iteration). It effectively distributes the error back through the network layers

- **Forward Pass:** Calculating the predicted output, moving the input data through the network layers
- **Loss Function:** Determining the error by comparing the predicted output to the actual output
- **Backward Pass:** Computing the gradient of the loss function with respect to each weight by the chain rule
- **Weight Update:** Adjusting the weights of the network in a direction that minimally reduces the loss (gradient descent)

Input Data → Forward Pass → Calculate Loss → Backward Pass → Update Weights

https://medium.com/@14prakash/back-propagation-is-very-simple-who-made-it-complicated-97b794c97e5c

Metrics

Quantitative measures used to evaluate and monitor the performance of the network, both during training and after deployment. Here are some common metrics used in DCNNs:

Accuracy: Defined as the ratio of correctly predicted observations to the total observations.

Precision: The ratio of correctly predicted positive observations to the total predicted positive observations. Important in scenarios where the cost of a false positive is high.

Recall (Sensitivity): The ratio of correctly predicted positive observations to all observations in the actual class. Crucial in situations where missing a positive instance is costly.

F1 Score: The harmonic mean of precision and recall. Trade-off between precision and recall.

Mean Squared Error (MSE): Commonly used in regression tasks.

Area Under the ROC Curve (AUC-ROC): Represents the degree or measure of separability achieved by the model.

Intersection over Union (IoU): Typically used in object detection and segmentation tasks. Measures the overlap between the predicted bounding box and the ground truth box.

How is metrics different from loss function?

- **Optimization vs. Evaluation:** Loss functions are used to optimize the model (i.e., to adjust the model parameters during training), whereas metrics are used to evaluate the model's performance.
- **Differentiability:** Loss functions need to be differentiable with respect to model parameters, but metrics do not have this requirement.
- **Interpretability:** Metrics are often more interpretable than loss functions and are chosen based on what makes the most sense for the specific application or domain.
- **Role in Model Training:** The loss function directly influences the training process, while metrics are more about assessing the outcome of that training.

How to not get lost?

- A vast array of network architectures ranging from Multilayer Perceptrons (MLPs) to Graph Neural Networks and Transformers
- Each architecture has unique characteristics suited for different types of data and tasks, ways to define the architecture, and so on
- Numerous methods to engineer loss functions
- Numerous ways to implement regularization

Problem-Centric Approach:

- Always start with the problem at hand: Analyze the nature of input data and desired output
- Choose a network architecture that aligns with the type and structure of your data
- Select a loss function that reflects the objective of the problem
- Metrics should be chosen based on what measures success for your specific task

Hyperparameter Tuning:

- Once the architecture and loss function are set, proceed to tune hyperparameters including network structure, optimizers, regularization, etc.
- Hyperparameter tuning should be guided by the chosen metrics and the nature of the problem

Some useful resources:

https://udlbook.github.io/udlbook/

https://dmol.pub/ml/classification.html

https://keras.io/examples/