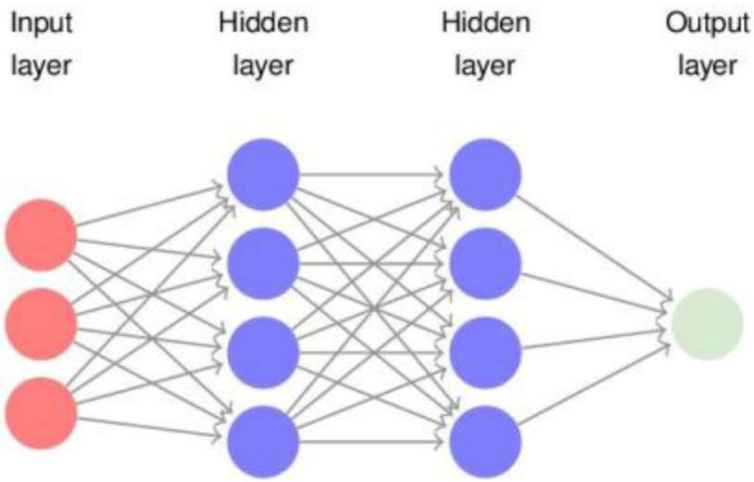
349:Machine Learning

Fall 2024

Neural Networks
Part 1

Terminology



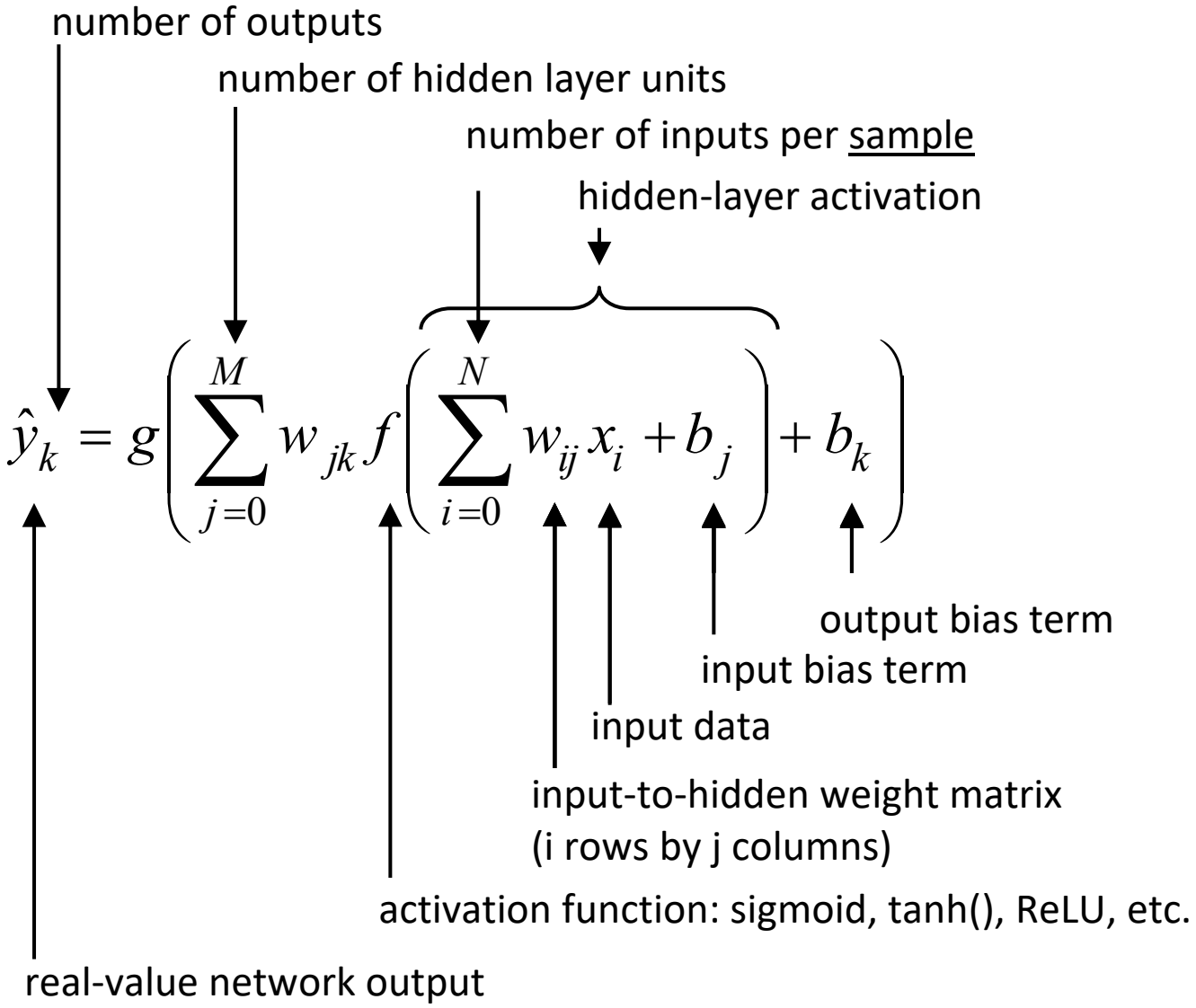
- Terms: Input layer, hidden layers, output layer
- A single hidden layer (of arbitrary width) can already be shown to be a **universal function approximator**
- Non-linear functions
  - are called activation functions in hidden layers
  - predict in the output layer and are used for the loss function

Feed Forward Neural Network: Basic Formulation

A feed forward neural network can be expressed by the following equation:

$$\hat{y}_k = g\left(\sum_{j=0}^{M} w_{jk} f\left(\sum_{i=0}^{N} w_{ij} x_i + b_j\right) + b_k\right)$$

Feed Forward Neural Network: Basic Formulation



Feed Forward Neural Network: Basic Formulation

… or more compactly in Linear Algebra form:

$$\hat{\mathbf{y}} = \sigma\left(\mathbf{W}_2 \cdot \tanh\left(\mathbf{W}_1 \cdot \mathbf{x} + \mathbf{b}_1\right) + \mathbf{b}_2\right)$$

$$\hat{y}_k = g\left(\sum_{j=0}^{M} w_{jk} f\left(\sum_{i=0}^{N} w_{ij} x_i + b_j\right) + b_k\right)$$

Characteristics

- Machine learners made of many simple functions connected by weight parameters
- Can model functions from $\mathbb{R}^d$ to $\mathbb{R}^k$ where d, k is very large (e.g. 10^5)
- Can learn arbitrary Boolean functions and very complex manifolds
- Often require lots of data (e.g. millions of examples)
- Use gradient descent and are thus susceptible to being stuck in local minima
- Opaque (internal representations are difficult to interpret)

Why We Care?

They are key technology that is enables state of the art performance in….

Image recognition (e.g. Facebook and Google image labeling)

Speech recognition (Cortana)

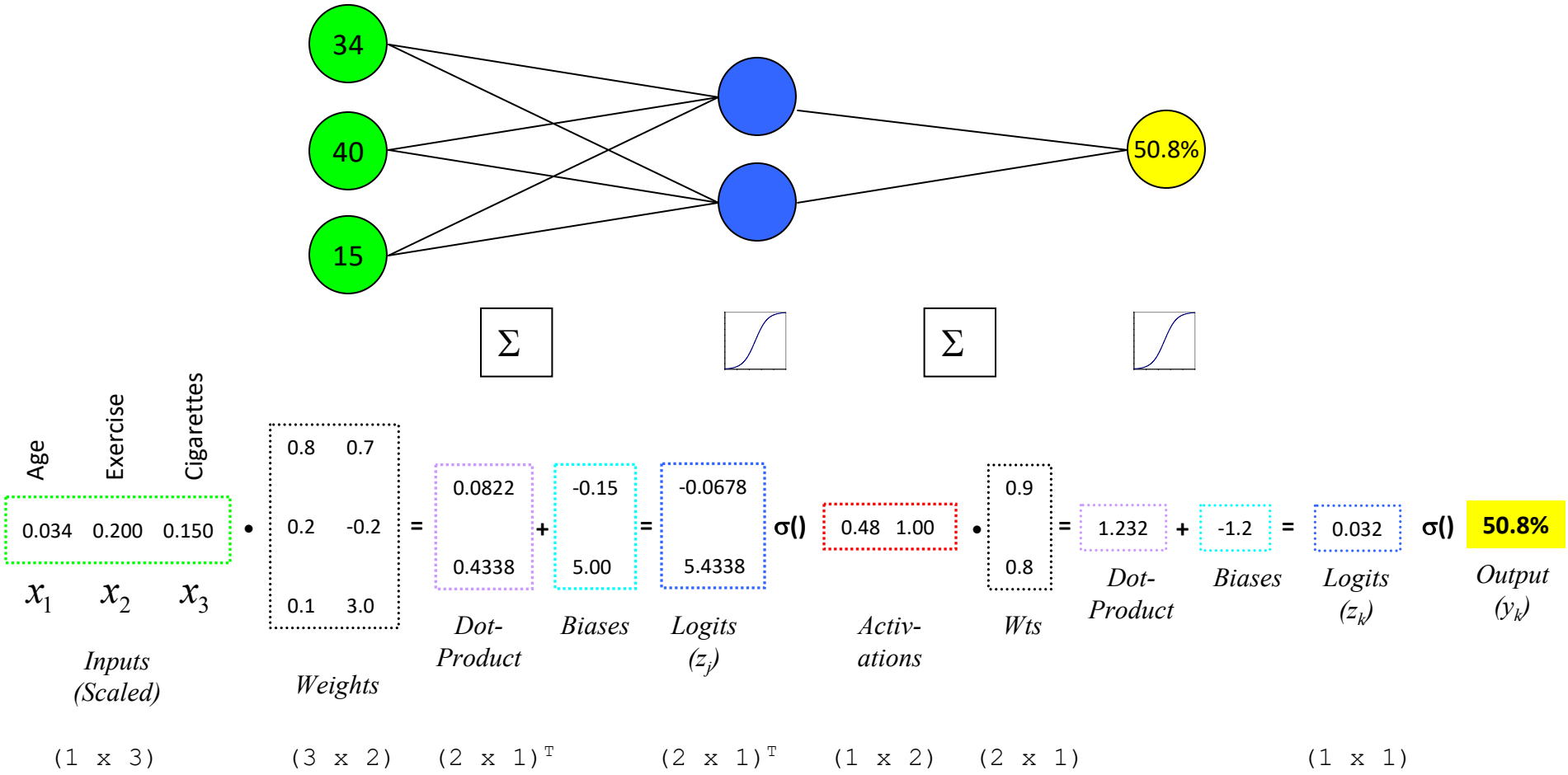
Machine translation (Google Translate)

Playing video games (Why are we automating this?)

Playing board games (e.g. AlphaGo)

An Example

Consider a simple neural network architecture that computes the probability of being alive at the end of five years given "life style" features:



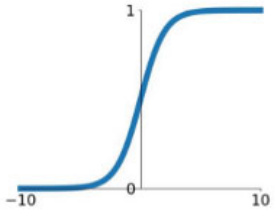
Design Choices

- Define the function you want to learn
- Determine an encoding for the data
- Pick a network architecture
  - Number of layers (between 3 and 100)
  - Activation functions function (tanh,ReLU, linear)
  - Select how units connect within and between layers
- Pick a gradient descent algorithm
- Pick regularization approach (e.g. dropout)

Activation Functions

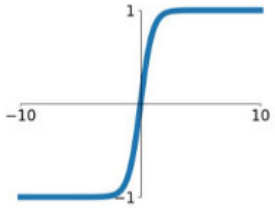
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



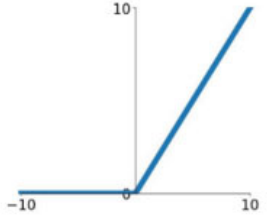
tanh

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$



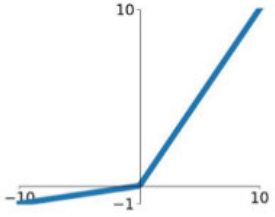
ReLU

$$\max(0, x)$$



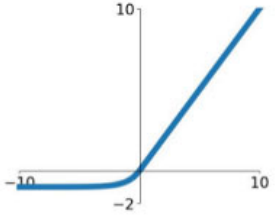
Leaky ReLU

$$\max(0.1x, x)$$



ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^{x} - 1) & x < 0 \end{cases}$$



Output Layers

A variety of output layers can be applied to the basic neural network architecture:

- Regression: Output is a real-valued number (e.g., linear regression in statistics, or previous example).

- Binary-Classification: Applies a threshold to a regression output to decide between classes :

$$class = \begin{cases} y_k > 0.50 \rightarrow \quad healthy \\ y_k \leq 0.50 \rightarrow \quad unhealthy \end{cases}$$

- Multi-Class Classification: The output-layer activation is replaced with the softmax function and a probability distribution is generated across k elements:

$$\hat{p}_k = \texttt{softmax}\left(\sum_{j=0}^{M} w_{jk} f\left(\sum_{i=0}^{N} w_{ij} x_i + b_j \right) + b_k \right) \quad \text{and} \quad \texttt{softmax}(z_k) = \frac{e^{z_k}}{\sum_{c \in C} e^{z_c}}$$