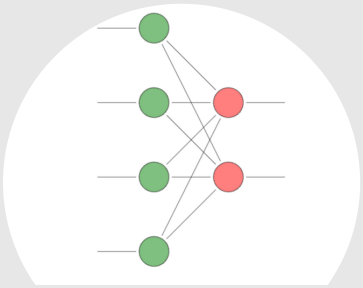


Foundations of Machine Learning

JuliaAcademy

Instructor Info

Chris Rackauckas

Dr. Chris Rackauckas is a mathematician and theoretical biologist at MIT.

His programming language of choice is Julia and he is the lead developer of the JuliaDiffEq organization dedicated to solving differential equations (and includes the package DifferentialEquations.jl). His research is in time stepping methods for solving stochastic differential equations (SDEs) and applications to stochastic partial differential equations (SPDEs) which model biological development.

Course Info

Course duration: 30 min total

Prereq: Basic Julia knowledge

Prereq: Basic Linear Agebra

Overview

In this course, we explore the fundamentals of Machine Learning in an interactive manner via Jupyter Notebooks. Over the course of 6 video lectures, totalling 30 minutes, we cover the Representation of Data with Models, Building Models, Model Complexity, What is Learning and What are Neurons. Through this series, we strive to achieve 5 learning objectives:

- To understand the practical applications of Machine Learning
- To know the intuition behind Machine Learning methods and why they work
- Understand motivations and theory behind Machine Learning mechanisms
- Gain practical experience training models
- Become able to apply software learned to Machine Learning projects

Course Summary

Representing Data with Models

To perform classification on an input image, we must first load it into computer-readable form, extract high-level features, and decode these features with a ML model. Images are parsed using matrices of its pixel values, while features encode information about various qualities (e.g. average amount of red pixels) of objects in the input image. Our model is a sequence of functions which map features to desired class inference output. Label outputs are chosen by decision boundaries out the output function, which strive to maximise inter-class variance and minimise intra-class variance. Output functions include the sigmoid function $\theta(x) = \frac{1}{1+e^{-wx+b}}$, which has a range of $(0, 1)$, suitable for binary classification problems.

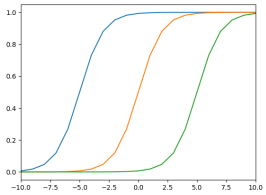


Figure 1: The effect of bias b on the sigmoid graph

Building Models

This lectures delves into the implementation details of a basic mathematical model, for classification of apples versus bananas. As this is a two-class binary classification problem, we utilise the sigmoid function as our output function via `σ(x, w, b) = 1 / (1 + exp(-w * x + b))`. while we fit the function to the dataset, new weights and biases are learned to derive a better decision boundary and hence higher classification accuracy.

Model Complexity

While training networks, we strive to define the most accurate possible architecture without overfitting. Adding model complexity is a great way of achieving better accuracy, as non-linear activations allow models to identify complex statistical relationships in data – relationships which linear architectures cannot model. However, if an overly complex model is defined, the decision boundary will fit too closely to samples in the training set, and fail to generalise to latent data in different distributions. To counter the overfitting problem, techniques such as Cross-Validation and Dropout are used.

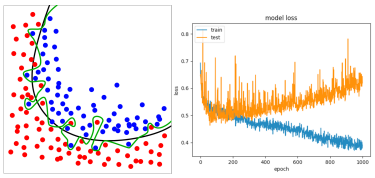
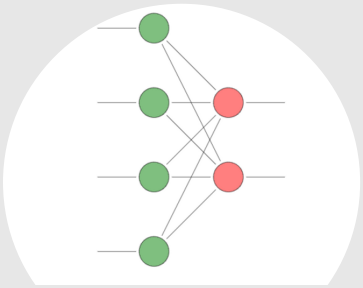


Figure 2: Models overfitting on training data



Foundations of Machine Learning

JuliaAcademy

Instructor Info

Chris Rackauckas

Dr. Chris Rackauckas is a mathematician and theoretical biologist at MIT.

His programming language of choice is Julia and he is the lead developer of the JuliaDiffEq organization dedicated to solving differential equations (and includes the package DifferentialEquations.jl). His research is in time stepping methods for solving stochastic differential equations (SDEs) and applications to stochastic partial differential equations (SPDEs) which model biological development.

Course Info

Course duration: 30 min total

Prereq: Basic Julia knowledge

Prereq: Basic Linear Agebra

Course Summary

What is Learning?

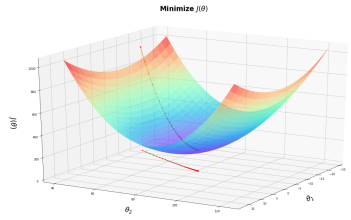


Figure 1: Optimisation with Gradient Methods

When optimising an ML model, one updates the trainable parameters of the network, including weights and biases. Non-trainable parameters include information such as the number of hidden layers and the minibatch size. Recall the effect of the bias b on the sigmoid decision boundary, as depicted in Figure 1. When training, we seek to iteratively search for the optimal values of weights and biases: we do this by defining an objective function and an optimisation algorithm. The objective function should quantify the loss of our model, and in other words measures how accurate our model is with the current parameters. The Sum of squares error is a commonly utilised loss function, defined as $C(x) = \sum_i C^2(x_i)$. To decrease the error in predictions, we must derive the appropriate update to be made to the bias. In Gradient Descent, we iterate in the direction of the negative gradient. For instance, the negative gradient for updating the bias is computed by $\frac{\partial C}{\partial b}$, while the update rule is defined as, s.t. α is the learning rate which determines the magnitude of oscillations:

$$b := b - \alpha \frac{\partial C}{\partial b} \tag{1}$$

What are Neurons?

Neurons are units which take in a group of weighted inputs, applies an activation function (e.g. Tanh, ReLU, Sigmoid) and returns an output. Each hidden layer of an ML model has its own neurons. Here is a visualisation of the a neuron:

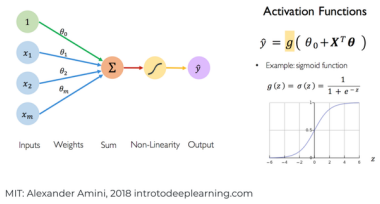


Figure 2: Neuron with sigmoid function

To understand how neurons map input vectors to their output shapes, we consider the following example: Given $\boldsymbol{a} = \sigma(\boldsymbol{x}) = \frac{1}{1+e^{-\boldsymbol{W}\bullet\boldsymbol{x}+\boldsymbol{b}}}$, if $\boldsymbol{x}$ and $\boldsymbol{a}$ are column vectors of size 10 (shape: `1, 10`), what are the corresponding shapes of $\boldsymbol{W}$ and $\boldsymbol{b}$? Since we know that the number of rows in $\boldsymbol{x}$ and number of columns in $\boldsymbol{W}$ must be equal; that the number of rows in $\boldsymbol{W}$ and $\boldsymbol{a}$ must be equal; that the dimensions of $\boldsymbol{b}$ must equal the dimensions of $\boldsymbol{a}$, we can infer the answer to be $\boldsymbol{W}$ `:(10,10)`, $\boldsymbol{b}$ `: (1,10)`.