Motivating example: handwritten digit recognition

We now turn to classification.

In machine learning and statistics, classification is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known. Examples are assigning a given email to the "spam" or "non-spam" class, and assigning a diagnosis to a given patient based on observed characteristics of the patient (sex, blood pressure, presence or absence of certain symptoms, etc.). Classification is an example of pattern recognition. In the terminology of machine learning, classification is considered an instance of supervised learning, i.e., learning where a training set of correctly identified observations is available.


The MNIST database (Modified National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems. The database is also widely used for training and testing in the field of machine learning. It was created by "re-mixing" the samples from NIST’s original datasets. The creators felt that since NIST’s training dataset was taken from American Census Bureau employees, while the testing dataset was taken from American high school students, it was not well-suited for machine learning experiments. Furthermore, the black and white images from NIST were normalized to fit into a 28x28 pixel bounding box and anti-aliased, which introduced grayscale levels. The MNIST database contains 60,000 training images and 10,000 testing images. Half of the training set and half of the test set were taken from NIST’s training dataset, while the other half of the training set and the other half of the test set were taken from NIST’s testing dataset.
Here is a sample of the images:

(Source (https://commons.wikimedia.org/wiki/File:MnistExamples.png))

We first load the data and convert it to an appropriate matrix representation. The data can be accessed with Flux.Data.MNIST.

```julia
# Julia version: 1.5.1
ENV["JULIA_CUDA_SILENT"] = true # silences warning about GPUs

using Images, ImageMagick
using Flux, Flux.Data.MNIST
```
In [2]:
images = MNIST.images()
labels = MNIST.labels()
length(labels)

Out[2]: 60000

For example, the first image and its label are:

In [3]: imgs[1]

Out[3]:

In [4]: labels[1]

Out[4]: 5

For now, we look at a subset of the samples: the 0’s and 1’s.

To find all such samples, we use the `findall` function with the boolean operator `||`.

In [5]: ?||

search: ||

Out[5]: x || y

Short-circuiting boolean OR.

In [6]: i01 = findall(l -> (l==0)|| (l==1), labels)
   imgs01 = imgs[i01]
   labels01 = labels[i01];

In this new dataset, the first sample is:
Next, we transform the images into vectors.

```plaintext
In [9]: X = reduce(hcat, [reshape(imgs01[i],:) for i = 1:length(imgs01)])
y = labels01;
```

The input data is now of the form \( \{(\mathbf{x}_i, y_i) : i = 1, \ldots, n\} \) where \( \mathbf{x}_i \in \mathbb{R}^d \) are the features and \( y_i \in \{0, 1\} \) is the label. Above we use the matrix representation \( X \in \mathbb{R}^{d \times n} \) with columns \( \mathbf{x}_i, i = 1, \ldots, n \) and \( y = (y_1, \ldots, y_n)^T \in \{0, 1\}^n \).

Our goal:

- to learn a classifier from the examples \( \{(\mathbf{x}_i, y_i) : i = 1, \ldots, n\} \), that is, a function \( \hat{f} : \mathbb{R}^d \rightarrow \mathbb{R} \) such that \( \hat{f}(\mathbf{x}_i) \approx y_i \).

This problem is referred to as binary classification (https://en.wikipedia.org/wiki/Binary_classification).

A natural approach to this type of supervised learning problem is to define two objects:

1. **Family of classifiers**: A class \( \widehat{F} \) of classifiers from which to pick \( \hat{f} \).
2. **Loss function**: A loss function \( \ell(\hat{f}, (\mathbf{x}, y)) \) which quantifies how good of a fit \( \hat{f}(\mathbf{x}) \) is to \( y \).

Our goal is then to solve

\[
\min_{\hat{f} \in \widehat{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(\hat{f}, (\mathbf{x}_i, y_i)),
\]

that is, we seek to find a classifier among \( \widehat{F} \) that minimizes the average loss over the examples.
For instance, in logistic regression, we consider linear classifiers of the form

$$\hat{f}(x) = \sigma(x^T \theta) \quad \text{with} \quad \sigma(t) = \frac{1}{1 + e^{-t}}$$

where $\theta \in \mathbb{R}^d$ is a parameter vector. And we use the cross-entropy loss

$$\ell(\hat{f}, (x, y)) = -y \log(\sigma(x^T \theta)) - (1 - y) \log(1 - \sigma(x^T \theta)).$$

In parametric form, the problem boils down to

$$\min_{\theta \in \mathbb{R}^d} -\frac{1}{n} \sum_{i=1}^{n} y_i \log(\sigma(x_i^T \theta)) - \frac{1}{n} \sum_{i=1}^{n} (1 - y_i) \log(1 - \sigma(x_i^T \theta)).$$

We will explain in a later notebook where this choice comes from.

The purpose of this topic's notebooks is to develop some of the mathematical theory and algorithms needed to solve this type of optimization formulation.

Optional reading

Material for this topic is partly covered in:

- Sections 2.1-2, 2.10-11, 3.1-4, 4.1-4 in [Wri](http://www.optimization-online.org/DB_FILE/2016/12/5748.pdf)

References

Parts of this topic's notebooks are based on the following references.


[DS100](https://www.textbook.ds100.org/intro.html) S. Lau, J. Gonzalez, D. Nolan, Principles and Techniques of Data Science


[Khu](https://search.library.wisc.edu/catalog/9910104042702121) A. I. Khuri, Advanced calculus with applications in statistics (2nd ed.), Wiley, 2003

[Wri](http://www.optimization-online.org/DB_FILE/2016/12/5748.pdf) S. J. Wright, Optimization Algorithms for Data Analysis, in: The Mathematics of Data, AMS, 2018