How to Train your AI: A Step-by-Step Study of an IDS

Cody Lewis

I will be using

- Python programming language
- Tensorflow: Neural network library, alternatives to consider are PyTorch or JAX
- Scikit learn: Includes simple machine learning algorithms and data processing functions
- Pandas: Easy management of tabular data
- Numpy: Fast and efficient linear algebra library
- Flower: Easy to use and powerful federated learning library

Background

Machine Learning - Generic Definition

- All machine learning models are a function, *f*, parameterized by weights, *w*, that maps the data samples, *x*, to a label, *y*.

$$f(w, x) = \hat{y}$$

- From this, we form a loss function that states how well *f(w, x)* maps to the true labels, *y*.

$$l(f, w, x, y) = d(f(w, x), y)$$

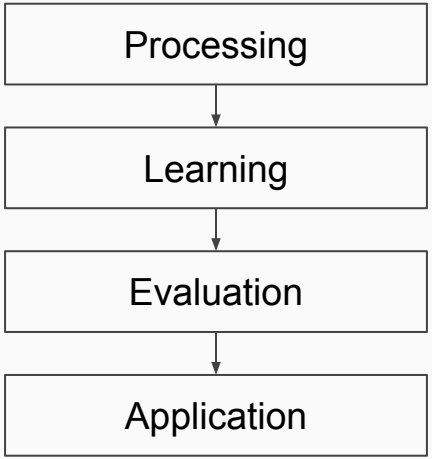
- With our loss function we can define machine learning as the minimization of loss with respect to the part of the model we can change, the weights.

$$\min_{w} l(f, w, x, y)$$

Data - Best Practices

- Dataset should be split into a testing and training dataset
- The testing dataset should never interact with the machine learning model outside of evaluation steps
- This ensures that the testing dataset is a best representative of how the model performs on new data

Machine Learning - Overall Process



Preprocessing Data

Preprocessing Data

- Gathering data and getting to know it
- Converting the data into a useful format
- Removing unneeded columns
- Fixing missing data (dropping, imputation, interpolation)
- Converting all data into numerical values
- Normalizing data

Gathering the UNSW-NB15 Data and About

- UNSW-NB15 can be found at https://research.unsw.edu.au/projects/unsw-nb15-dataset
- It is 100GB of raw network traffic data captured from a hybrid of real modern normal activities and synthetic contemporary attack behaviours
- The dataset is designed for training machine learning-based intrusion detection systems (IDS)
- Conveniently, the authors have already processed the dataset into training and testing sets in CSV format

Converting the Dataset into a Useful Format

- Often datasets will come in a format that is not directly useable within machine learning
- In those cases, we will want to convert it
- Best data for machine learning is typically has a semi-tabular format where each row/sample of the dataset is uniform in shape
- The process is subjective to the data we are dealing with

Converting the Dataset into a Useful Format

Before preprocessing:

```
train_data = pd.read_csv("data/UNSW_NB15_training-set.csv")
test_data = pd.read_csv("data/UNSW_NB15_testing-set.csv")
```

After preprocessing:

```
def extract_X_Y(data):
    Y = data.label.to_numpy()
    X = data.drop(columns="label").to_numpy()
    return X, Y

X_train, Y_train = extract_X_Y(train_data)
X_test, Y_test = extract_X_Y(test_data)
```

Removing Unneeded Columns

- Unneeded columns depends on what we are aiming to learn
- So this is just another element of knowing our dataset and task and dropping what is unnecessary

```
train_data = train_data.drop(columns=["id", "attack_cat"])
test_data = test_data.drop(columns=["id", "attack_cat"])
```

Fixing Missing Data

- Gaps in our dataset can cause undefined behaviour or bias in our data
- Solving this issue can be quite an advanced topic and is often subjective to the dataset and the target task of the machine learning model
- Three common tactics that get used are:
  - Dropping rows/samples with missing values
  - Imputation of values
  - Interpolation of values

Fixing Missing Data - Example

The are no missing values in our data, but if there were, the following code could help by filling in those values:

```
from sklearn.impute import SimpleImputer

imp = SimpleImputer(missing_values=np.nan, strategy='mean')
imp.fit_transform(train_data)
```

Converting Data to Numerical Values

- Machine learning models are mathematical in nature, so the data that is input must be numerical
- Usually this simply involves enumerating categorical values
- Although sometimes this implies the existence of intermediate categories which may not be possible, e.g. if cat = 0, dog = 1, then 0.5 implies cat-dog hybrid
- In those cases we may want to use one-hot encoding, where values are now n-vectors where a value of 1 means it is in the ith category of the n e.g. cat = [1, 0], dog = [0, 1]

Converting Data to Numerical Values

```
def encode_categoricals(data):
    categorical_cols = [c for c in data if data[c].dtype == "object"]
    data[categorical_cols] = skp.OrdinalEncoder().fit_transform(data[categorical_cols])

encode_categoricals(train_data)
encode_categoricals(test_data)
```

Normalizing the Data

- Normalization is the process of altering the range of numerical data
- The process needed is dependent on the nature of the data and what is viable
- A common form is min-max scaling where the data is brought into the range of [0, 1] by subtracting the min value of the data and dividing by the difference between the max value and min value
- Normalization is very useful for machine learning as it can speed up the search process for the weights that minimize the loss by shrinking the values that need to be searched as we are only dealing with smaller values
    - Note: While in practice, this often makes the learning faster, there is no guarantee that it is better

Normalizing the Data

```
def byterize(x):
    return 2**np.ceil(np.log2(x))

def min_max_scale(x, min_val, max_val):
    return (x - min_val) / (max_val - min_val)

def normalize(train_data, test_data):
    for col in train_data:
        min_val = np.floor(train_data[col].min())
        max_val = byterize(train_data[col].max())
        test_data[col] = test_data[col].clip(upper=max_val)
        train_data[col] = min_max_scale(train_data[col], min_val, max_val)
        test_data[col] = min_max_scale(test_data[col], min_val, max_val)

normalize(train_data, test_data)
```

Code Demo

Supervised and Unsupervised Machine Learning

Supervised Machine Learning

- Supervised machine learning is the learning of a task when the labels to be learned are included in the dataset
- This allows for a direct search process of *w*, that minimizes the loss function

Defining the Loss Function

The loss function to be chosen is dependent on whether the labels are categorical or a floating point, that is a classification or regression task respectively

- Common classification loss is cross-entropy:

$$l(f, w, X, Y) = \frac{1}{|Y|} \sum_{x \in X, y \in Y} -\Delta^y \log(f(w, x))$$

- Common regression loss is mean square error:

$$l(f, w, X, Y) = \frac{1}{|Y|} \sum_{x \in X, y \in Y} (y - f(w, x))^2$$

Choosing a Learning Model

- Binary trees/forest: Relatively simple algorithm that makes binary decisions based on threshold values found in the samples
- Bayesian Models: Applies Bayes' theorem to learn $P(y \mid x)$
- Neural networks: Simulates a classical model of the mammalian brain, constructing a linear equation that returns a value in y's range
- SVM: Finds the hyperplane or set of hyperplanes that separate samples into their classes
- And many more

Hyperparameters

- ML models tend to have static parameters that are not changed during the learning process, referred to as hyperparameters
- For example, the number of binary trees to include in a forest
- There is no precise method for determining hyperparameters, so usually it is simply a mixture of using knowledge of the dataset and exploratory testing
- Some algorithms exist for choosing hyperparameters at greater computation cost, e.g. grid search, and hypergradient descent

Choosing Hyperparameters for Neural Networks

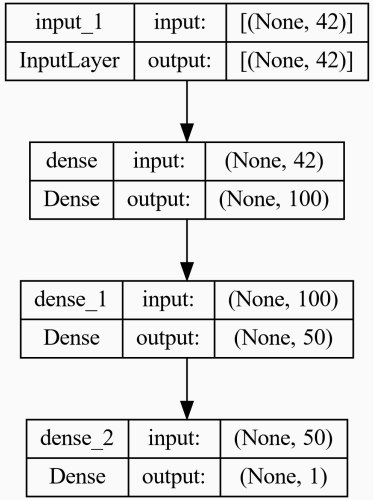
Neural networks have four main hyperparameters:

- Number of layers
- Type of layers
- Activation of layers
- Scale of the layers

Neural Networks: Number of Layers

Determines the order of the linear algebraic combinations that can be found by the network

- Note: A neural network with more than 1 layer is often called a deep neural network



Neural Networks: Type of Layers

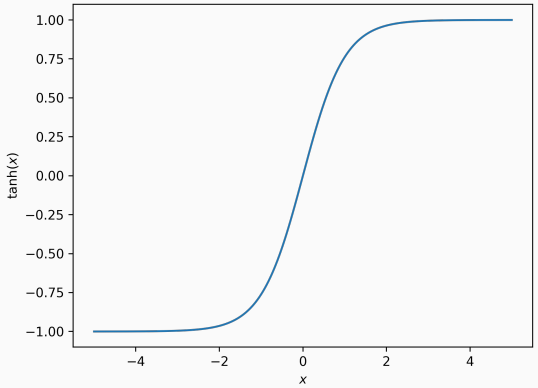
Determines the specific operations that are done within the model

- Dense: Maps every possible combination
- Convolution: Collects nearby elements of the vector to contribute to the output
- Recurrent: Past inputs impact the current one
- Batch normalization: Records the running mean and variance of the minibatch outputs from the previous layer and uses them to map them into the Gaussian normal distribution
- Dropout: Randomly drops a hyperparameter specified percentage of input elements

Neural Networks: Activation of Layers

Activation functions are functions that are applied element-wise to the outputs of the layer they are attached to. They alter the properties of the neural network including its gradients

- ReLU: $\max(x, 0)$
- Sigmoid: $\frac{1}{1+e^{-x}}$
- Tanh: $\tanh(x)$



Neural Networks: Scale of the Layers

Determines how much of the input the layer operates on, or the size of its output

- Dense: Number of neurons, i.e. how many combinations of inputs to create
- Convolution: Size of kernel and number of kernels (channels), i.e. how many nearby elements to take and how many collections to perform
- Recurrent: Hidden size, i.e. how much information should be used to encode the past inputs

Learning

- Sometimes the learning process itself has some parameters of its own to choose, such as which search algorithm to use and fine grained it should be
- For example, neural networks tend to use stochastic gradient descent where the minibatch gradient w.r.t. weights is subtracted from the current weights. Here a parameter is the learning rate, η, which states how fine grained an update should be.

$$w^+ = w - \eta \nabla_w l(f, w, x, y)$$

- There are also other options for neural network learning such as Adam, which uses the minibatch gradients to estimate the full batch gradients according to statistical moments

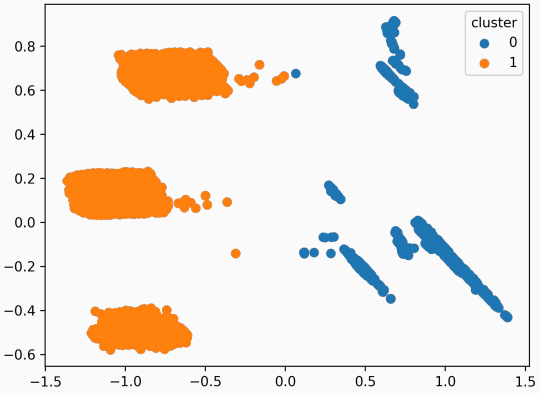
Evaluating

- To see how well our model has learned to predict, we evaluate on the testing dataset
- There are many metrics for measuring how well it performs, the most suitable is dependent on the goal task
  - Common classification metrics are accuracy, F1-score, or ROC AUC
  - Common regression metrics are mean square error, mean absolute, or R^2

Code Demo

Unsupervised Machine Learning

- Unsupervised learning is the learning of a task that does not include labels, or the labels are very limited
- It assumes there is some pattern in the dataset that correlate to the task to be learnt
- Most algorithms cluster the data into different groups



Choosing a Learning Model

- K-means: Forms K-clusters and iteratively recentres based on the mean value of the nearest samples until stability
- K-nearest neighbours: Forms graphs where samples are connected by an edge to the K samples with the smallest distance
- Autoencoder: Neural network with a double funnel structure, first the layers get smaller to produce a small vector representation of the input, then they get bigger to produce a vector the same shape as the input. The aim is to accurate reconstruct the input by deconstructing into the most minimal encoding possible
- etc.

Hyperparameters, Learning, Evaluating

- Choosing hyperparameters and a learning algorithm is the same as supervised learning
- The only mild difference here is in evaluation where for a classification task we will find which of the clusters found by the model maps best to each label, then evaluate based on this translation

Code Demo

A Brief Overview of Other ML Types

Semi-Supervised Machine Learning

- Semi-supervised learning combines both unsupervised and supervised learning into one task, it is used when there is only limited and labelled data
- Unsupervised learning is used to generate synthetic data that matches the distribution of the limited data present
- Supervised learning is used to train a model using both the synthetic data and real data. Where the synthetic data improves the learning of the model

Reinforcement Learning

- Places an agent in an environment, and scores them based on the actions they perform
- RL algorithms learn parameters for policies to determine which changed actions to take based on the scores they previously attained
- Notably, there have been recent advances in deep reinforcement learning which uses RL to train a deep neural network

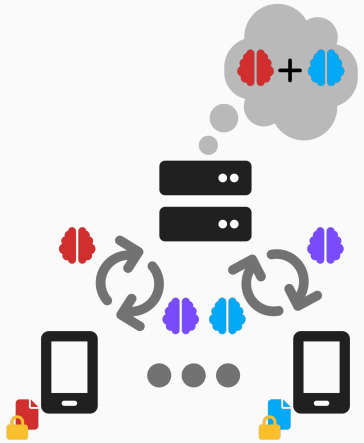
Genetic Algorithms

- Operates in the same environment as reinforcement learning
- Actions are instead determined by a “genome”, which are a set of parameters for the policy
- In each step of training, a pool of genomes are tested in the environment, only the best scoring ones are kept for the next step
- Between steps, new genomes are produced by mixing of those that already exist, additionally genomes are randomly changed in a mutation process

Federated Learning

Federated Learning

- Federated learning distributes neural network learning across many clients
- The main aim is to maintain data privacy by stopping any direct sharing of the data
- Clients train a copy of the neural network model and upload the resulting model to the server
- The server takes the average of these models



Federated Learning Client

- The client holds a local dataset that they want to keep private
- They first receive the global model from the server
- They train it for *E* steps, called epochs
- They then upload either the resulting model or the difference between it and the global model
    - The latter is called the gradient, since if the client uses SGD, it will be the sum of minibatch gradients

Federated Learning Server

- The server is relatively simple
- It sends the global model to each of the clients
- It then receives back trained models or gradients from each of the clients
- It concludes the “round” by aggregating the models or gradients, if gradients then the result is subtracted from the global model otherwise it is the new global model
- The steps are then repeated for many rounds of training

Hyperparameters

- Choosing hyperparameters is mostly similar to with machine learning, although there can be some differences due to how data is distributed
- For example, with ML of neural networks, batch normalization can be effective to maintain a set range for values throughout the network. While FL's unbalanced data can mean that the internal batch normalization statistics are offset, breaking outputs. So, it is more common for group or layer normalization to be used
- Additionally, there are four choices for the learning algorithms, the client-side learning algorithm, the server-side aggregation algorithm, and their respective parameters

Evaluating

Evaluation can be a little tricky within the federated setting, with mainly two options:

- Centralized evaluation: The server has a testing dataset, and evaluates as normal using the global model
- Federated analytics: Client each evaluate a copy of the global model on their local testing dataset, and submit the result to the server, the server aggregates those results to produce an overall evaluation

Code Demo

Code resources and Q&A

The code used in this presentation is publicly available at https://github.com/codymlewis/ids_presentation

Any Questions?