Machine Learning 101 Fundamentals by Michaël Bettan

Definitions & Scope

- **Artificial Intelligence (AI):** Computer think/act like humans.
- **Machine Learning (ML):** Scalably solve problems using data examples (not rules).
- **Deep Learning (DL):** Subtype that works even when the data consists of unstructured data, like images, speech, video, natural language text and so on. Approach based on Neural Networks.

CORE DISTINCTION

The basic difference between Machine Learning and other techniques in AI is that in Machine Learning machines learn.

Standard Workflow & Lifecycle

Problem Definition

What business problem are you solving for? What is the outcome you want to achieve?

Data Extraction

Determine the data you need for training and testing your model based on the use case.

Data Preparation

Your data needs to be properly formatted before and after data import (iteratively).

Model Training

Set parameters and build your model.

Model Troubleshooting

Troubleshoot the performance.

Model Evaluation

Review model metrics and performance.

Model Tuning

Adjust parameters not learned from data.

Model Testing

Try your model on test data.

Model Serving

Make your model available to use.

Model Monitoring

Keep your model accurate.

ML Problem Types

Supervised Learning

Learn from examples, apply label to data.

- **Binary classification model:** predicts a binary outcome (one of two classes) → yes or no questions. Example: Credit card transactions (fraud or not).
- **Multi-class classification model:** predicts one class from three or more discrete classes (one of a set). Example: segment customers into different personas.
- **Regression model:** predicts a continuous value. Example: forecast customer spending over the next month.
- **Matrix Factorization:** (matrix decomposition) used in recommender systems. Example: Netflix.

Unsupervised & Reinforcement

Unsupervised Learning: detect unknown patterns w/o labeled examples.

- **Clustering model:** identifying similarities in groups.
- **Anomaly Detection:** identifying abnormalities in data.

Reinforcement Learning:

- Learn from the environment via exploration and exploitation. Use positive/negative reinforcement to complete a task. Example: chess, maze, etc.

Data Extraction & Preparation

Data Extraction

- **Select relevant features:** A feature is an input attribute used for model training. Features are how your model identifies patterns to make predictions, so they need to be relevant to your problem.
- **Include enough data:** The more training examples you have, the better your outcome. Amount of example data required scales with the complexity of the problem you're solving for.
- **Capture variation:** Your dataset should capture the diversity of your problem space. The more diverse examples a model sees during training, the more readily it can generalize to new or less common examples.

DATA PREPARATION PITFALLS

Prevent data leakage: when you use input features during training that "leak" information about the target that you are trying to predict which is unavailable when the model is actually served. This can be detected when a feature that is highly correlated with the target column is included as one of the input features. Strong model performance during testing, but not when deployed in production. **Prevent prediction-serving skew:** avoiding a mismatch between the data or processing used during model training and what is available at prediction (inference) time. **Clean up data:** clean up missing, incomplete, and inconsistent data to improve data quality before using it for training purposes.

Model Training Fundamentals

- **Feature engineering:** transforming input features to be more useful for the models. e.g. mapping categories to buckets, normalizing between -1 and 1, removing null.
- **Model selection and iterations.**

Model Troubleshooting

Underfitting

Poor performance on training and test dataset.

- Increase the complexity of model
- Increase the training time

Overfitting

Good perf on training, poor perf on test dataset.

- Specific to the trained data, does not generalize.
- **Reasons:** not enough training data, too many features (redundant), features not useful (noise).
- **Combat via:** Regularization which limits information captured.
- **Combat via:** Early Stopping (halting when validation loss increases).
- **Combat via:** L1 / L2 Regularization.
- **Combat via:** Dropout (randomly dropping nodes).
- **Combat via:** Max-Norm Regularization.
- **Combat via:** Data Augmentation (artificially boosting dataset diversity).

Bias

Missing relationship between features and labels → training data is not generalized.

Ideal state: Low bias, low variance.

Variance

Sensitivity in small fluctuations in the training dataset → difference in a set of predictions.

Ideal state: Low bias, low variance.

Model Evaluation

Model evaluation metrics are based on how the model performed against a slice of your dataset (the test dataset).

Classification Metrics

Score thresholds to adjust the confusion matrix (true positive, true negative, false positive and false negative).

- **Accuracy:** % of total true positive & true negative / total population.
- **Precision:** % of total positive class correctly classified by TP / (TP + FP).
- **Recall:** % of total actual positive labels identified by TP / (TP + FN).
- **F1 score:** 2 * (Precision * Recall) / (Precision + Recall).
- **AUC PR, AUC ROC, Log loss:** Common classification performance metrics.

Regression Metrics

Mean squared error:

- MAE (Mean Absolute Error)
- RMSE (Root Mean Squared Error)
- RMSLE (Root Mean Squared Logarithmic Error)

Gradient Descent & Backpropagation

- **Gradient Descent:** technique to minimize the loss in a neural network by calculating gradient (slope) → optimal weight.
- **Batch Gradient Descent:** → slow large dataset.
- **Stochastic Gradient Descent:** → random process.
- **Mini-Batch Gradient Descent:** → mixed mode.
- **Backpropagation:** algorithm mapping from the input-output pair to calculate the gradient descent → efficient way.

MODEL TUNING & TESTING

Hyperparameter tuning: Adjusting the parameters of our models not from the data but the model (e.g., # layers, learning rate, etc.). **Model Testing:** Evaluating your model metrics is primarily how you can determine whether your model is ready to deploy, but you can also test it with new data. Try uploading new data to see if the model's predictions match your expectations. Based on the evaluation metrics or testing with new data, you may need to continue improving your model's performance. *Consideration: Never test with training data.*

Model Serving & Monitoring

Batch prediction

Is useful for making many prediction requests at once. Batch prediction is asynchronous, meaning that the model will wait until it processes all of the prediction requests before returning the predicted values.

Online prediction

Is synchronous (real-time), meaning that it will quickly return a prediction, but only accepts one prediction request per API call. Online prediction is useful if your model is part of an application and parts of your system are dependent on a quick prediction turnaround.

Model Monitoring

- **Metrics to monitor:** traffic patterns, error rates, latency, resource utilization with Cloud Monitoring alerting.
- **Data Drift:** the input data changes over time → refresh models. (Not to be confused with *training-serving skew*, a mismatch between training and serving data.)
- Watch for changes in dependencies upstream in the pipeline.
- Assess model prediction quality.
- Test for unfairness (bias unintentionally).

User Personas and User Stories

- **Product Managers:** Insights and objectives.
- **Data Analyst:** Query and analyze.
- **Data Engineer:** Get clean and useful data.
- **Data Scientist:** Models that work.
- **ML Developer:** Intelligent applications.
- **ML Engineer:** Models in production.

ML OPS & OPERATIONALIZE

"DevOps" / automated operations for machine learning.

Neural Networks

Architecture & Components

- **Neural Networks:** = input layer + hidden layer to output. Model composed of layers, consisted of neurons.
- **Neuron:** node → combine input values to one output value.
- **Epoch:** single pass through training dataset.
- **Weight:** multiplication of input values.
- **Bias:** value of output given a weight of 0.
- **Hidden layers:** set of neurons operating from same input set.

Network Types

- **Feedforward neural network (FFNN):** information moves strictly forward from input to output.
- **Recurrent neural network (RNN):** optimized for sequential data, where previous runs feed into the next.
- **Deep Neural networks:** = generalization, many hidden layers.
- **Wide Neural networks:** = memorization, many features.
- **Deep and Wide networks:** generalization + memorization. Good fit for recommendation engines.

Glossary

Core Concepts

- **Inference:** = scoring = predictions: applied trained model to unlabelled examples.
- **Label:** The variable we are predicting (target).
- **Features:** Input data used by the ML model.
- **Feature Store:** Rich feature repository to serve, share and re-use ML features.
- **Training-Serving Skew:** Mismatch between input features at training time and serving time.

Data Splits

- **Training Set:** Labeled examples used to optimize the model.
- **Validation Set:** Disjoint subset used to adjust hyperparameters and prevent overfitting.
- **Test Set:** Subset used to provide final results on new data (contains labels, model never learns from them).

3 Types of "Bias"

- **Bias (Math):** Intercept or offset from an origin (the b in y = wx + b).
- **Prediction Bias:** Difference between average of predictions and average of labels in the dataset.
- **Bias (Ethics/Fairness):** Unintentional unfairness or stereotyping in algorithms or data.

Neural Networks & Optimization

- **FFNN:** Neural network without recursive connections; information moves strictly forward.
- **RNN:** Network optimized for sequential data where previous runs feed into the next.
- **Activation Functions:** Mathematical functions (e.g. RELU, tanh) that introduce non-linearity.
- **Optimizer:** Operation that changes weights and biases to reduce loss e.g. Adagrad or Adam.
- **Gradient Descent:** Technique to minimize loss by calculating gradient (slope) to find optimal weight.
- **Backpropagation:** Efficient algorithm to calculate gradient descent in neural networks.
- **Learning Rate:** Rate at which optimizers adjust weights and biases; risks non-convergence if too high.
- **Converge:** State where loss stabilizes and the algorithm reaches an optimal answer.

Preventing Overfitting in DNNs (Regularization)

- **L1 Lasso Regularization:** Penalizes absolute weight values; drives least useful features to 0.
- **L2 Ridge Regression:** Penalizes squared weights; keeps weights approximately equal in size.
- **Dropouts methods:** Randomly dropping nodes during training to improve generalization error.
- **Early Stopping:** Ending training when validation loss starts to increase to prevent overfitting.
- **Max-Norm Regularization:** Limiting the absolute magnitude of network weights.
- **Data Augmentation:** Transforming existing examples (e.g. rotating images) to artificially boost dataset diversity.

Advanced Techniques & Data Processing

- **Ensemble Learning:** Combining predictions from multiple distinct models to solve the same problem.
- **Neural Architecture Search (NAS):** Automated approach to designing and selecting the best model ensemble.
- **Embeddings:** Mapping discrete objects (like words) to vectors of real numbers.
- **One-Hot encoding:** Mapping attribute values to a bit in a binary array.
- **Normalization:** Converting numeric values to a standard range, e.g. -1 and 1.
- **Tensor:** N-dimensional arrays of numbers; primary data structure in ML.