Machine Learning with Go

Ephraim Berkovitch, January 2021

Introduction

- Ephraim Berkovitch - senior software engineer in ZipRecruiter
  - Building full-stack solutions in Go and Python
  - Contributes to open-source (HaSadna, Ben-Yehuda Project, Yiddish apps)
- By the end of this session you will understand
  - how simple ML algorithms work
  - how to run them in Go
  - how different ML Golang libraries compare with their Python counterparts

Agenda

- Why Go?
- ML using Go vs Python
- ML Steps
- Gorgonia
- Flower species prediction by classification
- Linear regression
- Gota - counterpart of pandas
- Gonum - counterpart of numpy and matplotlib
- Gophernotes - Jupyter kernel for Go

Accompanying Materials

+ https://github.com/ephraimberkovitch/golang_ml
+
+ https://github.com/go-gota/gota
+ https://github.com/gorgonia/gorgonia
+ https://github.com/gonum/gonum

Why Go?

+ simple syntax
    - clearly describe complex algorithms
    - does not obscure developers from understand how to run efficient optimized code
+ easy to deploy and code
+ easy to understand and debug
+ have its performance measured
+ great tooling
    - test coverage
    - presentations

ML using Go vs Python

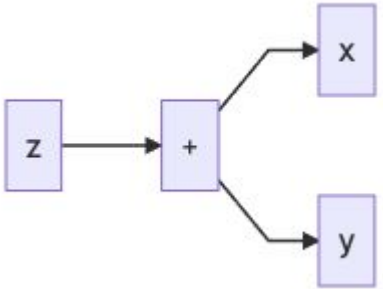
+ ML frameworks use languages like C/C++ for actual computation
    - all offer a Python interface
+ To actually run a production machine learning API at scale, we need an infrastructure to
    - Autoscaling, so that traffic fluctuations don't break our APIs (and our AWS stays manageable)
    - API management, to handle multiple deployments
    - Rolling updates, so that we can update models while still serving requests
+ Reliability - first line of defense, to avoid silly type errors - static typing, compilation step, no dead code, no unused imports
+ **Python for data science and scripting, Go for infrastructure**

ML Steps

- Gather data
- Perform exploratory data analysis
- Determine the correct machine learning solution to use
- Build a model
- Train the model
- Test the model

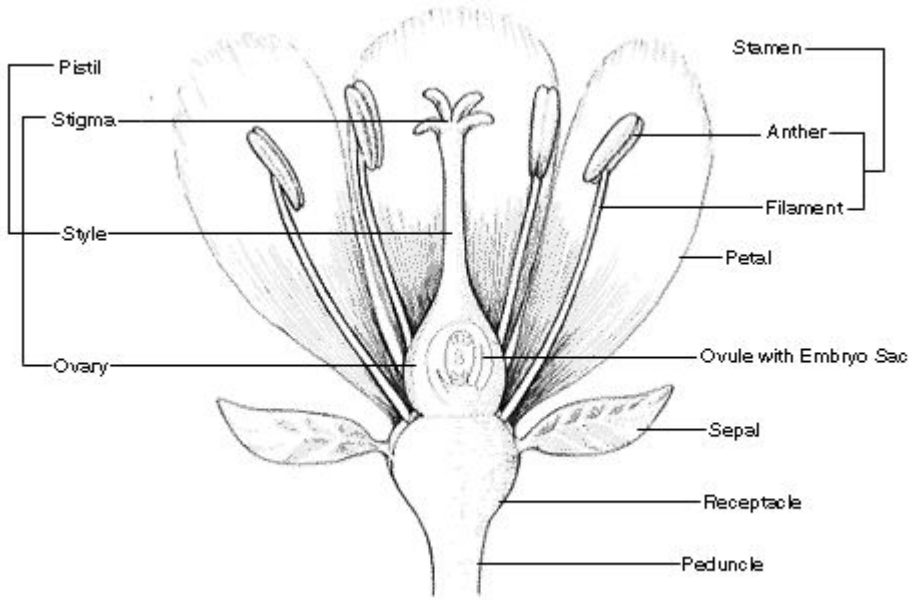
Gorgonia

+ Deep learning in Go
    - go get gorgonia.org/gorgonia
+ Write and evaluate math equations with multi dimensional arrays
    - g := gorgonia.NewGraph() // expression/computation graph
+ VM - understands graphs and computes it
    - compile-once, run-many-times
+ *Demonstration 1 - Gorgonia 1st steps*



Flower Species Prediction by Classification

+ Iris dataset (1935) - https://gist.github.com/curran/a08a1080b88344b0c8a7
+ features - X
    - sepal_length - אורך עלי הגביע
    - sepal_width - רוחב עלי הגביע
    - petal_length - אורך עלי הכותרת
    - petal_width - רוחב עלי הכותרת
+ target - y
    - species - setosa, versicolor, virginica



Flower Species Prediction by Classification

Linear Regression

+ encode the data (the true facts from observation of different flowers) into a matrix
    - containing 5 columns (sepal length, sepal width, petal length, petal width and 1 for the bias).
+ A row of the matrix represents a flower

$$y = \theta_0 + \theta_1 * sepal\_length + \theta_2 * sepal\_width + \theta_3 * petal\_length + \theta_4 * petal\_width$$

$$x = [\, sepal\_length \quad sepal\_width \quad petal\_length \quad petal\_width \quad 1 \,]$$

$$\Theta = [\, \theta_4 \theta_3 \theta_2 \theta_1 \theta_0 \,]$$

$$y = x \cdot \Theta$$

LInear Regression

- Encode categorical data
    - setosa = 1.0
    - virginica = 2.0
    - versicolor = 3.0
- Other encoding techniques:
    - species_setosa = 1/0: species_virginica=0/1: species_versicolor=0/1
- Learning phase cost: $cost = \frac{1}{m} \sum_{i=1}^{m} (X^{(i)} \cdot \Theta - Y^{(i)})^2$
- Use **gradient descent** to lower the cost

Gota - counterpart of Pandas

+ Data wrangling library
+ Generate / load the training / test set with Gota
    - dataframe
    - series
+ Pre-processing
+ Generate training / validation data split
+ *Demonstration 2 - gota 1st steps*

Gonum - counterpart of numpy and matplotlib

+ Scientific and plotting library
    - go get gonum.org/v1/gonum
    - go get gonum.org/v1/plot
+ *Demonstration 3 - gonum/plot 1st steps*

Demonstrations 4 & 5

- iris
    - training_iris
    - cli_iris
- congress votes (~1980s)
    - training_congress
    - cli_congress

Jupyter kernel for Go

+ Gophernotes
    - docker run -it -p 8888:8888 -v $(pwd):/usr/share/notebooks gopherdata/gophernotes:latest-ds
+ *Demonstration 6 - gophernotes*

Comparisons

- Gorgonia
  - https://golangdocs.com/golang-machine-learning-libraries
- Gota
  - performance - https://github.com/go-gota/gota/issues/16
  - other libraries - https://mungingdata.com/go/dataframes-gota-gframe
- gophernotes
  - https://github.com/gopherdata/gophernotes - backend - interpreter
  - alternatives - lgo https://github.com/yunabe/lgo - backend - official compiler, elder versions

More

+ TensorFlow for Go is not mature yet
    - https://www.tensorflow.org/install/lang_go - The TensorFlow Go API is not covered by the TensorFlow API stability guarantees
    - **galeone/tfgo** has its fixes to official TensorFlow Go bindings
+ Cortex - machine learning model serving infrastructure
    - https://github.com/cortexlabs/cortex
    - take a trained model and automatically implement all the infra features needed—like reproducible deployments, scalable request handling, automated monitoring, etc—to deploy it as an API

References

+ **Michael Bironneau, Toby Coleman**, *Machine Learning with Go Quick Start Guide*, Packt Publishing, 2019
+ **Xuanyi Chew**, *Go Machine Learning Projects*, Packtpub, 2018
+ https://towardsdatascience.com/why-we-deploy-machine-learning-models-with-go-not-python-a4e35ec16deb
+ https://towardsdatascience.com/understanding-regression-using-covid-19-dataset-detailed-analysis-be7e319e3a50
+ https://towardsdatascience.com/go-for-data-science-lets-try-46850b12a189

References

+ https://tech.travelaudience.com/training-tensorflow-models-in-python-and-serving-with-go-1b2a9386b0ff
+ https://jinglescode.github.io/2019/05/07/why-linear-regression-is-not-suitable-for-classification/
+ https://mungingdata.com/go/dataframes-gota-qframe/#:~:text=Go%20has%20great%20DataFrame%20libraries,manipulation%20and%20grouping%20functionality%20natively
+ https://towardsdatascience.com/a-gentle-journey-from-linear-regression-to-neural-networks-68881590760e
+ https://github.com/RedHatOfficial/GoCourse