Machine Learning in the Wild

Semih Yagcioglu
@semihyagcioglu

STM

Dec 10, 2018
CS491 - Bilkent University
What is Machine Learning?

Field of study that gives computers the ability to learn without being explicitly programmed.

Arthur Samuel, 1959
What is Machine Learning?

A computer program is said to learn from experience $E$ with respect to some class of tasks $T$ and performance measure $P$, if its performance at tasks in $T$, as measured by $P$, improves with experience $E$.

Tom Mitchell, 1997
AI, Deep Learning and Machine Learning

Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.
Types of ML

- **Supervised Learning**
  - Classification
  - Regression
  - Ranking

- **Unsupervised Learning**
  - Clustering
  - Association Mining
  - Segmentation
  - Dimension Reduction

- **Reinforcement Learning**
  - Decision Process
  - Reward System
  - Recommendation Systems
ML vs Traditional Programming
A Machine Learning System

THIS IS YOUR MACHINE LEARNING SYSTEM?

YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.

WHAT IF THE ANSWERS ARE WRONG?

JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.

Credit: xkcd.com/1838
Typical ML Workflow
A Typical ML Model

```python
>>> from sklearn import svm
>>> from sklearn import datasets
>>> clf = svm.SVC(gamma='scale')
>>> iris = datasets.load_iris()
>>> X, y = iris.data, iris.target
>>> clf.fit(X, y)
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)

>>> import pickle
>>> s = pickle.dumps(clf)
>>> clf2 = pickle.loads(s)
>>> clf2.predict(X[0:1])
array([[0]])
>>> y[0]
0
```
Experimenting vs Production

In [5]: model.fit(train[features], y)
Out[5]:
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
 max_depth=None, max_features='auto', max_leaf_nodes=None,
 min_samples_leaf=1, min_samples_split=2,
 min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=2,
 oob_score=False, random_state=None, verbose=0,
 warm_start=False)

In [6]: model.predict(test[features])
Out[6]:
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,
 2, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])

In [7]:

NameError Traceback (most recent call last)
<ipython-input-1-fa20f16f8557> in <module>()
----> 1 model.predict(test[features])
NameError: name 'model' is not defined

In [2]:
Priorities and Requirements

“Might Works well for KAGGLE!
But Kaggle isn’t real world Machine learning!”

Interpretability + Low Complexity + Speed >>>> Accuracy

Accuracy = 0.81
Speed = 30ms

Accuracy = 0.91
Speed = 3s
ML is a Tool to Solve a Problem

Machine Learning != Your Product
A Few Principles

1. Don’t build ML for the sake of it
2. Do you need ML in your MVP to test product market fit?
3. Is your ML mission critical?
Be Agile

Build > Measure > Learn

Eric Ries - The Lean Startup
ML Checklist

Systematic Process

1. Define Problem
2. Prepare Data
3. Spot Check Algorithms
4. Improve Results
5. Present Results

Capabilities of ML Platform or Tools

- ML Problem checklist
- Data Visualization methods...
- Data Selection
- Feature Selection methods...
- Feature Engineering methods...
- Data Transformation methods...
- Test Harness...
- Perform Measure...
- Evaluate accuracy of different algorithms...
- Algorithm Tuning methods...
- Ensemble methods...
- ML Presentation checklist
A Few Useful Things to Know About ML

1. Learning = representation + evaluation + optimization
2. It is generalization that counts
3. Overfitting has many faces
4. Intuition fails in high dimensions
5. Feature engineering is the key
6. More data beats a cleverer algorithm
7. Learn many models, not just one
8. Representable does not imply learnable
9. Correlation does not imply causation

Credit: Pedro Domingos (2012)
A Few Useful Things to Know About ML

Learning = representation + evaluation + optimization

All machine learning algorithms have three components:

- **Representation** for a learner is the set if classifiers/functions that can be possibly learnt. This set is called hypothesis space. If a function is not in hypothesis space, it can not be learnt.
- **Evaluation** function tells how good the machine learning model is.
- **Optimisation** is the method to search for the most optimal learning model.

Credit: Pedro Domingos (2012)
A Few Useful Things to Know About ML

Its Generalization That Counts

The fundamental goal of machine learning is to generalize beyond the training set. The data used to evaluate the model must be kept separate from the data used to learn the model. When we use generalization as a goal, we do not have access to a function that we can optimize. So we have to use training error as a proxy for test error.
A Few Useful Things to Know About ML

Overfitting Has Many Faces

One way to interpret overfitting is to break down generalization error into two components: bias and variance. Bias is the tendency of the learner to constantly learn the same wrong thing. Variance is the tendency to learn random things irrespective of the signal.

Credit: Pedro Domingos (2012)
A Few Useful Things to Know About ML

Intuition Fails In High Dimensions

After overfitting, the biggest problem in machine learning is the curse of dimensionality. Generalizing correctly becomes exponentially harder as the dimensionality (number of features) of the examples grow, because a fixed-size training set covers a dwindling fraction of the input space.

Credit: Pedro Domingos (2012)
A Few Useful Things to Know About ML

Feature Engineering Is The Key

At the end of the day, some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used. If you have many independent features that each correlate well with the class, learning is easy. On the other hand, if the class is a very complex function of the features, you may not be able to learn it.

Credit: Pedro Domingos (2012)
A Few Useful Things to Know About ML

More Data Beats a Cleverer Algorithm

data means more scalability issues. Fixed size learners (parametric learners) can take advantage of data only to an extent beyond which adding more data does not improve the results. Variable size learners (non-parametric learners) can, in theory, learn any function given sufficient amount of data. Of course, even non-parametric learners are bound by limitations of memory and computational power.

Credit: Pedro Domingos (2012)
A Few Useful Things to Know About ML

Learn Many Models, Not Just One

In early days of machine learning, the model/learner to be trained was pre-determined and the focus was on tuning it for optimal performance. Then the focus shifted to trying many variants of different learners. Now the focus is on combining the various variants of different algorithms to generate the most optimal results. Such model ensembling techniques include bagging, boosting and stacking.
A Few Useful Things to Know About ML

**Representation Does Not Imply Learnable**

Just because a function can be represented, does not mean that the function can actually be learnt. Restrictions imposed by data, time and memory, limit the functions that can actually be learnt in a feasible manner. For example, decision tree learners can not learn trees with more leaves than the number of training data points. The right question to ask is "whether a function can be learnt" and not "whether a function can be represented".

Credit: Pedro Domingos (2012)
A Few Useful Things to Know About ML

**Correlation Does Not Imply Causation**

Correlation may hint towards a possible cause and effect relationship but that needs to be investigated and validated. On the face of it, correlation can not be taken as proof of causation.

Credit: Pedro Domingos (2012)
ML in Practice

1. **Understand the domain, prior knowledge and goals.** Talk to domain experts. Often the goals are very unclear. You often have more things to try then you can possibly implement.

2. **Data integration, selection, cleaning and pre-processing.** This is often the most time consuming part. It is important to have high quality data. The more data you have, the more it sucks because the data is dirty. Garbage in, garbage out.

3. **Learning models.** The fun part. This part is very mature. The tools are general.

4. **Interpreting results.** Sometimes it does not matter how the model works as long it delivers results. Other domains require that the model is understandable. You will be challenged by human experts.

5. **Consolidating and deploying discovered knowledge.** The majority of projects that are successful in the lab are not used in practice. It is very hard to get something used.
A Typical ML Deployment Scenario
Rules for ML Systems in the Wild

1. Don’t be afraid to launch a product without machine learning.
2. First, design and implement metrics.
3. Choose machine learning over a complex heuristic.
4. Keep the first model simple and get the infrastructure right.
5. Test the infrastructure independently from the machine learning.
6. Detect problems before exporting models.
7. Watch for silent failures.
8. Plan to launch and iterate.
9. You are not a typical end user.
10. Measure the delta between models.
Do Not Reinvent the Wheel
Structure Your Project

- Makefile <- tasks
- config.yml <- config file in YAML, can be exported as env vars if needed
- config-private.yml <- config file with private config (password, api keys, etc.)
- data
  - raw
  - intermediate
  - processed
  - temp
- results
  - outputs
  - models
- logs
- checkpoints
- documents
  - docs
  - images
  - references
- notebooks <- notebooks for explorations / prototyping
- src <- all source code, internal org as needed
Know Your Data

A

Raw data

Clean data

Features

Model

Results

Pre-processing

Feature extraction

Training

Evaluation

B

Supervised

Unsupervised

x → y

x

- Linear regression
- Logistic regression
- Random Forest
- SVM
-...

- PCA
- Factor analysis
- Clustering
- Outlier detection
-...

C

Raw data

Discriminative features

Intron

Exon

Feature extraction

D

Label

Layer 2
- TSS
- Intron
- Exon

Layer 1

Raw data
Start Small

Always start simple, small and easy.
Always Monitor

TensorBoard

SCALARS  IMAGES  AUDIO  GRAPHS  DISTRIBUTIONS  HISTOGRAMS  EMBEDDINGS  TEXT

Write a regex to create a tag group

gradient

parameter

Histogram Mode

OVERLAY  OFFSET

Offset Time Axis

STEP  RELATIVE  WALL

Runs

Write a regex to filter runs

n_samples_1/20170530_141631
n_samples_5/20170530_141605

log
Always Explore
Know Your Boundaries
Learn from Your Mistakes

Confusion Matrix

- True Class
  - class 1
  - class 2
  - class 3
  - class 4
  - class 5
  - class 6
  - class 7

- Predicted Class

Confusion Matrices are heat maps to visualize the accuracy of a binary or multiclass classifier.

Credit: Chris Albon
Be Careful About the Adversarial Attacks

Adversarial sample crafting exploits the decision boundary:
- bypassing it (evasion)
- modifying it (poisoning)

Credit: Goodfellow et al. (2015)
Be Careful About the Adversarial Attacks

These attacks can be used to manipulate self driving cars!
Be Careful About the Adversarial Attacks

<table>
<thead>
<tr>
<th>CVE-ID</th>
<th>Vulnerability</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>2016-1516</td>
<td>Heap Corruption in FE</td>
<td>Code Execution</td>
</tr>
<tr>
<td>2016-1517</td>
<td>Heap Corruption in FE</td>
<td>DoS</td>
</tr>
<tr>
<td>n/a</td>
<td>Inconsistent rendering in FE</td>
<td>Evasion</td>
</tr>
</tbody>
</table>

Bugs in ML software can be exploited for various attacks, such as using OpenCV’s bugs to bypass face identification.

Credit: Octavian Suciu
Debug Your Learning Algorithm

Suppose you have implemented regularized linear regression to predict housing prices.

\[ J(\theta) = \frac{1}{2m} \left[ \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{m} \theta_j^2 \right] \]

However, when you test your hypothesis on a new set of houses, you find that it makes unacceptably large errors in its predictions. What should you try next?

- Get more training examples
- Try smaller sets of features
- Try getting additional features
- Try adding polynomial features \((x_1^2, x_2^2, x_1x_2, \text{etc.})\)
- Try decreasing \(\lambda\)
- Try increasing \(\lambda\)
Debug Your Learning Algorithm

Suppose you have implemented regularized linear regression to predict housing prices. However, when you test your hypothesis in a new set of houses, you find that it makes unacceptably large errors in its prediction. What should you try next?

- Get more training examples  $\rightarrow$ fixes high variance
- Try smaller sets of features  $\rightarrow$ fixes high variance
- Try getting additional features  $\rightarrow$ fixes high bias
- Try adding polynomial features $(x_1^2, x_2^2, x_1x_2, \text{etc})$  $\rightarrow$ fixes high bias.
- Try decreasing $\lambda$  $\rightarrow$ fixes high bias
- Try increasing $\lambda$  $\rightarrow$ fixes high variance
Test Your ML Code

```
def make_convnet(input_image):
    net = slim.conv2d(input_image, 32, [11, 11], scope="conv1_11x11")
    net = slim.conv2d(input_image, 64, [5, 5], scope="conv2_5x5")
    net = slim.max_pool2d(net, [4, 4], stride=4, scope='pool1')
    net = slim.conv2d(input_image, 64, [5, 5], scope="conv3_5x5")
    net = slim.conv2d(input_image, 128, [3, 3], scope="conv4_3x3")
    net = slim.max_pool2d(net, [2, 2], scope='pool2')
    net = slim.conv2d(input_image, 128, [3, 3], scope="conv5_3x3")
    net = slim.max_pool2d(net, [2, 2], scope='pool3')
    net = slim.conv2d(input_image, 32, [1, 1], scope="conv6_1x1")
    return net
```

Can you spot the bug?
Wrap Up

- Using Machine Learning is a big challenge in product development.
- You would need to consider many aspects of theoretical and practical Machine Learning.
- You would need to solve both research and engineering problems.
- You will have many constraints such as speed, accuracy, consistency, security, robustness and customer satisfaction, etc.
- Do not reinvent the wheel, there are many resources available online.
- Use a consistent Machine Learning pipeline.
- Prefer simple over complex.
- Always iterate.
Quiz Questions

1. Which process **does not belong to** standard ML product development pipeline?
   a. Deploying a model
   b. Developing a model
   c. Developing a mobile application
   d. Monitoring model performance

2. Which **is not considered** as True?
   a. Dimensionality Reduction is a supervised learning method.
   b. Clustering is an unsupervised learning method.
   c. Classification is a supervised learning method.
   d. Regression is a supervised learning method.
Bonus
Case Study - Neural Program Synthesis

Traditional synthesis

Task
- Specification
  \[ \forall i, \text{in}[i] \in \text{out} \land \text{out}[i] \in \text{in} \]
  \[ \forall i < j, \text{out}[i] \leq \text{out}[j] \]
  \[ [1,3,2] \rightarrow [1,2,3] \]
  \[ [20,47,12,31] \rightarrow [12,20,31,47] \]
  ...

Source code
- def bubblesort(input):
  - for i in range(len(input)):
  -     for j in range(len(input) - 1 - 1):
  -         if input[j] > input[j + 1]:
  -             temp = input[j]
  -             input[j] = input[j + 1]
  -             input[j + 1] = temp

Data types
- bool, int, int[], bit vectors, string, float

Methods
- Enumerative search
- Constraint solving

Deep learning

Task
- Examples
  \[ 5 \rightarrow 5 \]
  \[ 3 \rightarrow 3 \]

Black-box function

Data types
- Perceptual
- Big data

Methods
- Gradient descent
Case Study - Neural Program Synthesis

```python
def run():
    move()
    if leftIsClear():
        turnLeft()
    if MarkersPresent():
        pickupMarker()
    else:
        move()
```
Questions?

Semih Yagcioglu
@semihyagcioglu

Dec 10, 2018
CS491 - Bilkent University