Black Box Machine Learning

David S. Rosenberg

Bloomberg ML EDU

May 31, 2018

Overview

- What is machine learning for?
- What is machine learning?
- How do I do it? (e.g. properly use an ML library)
- What can go wrong?
- Case study

Machine Learning Problems

Common theme is to solve a prediction problem:

- given an **input** *x*,
- predict an "appropriate" output y.

We'll start with a few canonical examples...

Example: Spam Detection

• Input: Incoming email



- Output: "SPAM" or "NOT SPAM"
- A binary classification problem, because only 2 possible outputs.

Example: Medical Diagnosis

- Input: Symptoms (fever, cough, fast breathing, shaking, nausea,...)
- Output: Diagnosis (pneumonia, flu, common cold, bronchitis, ...)
- A multiclass classification problem: choosing one of several [discrete] outputs.

How to express uncertainty?

• Probabilistic classification or soft classification:

 $\begin{aligned} \mathbb{P}(\text{pneumonia}) &= 0.7\\ \mathbb{P}(\text{flu}) &= 0.2\\ \vdots & \vdots \end{aligned}$

Example: Predicting a Stock Price

- Input: History of stock's prices
- Output: Predict stock's price at close of next day
- A regression problem, because the output is a number.

(Regression is **not** just "linear regression" from basic statistics.)

- A prediction function takes input x and produces an output y.
- We're looking for prediction functions that solve particular problems.
- Machine learning helps find the best prediction function.

What is Machine Learning?

What is not ML: Rule-Based Approaches

- Consider medical diagnosis.
 - Consult textbooks and medical doctors (i.e. "experts").
 - ② Understand their diagnosis process.
 - Implement this as an algorithm (a "rule-based system")
- Doesn't sound too bad...
- Very popular in the 1980s.

(To be fair, these "**expert systems**" could be much more sophisticated than they sound here. For example, through "inference" they could make new logical deductions from knowledge bases.)

Rule-Based Approach



David S. Rosenberg (Bloomberg ML EDU)

Fig 1-1 from Hands-On Machine Learning with Scikit-Learn and TensorFlow by Aurelien Geron (2017).

Issues with rule-based systems:

- Very labor intensive to build.
- Rules work very well for areas they cover
 - But cannot generalize to unanticipated input combinations.
- Don't naturally handle uncertainty.
- Expert systems seen as "brittle"

- Don't reverse engineer an expert's decision process.
- Machine "learns" on its own.
- We provide "training data", i.e.
 - many examples of (input x, output y) pairs.
 - e.g. A set of videos, and whether or not each has a cat.
 - e.g. A set of emails, and whether or not each is SPAM.
- Learning from training data of this form is called supervised learning.

Machine Learning Algorithm

- A machine learning algorithm:
 - Input: Training Data
 - "Learns" from the training data.
 - **Output**: A "prediction function" that produces output y given input x.

Machine Learning Approach



David S. Rosenberg (Bloomberg ML EDU)

Fig 1-2 from Hands-On Machine Learning with Scikit-Learn and TensorFlow by Aurelien Geron (2017).

Key Concepts

• most common ML problem types

- classification (hard or soft/probabilistic)
- multiclass (hard or soft/probabilistic)
- regression

prediction function

- predicts output *y* given input *x*
- training data
 - a set of (input x, output y) pairs
- supervised learning algorithm
 - takes training data and produces a prediction function

Elements of the ML Pipeline

Feeding Inputs to ML Algorithms

- Raw input types can be
 - Text documents
 - Variable-length time series
 - Image files
 - Sound recordings
 - DNA sequences
- But most ML prediction functions like their input as
 - fixed-length arrays of numbers
 - double[d] for the computer scientists
 - \mathbf{R}^d for the mathematicians

Feature Extraction

Definition

Mapping raw input x to double [d] is called feature extraction or featurization.



- Better features \implies less "smart" ML needed (makes things easier)
 - Limiting case: a single feature is already the correct output
- Feature vectors are often called input vectors.

Example: Detecting Email Addresses

- Task: Predict whether a string is an email address
- Could use domain knowledge and write down:



• This was a bit ad-hoc. Could we be more systematic? Yes ...

David S. Rosenberg (Bloomberg ML EDU)

From Percy Liang's "Lecture 3" slides from Stanford's CS221, Autumn 2014.

Feature Template: Last Three Characters Equal

- Don't think about which 3-letter suffixes are meaningful...
- Just include them all.



From Percy Liang's "Lecture 3" slides from Stanford's CS221, Autumn 2014.

- one-hot encoding: a set of binary features that always has exactly one nonzero value.
- categorical variable: a variable that takes one of several discrete possible values:
 - NYC Boroughs: "Brooklyn", "Bronx", "Queens", "Manhattan", "Staten Island"
- Categorical variables can be encoded numerically using one-hot encoding.
 - In statistics, called a dummy variable encoding
- Concept Check: How many features to one-hot encode the boroughs?

Labeled Data

• Package feature vectors together with output "labels":

Ftrl	Ftr2	 FtrD	Y
0	1.54	 932	False
1	-1.9	 200	True
0	2.3	 0	False

- Each row is an "example" or "labeled datum".
- The last column is the **output** or "**label**" column.

Unlabeled Data

• Just the feature vectors:

Ftrl	Ftr2	 FtrD	Y
0	1.54	 932	?
1	-1.9	 200	?
0	2.3	 0	?

• We want to be able to predict the missing labels.

Prediction Functions

A prediction function has

- input: a feature vector (a.k.a. "input vector")
- output: a "label" (a.k.a. "prediction", "response", "action", or "output")

[Unlabeled] Input Data





The prediction function is what gets deployed.

Learning Algorithm

A learning algorithm has

- input: labeled data (i.e. the training set)
- output: a prediction function



A training set of labeled data

Today is about what's outside the "purple box".

Key Concepts

• feature extraction

- maps raw inputs into arrays of numeric values
- ideally, extracts essential features of the input
- one-hot encoding for categorical variables
- labeled data / unlabeled data

Evaluating a Prediction Function

Evaluating a Prediction Function

- Brilliant data science intern gives you a prediction function.
- How do we evaluate performance?
- Very important part of machine learning.
 - It can be subtle.
 - Evaluation should reflect business goals as closely as possible.

Evaluating a Single Prediction: The Loss Function

- A loss function scores how far off a prediction is from the desired "target" output.
 - loss(prediction, target) returns a number called "the loss"
 - Big Loss = Bad Error
 - Small Loss = Minor Error
 - Zero Loss = No Error

Classic Loss Functions

- Classification loss or "0/1 Loss"
 - Loss is 1 if prediction is wrong.
 - Loss is 0 if prediction is correct.
- Square loss for regression
 - loss = $(predicted target)^2$

Evaluating a Prediction Function

- Data science intern gives you a prediction function f(x).
 - "Average classification loss on training data was 0.01" (i.e. 1% error)
- $\bullet\,$ Product manager says "we can deploy if $\leqslant 2\%$ error."
- Deploy this prediction function?
 - No!
- Prediction function needs to do well on new or "out-of-sample" inputs
 - out-of-sample means not in the training settings
- (Don't test somebody with problems they've seen in advance.)

- A "test set" is labeled data that is independent of training data.
- e.g. Split labeled data randomly into 80% training and 20% test.
- Training set: only for training prediction functions.
- Test set: only for for assessing performance.
- Larger test set gives more accurate assessment of performance.
- How big? We could review "confidence intervals" from statistics.

Train/Test vs. Train/Deploy

- Train/Test:
 - Build model on training data (say 80% of all labeled data).
 - Get performance estimate on test data (remaining 20%).
- Train/Deploy:
 - Build model on all labeled data.
 - Deploy model into wild.
 - Hope for the best.
- A large part of real-world machine learning is ensuring that
 - Test performance is a good estimate of deployment performance.
- How can we do this, and what can go wrong?

- Train/Test setup should represent Train/Deploy scenario as closely as possible.
- Random split of labeled data into train/test is usually the right approach.
 - (why random?)
- But consider time series prediction: 1000 days of historical data
 - Should we randomly split the days into training and test?

Train/Test Split for Time Series

- Consider Train/Deploy scenario:
 - Prediction function trained on days occurring before deployment time period.
- Consider Train/Test scenario with random splitting:
 - Some test days occur before some training days.
 - No good!
- What can go wrong with random splitting of time series?
 - Suppose time series changes slowly over time.
 - To predict at test day d, just predict value at training day closest in time.
 - That trick won't work for very long during deployment.
- Create train/test split by splitting in time:
 - Training set is everything before time T
 - Test set everything after time T

Train

Test

David S. Rosenberg (Bloomberg ML EDU)

Jatin Garg (https://stats.stackexchange.com/users/123886/jatin-garg), Using k-fold cross-validation for time-series model selection, URL (version: 2017-03-22): https://stats.stackexchange.com/q/268847

Summary: What to Give your Data Science Intern

- Split data into train and test.
- Give training set to intern, you keep the test set.
- Intern gives you a prediction function.
- You evaluate prediction function on test set.
- No matter what intern did with training set,
 - test performance should give you good estimate of deployment performance.

What Should the Intern Do?

- Intern wants to try many fancy ML models.
- Each gives a different prediction function.
- Intern needs her own test set to evaluate prediction functions.
- Intern should randomly split data again into
 - training set and
 - validation set (also called the development or "dev" set)
- This split could again be 80/20.
- Validation set is like test set, but used to choose best among many prediction functions.
- Test set is just used to evaluate the final chosen prediction function.

k-Fold Cross Validation

- Suppose test set too small for good performance estimate.
- Use *k*-fold cross validation:
 - Randomly partition data D into k "folds" of equal size:
 D₁,..., D_k.

2 For
$$i = 1, ..., k$$
:

- 1 Train model M_i on $\mathcal{D} \mathcal{D}_i$.
- Let T_i be M_i's performance on D_i.
- **3** Report $\hat{T} \pm SE(\hat{T})$ where

$$\hat{T} = \text{Mean}(T_1, \dots, T_k)$$

 $SE(\hat{T}) = SD(T_1, \dots, T_k)/\sqrt{k}.$

Provost and Fawcett Data science for Business, Figure 5-9.



Forward Chaining (Cross Validation for Time Series)



David S. Rosenberg (Bloomberg ML EDU)

Jatin Garg (https://stats.stackexchange.com/users/123886/jatin-garg), Using k-fold cross-validation for time-series model selection, URL (version: 2017-03-22): https://stats.stackexchange.com/q/268847

Key Concepts

loss functions

- e.g. 0/1 loss (for classification)
- e.g. square loss (for regression)
- training set, validation set, test set
 - train/test should resemble train/deploy as closely as possible
 - random split often reasonable
 - for time series, split data in time, rather than randomly
 - validation and test sets are often called "hold-out data"
- k-fold cross validation for small datasets

Other Sources of Test \neq Deployment

- Leakage: Information about labels sneaks into features.
- Examples:
 - identifying cat photos by using the title on the page
 - including sales commission as a feature when ranking sales leads
 - using star rating as feature when predicting sentiment of Yelp review

Sample Bias

- Sample bias: Test inputs and deployment inputs have different distributions.
- Examples:
 - create a model to predict US voting patterns, but phone survey only dials landlines
 - building a stock forecasting model, but training using a random selection of companies that exist today what's the issue?
 - US census slightly undercounts certain subpopulations in a way that's somewhat predictable based on demographic and geographic features.
 - If predictable, can it be corrected? Hotly debated topic ~2000 some of the world's top statisticians couldn't agree (Stephen Fienberg vs David Freedman).)

- Nonstationarity: when the thing you're modeling changes over time
- Nonstationarity often takes one of two forms:
 - Covariate shift: input distribution changed between training and deployment.
 - (covariate is another term for input feature)
 - e.g. once popular search queries become less popular new ones appear
 - mathematically similar to sample bias
 - Concept drift: correct output for given input changes over time
 - e.g. season changes, and given person no longer interested in winter coats
 - e.g. last week I was looking for a new car, this week I'm not

Model Complexity & Overfitting

Toy Example

• Green line is truth; Blue points are our noisy data



• What's the input? What's the output?

From Bishop's Pattern Recognition and Machine Learning, Ch 1.

Polynomial Curve Fitting (an ML algorithm)

• Fit data with a polynomial.

$$f(x) = w_0 + w_1 x + w_2 x^2 + \dots + w_M x^M$$

• Concept Check: What is f(x) in our ML vocabulary?

Polynomial Curve Fitting (an ML algorithm)

- Fit with polynomial $f(x) = w_0 + w_1 x + w_2 x^2 + \dots + w_M x^M$
- Imagine a learning function:

double[] fitPolynomial(Data data, int M)

- This function does the "learning".
- Returns array of **parameters** w_0, w_1, \ldots, w_M .
- With parameters and *M* we can create **prediction function**:

double predictPolynomial(double[] w, int M, double x)

Polynomial Curve Fitting (an ML algorithm)

- A polynomial model $f(x) = w_0 + w_1 x + w_2 x^2 + \dots + w_M x^M$
- Learning algorithms find the best parameters w_0, w_1, \ldots, w_M .
- A hyperparameter is a parameter of the ML algorithm itself.
 - Here, *M* is a hyperparameter.
- Generally, the data scientist adjusts the hyperparameters.
- Though it can also be chosen by an ML algorithm.

• Green curve is truth



From Bishop's Pattern Recognition and Machine Learning, Ch 1.

• Fit with M = 0:



UNDERFIT (not fitting data well enough)

From Bishop's Pattern Recognition and Machine Learning, Ch 1.

• Fit with M = 1



UNDERFIT (not fitting data well enough)

From Bishop's Pattern Recognition and Machine Learning, Ch 1.

• Fit with M = 3



PRETTY GOOD!

From Bishop's Pattern Recognition and Machine Learning, Ch 1.

• Fit with M = 9



OVERFIT (fits data too well)

From Bishop's Pattern Recognition and Machine Learning, Ch 1.

- *M* controls the model complexity.
- Bigger *M* allows more "complex" prediction functions.
 - i.e. more "squiggly" functions
- Larger model complexity means
 - Better fit to training data
 - NOT necessarily better performance on test data

- Loosely speaking, we say a model overfits when
 - training performance is good but
 - test/validation performance is poor.
- Fix overfitting by
 - Reducing model complexity
 - Getting more training data

• Fit with M = 9 (more data)



Pretty good - slightly overfit?

From Bishop's Pattern Recognition and Machine Learning, Ch 1.

• Fit with M = 9 (even more data)



NAILED IT?

From Bishop's Pattern Recognition and Machine Learning, Ch 1.

David S. Rosenberg (Bloomberg ML EDU)

Hyperparameters (or "Tuning Parameters")

- Almost every learning algorithm has
 - at least one "hyperparameter" or "tuning parameter"
- You (the data scientist) must tune these values.
- Hyperparameter control various things
 - model complexity (e.g. polynomial order)
 - type of model complexity control (e.g. L1 vs L2 regularization)
 - optimization algorithm (e.g. learning rate)
 - model type (e.g. loss function, kernel type,...)

Overall Machine Learning Workflow

Basic Machine Learning Workflow

- Split labeled data into training, validation, and test sets.
- ② Repeat until happy with performance on validation set:
 - Build / revise your feature extraction methodology.
 - Ochoose some ML algorithm.
 - Train ML model with various hyperparameter settings.
 - Evaluate prediction functions on validation set.
- Setrain model on (train + validation)
- Several equation of the set of the set of the set of the second secon
- S Retrain on all labeled data (training + validation + test).
- O Deploy resulting prediction function.

Case Study: Cell Phone Churn Prediction

- Cell phone customers often switch carriers. Called "churn".
- Often cheaper to retain a customer than to acquire a new one.
- You can try to retain a customer by giving a promotion, such as a discount.
- If you give a discount to somebody who was going to churn, you probably saved money.
- If you give a discount to somebody who was NOT going to churn, you wasted money.

- Suppose you have 2 years of customer data.
- For each customer, you know whether they "churned" (i.e. changed service), and the date of churn if they did churn.
- How can we use machine learning to find the most likely churners?

Lift Curves for Predicting Churners



David S. Rosenberg (Bloomberg ML EDU)