Review of Supervised Machine Learning

3 Stages of Understanding ML

1) Capable of using packages to implement
2) Comprehend underlying mathematical methodology/theory } Our Goal
3) Construct your own implementation ← NOT our goal

Overview from Bank Executives

1) Always compare results with a "base case" (linear regression) to demonstrate value added
2) True data analytics is cross-disciplinary
3) Big data can give clean + precise results, but can cause massive failures as well

Review

Regression

Goal is to establish the relationship between dependent variables Y and independent variables X

Can be parametric (ex: linear regression)
or nonparametric (ex: random forest)

Parametric Regression

Begin with some parametric family of functions $f(\cdot, \beta)$

Goal: Estimate the parameters β from data

↳ Given paired data (x_i, y_i), want β so that $y_i \approx f(x_i, \beta)$

Generally write $y_i = f(x_i, \beta) + e_i$ for errors/residuals e_i

How do we define the best β? Minimize the errors

↳ No unique loss function

- $\hat{\sum_{i=1}} (y_i - f(x_i, \beta))^2$ ← attempts to regress the mean
- $\hat{\sum_{i=1}} |y_i - f(x_i, \beta)|$ ← attempts to regress the median
- $\max_i |y_i - f(x_i, \beta)|$

Choice of loss function can greatly impact the regression estimated

Q: How do you estimate the quantile of the data?

A: $(\alpha - 1) \sum_{y_i < f(x_i, \beta)} (y_i - f(x_i, \beta)) + \alpha \sum_{y_i > f(x_i, \beta)} (y_i - f(x_i, \beta))$

Think about the problem you are interested in to determine the loss function

Linear Regression

Want to predict y using data x

↳ $y = \beta_0 + \beta^T x +$ error

Often estimate with least squares error

↳ 1) Mathematically analytical solution exists

$$\begin{pmatrix} \beta_0 \\ \beta \end{pmatrix} = (X^T X)^{-1} X^T y \quad \text{for} \quad X = \begin{pmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & & \vdots \end{pmatrix}, \; y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \end{pmatrix}$$

2) Computationally easy

3) Encodes a tradeoff between minimizing outlier errors & controlling near the "average"

Can easily extend to polynomial regression

ex: $y = \beta_2 x^2 + \beta_1 x + \beta_0 + \text{error}$

is a linear regression on $((x, x^2), y)$

Be wary of overfitting if using high dimensional models without an underlying meaning (ex: in physics for laws of motion)

Can also be used for autoregressive models

↳ Typical in time series analysis (FA 542 / FIN 620)

↳ Output desired is "tomorrow's" data

Ridge & LASSO Regression

Goal is to "shrink" the model

Ridge: For tuning parameter $\lambda \geq 0$, we seek to minimize $RSS + \lambda \sum_{j=1}^{p} \beta_j^2$

LASSO: minimize $RSS + \lambda \sum_{j=1}^{p} |\beta_j|$ → Produces sparse models (many predictors have $\beta_j = 0$)

Classification

Goal is to predict a (discrete) class Y from data X

Can view as a regression with discrete output space

Basics

Classifier f takes input data + provides a "guess" for the class label

ex: $f(x) = \begin{cases} 1 & \text{if } \eta(x) \geq 1/2 \\ 0 & \text{if } \eta(x) < 1/2 \end{cases}$

Confusion Matrix for binary classification

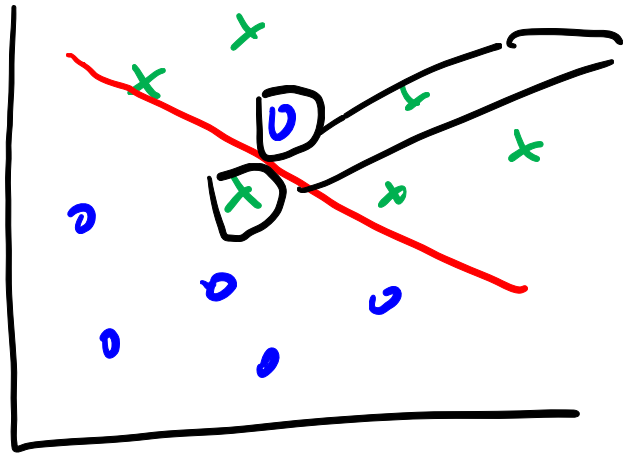
Predicted class \ Actual Class	1	0
1	TP	FP
0	FN	TN

A large number of accuracy scores are derived from the confusion matrix

Linear Discriminant Functions

Idea: Want a linear function $\beta_0 + \beta^T x$ so that

$$f(x) = \begin{cases} 1 & \text{if } \beta_0 + \beta^T x \geq 0 \\ 0 & \beta_0 + \beta^T x < 0 \end{cases}$$



False positive/negative

Related to the logistic regression:

Assume there is a linear relationship between x & "log-odds" of $y=1$

$\hookrightarrow l = \log\left(\frac{p}{1-p}\right) = \beta_0 + \beta^T x \Rightarrow p = \left(1 + \exp(-[\beta_0 + \beta^T x])\right)^{-1}$

Note: $p \geq 1/2 \iff \beta_0 + \beta^T x \geq 0$

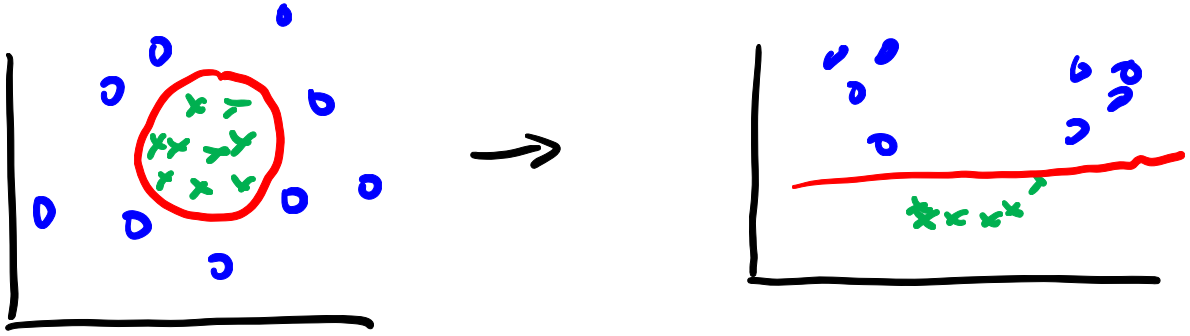
Support Vector Machines (SVM)

Seek to create the "widest" gap between the 2 classes
while minimizing the error from misclassifications (in training data)

"Plain" SVM finds a linear classifier

Use Kernel trick for nonlinear classifiers

↳ remap/change dimensions of the original input
so that the classes are linearly separable in the new space



Decision Tree & Random Forest

Idea [Decision Tree]: Sequentially partition data until the class is "clear"

Benefit: very easy to interpret

Cost: generally very sensitive to training data & subject to overfitting

↳ Random Forest

Idea: Create many different decision trees from training data
Choose the class most commonly predicted

This is a type of ensemble model (model with the vote of many algorithms)

Validation + Backtesting

Want: Test the model performance on randomly selected data

A common approach is K-fold Cross-Validation

Idea: Split data into K (roughly) equally-sized groups (randomly)

Train the model on K-1 groups & test on the last group

↳ Keep the test evaluation score

Repeat K times (with different group removed for testing)

Model "accuracy" is the average of evaluation scores

Why do we do this?

Want to evaluate how the results of statistical analysis will generalize to new data

↳ This can assist in tuning hyperparameters to avoid overfitting

↳ Can also provide approximate information about the accuracy of the model on new data

If scores are "good" then train the model on the full dataset to prepare for new data

For time series models

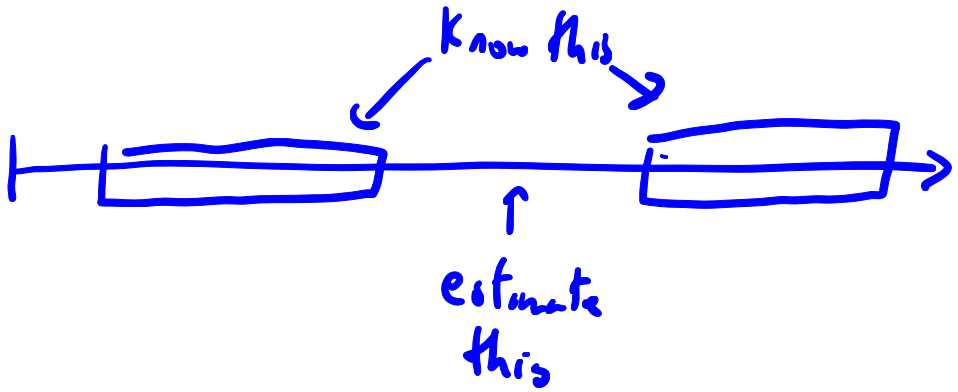
If inputs are $(x_{t-1}, x_{t-2}, \ldots, x_{t-p})$ with output y_t

↳ Full data set is $((x_{t-1}, \ldots, x_{t-p}), y_t)_{t=p}^{T}$ [fixed length memory]

Then cross-validation is useful

For models with "memory". Then K-fold CV does NOT work

↳ If we split our data randomly, we will be using the "future"
to train & estimate the "past"



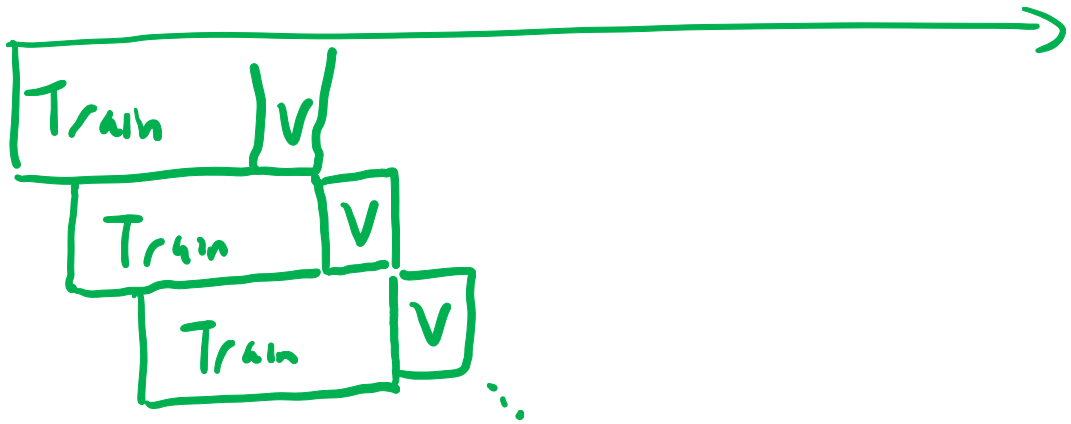
This is NOT helpful
when applying to new data

Two Validation Approaches

1) **Rolling Window** : Take T prior data points to predict the next one (for example)

Move forward in time + apply again

Each window of time, a new model is trained + tested



The window of training data constantly moves forward in time

Since financial markets change over time, this method realistically

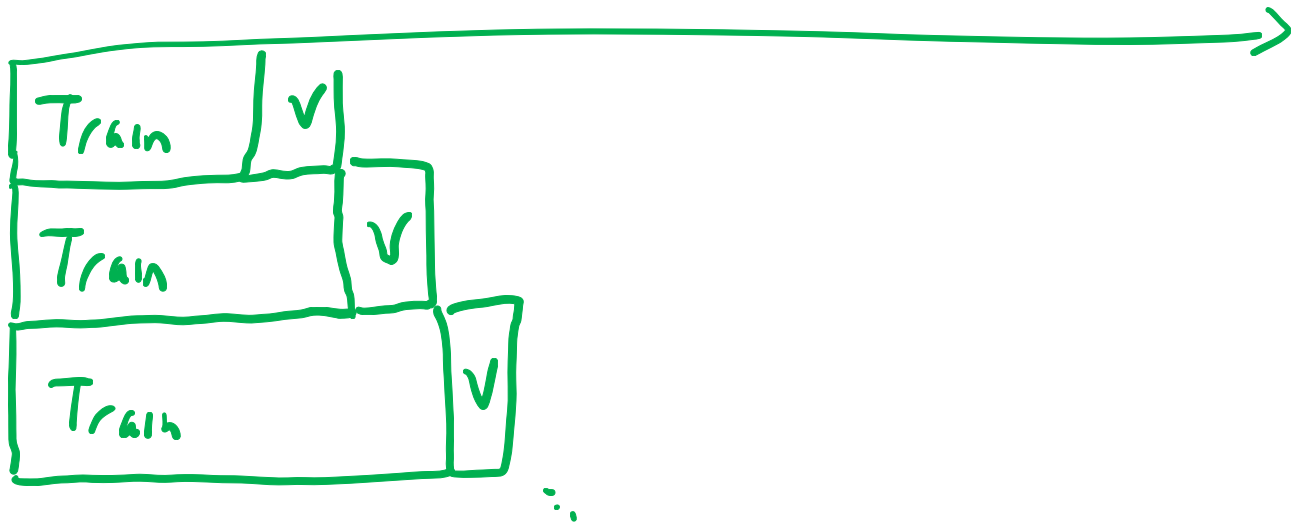
Considers cutting off past data (do you care about data in 1940?)

Note: Choice of T matters

2) Chaining: Constantly expand the window of training data to predict the next point

Move forward in time & apply again

Each window in time is trained & tested



Never "throws out" old data

Common if you want to make sure you include specific events

↳ ex: 2008 or Spring 2020

Especially useful for risk management