AI Exam II Review

April 5, 2018

Contents

1 The Machine Learning Landscape **1**
1.1 The Machine Learning Landscape 1
1.2 Training Mode 1
1.3 Type of Model 1
1.4 Challenges of ML 1

2 End to End Machine Learning Project **2**
2.1 Look at the big picture 2
2.2 Get the Data 2
2.3 Discover and visualize the data to gain insights 3
2.4 Prepare the data for Machine Learning algorithms 3
2.5 Select a model and train it 3
2.6 Fine-tune your model 3

3 Classification **3**
3.1 Performance Measures 3
3.2 Multi class Classifier 4
3.3 Error Analysis 4
3.4 Multi label Classification 4
3.5 Multi output Classification 4

4 Training Models **5**
4.1 Linear Regression 5
4.2 Polynomial Regression 6
4.3 Learning Curves 7
4.4 Bias/Variance Trade off 7
4.5 Regularized Linear Model 7
4.6 Logistic Regression 8

5 Decision Trees **8**
5.1 Making Predictions 8
5.2 Estimating Class Probabilities 9
5.3 CART Training Algorithm 9
5.4 Computational Complexity 9
5.5 Gini Impurity or Entropy 9
5.6 When to Stop? 9
5.7 Limitations of Decision and Regression Trees 9

6 Ensemble Learning and Random Forests **10**

7 Perceptions, Networks, and Activation Functions **10**

8 YouTube, The Great Radicalize **10**

9 Asking the Right Questions about AI **10**

1 The Machine Learning Landscape

1.1 The Machine Learning Landscape

Supervised Learning Correct outputs given for training examples.

> **Input** Examples and features
>
> **Output Labels** Classification vs Regression
>
> **Regression** Given a set of features, used to predict a target numeric value.

Unsupervised Learning Training data unlabeled. Examples: clustering

Semi-Supervised Learning Partial labeled training data.

Reinforcement Rewards/penalties for guesses given.

1.2 Training Mode

Online Incremental, not necessarily live, learning rate matters: how fast they should adapt to changing data.

Batch Uses all available data done offline, to replace: train from scratch with all available data then replace old one.

1.3 Type of Model

Instance Based System learns the examples by heart, then generalizes to new cases using a similarity measure.

Model Based Cost function can be adjusted to minimize some measure of error/cost.

1.4 Challenges of ML

Insufficient quantity of training data

Non representative data This is why domain knowledge is important!.

Poor-quality data Outliers, missing features.

Irrelevant features Choosing the right features can make a learning task much easier.

Overfitting Essentially memorizing the training data (including noise), which does not generalize well. Solutions include using simpler model, fewer degrees of freedom, regularize, get more data.

Underfitting Model not complex enough. Solution include using more complex model (more degrees of freedom), better features.

Testing and Validating It is important to separate testing, training, and validating data.

  Training set and Test set Split the data set into training and testing sets.

    Generaliation Error How well does it do on the test set?

    Rule of thumb 80% of data for training, 20% for testing.

  Problem: You can't look at the test set untill the end! If you do (while you are trying different models), you might be subconsciously fitting the testing data; this is "data snooping". Solution is to have a third validation set.

No Free Lunch Theorem David Wolpert demonstrated that if you make absolutely no assumptions about the data, then there is no reason to prefer one model over any other.

2 End to End Machine Learning Project

2.1 Look at the big picture

- Frame the Problem
- Select a performance measure

  Root Mean Squared Error Euclidean norm, measures the standard deviation of the errors the system makes in its predictions

$$RMSE(X,h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(h\left(x^i\right) - y^i\right)^2}$$

  Mean Absolute Error Manhattan norm, less sensitive to outliers.

$$MAE(X,h) = \frac{1}{m}\sum_{i=1}^{m}\left|h\left(x^i\right) - y^i\right|$$

- Check assumptions

2.2 Get the Data

- Work through the process of cloning from Github to PyCharm
- Take a quick look at the data structure
- Create a test set

2.3 Discover and visualize the data to gain insights

- Consider creating new features
- Look for correlations

2.4 Prepare the data for Machine Learning algorithms

Data Cleaning Dealing with missing features

Convert text and categorical attributes One-hot encoding

Feature Scaling Tweak the datasets to prepare for machine learning.

> **Min-Max** Values are shifted and rescaled so that they end up ranging from 0 to 1.
>
> **Standardization** Does not bound values to a specific range(mean zero, variance 1).

2.5 Select a model and train it

Cross-validation

2.6 Fine-tune your model

Grid Search Exploring relatively few combinations

Randomized search Evaluates a given number of random combinations by selecting value form each hyper parameter at every iteration.

Test Evaluate system on test set.

3 Classification

3.1 Performance Measures

Accuracy
- Cross-validation.
- Compare to dumb constant classifier.
- Not the best measure for highly skewed data.

Confusion Matrix Matrix to determine performance of model.

- Rows are actual
- Columns are predicted

	Predicted	
Actual	True Negative (TN)	False Positive (FP)
	False Negative (FN)	True Positive (TP)

Accuracy=$\frac{TP+TN}{TP+FP+TN+FN}$ Like many of these measures, ranges form 0 to 1.

Precision=$\frac{TP}{TP+FP}$ When you predict yes, how often were you right¿ IF set a very high stander for predicting yes, you can get precision 1.

Recall =$\frac{TP}{TP+FN}$ When the answer was yes, how often were you right? Ration of positive instance that are correctly detected by the classifier. If you st a very low standard of predicting yes, you can get recall 1.

Precision/Recall Tradeoff precision and Recall are mutually exclusive, and there is a trade off between the two.

- Decision function compared to threshold.
- You can plot both as a function of threshold
- Alternately, plot them against each other (PR curve)

ROC Curve Receiver Operating Characteristic

- Plot $\frac{FP}{FP_TN}$ vs $\frac{TP}{TP+FN}$.
- Random classifier produces diagonal line from $(0,0)$ to $(1,1)$.
- Perfect classifier has same endpoints but hits upper left corner.
- You can use area under this curve as a measure
- Use PR curve "whenever the positive class is rare or when you care more about positives then the false negatives"

3.2 Multi class Classifier

OvA Train a detector for each class, then the one with the highest score (one v all).

OvO Train a classifier for each pair, take the one that winds the most "duels" (one v one).

OvA is usually faster because OvO requires training more classifiers.

3.3 Error Analysis

Plot confusion matrix. If its brightest along diagonal, good! If you normalize each row (divide by total instances in row) and zero out diagonals, you can see what gets confused with what.

3.4 Multi label Classification

May tag each example with more than one class

3.5 Multi output Classification

Example: noise removal.

4 Training Models

4.1 Linear Regression

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

- $\hat{y}$ is the predicted value.
- n is the number of features.
- x_i is the i^{th} feature value.
- θ_j is the j^{th} model parameter (including the bias term θ_0 and the feature weights).

$$\hat{y} = h_\theta(x) = \theta^T \cdot x = \sum_{i=0}^{n} \theta_i x_i$$

Training means searching for parameters that minimize error (RMSE, or just MSE).

$$MSE\left(x, h_\theta\right) = \frac{1}{m} \sum_{i=1}^{m} \left(\theta^T \cdot x^i - y^i\right)^2$$

Normal equation;

$$\hat{\theta} \left(X^T \cdot X\right)^{-1} \cdot X^T \cdot y$$

- $\hat{\theta}$ is the value of θ that minimizes the cost function.
- y is the vector of target values containing y^i to y^m.

Gradient decent Tweak parameters iteratively in order to minimize a cost function.

Learning rate Too small, many iterations and thus too long. Too large, might make the algorithm diverge, failing to find a solution.

Local minima If the random initialization starts the algorithm on the left, then it will converge to a local minimum, which is not as good as the global minimum. If it starts on the right, then it will take a very long time to cross the plateau, and if you stop too early you will never reach the global minimum.

Batch Gradient Descent Concept of the gradient

$$\frac{\partial}{\partial \theta_j} MSE(\theta) = \frac{2}{m} \sum_{i=1}^{m} \left(\theta^T \cdot x^i - y^i \right) x)j^i \tag{1}$$

Computes the partial derivative of the cost function with regards to parameter θ_j.

$$\nabla_\theta MSE(\theta) = \begin{pmatrix} \frac{\partial}{\partial \theta_0} MSE(\theta) \\ \frac{\partial}{\partial \theta_1} MSE(\theta) \\ \vdots \\ \frac{\partial}{\partial \theta_n} MSE(\theta) \end{pmatrix} = \frac{2}{m} X^T \cdot (X \cdot \theta - y) \tag{2}$$

To compute all gradients in one go.

At each step, move in the direction opposite the gradient. Stop when the gradient gets too small. Batch mode is slow for large data sets. Want to limit the number of iterations so that grid search can eliminate models that take too long to converge.

Stochastic Gradient Descent Picks a random instance in the training set at every step and computes the gradients based only on that single instance. One instance at a time (in random order). Doe to its stochastic (i.e. random) nature, this algorithm is much less regular than Batch Gradient Descent: instead of gently decreasing until it reaches the minimum, the cost function will bounce up and down, decreasing only on average. Over time it will end up very close to the minimum, but once it gets there it will continue to bounce around, never settling down. So once the algorithm stops, the final parameter values are good, but not optimal. Don't have to fit entire data set in memory at once. Bouncing good for escaping local minimum, but bad for final optimization. A solution is to gradually reduce the learning rate.

Mini-Batch Gradient Descent Computes the gradients on small random sets of instances called mini batches. The main advantage is that you can get a performance boost from hardware optimization of matrix operations, especially when using GPUs.

4.2 Polynomial Regression

Use a linear model to fit nonlinear data. A simple way to do this is to add powers of each feature as new features, then train a linear model on this extended set of features.

Degree too low Underfitting

Degree too high Overfitting

4.3 Learning Curves

Plot of the model's performance on the training set and the validation set as a function of the training set size. Compare curves for training vs validation set.

- Training error stats low and increases to a plateau.
- Validation error starts high and decreases to a plateau.
- Training error is lower than validation error.
- If both curves reach the same plateau, underfitting
- If there is a big gap, overfitting.

4.4 Bias/Variance Trade off

Bias This part of the generalization error is due to wrong assumptions, such as assuming that the data is linear when it is actually quadratic. A high-bias model is most likely to under fit the training data.

Variance This part is due to the model's excessive sensitivity to small variations in the training data. A model with many degrees of freedom is likely to have higher variance, and thus to over fit the training data.

4.5 Regularized Linear Model

Reduce over fitting by construing the model, fewer degrees of freedom, limiting how much the weights can vary.

Ridge Regression Forces the learning algorithm to not only fit the data but also keep the model weights as small as possible.

$$J(\theta) = MSE(\theta) + \alpha \frac{1}{2} \sum_{i=1}^{n} \theta_i^2 \tag{3}$$

Add a regularization term to cost function. Only do this during training, not during validation/ testing. Hyper parameter α determine how much regularization (0 none). The bias term is not regularized. The regularization term is just the l^2 norm of the weight vector. For gradient descent, just add α·weight vector to gradient vector. It is important to scale the data so one feature isn't more heavily regularized.

Lasso Regression Least absolute shrinkage and selection operator regression (LASSO). It adds a regularization term to the cost function, but it uses the l^1 norm of the weight vector instead of half the square of the l^2 norm.

$$J(\theta) = MSE(\theta) + \alpha \sum_{i=1}^{n} |\theta_i| \tag{4}$$

Tends to shrink "unimportant" weights to zero. Sum fudging is necessary because this isn't differential everywhere.

Elastic Net A weighted average of ridge and lasso.

$$J(\theta) = MSE(\theta) + r\alpha \sum_{i=1}^{n} |\theta_i| + \frac{1-r}{2}\alpha \sum_{i=1}^{n} \theta_i^2 \tag{5}$$

Early Stopping After many epochs with a powerful model, training error continues to slowly decrease but validation error starts to go back up. To avoid over fitting, just stop at the lowest validation error.

4.6 Logistic Regression

Computes a weights sum of the input features (plus a bias term), but instead of outputting the result directly like the Linear Regression model does, it outputs the logistic of this result

$$\hat{p} = h_\theta(X) = \sigma\left(\theta^T \cdot x\right) \quad \text{Vectorized form} \tag{6}$$

$$\sigma(t) = \frac{1}{1+e^{-t}} \quad \text{Logistic function} \tag{7}$$

Pass weighted sum to logistic function.

- $\sigma(large) \to 1$
- $\sigma(0) \to 0.5$
- $\sigma(-large) \to 0$

Decision boundaries.

Softmax Regression For multi class classification. Find a weighted sum for each class. Estimate probabilities using softmax function.

$$\hat{p_k} = \sigma(s(x))_k = \frac{e^{s_k(x)}}{\sum_{j=1}^{k} e^{s_j(x)}} \tag{8}$$

A differentiable approximation to max. Use actual argmax when classifying, but softmax when training. Preferred cost function: cross entropy, for measuring distance between two distributions, equivalent to logistic regression cost function when there are two classes.

5 Decision Trees

5.1 Making Predictions

Samples A node's samples attribute counts how many training instances it applies to.

Value A node's value attribute tells you how many training instances of each class this node applies to.

Gini A node's gini attribute measures its impurity: a node is "pure" ($gini = 0$) if all training instances it applies to belong to the same class.

$$G_i = 1 - \sum_{k=1}^{n} p_{i,k}^2 \tag{9}$$

5.2 Estimating Class Probabilities

A decision tree can also estimate the probability that an instance belongs to a particular class k: first it traverses the tree to find the leaf node for this instance, and then it returns the ratio of training instances of class k in this node.

5.3 CART Training Algorithm

Classification and Regression Tree (CART).

The algorithm first splits the training set in two subsets using a single features k and a threshold t_k. It searches for the pair (k, t_k) that produces the purest subsets (weighted by their size). The cost function that the algorithm tries to minimize is given by:

$$J(k, t_k) = \frac{m_{left}}{m} G_{left} + \frac{m_{right}}{m} G_{right} \tag{10}$$

5.4 Computational Complexity

$$O(n \cdot m \log m) \tag{11}$$

5.5 Gini Impurity or Entropy

A set's entropy is zero when it contains instances of only one class

$$H_i = - \sum_{\substack{k=1 \\ p_{i,k} \neq 0}}^{n} p_{i,k} \log\left(p_{i,k}\right) \tag{12}$$

5.6 When to Stop?

Definitely stop when impurity cannot be reduced. Hyper parameters such as max depth, min samples per leaf. Stop if improvement is not statistically significant.

5.7 Limitations of Decision and Regression Trees

Sensitive to individual training points. Not good at finding interactions between features.

6 Ensemble Learning and Random Forests

Ensemble Learning Train several classifiers and have them vote.

Voting Classifiers Different methods for voting.

- **Hard voting** Aggregate the predictions of each classifier and predict the class that gets the most votes
- **Soft voting** Predict the class with the highest class probability, averaged over all the individual classifiers. Often achieves higher performance than hard voting because it gives more weight to highly confident votes.

Bagging and Pasting Take several samples of the data, train a model on each one, then vote. Bagging samples with replacement, pasting without. This reduces both bias and variance. For validation, just use, for each data point, the votes of those models not trained on that particular data point.

Random Subspaces Instead, give each model a different subset of the features.

Random Patches Take random subsets of both features and data points.

Random Forest An ensemble of bagged decision trees, with a random subset of features considered at each node.

Extra-Trees Choose thresholds randomly, creates even more randomness, but faster to train because choosing thresholds takes time.

Feature Importance Across the forest, at what depth does a feature appear (Shallower is more important)

7 Perceptions, Networks, and Activation Functions

8 YouTube, The Great Radicalize

YouTube's algorithm aims to keep you on the site, so it will often feed you more extreme videos, to keep your attention. Politically, you will be drawn more and more towards one side.

9 Asking the Right Questions about AI