Stockholm University, Research Topics in Datascience: DAMI2 *Hand-In-Exam*

Andreas Ährlund-Richter,anah4939,8909080379

June 2019

1 Question 1, Variance and Bias

1.0.1 (a) Variance and bias tradeoff

When we do modeling we try to create a model *f'(x)*, that predicts values *y'* as close as possible to the real values *y*, we want to minimize distance or error between our predicted value for each predicted value y' and the real values, y. The cause of error for prediction models are *bias* and *variance* (also error or noise in the real model, but we cant change that). Variance is the expected squared distance of a point around the mean for the model. Bias is the squared distance of our models mean to the true mean (Hastie et al. (2001)).
For example, kNN, a high-variance model, have a high VC-count (can fit more complicated real models), use many variables(each datapoint), but overfit and change(have model variance) when the training data is changed. kNN doesent make any assumptions on distribution, but just fits training data directly. In contrast, a non-kernel based SVM, a high-bias learner, would instead always assume linearity, and not change much for different training data.

1.0.2 (b) Compare bagging, boosting and stacking

As bagging takes its duplicates and data randomly from the training data, it creates some variance in its predictors, but still has some unadjusted bias. Adaboost on the other hand will focus on more difficult samples by giving them higher weight, and these can include noise and outliers(Dietterich (2001)). Therefore, Adaboost leans towards higher variance than bagging. On noisy datasets, adaboost is proven to perform less good than bagging (T.G. (2000)). Stacking lets the later-stage classifier use earlier predictions misstakes, correcting the bias error of the finished predictors further down the stack. The last-level generalizer cannot be too simple however, then the results will be similar to using cross-evaluation (Wolpert (n.d.)).

1.0.3 c) bias/variance trade-off in diagnosing models.

High-bias models are not so sensitive to irregularities in the dataset, like outliers and noise. When we increase the amount of data, outliers and error has less weight statistically, so depending on the amount of data, or if you have a distribution fitting your biased model, will tilt the tradeoff in eithers favour. Also, important to note, if the actual distribution doesent follow the biased models assumptions, it might never learn even through ensembles, while a high-variane model can become quite good if used in an ensemble the right way and there is sufficient training data to minimize the effect of noise and error.

2 Question 2, Activation Functions

2.0.1 a) Activation functions

The sigmoid function(SF) Returns values between 1 and 0, but its slope/derivative reaches 1 and 0 infinitely slowly. Its a sigmoid curve(Hinkelmann, 2003).
Rectified Linear(RELU) RELU outputs positive values of the function, or 0 if the value is negative(Hinkelmann, 2003).
Leaky Rectified Linear (LRELU) has the negative output weighted by 0.01, and outputs the positive input values in their entirety unweighted Maas (2003).

2.0.2 b) Activation functions pros and cons

The sigmoid function(SF)is popular because its non-linear, letting the network learn non-linear functions. But SF has problem reaching 1, making backpropagation error vanish slowly, making deep networks difficult. On the other hand SF will not generate dead neurons, as it rarely reaches 0, around 0 and 1 for input is were its derivate or slope is the strongest. Experimental results however showed that dead neurons did not prevent learning and back-propagation as much as expected, as some paths usually stayed non-zero (Glorot, n.d.).
RRELU is non linear as it cant have negative values. This makes it possible to have dead neurons which can be good as there is some benefit to sparse networks. Neurons can be removed to make computation quicker, and some problem-spaces are naturally sparse(written characters are mostly white-space etc) (Glorot, n.d.). That RELU doesent taper of like SF and have vanishing gradient, makes backpropagation stronger, and learning faster.
Leaky Rectified Linear (LRELU) is a way to try to avoid dead neurons, yet not reach vanishing gradient problem, while at the same time LRELU not linear, as the negative output is weighted. However (Maas, 2003) found no significant differences between LRELU and RELU when applied to prediction of audio data.

2.0.3 c) Why Afs?

Without activation functions using weights and previous inputs in a non-linear fashion we will simply have the composition of linear functions, which is also a linear function.

The composition f(g(x)) is:

$$h(x) = f(g(x)) \tag{1}$$

when

$$f(x) = 3x - 1 \tag{2}$$

$$g(x) = -x + 2 \tag{3}$$

then

$$h(x) = 3(-x + 2) - 1 \tag{4}$$

$$h(x) = -3x + 5 \tag{5}$$

(Wilson, n.d.)
This cannot represent the nonlinear functions.

3 Question 3, Time Series Mining

3.0.1 a) Why Dynamic Time Warping Distance (DTW) is useful and different from Euclidean Distance

Dynamic Time Warping calculates the closest measurement from one sequence to another, by plotting both series against eachother, as x and y, and calculating a distance matrix. The shortest distance path is detected, and the closest coordinates in a series are matched between the two time-series for each attribute (Maas, 2003). Euclidean just measures each measurement at the same time in both series to eachother, sums them up, and calculates the total distance between the two series. The constraints that optimize DTW are covered in 3.b).
DTW is very useful for matching up time series that potentially follow the same pattern, this can be audio or other data-sequence analysis like market-trends. It can handle time-series being out of phase, because the closest points can be stretched in the series (Tsiporkova, 2012).

3.0.2 b) a dynamic programming algorithm for finding an optimal warping path between T and Q

Theres an exponential amount of possible paths to test, this is not good. There are several constraints commonly used.
Monotonicity. This means that for series X and Y, $i_x - 1$ cant map to a higher point than i_x (Maas, 2003), and This means that for series X and Y, $i_y - 1$ cant map to a higher point than i_y, this prevents the mappings going back in time, and eliminates those possible paths. Usually we are interested in similar patterns that are shifted on the x-axis, not actually the highest similarity for all points, so this is fine (Maas, 2003).
Continuity $i_s - i_{s-1} < 1$ and $j_s - j_{s-1} < 1$ This means that every point in the series for the both series is used at one time. In the shortest part in the distance matrices, there are no holes. This does reduce the amount of possible paths to try, and also removes distance measurements were a bit of one time-series is just left out. We want warping/shift adjustments, not "most similar area" (Maas, 2003).
Boundary conditions. $i_1 = 1$, $i_k = n$ where i_k is the last mapped point in the series, and n is the last measurement of the i-series. This must hold for both time series. Basically the series must start in the lower left corner, and end in top right. This makes suer the end and start of a series isnt skipped when making the shortest path(highest similiarity), also our similarity measurement becomes fair and covers the whole series.
Warping window This is usually represented by the variable r. Two lines are

added to the distance matrix, $y = x + r$ and $y = x - r$. This is also known as a Sakoe-Chiba band (Chiba, 1978). In their research Sakoe and Chiba found that a symmetric distance and diagonal with the slope of 1 was superior to a diagonal constraint that was assymmetric (Chiba, 1978). All paths are forced to stay within these diagonals. Itakura has a stricter boundary using a parallelogram ending and starting at lower left ,top right. The slopes are 2 respective 0.5 for the edges of the parallelogram, but in practice this boundary is implemented in how the shortest path is calculated in the DTW algorithm used by Itakura(Mizutani, 2006). Lower bounding methods can then utilize how the width of the band is decided (see answer c)).

3.0.3 c) Lower-Bounding

Lower bounding is a way to define some measurement for maximum distance between a query and a sequence, usually expressed like $D(P, Q)$ with Query = Q, Sequence = P. The usefulness lies in being able to prune sequences when looking in a database for the most similar sequence to a query. First the best lowest bound is set to infinity, but as new sequences are checked, the best lowest bound is updated, and any sequence with a worse lowest bound is pruned(Nath, 2014). These are approximated to have worse DTW than the current lowest distance. For this however to work, the lower-bound measurement must
1), be easier to calculate.
And 2), have a tight lower bound, with a low ratio to the actual DTW distance.
Another 3) important factor is that good matches should not be pruned by misstake.
4 Triangular inequality between query, a segment, and another segment should preferably also hold. I will describe and compare the bounding by Yi and Kim. Yi uses a very computationally simple lower bound. The distance for all elements that are above the queries max, and belows the queries min are have their euclidean distance to max (for above max items) and to min(for below min items) summed up(Yi, 1997). The computational complexity is linear O(N) (Nath, 2014). Calculating all these positions is computationally taxing however. Also, triangular inequality doesent hold for the measurement (Nath, 2014). However, the lowest bound by Kim holds the inequality, and also is slightly quicker to compute (Kim, 2001). The algorithm only uses the first, last, max and min positions, and compare these. That makes computation much quicker. According to a comparison study, Yi sometimes outperforms Kims lower bound, however as stated earlier it is a bit more demanding computationally (Nath, 2014).

4 Question 4 T-tests

4.0.1 a) Paired t-test

When comparing means of BOSS and HIVE-COTE accuracy, we know the populations are dependent, so we use a paired t-test.
The null hypothesis is that the means between the results are not statistically

significantly different. The alternative hypothesis is that they are. Using Python package numpy and scipy.stats method for paired t-test called "ttest_rel" we get the follow result: $Ttest\_rel.Result(statistic = 10.614603477514066, pvalue = 3.45566254971307e - 17)$ The high negative exponent means we are well below our α of 0.01. **We reject the null hypothesis.**
However, the formula is still supposed to be included in this exam, see below: The way the students t-test is performed is as such, We will use the following formula (https://www.statisticshowto.datasciencecentral.com/probability-and-statistics/t-test/):

$$t = \frac{\frac{(\Sigma_D)}{N}}{\sqrt{\frac{\Sigma_D^2 - (\frac{(\Sigma_D)^2}{N})}{(N-1)(N)}}} \tag{6}$$

Σ_D: Sum of the differences (Sum of X-Y from Step 2)
Σ_D^2 Sum of the squared differences (from Step 4)
$(\Sigma_D)^2$: Sum of the differences (from Step 2), squared.
N: Number of samples in each population.
From Python https://github.com/AndreasAAR/DAMI2ExamCalc these will be:
Σ_D: 9.697089319
Σ_D^2: 1.9394178638
$(\Sigma_D)^2$: 94.1341650426555
N: 85

$$t = \frac{\frac{(9.6)}{85}}{\sqrt{\frac{1.9 - (\frac{94}{85})}{(85-1)(85)}}} t = 10.61 \tag{7}$$

Corresponds perfectly to our built-in statistic function. I have not found a lookup table up to 10.61 in value and Df = n-1 (85-1), but likely it would correspond to our built in functions value of 3.45e-17.

4.0.2 b) Friedman test

We will compare BOSS, HIVE-COTE, Flat-COTE, LS, DDTWR11NN and TSBF using the friedmans test https://www.statisticshowto.datasciencecentral.com/friedmans-test/. The null hypothesis is that there is no statistical difference between the methods compared. The alternate is that there are statistical differences. This can then be investigated with a post-hoc test. One can discuss if Friedmans is correct as the pairs are related, as the datasets are the same. However, (Demsar, 2006) did find the Friedman test to be the most reliable test overall in a study comparing different methods for comparing machine learning classification models. Using python,
https://github.com/AndreasAAR/DAMI2ExamCalc
{statistic=203.92664872139994, pvalue=4.104817195615645e-42 } we find that our result is statistically significant, there IS a difference between the different classifiers. Before performing the post-hoc test however we need to describe the formula and the test.
We will use the following formula https://www.statisticshowto.datasciencecentral.com/friedmans-test/

We need:
1. n: the number of subjects 85
2. k: the number of treatments 39 (classifiers)
3. R: The total ranks for each of the three columns (From python calculate below)
HIVE-COTE: 471.0
Flat-COTE: 231.0
TSBF: 288.0
BOSS: 185.0
$DDTW\_R1\_1NN$; 220.0
LS: 384.0 And Squaredsum: 588227.0 Formula to get statistic:

$$FM = ((\frac{n}{n*k*(k+1)}))*\Sigma R^2 - [3*N*(k+1)] \quad (8)$$

When calculated:

$$FM = ((\frac{n}{n*k*(k+1)}))*221841.0+53361.0+82944.0+34225.0, 48400.0, 147456.0-[3*N*(k+1)] \quad (9)$$

References

Chiba, S. (1978), 'Dynamic Programming Algorithm Optimization for Spoken Word Recognition'.

Demsar, J. (2006), 'Statistical Comparisons of Classifiers over Multiple Data Sets.', *Journal of Machine Learning Researc* .

Dietterich (2001), *Ensemble Methods in Machine Learning*, Springer Series in Statistics, Springer New York Inc., New York, NY, USA.

Glorot, X. (n.d.), 'Deep Sparse Rectifier Neural Networks.'.

Hastie, T., Tibshirani, R. and Friedman, J. (2001), *The Elements of Statistical Learning*, Springer Series in Statistics, Springer New York Inc., New York, NY, USA.

Hinkelmann, K. (2003), 'Neural networks, p. 7'.
URL: *http://didattica.cs.unicam.it/lib/exe/fetch.php?media=didattica:magistrale:kebi:ay*$_1718 : ke - 11_neural_networks.pdf$

Kim, S.-W. (2001), 'An index-based approach for similarity search supporting timewarping in large'.
URL: *https://www.researchgate.net/publication/3892934*$_An_index - based_approach_for_similarity_search_supporting_timewarping_in_large_sequence_databases$

Maas, A. L. (2003), 'Rectifier nonlinearities improve neural network acoustic models'.
URL: *https://pdfs.semanticscholar.org/367f/2c63a6f6a10b3b64b8729d601e69337ee3cc.pdf*

Mizutani, E. (2006), ‘The dynamic time warping algorithms’.
URL: *http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.561.3438rep=rep1type=pdf*

Nath, H. (2014), ‘Evaluation of Lower Bounding Methods of Dynamic Time Warping (DTW)’.

T.G., D. (2000), ‘Ensemble Methods in Machine Learning. In: Multiple Classifier Systems.’, *Lecture Notes in Computer Science, vol 1857.* .

Tsiporkova, E. (2012), ‘Dynamic time warping algorithm’.
URL: *http://www.mathcs.emory.edu/ lxiong/cs730*$_s$13*/share/slides/searching*$_s$*igkdd*2012$_D$*TW.pdf*

Wilson, J. (n.d.), ‘Exploring the Compositions of Linear Functions.’.

Wolpert, D. H. (n.d.), ‘STACKED GENERALIZATION Classifier Systems.’.

Yi, B.-K. (1997), ‘Efficient Retrieval of Similar Time Sequences Under Time Warping’.