Machine Learning Methods for Climbing Route Classification

Alejandro Dobles
Mathematics
adobles@stanford.edu

Juan Carlos Sarmiento
Management Science & Engineering
jcs10@stanford.edu

Peter Satterthwaite
Electrical Engineering
psatt99@stanford.edu

Abstract

Classifying the difficulty of climbing routes is a challenging and subjective task, even for experienced climbers. This project builds on previous work [6] and evaluated several algorithms to aid in this process. Data was sourced from a standard climbing wall (the Moonboard) found in gyms around the world. Three different models were explored for classification: Naive Bayes, softmax regression, and a convolutional neural-network (CNN) with an ordinal output. While all models achieved $\sim 35\%$ top-1 accuracy, the CNN was found to have the best combination of accuracy and accurate reproduction of underlying distribution of the three classifiers, though all models lag behind human classification performance.

1 Introduction

Determining the difficulty grade of climbing routes is notoriously subjective and difficult to do accurately and consistently. Climbers use these grades to benchmark their performance on climbs and find inaccurately graded routes to be frustrating and detrimental to their ability to accurately assess their performance. Several grading systems (Hueco, Fontainebleau, UIAA) exist for classifying the difficulty of bouldering routes, short climbing routes where the climber's fall is arrested by a cushioned pad under the route. The Fontainebleau system is used for this work because it is the most widely used internationally. Typical grades range on a alphanumeric scale in increasing difficulty from 6B+ to 8B+, here represented with integers ranging 1-13.

This project explored different machine learning methods to create a classifier to determine the difficulty of climbing routes. Data was sourced from routes created for the Moonboard. The Moonboard is a standardized climbing wall located in many gyms around the world. The company who sells it has built a platform so that users can generate and share routes to climb. After the users upload the routes they've come up with, they are graded on their difficulty by both the route-setter and the community. More details on the dataset can be found in Section 3.

2 Related Work

The existing literature on machine learning in climbing applications is limited. The major work in the field was published by Phillips et al. in 2011 [6]. This work looked into using chaotic variations in order to design novel climbing routes. The resulting software, named Strange Beta, was used to generate routes in cooperation with an experienced route-setter and the quality of these routes was benchmarked against routes created without the use of this software. The authors claim that the use of Strange Beta resulted in higher quality routes than those created without the software.

Subsequent work on the intersection between climbing and machine learning looking into identification of known climbing routes using information from wearables [3]. These previous works left open several directions for novel research. Phillips et al. only looked into routes in two difficulty categories, and used a general language for route description, preventing them from assessing specific routes. Kosmalla et al. used wearables to distinguish between five known routes, without learning anything about the difficulty of the routes. Despite recent advances in classifiers [4], particularly those for ordinal data [2], accurate classification of climbing route difficulty has yet to be demonstrated.

3 Dataset and Features

Data was sourced from the moonboard website (`www.moonboard.com`). The moonboard consists of $n = 142$ holds arranged on a 18x11 grid. A route is specified by designating a subset of these holds which the user can use to get from specified start holds to end holds. At the time when the data was scraped, the database of routes had 13,871 entries. Each route has a difficulty specified by its setter, and 42% of the routes also have a difficulty determined by the community. Difficulties are divided into 13 ordinal grades. The distribution of problems of different grades was uneven, with 31 % of the routes having the easiest grade, and only 1 % percent of the routes having a grade of 10-13 (See Figure 5 for full distribution).

Routes were randomly divided into 11,095 training routes, 1,388 validation routes and 1,388 test routes, under the restriction that all three sets had the same distribution of difficulty grades. Figure 1 shows the frequency with which each of the 142 holds are used on routes of different difficulty, from easy to hard. From this figure it is clear that for the hardest and easiest routes, certain holds are used more frequently than others, whereas all holds are used roughly equally on intermediate difficulty routes. This implies that identifying intermediate difficulties will be more difficult than extreme difficulties because intermediate difficulties lack characteristic holds.

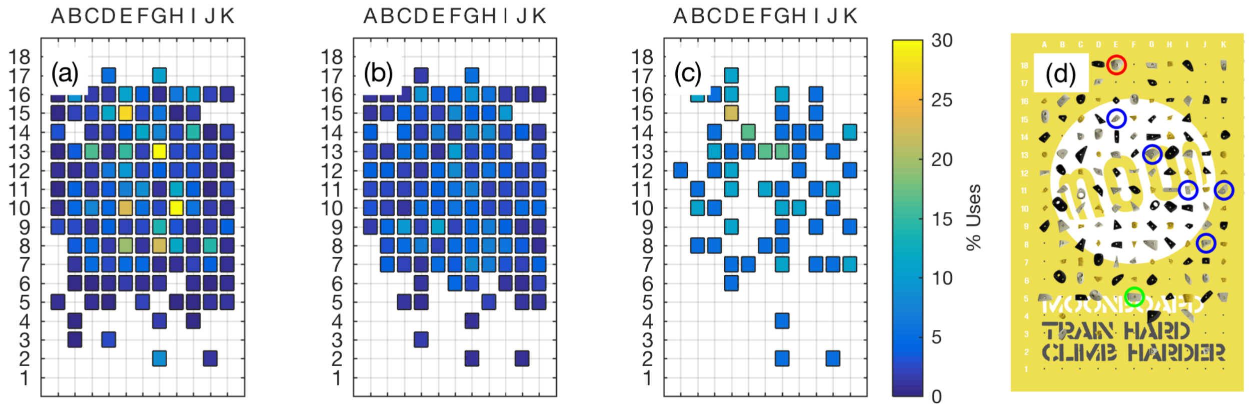

Figure 1: Frequency with which routes of different difficulties use certain holds **(a)** 6B+ (1) routes **(b)** 7C+ (9) routes and **(c)** 8B+ (13) routes. All panels use the same color scale. Only intermediate holds are shown for clarity, start and end holds are excluded. **(d)** Example problem from moonboard database. Green circles indicate start holds, blue circles indicate intermediate holds and red circles indicate end holds

For the Naive Bayes and softmax classifiers which were tested, the hold grid for each route was flattened into a vector, $x^{(i)} \in \{0,1\}^n$ and used as the feature space. For the CNN the hold grid was mapped to a matrix, $x^{(i)} \in \{0,1\}^{18\times 11}$. Graphical models were explored, where the different holds were connected with edges weighted by the physical distance between the holds, however none had a performance which improved upon the tested classifiers.

4 Methods

The performance of three classifiers was evaluated for this project: (1) Naive Bayes classifier, (2) softmax regression classifier and (3) convolutional neural network (CNN) classifier. A state vector machine (SVM) using several kernels was also implemented, however this method had lower validation set accuracy than the three classifiers explored here so it was not explored further.

4.1 Performance Baseline - Naive Bayes and Softmax Classifiers

A Naive Bayes classifier using a Bernoulli event model was implemented in order to set a baseline for performance. Its input was a flattened hold matrix with values in $\{0,1\}$, where a 1 represents the presence of the corresponding hold in the climbing route. An important observation is that the Naive Bayes assumption is an incorrect one in the context of climbing routes. The presence or absence of certain holds is not independent when conditioning on a climbing route's difficulty, climbing consists of a series of connected movements between holds.

A softmax regression classifier was also implemented. The features fed into this classifier were the flattened hold vectors $\in \{0,1\}^n$. Batch gradient descent was used to train the classifier. As discussed in detail in the next section, this classifier tended tended to over-predict the more common categories. The following weighting was explored to encourage it to pick less common categories:

$$w^{(i)} = a\frac{m}{\sum_{j=1}^{m} \mathbf{1}\{y^{(i)} = y^{(j)}\}} + 1 - a \tag{1}$$

This weighting function penalized the classifier more for mis-predicting less common categories. The value of a is an empirically determined parameter $\in [0,1]$. With the softmax classifier, all values of $a > 0$ led to the classifier over-fitting the less common categories meaning that this weighting scheme did not generalize well, and was ineffective at improving the classifier performance.

4.2 Convolutional Neural Network Classifier

In light of recent progress in image classification using convolutional neural networks (CNN) [4], a CNN was implemented to classify route difficulty. The implemented CNN is shown in Figure 2. The features for this network were the un-flattened hold matrices $\in \{0,1\}^{18\times 11}$. One convolutional layer with stride one was implemented having four filters of size 11x7 (2.2mx1.4m on the moonboard), and one filter of size one which fed hold information directly to the next layer. Convolution was only performed when filters were centered on a hold, and set to zero otherwise, so that the filters looked at the context of each hold. The subsequent hidden layer of size $5(18 \times 11)$ had a sigmoid activation function and flattened the information from the convolutional layer. The second fully connected hidden layer of size 50 also used sigmoid activation functions. The output layer used an ordinal regression as the activation function. With k categories, the hypothesis function $h(x^{(i)}) \in [0,1]^k$of the ordinal regression can be described as follows [8]:

$$h(x^{(i)})_j = p(y^{(i)} \leq j) - p(y^{(i)} \leq j-1) \tag{2}$$

This takes advantage of the ordering of the categories being fit, knowing, for example, that routes of grade 1,2,3,4 are all easier than a route of difficulty 5. The individual probabilities are calculated as follows:

$$p(y^{(i)} \leq j) = \begin{cases} 0, \ j=0 \\ s(b_j - w^T x^{(i)}), \ 0 < j < k \\ 1, \ j=k \end{cases} \tag{3}$$

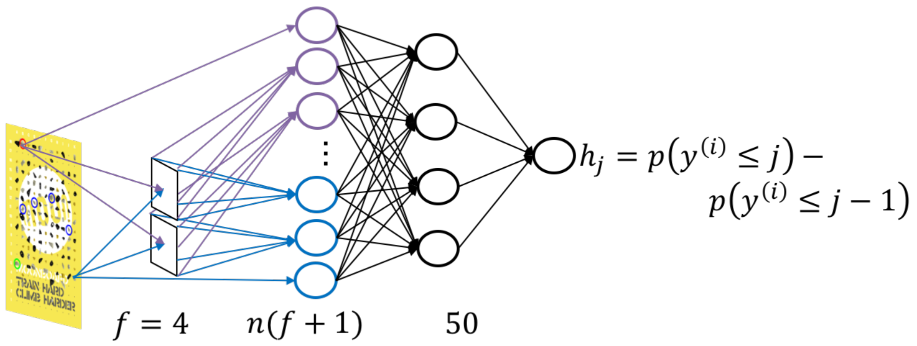

Figure 2: Diagram of implemented convolutional neural network. Example problem is shown on the left.

Where $s(z)$ is the sigmoid function. In addition to taking advantage of the ordering of the categories, the ordinal regression reduces the $(k-1)(n+1)$ parameters used by the softmax regression to $n+k-1$ parameters, by projecting the data from all categories onto a single w vector, as opposed to the $k-1$ different directions required by the softmax regression. This can also prevent over-fitting of categories with only a few examples, by restricting all categories to be projected on the same direction. The appropriate loss function to be minimized for this regression is:

$$L(w,b) = -\sum_{i=1}^{m} w^{(i)} \sum_{j=1}^{k} \mathbf{1}\{y^{(i)} \leq j\} \log p(y^{(i)} \leq j | x^{(i)}; w, b) \tag{4}$$

Where $w^{(i)}$ is the weighting assigned to each training example. The weighting function used for the CNN was the same as Equation 1 used for the softmax regression. A CNN with a softmax output neuron was also implemented in Python using the Keras library with cross-entropy loss, however its performance did not match that of the ordinal CNN shown here, so it was not explored further.

5 Experiments and Discussion

The performance of the three tested classifiers was evaluated using three metrics. Accuracy was evaluated based on the percent of samples from the validation set where the prediction matched the ground truth and the mean absolute error (MAE) between the predictions and the true values. The similarity between the distribution of predictions and the true distribution of difficulty grades is measured using the Kullback-Leibler (KL) divergence of the predictions relative to the true underlying distribution [5].

This performance is benchmarked against human performance which compares the user ratings to that of the setter. Human performance and the performance of the algorithms on the validation set are summarized in Table 1. The

Table 1: Validation Set Performance

	User Ratings	Naive Bayes	Softmax	CNN
Accuracy	93.4%	34.0%	36.5%	34.0%
MAE	0.12	1.73	1.37	1.40
KL-divergence	0.0071	0.41	0.078	0.020

performance of the three tested algorithms lags significantly behind human classification performance on all three metrics. It should, however, be noted that the human generated ratings were not generated merely by looking at the route, the users also climbed the routes to asses their difficulty. It is the author's opinion that even expert climbers would have significantly lower classification accuracy if they were constrained to only looking at the routes and not climbing them. The figure of 93% accuracy should thus be view not as a metric to beat, but rather a measure of the noise in the dataset. As seen in Table 1, all three tested algorithms have comparable accuracy. For all algorithms, training set and validation set accuracy agreed within a few percent, indicating that bias was a stronger limit on performance than variance. Significant differences can be seen in the KL divergence of the tested classifiers. This difference can be seen in the KL-divergence values in Table 1 and is reflected in Figure 5 which clearly shows that the distribution of the predictions of the CNN much more closely matches the underlying distribution, than do the predictions of the Naive Bayes and softmax classifiers. In fact, neither the softmax nor the Naive Bayes classifiers ever predicted that a route is more difficult than 8, due to the skewed nature of the dataset towards easier grades. Up-sampling the less common categories was not practical with this dataset because no new data could be acquired.

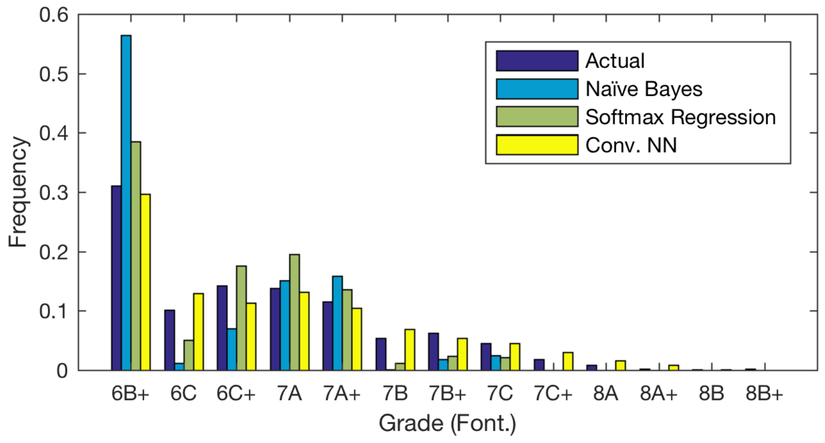

Figure 3: Frequency with which the different classifiers predict different categories, compared to underlying distribution. Data is from validation set.

The over-prediction of more common categories is inherent to the structure of the Naive Bayes and softmax classifiers. The Naive Bayes classifier does this because according to Baye's rule, $p(y^{(i)}|x^{(i)}) \propto p(y^{(i)})$, biasing it to predict more common categories. The softmax classifier over-predicts common categories because it's loss function weights all examples equally, so in the face of uncertainty it will have a lower loss by picking the most common category. When a different weighting function was used, where the softmax algorithm is incentivized to get the highest accuracy averaged across grades (setting $a > 0$ in equation 1), it over-fit the less common categories and did not generalize to the validation set. A CNN with an ordinal regression did, however, allow for some weighting, which reduced the KL divergence, without over-fitting the less common categories, due to the reduced parameter space of the output neuron. It was experimentally determined that setting $a = 0.125$ in equation 1 minimized the validation set KL divergence. This improvement can be seen in the confusion matrices plotted in Figure 5. Whereas the Naive Bayes and softmax classifiers tended to predict the easier categories more commonly (no matter the actual difficulty of a route, some of the time the Naive bayes classifier predicted it to have difficulty 1), never predicting above 8, the CNN has a more diagonal confusion matrix indicating that it achieved comparable accuracy to the other classifiers without "cheating" by over-predicting common categories.

The principle limitation to implementing a more complex CNN was the relatively small training set. The implemented CNN with four layers already had $\sim 50,000$ independent parameters (with $\sim 10,000$ training examples) and was difficult to get to converge while avoiding local minimums such as always predicting 1. Mini-batch stochastic gradient descent, with dropout, was used to improve convergence [7]. Training curves for the CNN are shown in Figure 5, demonstrating high noise. The difficulty in training was justified by the two fold advantages of the CNN. First of all, it took advantage of spatial data, analyzing each hold in its context. This mapped the feature space from $\{0, 1\} \to \mathbb{R}$ giving the output neuron better data on which to make a prediction. The non-linearities introduced by the hidden layers also allowed an ordinal regression to be used. Performance of this ordinal regression was poor when applied directly to the flattened hold-matrices, because route difficulty is not inherently linearly separable.

Once the parameters of the CNN were finalized, its performance on the test set was analyzed. On the test set, it achieved an accuracy of 31.8%, and MAE of 1.45, slightly worse than the validation set performance, with a KL divergence of 0.015, slightly better than validation set performance. The close agreement between training, validation and test set performance indicates that the CNN classifier generalizes well.

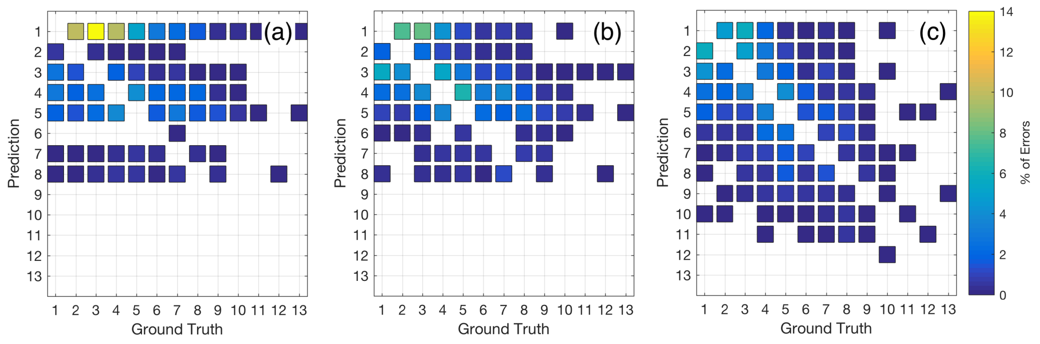

Figure 4: Confusion matrices for **(a)** Naive Bayes classifier **(b)** Softmax regression classifier and **(c)** Convolutional neural network classifier. All panels use the same colorbar which represents the percent of errors attributed to that ground/truth prediction combination. Correct predictions are excluded from figure for clarity.

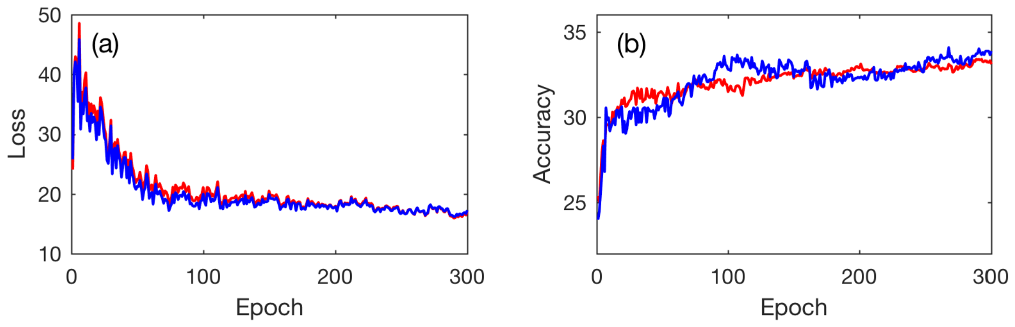

Figure 5: Training curve for CNN. Learning rate was 0.06. Red curves represent training set, blue curves represent validation set. Best performance is found at epoch 291.**(a)** Loss function vs. epoch **(b)** accuracy vs. epoch

6 Conclusions/Future Work

This work analyzed the performance of three different algorithms for classifying the difficulty of climbing routes, a Naive Bayes, softmax and CNN classifier. All tested algorithms had comparable accuracy and mean absolute error, though the CNN classifier with an ordinal output was able to achieve comparable accuracy without over-predicting common categories, resulting in a lower KL divergence relative to the underlying distribution. Though further data mining is infeasible, several other directions can be pursued to improve performance. One option would be to get experienced climbers to rate how difficult it is to use each hold on the board, and incorporating that data into the algorithm, to increase the feature space and improve performance. Alternative cost functions could also be explored to improve on current algorithm performance, balancing over-predicting common categories and over fitting the less common categories. Finally, generative adversarial networks can be implemented to create a route generator, which can improve the moonboard user's experience by expanding the route variety available to them [1].

Contributions

Peter Satterthwaite wrote the web-scraper, co-wrote Softmax classifier, and plotted results. Alejandro Dobles wrote Naive-Bayes classifier. Juan Carlos Sarmiento co-wrote Softmax classifier. Ordinal CNN was implemented by Peter Satterthwaite based on starter code for CS 229 Problem Set 4, Problem 1, Keras CNN was implemented by Juan Carlos Sarmiento and Alejandro Dobles. All authors contributed to tuning CNN. All authors contributed to writing this report.

References

[1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. *Advances in Neural Information Processing Systems 27*, pages 2672–2680, 2014.

[2] Pedro Gutierrez, Maria Perez-Ortiz, Javier Sanchez-Monedero, Francisco Fernandez-Navarro, and Cesar Hervas-Martinez. Ordinal regression methods: survey and experimental study. *IEEE Transactions on Knowledge and Data Engineering*, 4347(c):1–1, 2015.

[3] Felix Kosmalla, Florian Daiber, and Antonio Krüger. ClimbSense: Automatic Climbing Route Recognition using Wrist-worn Inertia Measurement Units. *Proceedings of the ACM CHI'15 Conference on Human Factors in Computing Systems*, 1:2033–2042, 2015.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances In Neural Information Processing Systems*, pages 1–9, 2012.

[5] Leibler Richard A. Kullback, Solomon. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 12 1951.

[6] C. Phillips, L. Becker, and E. Bradley. Strange Beta : An assistance system for indoor rock climbing route setting. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 22(1):013130, 2012.

[7] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[8] Christopher Winship and Robert D Mare. Regression Models with Ordinal Variables. *American Sociological Review*, 49(4):512–525, 1984.