Six dubious ways to estimate the difficulty of a chess puzzle

Jan Zyśko
Motivation: chess training process

1. Play a tournament
2. Find all the mistakes in the games
3. Prepare training materials and exercises
4. Train on the materials and exercises
5. Repeat

How to perform point 3 without a coach?

1) Training materials - game diagnostics
2) Exercises: find puzzles with positions similar to the one analyzed in character and difficulty

our task here!

Mark Dvoretsky, the most famous chess coach in the world
Anatomy of a lichess puzzle

Exercise rating (based on how many people previously solved it correctly)

Player rating (based on her previous performance)

The better the player is doing, the more difficult tasks they are assigned.
Lichess puzzle dataset

Total: over 1.9 million puzzles
Benchmarks

Median prediction

Results:
mean: 361
std: 256

Expert knowledge

Results:
mean: 278
std: 227
Method 1: Use activations from an existing model (transfer learning)

- Value head
- Policy head

Resnet
- 20 layers

Resnet

Input (present and previous states)

The value head
- tanh non-linearity
- Fully connected layer
- Rectifier non-linearity
- Fully connected layer
- Rectifier non-linearity
- Rectifier non-linearity
- Batch normalization
- 1 convolutional filter (3x3)
- Input

The policy head
- Fully connected layer
- Rectifier non-linearity
- Batch normalization
- 2 convolutional filters (1x1)
- Input

19 x 19 + 1 (for pass) move logit probabilities
Method 1: Use activations from an existing model

Best layer errors:
mean: 331
std: 243
Method 1: Use activations from an existing model

Predicted ratings rise (on average) as the variance is big and the rise

Observations:
- Predicted ratings rise (on average) as the variance is big and the rise
Method 2: Use all the activations, averaged by channel

8x8x112 -> 8x8x1
concatenated into 8x8x21

Results:
mean: 338
std: 246
Method 3: Use policy layer

Results:
mean: 350
std: 249
Method 4: Policy layer entropy

Results:
- mean: 352
- std: 251
- Pearson correlation: -0.08
Method 5: Relearning weights for the rating prediction task

Results:
mean: 527
std: 396
Method 6: Successful engine runs

Results:
mean: 275
std: 200
Method 6: Successful engine runs

% of correct solutions

rating_intervals

(500, 600] (1000, 1100] (1500, 1600] (2000, 2100] (2500, 2600]
## Summary of the results

<table>
<thead>
<tr>
<th>models</th>
<th>mean error</th>
<th>error std</th>
<th>train size</th>
<th>test size</th>
<th>model type</th>
</tr>
</thead>
<tbody>
<tr>
<td>engine ensembling</td>
<td>275</td>
<td>200</td>
<td>35000</td>
<td>5000</td>
<td>XGBoost</td>
</tr>
<tr>
<td>transfer learning from activations</td>
<td>331</td>
<td>243</td>
<td>60000</td>
<td>20000</td>
<td>XGBoost</td>
</tr>
<tr>
<td>ensembling activations from different layers</td>
<td>338</td>
<td>246</td>
<td>60000</td>
<td>20000</td>
<td>XGBoost</td>
</tr>
<tr>
<td>relearning weights for prediction task</td>
<td>527</td>
<td>396</td>
<td>75000</td>
<td>5000</td>
<td>DNN</td>
</tr>
<tr>
<td>transfer learning from policy layer</td>
<td>350</td>
<td>249</td>
<td>60000</td>
<td>20000</td>
<td>XGBoost</td>
</tr>
<tr>
<td>policy entropy</td>
<td>352</td>
<td>251</td>
<td>90000</td>
<td>10000</td>
<td>Linear Regression</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>benchmarks</th>
<th>mean</th>
<th>error std</th>
<th>train size</th>
<th>test size</th>
<th>model type</th>
</tr>
</thead>
<tbody>
<tr>
<td>median benchmark</td>
<td>361</td>
<td>256</td>
<td>0</td>
<td>100000</td>
<td>mean</td>
</tr>
<tr>
<td>aimchess</td>
<td>278</td>
<td>227</td>
<td>0</td>
<td>1000</td>
<td>ensemble model</td>
</tr>
</tbody>
</table>
Future work

- Using more data and bigger models
- Ensembling more engines
- Analyzing MCTS tree:
  - evolution of the predicted best move
  - distribution of Q in the tree