Learning Chess from Data

Tom Ron
Niv Mizrahi

https://github.com/nivm/learningchess

Everyone wants to make computer play Chess smarter..

We just want to make computer play chess

What's on our mind?

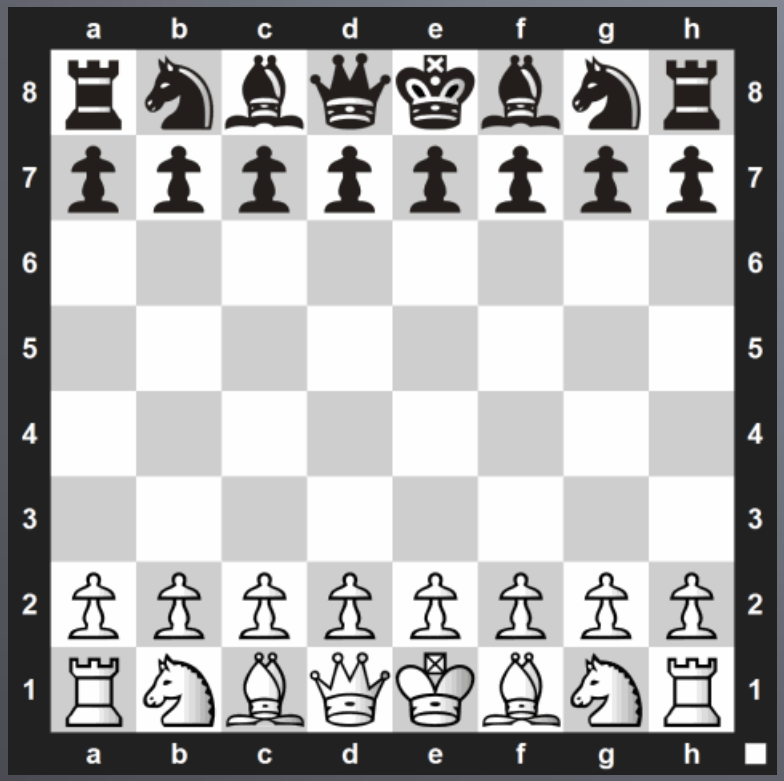
- Can a computer learn Chess just by looking on Chess games?
- Game on - given a board state, is a specific move legal?
- Game over - given a board state, is it Checkmate?
- If those are possible, what else can we learn empirically?
- Work on progress

Undressing chess

- 2 parties
- Game end – one winner or tie
- 8 X 8 board
- Different pieces have different unknown properties (moving, eating other pieces, promotion, getting eaten..)

Dataset

- Data based on



Dataset

- Ignoring meta data (players, ranking, location, etc.)
- 103925 games – full or partial description
- 8,302610 moves -

Pawn	22001860	Rook	1352388
Bishop	1290920	Knight	1462645
King	996128	Queen	1003169

Stack

- Chess - parsing chess algebric notation data.
- PyLab – Numpy, Scipy, Matplotlib

- Used Map-Reduce architecture but the data is small enough to process all on single machine

Simple move

Have we seen that move – board status + move before?

Yes – Good, No – Try again

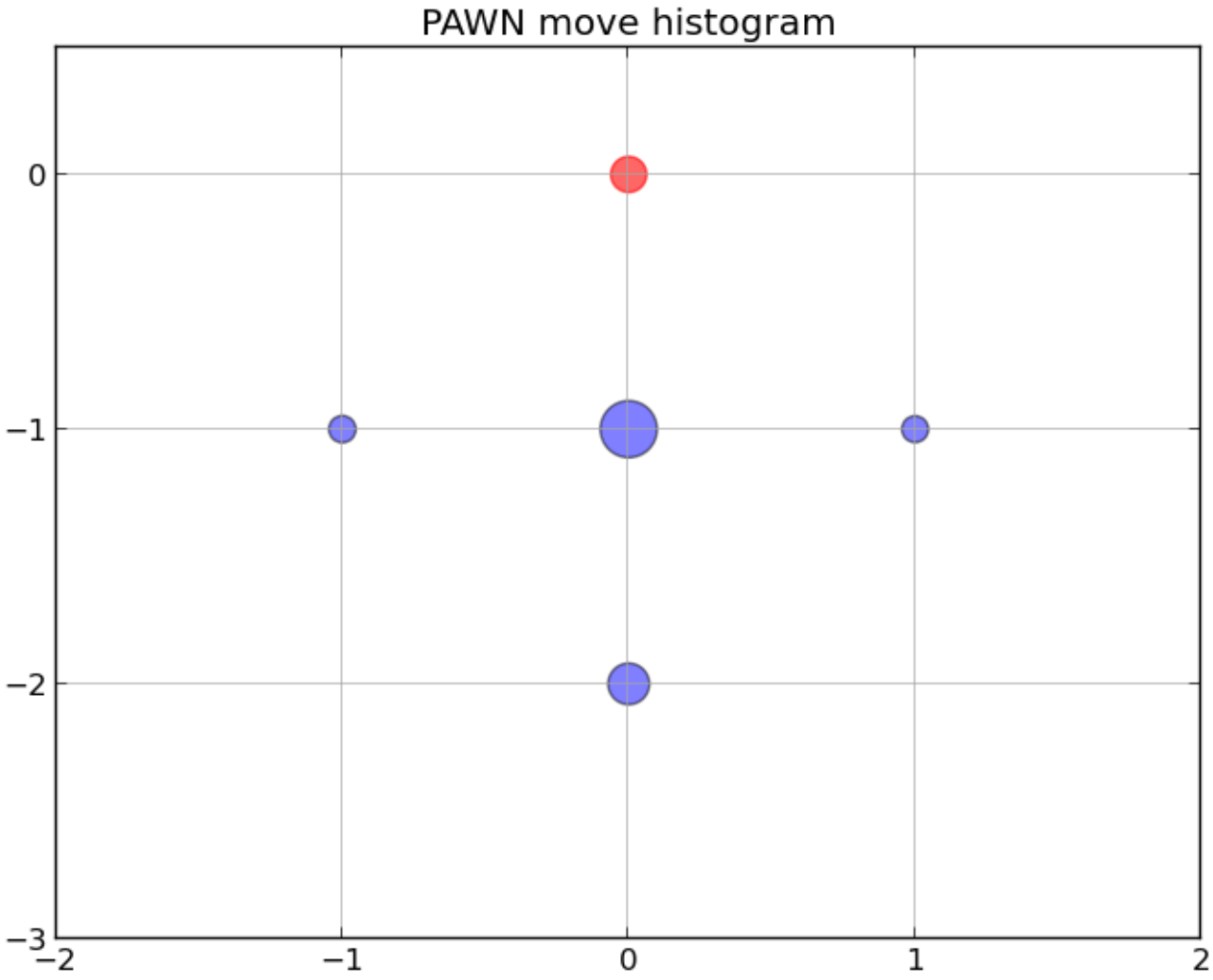
Consumes a lot of memory and running time – 10^80 estimated chess moves

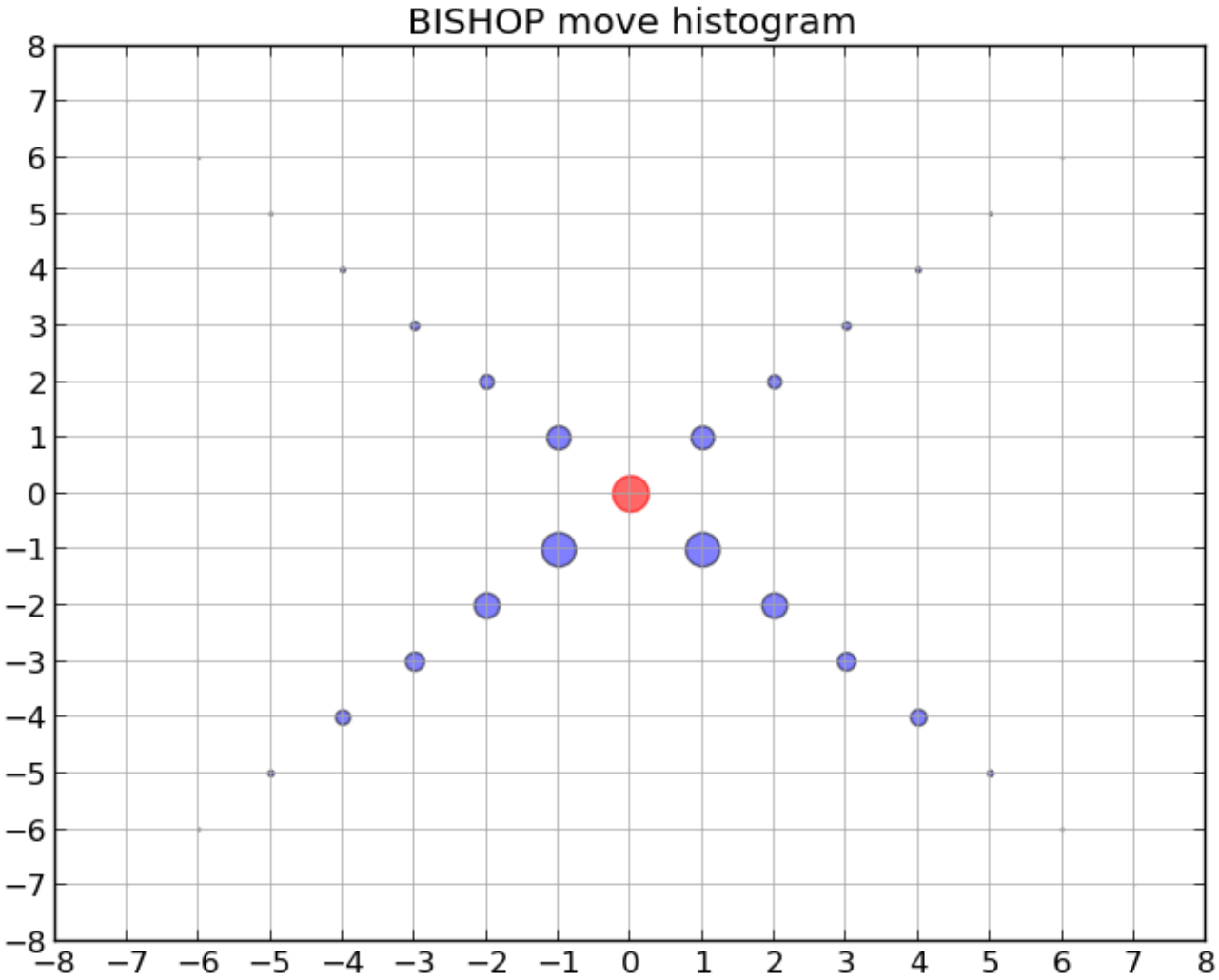
Simple move 1.0

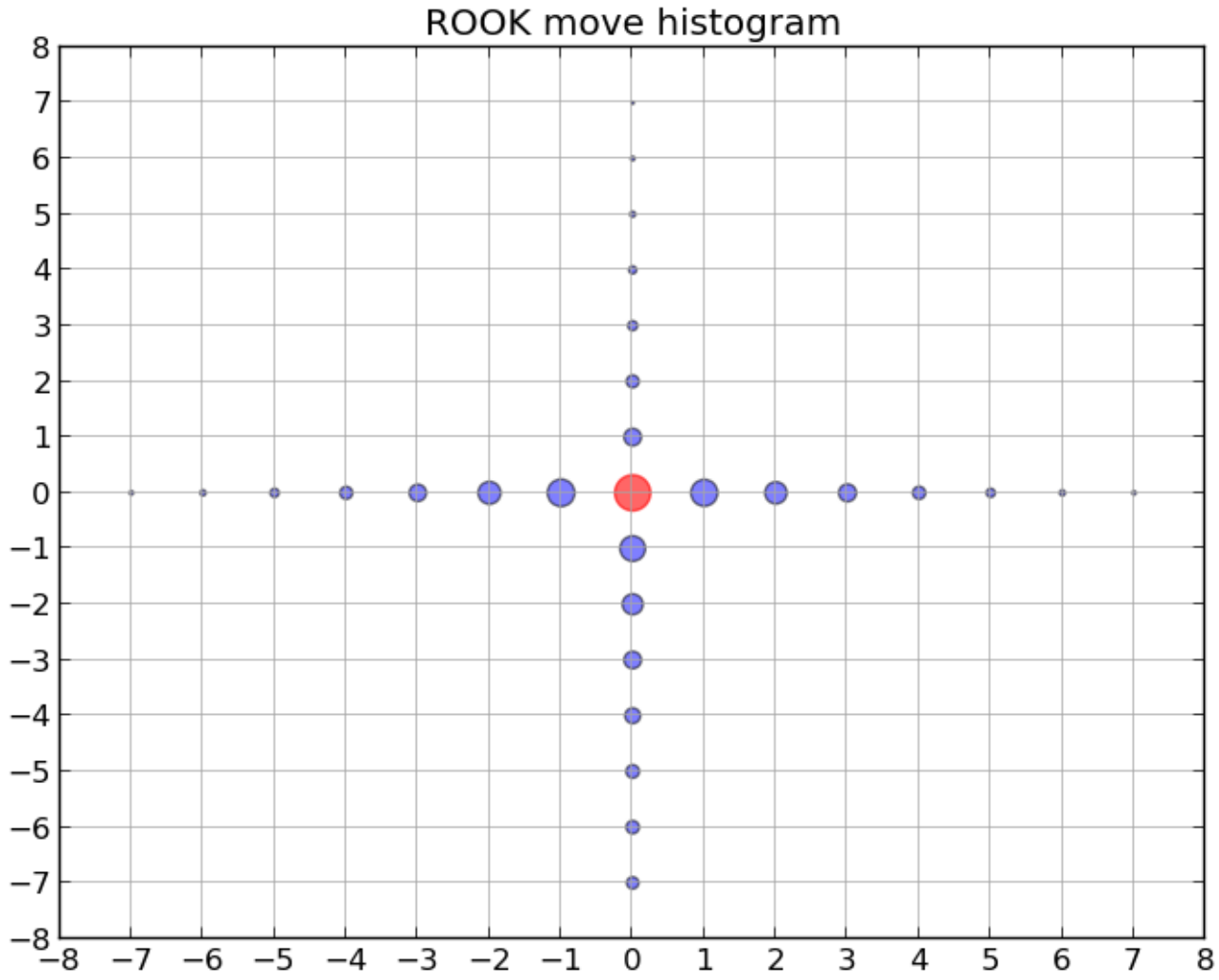
For each move output move x, y difference. Aggregate all the moves of specific piece.

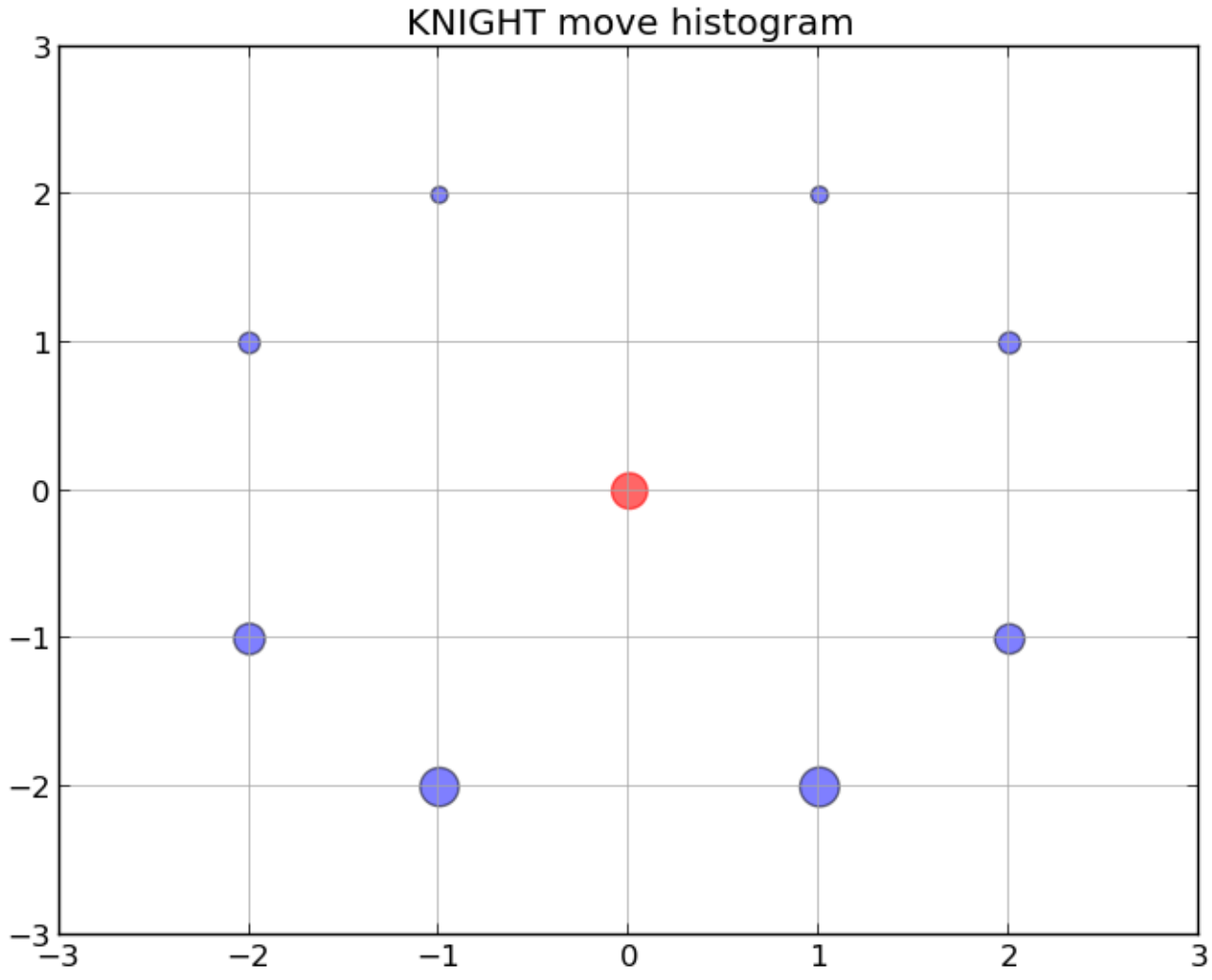
White Pawn moves from b2 to b4 -
X diff – 0, Y diff – 2

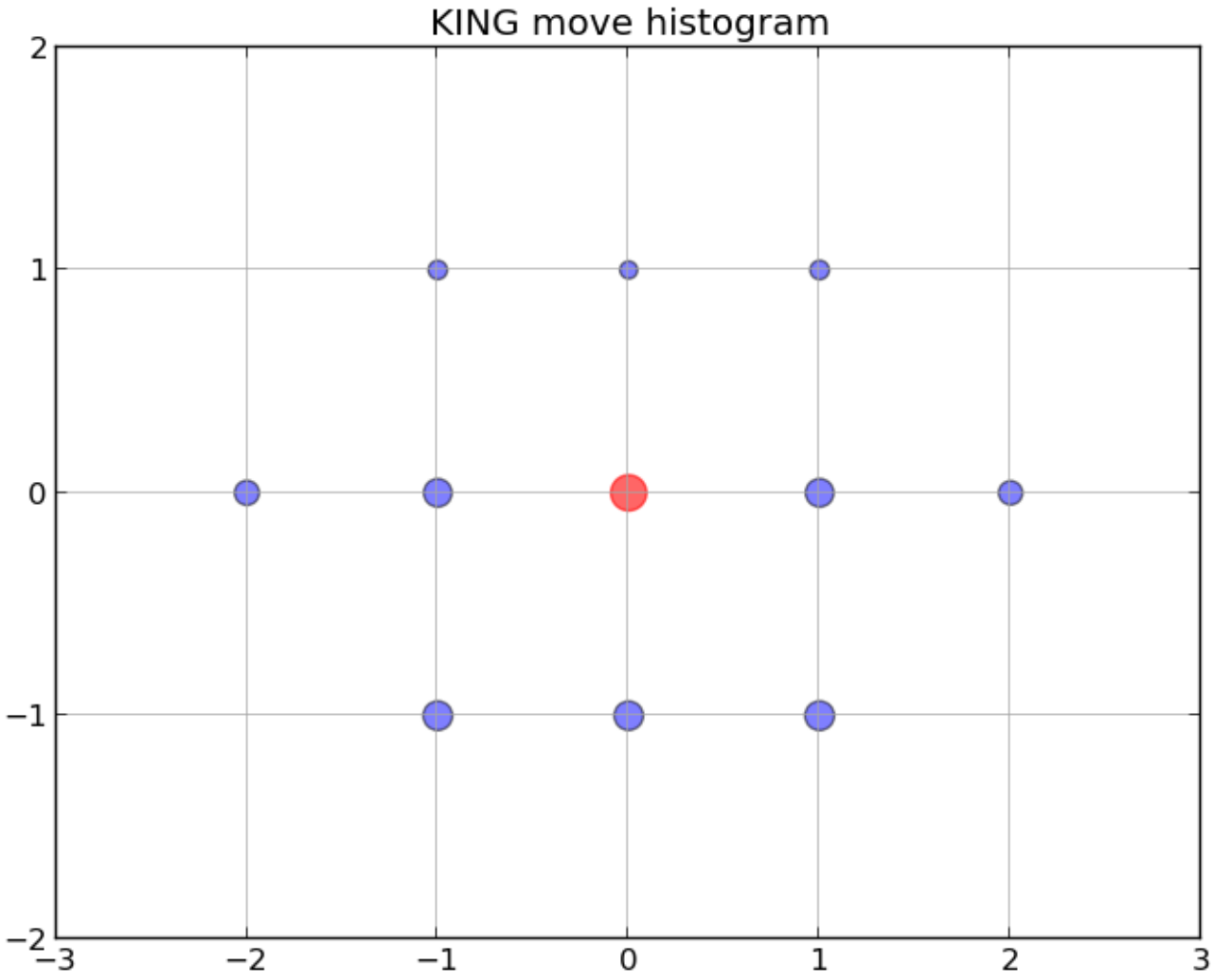
Relative black and white adjustments.


PAWN move histogram
0
−1
−2
−3
−2
−1
0
1
2


BISHOP move histogram
8
7
6
5
4
3
2
1
0
−1
−2
−3
−4
−5
−6
−7
−8
−8 −7 −6 −5 −4 −3 −2 −1 0 1 2 3 4 5 6 7 8


ROOK move histogram
8
7
6
5
4
3
2
1
0
−1
−2
−3
−4
−5
−6
−7
−8
−8 −7 −6 −5 −4 −3 −2 −1 0 1 2 3 4 5 6 7 8


KNIGHT move histogram
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3


KING move histogram
2
1
0
−1
−2
−3
−2
−1
0
1
2
3

Simple move 1.0

Pros –

- Good for common moves.
- Getting better as data size grows (diminishing return)
- Time and memory efficiency

Simple move 1.0

Cons –

- Does not take into account board status.
- Not good for rare move

Conclusion - almost always necessary condition but not sufficient

Simple move 2.0

For each move output move diff + surround histogram of radius 1

Options – occupied, free, out of the board

0	1	2
3	B,R,K,N,Q,P	5
6	7	8

Simple move 2.0

Piece	Move diff	Relative location	Content
Queen	(-2, -2), (-3, -3), (-4,-4), (-5, -5), (-6,-6), (-7, -7)	0	Free

Free	1	2
3	**Q**	5
6	7	8

Simple move 2.0

Piece	Move diff	Relative location	Content
Queen	(-7,7)	2 6	Out of the board Free

0	1	**Out of the board**
3	**Q**	5
Free	7	8

Simple move 2.0

Piece	Move diff	Relative location	Content
King	(2,0)	5	Free

0	1	2
3	**K**	**Free**
6	7	8

Simple move 2.0

Piece	Move diff	Relative location	Content
Pawn	(0,2), (0,1)	7	Free

0	1	2
3	**P**	5
6	**Free**	8

Simple move 2.0

Piece	Move diff	Relative location	Content
Knight			

0	1	2
3	**N**	5
6	7	8

Simple move 2.0

Pros -

- Efficiency
- Takes surrounding into account

Cons –

- Assume different moves are independent of one another \ history.
- Less generalized.

Learning checkmate

Checkmate – game over

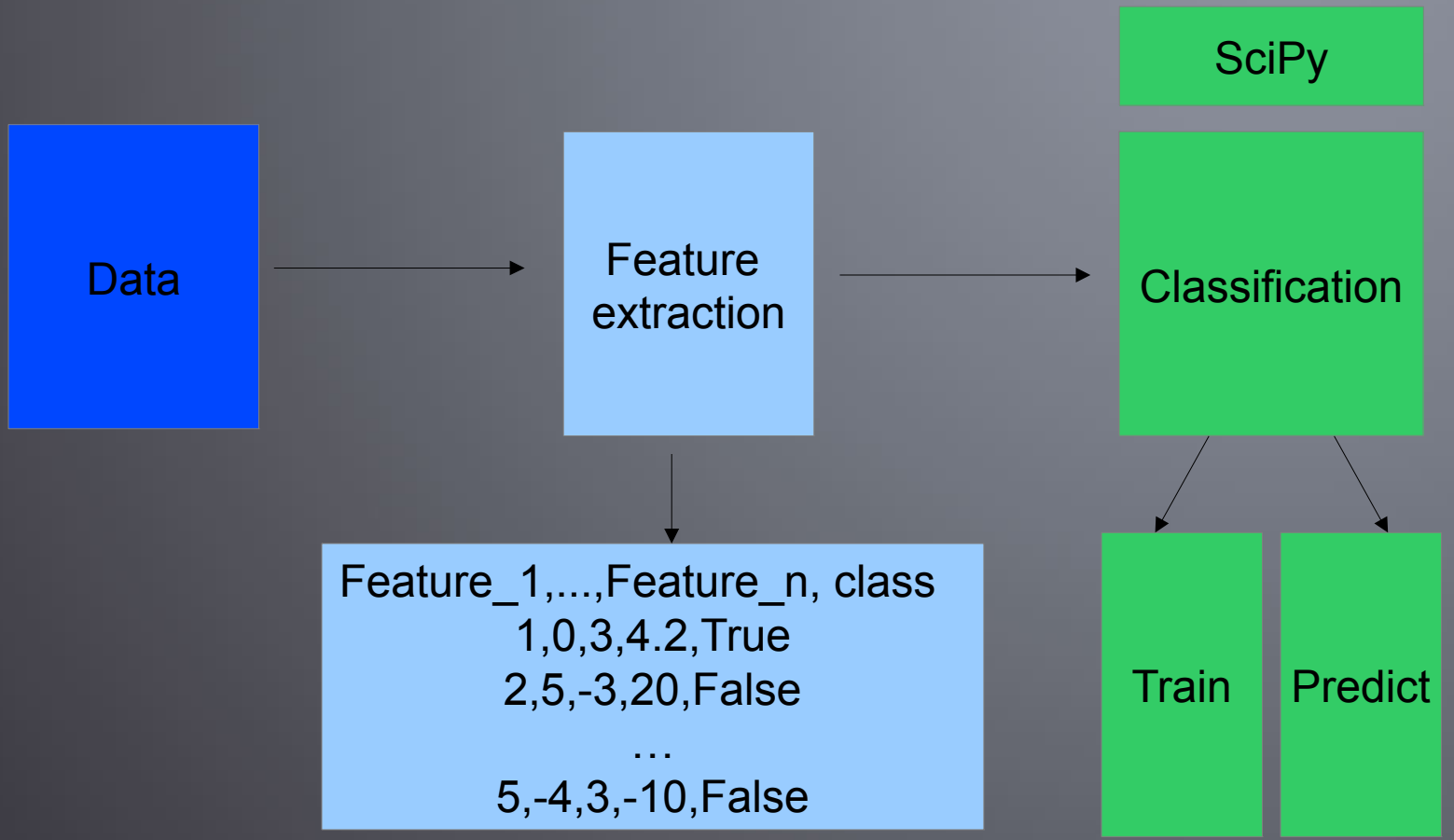
Datasets – 10k, 30k, 800k

Training set – 40%, test set – 60%

50-50 True-False samples

Classifier – SVM with linear kernel

Classification crash course



Features

First try – simple count features

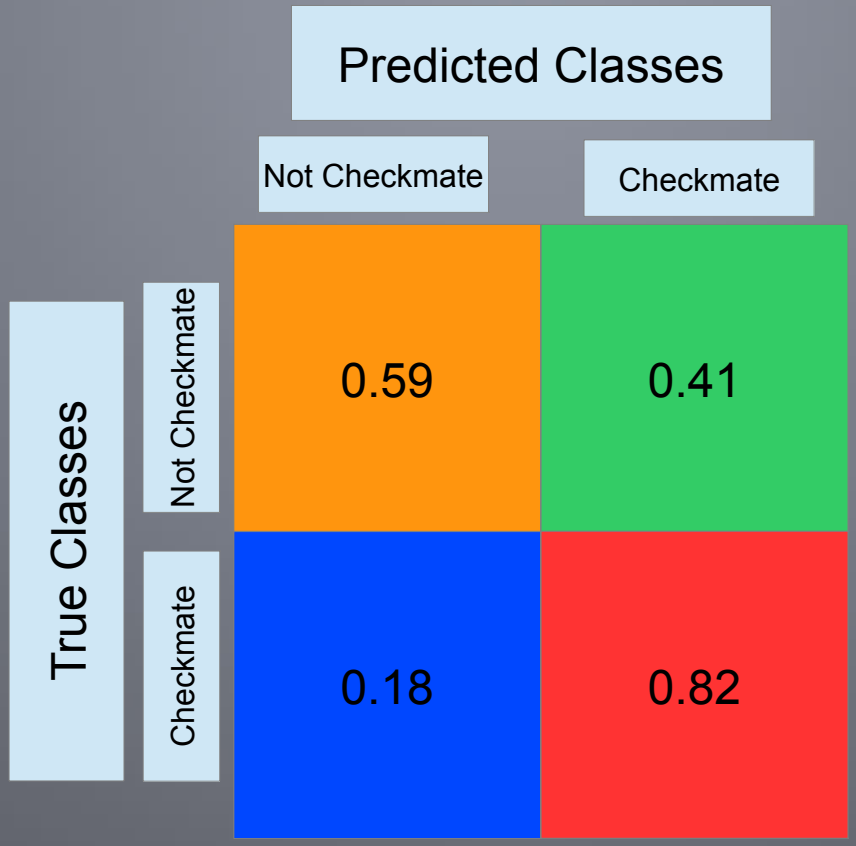
- Total - # pieces on the board
- White - # white pieces on the board
- Black - # black pieces on the board
- The total amount of pieces of each type
- The total amount of pieces of each type of each side

Results

Correct classification -
- Checkmate – 0.82
- Not checkmate – 0.59

Wrong classification -
- Checkmate – 0.18
- Not checkmate – 0.41

Accuracy – 0.705



Features

Second try – previous features feature + First degree neighbors

- # empty \ same \ other pieces around each piece, each side

Out of the board is not counted

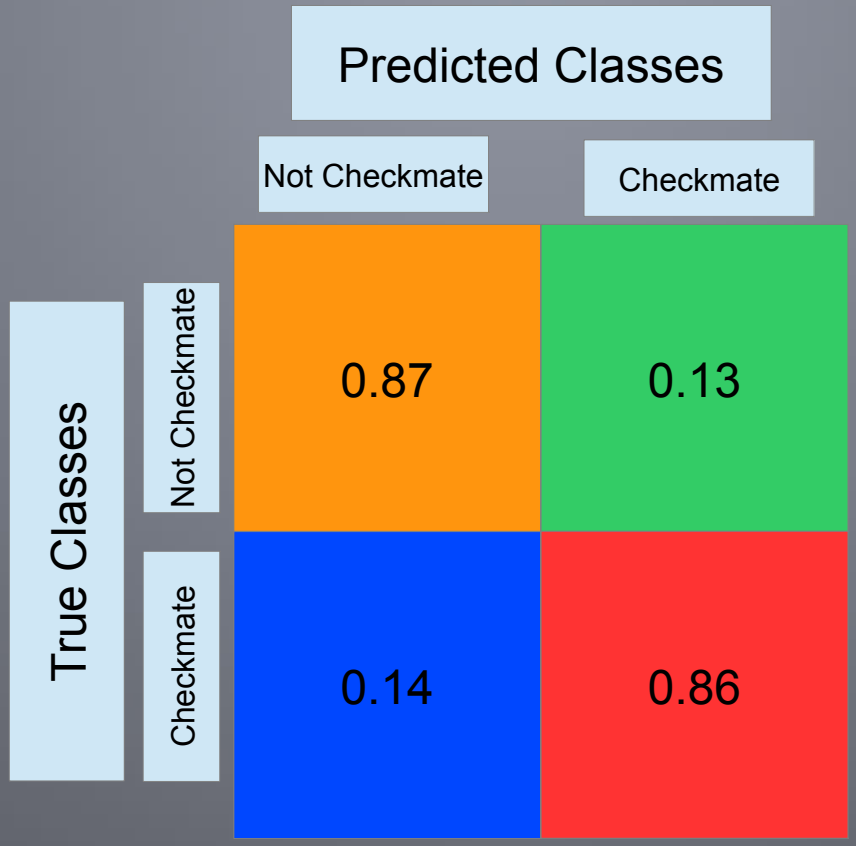
- Boolean features -
- Single side > other side, mostly empty, ..

Results

Correct classification -
- Checkmate – 0.86
- Not checkmate – 0.87

Wrong classification -
- Checkmate – 0.14
- Not checkmate – 0.13

Accuracy – 0.865



Features

Third try – previous features feature + Second + Third degree neighbors

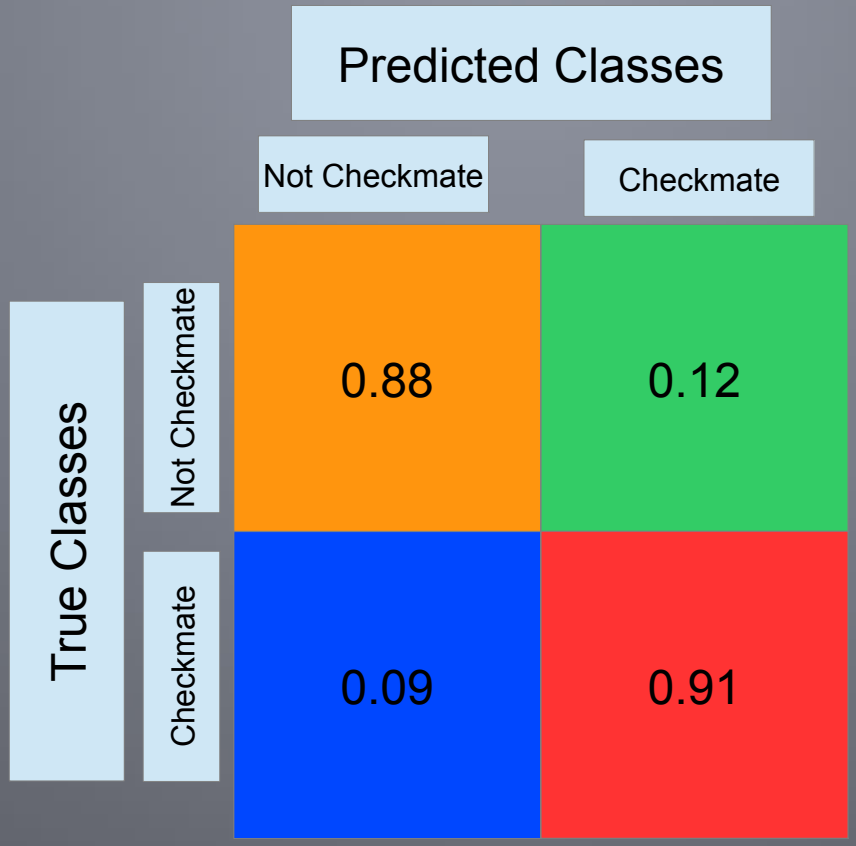
300~ features

Results

Correct classification -
- Checkmate – 0.91
- Not checkmate – 0.88

Wrong classification -
- Checkmate – 0.09
- Not checkmate – 0.12

Accuracy – 0.895



Future

- Test different classifiers (deep learning)
- Integrate out of the board into the count
- Winner – multi class classification – white won, black won, no checkmate
- Chess classifier
- Complex move detection – history arguments
- More efficient parsing
- Scaling

Q & A

Thank you!

http://cheezburger.com/8067088640