**Key**

To Do

**Datasets**

* Regular season games: 2014-2015 to 2017-2018
  + Need to get 2018-2019 and 2019-2020 games and format them for use in our classifier (teams, dates, results)
* Playoff matchups + results: 2015 - 2019

**Results**

* Linear Classifier
  + Playoffs
    - Run 1: No validation set, games in sequential order
      * Parameters
        + Train: Regular season games
        + Test: Playoff matchups (fill out the brackets, etc.)
        + Step size: 1E-3
        + Convergence Criteria: 1E-4
      * Results (Playoff accuracy)

|  |  |
| --- | --- |
| **Year** | **Test Accuracy** |
| 2014-2015 | 53% |
| 2015-2016 | 53% |
| 2016-2017 | 87% |
| 2017-2018 | 67% |

* + - Run 2: Add a validation set, randomly shuffle the games
      * Parameters
        + Train: 80% Regular season games
        + Validation: 20% Regular season games
        + Test: Playoff matchups (not shuffled)
        + Step size: 1E-3
        + Convergence Criteria: 1E-4
      * Results (Dev accuracy, playoff accuracy)

|  |  |  |
| --- | --- | --- |
| **Year** | **Dev Accuracy** | **Test Accuracy** |
| 2014-2015 | 54.3% | 53.3% |
| 2015-2016 | 60% | 80% |
| 2016-2017 | 56.4% | 60% |
| 2017-2018 | 59.2% | 73.3% |

* Parameters
  + Max values of theta:
* General Game Prediction
  + - * + Run 1: Randomly shuffled games across all seasons

Parameters

Train: 80% regular season games

Validation: 10% regular season games

Test: 10% regular season games

Step size: 1E-3

Convergence Criteria: 1E-4

Results

Dev accuracy: 66.8%

Test accuracy: 61.5%

Notes: converged in < 500 iterations

* + - * + Run 2: stricter convergence criteria

Parameters:

Train: 80% regular season games

Validation: 10% regular season games

Test: 10% regular season games

Step size: 1E-3

Convergence Criteria: 1E-5

Results

Dev accuracy: 61.5%

Test accuracy: 64.4%

* + - * Notes: different runs seems to give different results for the playoffs. Not a very stable ‘prediction’.
    - Quadratic Classifier
      * Running it on 2014-2015 data with mini-batch.
        + Step-size: 1E-5, eps: 1E-4. Error is fluctuating and not converging. Batch size of 32.
        + Batch size of 64 (and same parameters) also did not work.
        + Batch size of 128 also does not seem to work. Hmm…
        + Batch size of 128 with step-size: 1E-6 – does not converge.
        + Batch size of 128, step-size: 1E-4 – also not converging
        + Changing norm in project function to 2-norm (default is frobenius, which is not necessarily what we want?). Batch size of 128, step-size of 1E-5. Does not converge.

Error starts at 1.1E-3 and fluctuates around 1.1 – 1.4E-3.

* + - * + Re-test Bai’s (almost) conditions: first 50 data points, step-size: 1E-5, eps: 1E-4, 2-norm in project function, w/o mini-batch.

Error starts at 9E-4 and decreases. Seems to be converging.

From Bai’s graph – about 360 iterations for convergence.

* + - * + Randomly shuffle first 50 data points, step-size: 1E-5, eps: 1E-4, 2-norm in project function, w/o mini-batch. Should also converge…and it does! Or has started to decrease at least.
        + Randomly shuffle first 50 data points, step-size: 1E-5, eps: 1E-4, 2-norm in project function, w/mini-batch (32).

Error starts at 6.8E-4 and it seems to be converging slowly.

Final number of iterations:

So for 50 data points, batch gradient descent seems to be better. Mini-batch seems to be slower. What happens if the data size is larger though?

* + - * + Randomly shuffle first 50 data points, step-size: 1E-4, eps: 1E-4, 2-norm in project function, w/mini-batch (32)

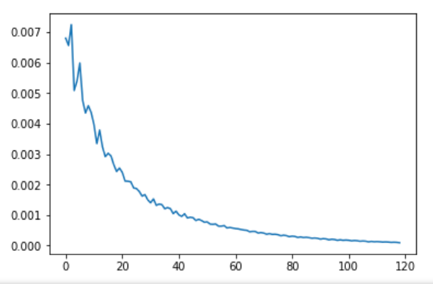
Seems to be going faster but error starts at 6E-3 now.

* + - * + Running the following experiments with max\_iter = 150, randomly shuffle first 50 data points, eps: 1E-4, mini-batch of 32

Step\_size: 1E-5

Error 1: 6.8E-4

Error 90: roughly 4E-4

Step\_size: 1E-4

Error 1: 6.8E-3

Error 30: 1.4E-3

Error 54: 7E-4

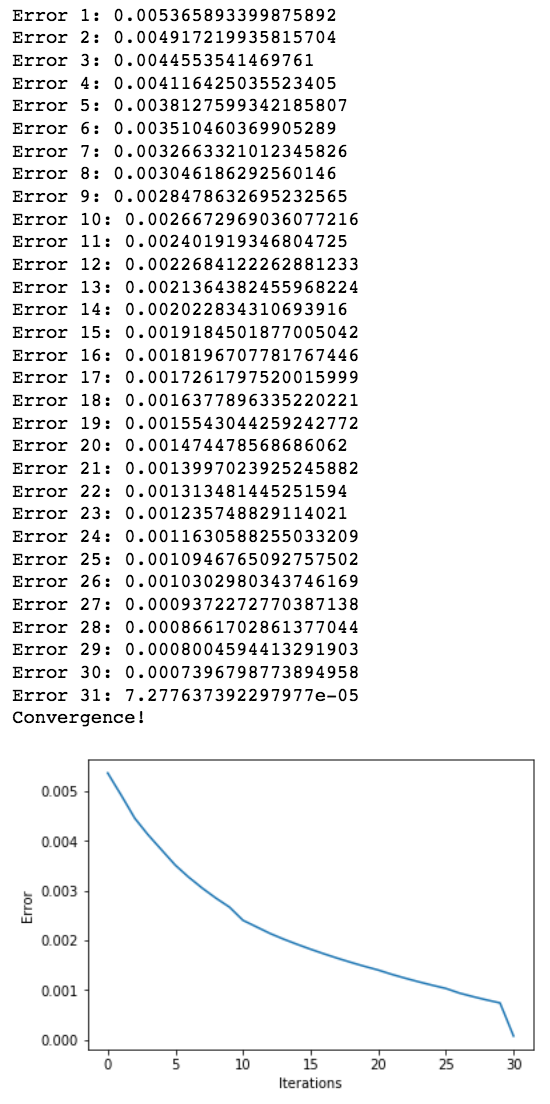
Error 80: 3E-4

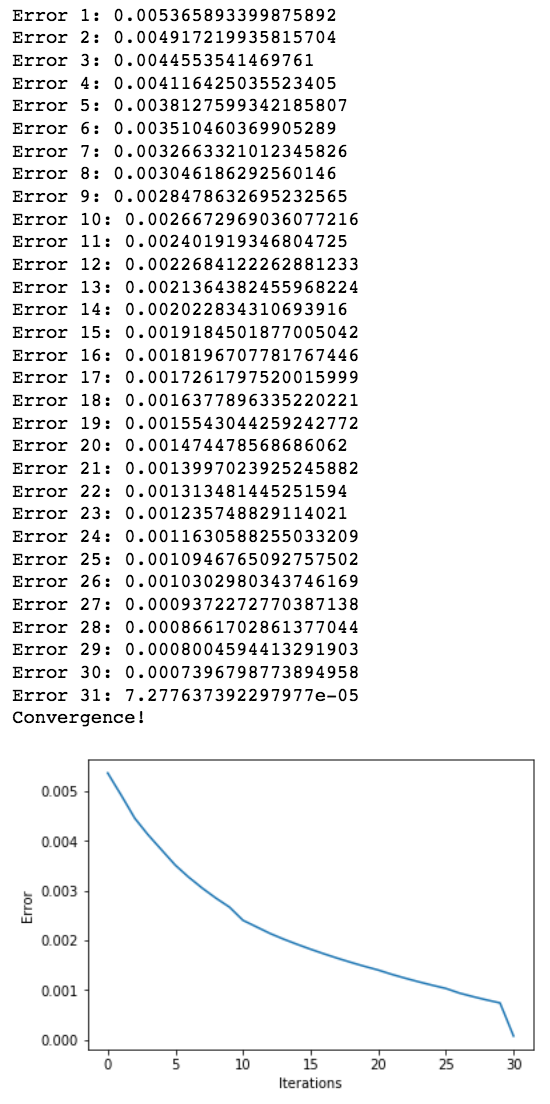
Error 90: 2.3E-4

Converged at Error 119!!!

* Transition to larger data set (50 to 100)
* Step\_size: 1E-4 (seems to be decreasing)
  + Error 1: 5.8E-3
  + Error 30: 3.5E-3
* Step\_size: 1E-3
  + Error 1: 5.8E-2
  + Started cycling between points so stopped it.
* Corrected the equation for grad, vectorized it with pytorch (omg)
* Wrote learning\_rate\_tune function
* Running learning\_rate\_tune on individual seasons
  + Train/dev split: 80%, 20%
  + Test set: Play-off matches
  + 2014-2015 season
    - 1e-3 blows up
    - 1e-4 – error doesn’t go below 0.016 (max\_iters)
* 2015-2016 season
* 2016-2017 season
* 2017-2018 season
* Running learning\_rate\_tune on all seasons
* Train/dev split: 80%, 20%

Mini-batch (32), 2014-2015 season, 1E-4 Convergence – dev\_accuracy: 53%





Tested few different sizes + convergences. Mini-batch size of 64 seems to do slightly better? 1E-4 convergence. Dev\_accuracy of 58%.

Seeing similar accuracies on quadratic\_classifier so probably just going to use the torch version.

Question is whether to use mini-batch or not…Probably do both eventually. Maybe do full set first with lower max\_iters?

Full set (no mini-batch), first 100 data points, step\_size: 1e-4, 25 iterations

No pytorch – 79% training accuracy,

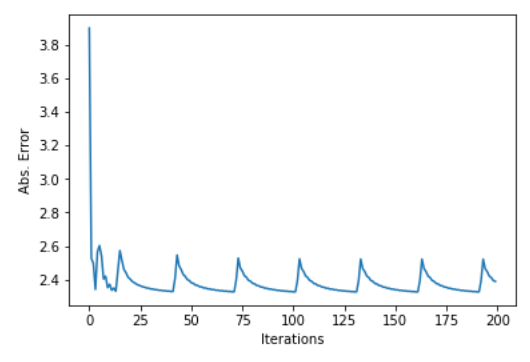
Pytorch – 65.7% training accuracy, 48% dev accuracy (LOL)

FIXED LOSS FUNCTION -- MAJOR CHANGE

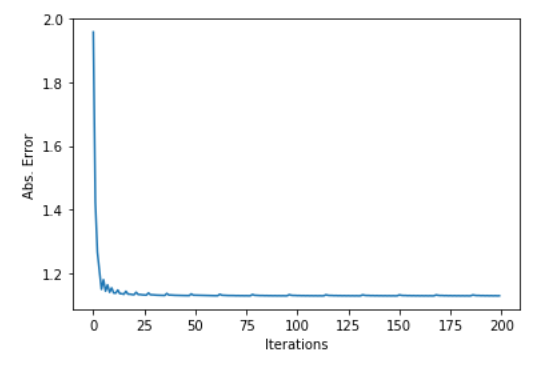
**Final Experiments**

Quadratic Classifier (Pytorch)

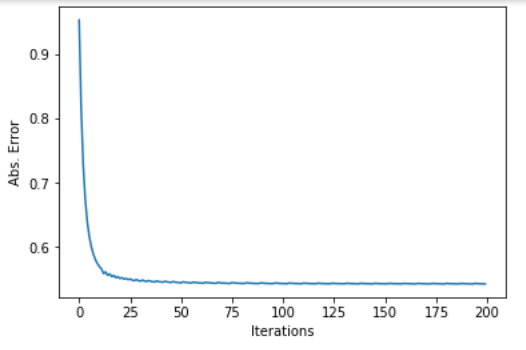
* Full dataset (Train: 3935, Test: 984), convergence criteria = 1E-4
  + lr = 1E-3, 1E-4, 5E-5, 1E-5, 5E-6 – do not converge
  + lr = 2E-6



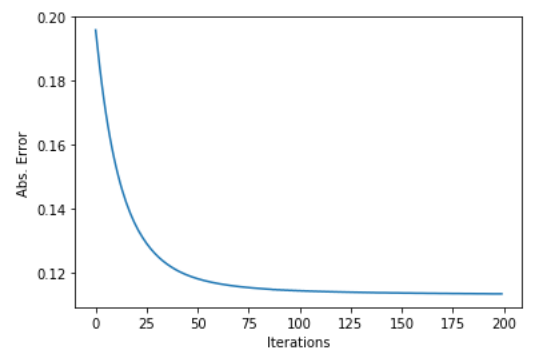
* + lr = 1E-6



* lr = 5E-7

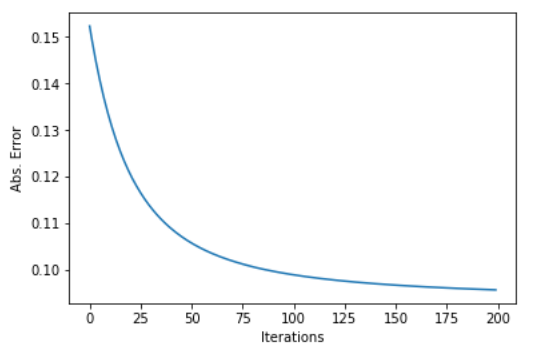


* lr = 1E-7

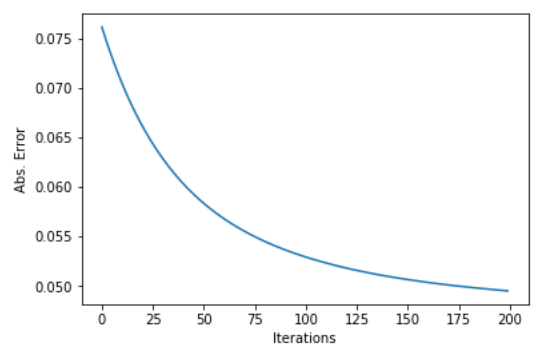


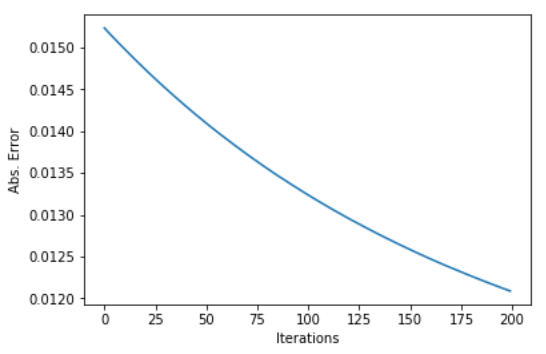


* Individual Seasons
* Learning the lr with 2014-2015 season dataset
  + lr = 1E-6

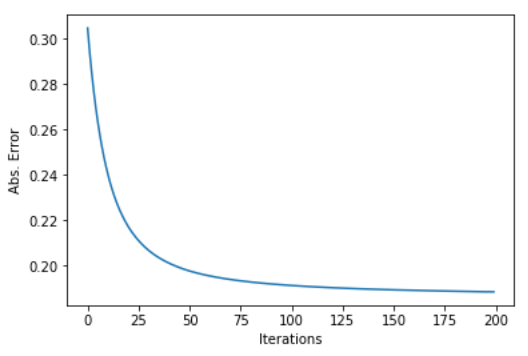




* lr = 5E-7
* lr = 1E-7 (probably needs to be run longer…)



* lr = 2E-6



* lr = 4E-6

