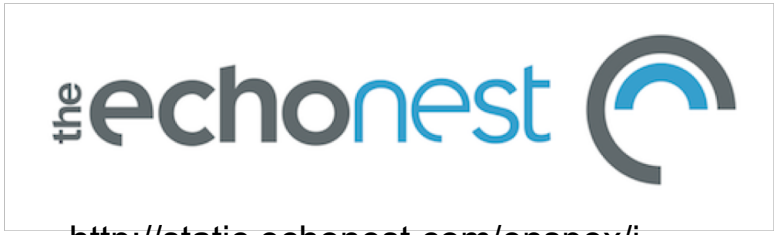
Music Decade Prediction

Shahriar Hooshmand, Shiny Patel, Joel Pepper

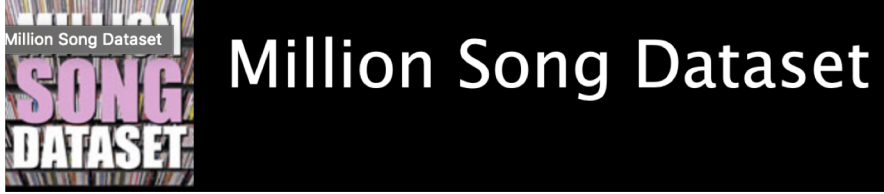
Background

- Music data mining becoming increasingly more prevalent due to companies such as Spotify and Shazam
- Waveforms can be segmented into numerous features such as pitch, tempo and loudness, as well as contrived metrics like “danceability” and “energy”
- Machine learning analysis of these features is of great interest to the industry for determining things like target market demographics and suggested songs lists



http://static.echonest.com/enspex/images/logo.png

http://static.echonest.com/enspex/images/spotify.png



Dataset

- Columbia's Labrosa laboratory maintains software to segment audio as well as a premade dataset of 1 million entries with the following features of interest:
  - Bar and beat starts, danceability, energy, key, loudness, section starts, timbre, tatums and tempo
- Also includes metadata such as artist, song title and year
- Many features are arrays of values in temporal space, makes direct analysis somewhat difficult
- Code for generating dataset has lost functionality (6-8 year old code, many of API's used have gone defunct)
  - Could rewrite to use updated API's, but would be non trivial

*https://labrosa.ee.columbia.edu/millionsong/

Problem Formulation

- Our project focused on the classification of songs by the decade in which they were released
- Split classes up by decade, as performing classification by discrete years unsurprisingly warranted very poor results
- 10 classes, 1920’s – 2010’s
- 515345 songs within the Million Song DB have years associated with them

https://sites.google.com/site/50sflashback/_/rsrc/1353371375812/home/events/sock-hops---haley-shearer/sock%20hop.jpg

https://www.timeoutdubai.com/sites/default/files/tod/styles/full_img/public/images/2018/01/09/2017_Moes_on_the_5th_dubai_base_1.jpg?itok=TyeLFSy3

Overall Methodology

- Did cross validation and feature selection with multiple types of classifiers
- Used the following types of classifiers as implemented by the sklearn python library:
  - Neural network
  - Adaboost with decision stumps
  - Multiclass SVM
- Since many of the features were variable length arrays, we took the mean squared displacement of their values
- Used *GridSearchCV* from sklearn to optimize metaparameters such as loss type, number of classifiers and learning rate

Neural Network Analysis

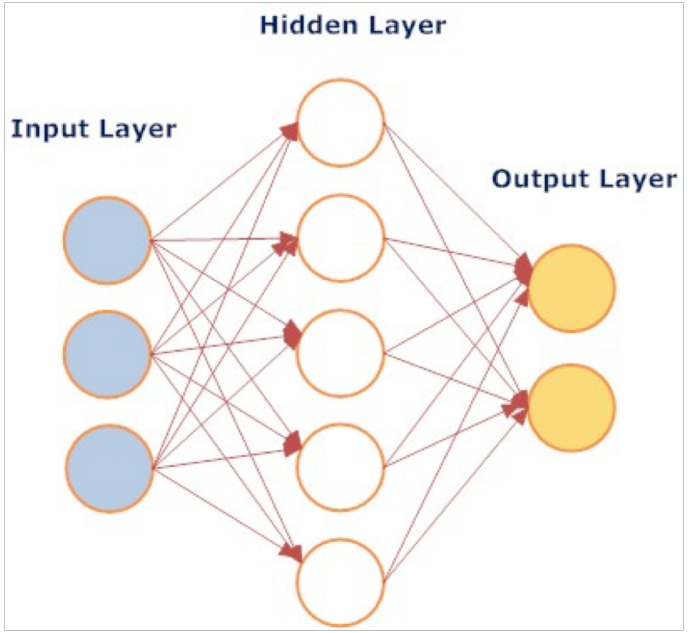
- Stochastic gradient based optimizer
  - by Kingma et.al
- Tolerance for optimization: **1e-4**
- Activation : **Logistic sigmoid**

Parameter Tuning:

1. Number of layers and nodes
2. L2 regularization term
3. Learning rate

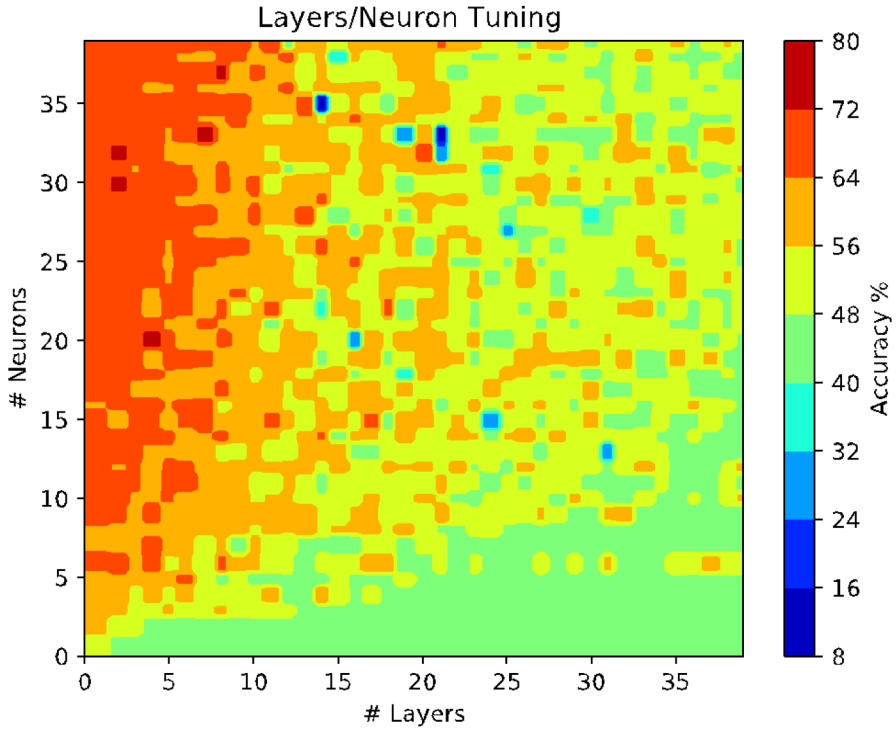
Criterions:

1. Number of Epochs
2. Training error
3. Cross-validation error



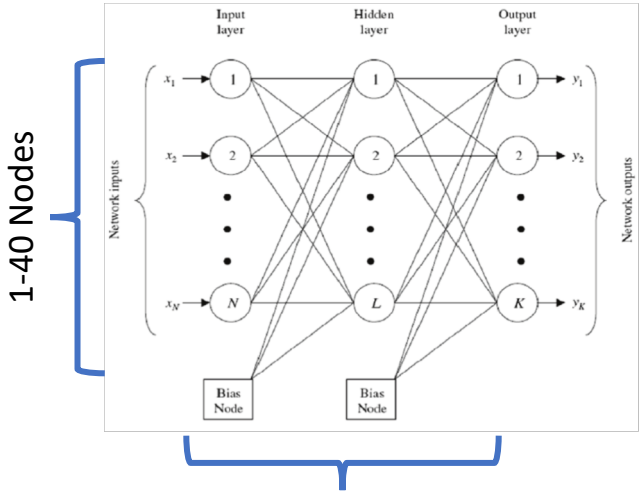
Schematic of NN

Parameter Tuning: Layers and Nodes

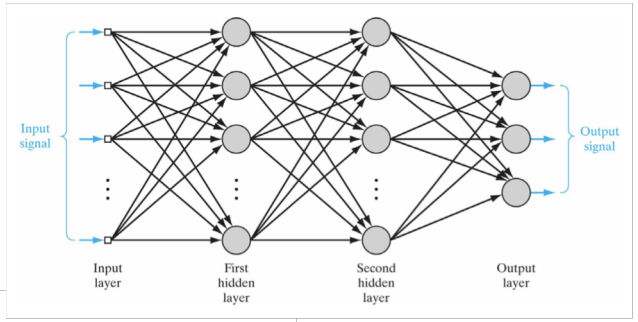


Dataset size: 515,345
Toy Dataset for PT: 500
Optimum parameters:

2 layers with 39 neurons



Parameter Tuning: Learning Rate and Momentum



Back propagation Algorithm

$$\delta_j(n) = a y_j(n)[1 - y_j(n)] \sum_k w_{kj}(n)\delta_k(n)$$

Back-prop gradient

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n)$$

Weight update

Gradient descent hyperparameter

Helps ease of oscillating weights due to large learning rate

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) + \alpha \Delta w_{ji}(n-1), \qquad 0 < \alpha < 1$$

Weight update (current)

Momentum

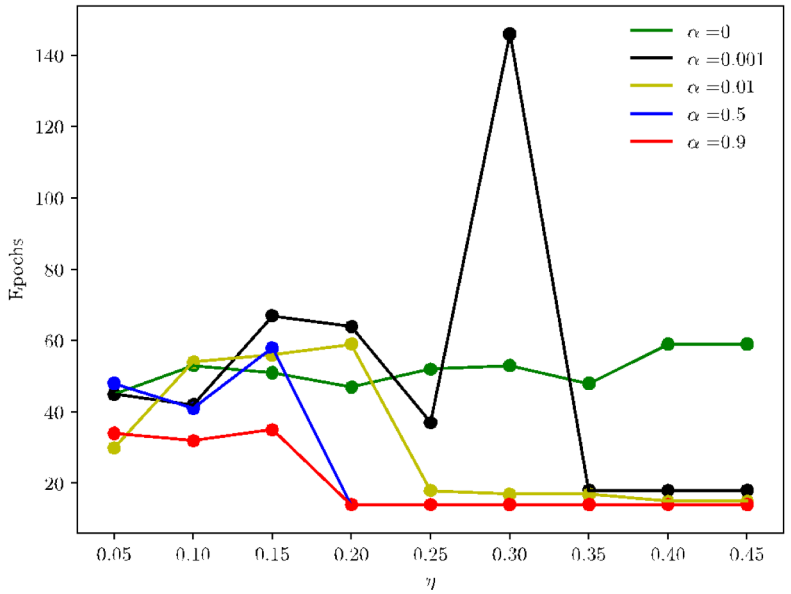
Weight update (previous time step)

In downhill situation: accelerating learning by a factor of $1/(1-\alpha)$

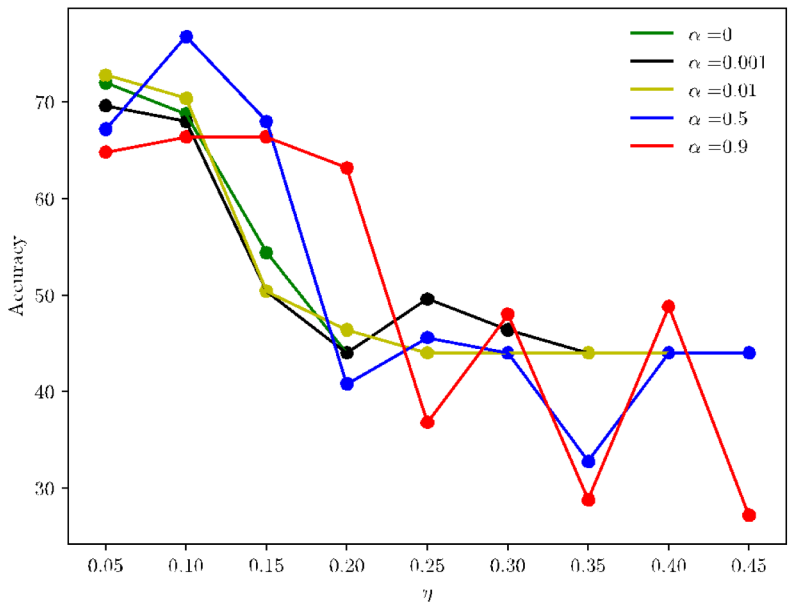
Courtesy to Prof. Deliang Wang and Prof. Jeremy Morris

Parameter Tuning

Optimizing over Epochs



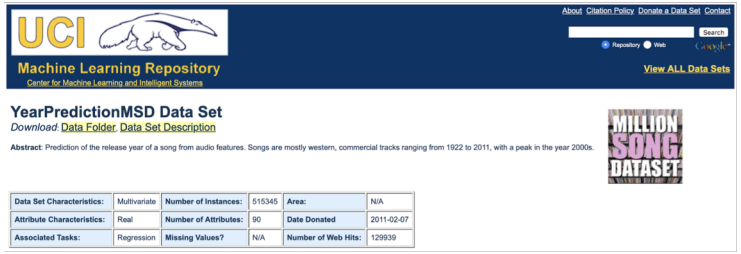
Optimizing over training accuracy

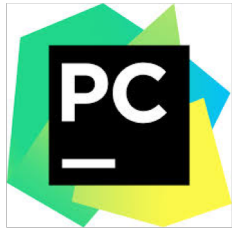


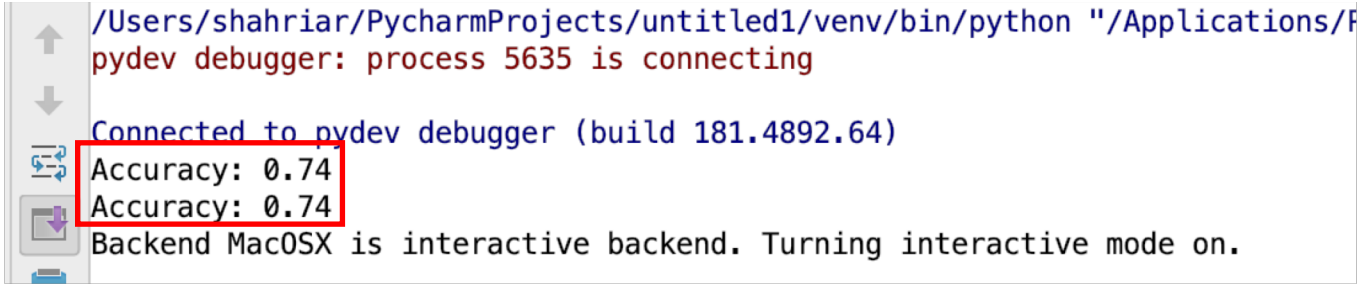
$\boldsymbol{\eta = 0.1, \alpha = 0.5}$ Parameters resulted in 70-75% score accuracy on training data

Model Accuracy

- Cross-validating settings: Train on 75% of data and test on 25% unseen data
- 2 Layer with 39 Neurons in each and learning rate of 0.1 with regularization parameter of 0.5
- With the accuracy of **74%,** the decade of each song can be predicted







SVM

Design choices

- Dataset size: 515345 songs
- UCI recommended splits (to avoid producer effect)
- Training set: first 463715 songs
- Validation set (for parameter tuning): subset of training set (500 songs)
- Test set: last 51630 songs
- SVM: Linear SVC
    - implements 1 vs rest; preferred over 1 vs 1
    - similar results for significant less running time

SVM Parameter-Tuning

- C: powers of 2 in range [-4, 8)
- Max iterations: 1e6
- Scoring metric: accuracy
- k: 3
- Squared hinge loss (primal) vs hinge loss (dual)

	-4	**-3**	**-2**	**-1**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
loss=sq_hinge dual=False	.42	.42	.44	.45	.44	.44	.44	.44	.44	.44	.44	.44
loss=hinge dual=True	.38	.37	.41	.41	.42	.42	.42	.42	.42	.43	.43	.43

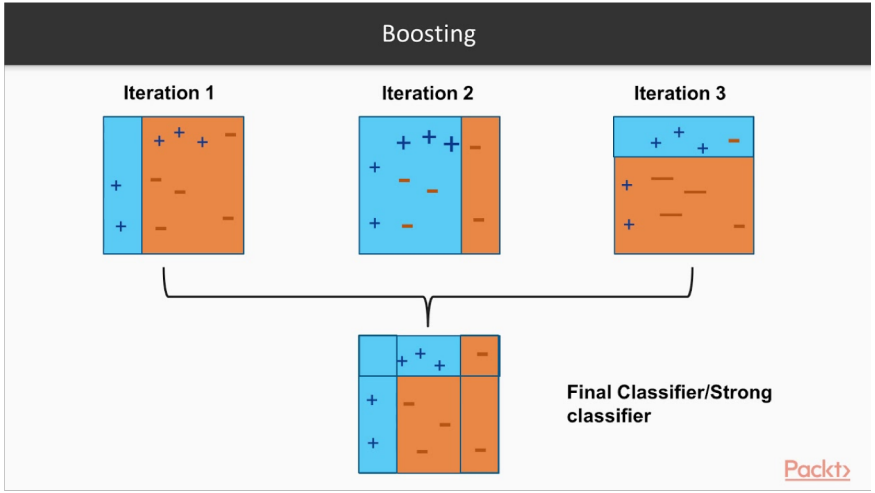
SVM Detailed Classification Report

- Accuracy (on entire test data): 56%
- Total running time: ~8 min

	precision	recall	f1-score	support
1920	0.02	0.65	0.05	20
1930	0.01	0.54	0.02	13
1940	0.02	0.49	0.05	61
1950	0.05	0.45	0.09	275
1960	0.16	0.35	0.22	1166
1970	0.26	0.35	0.30	2396
1980	0.36	0.44	0.40	4201
1990	0.51	0.11	0.18	12580
2000	0.73	0.81	0.77	29885
2010	0.00	0.00	0.00	1033

Adaboost

- Same design setup as SVM
- Results entirely dependent (within 1% accuracy) of the data split, not the number of estimators when doing 500 sample cross validation
  - Tested with 10, 50, 100, 300, 500 and 1000 classifiers to confirm this
- Results ranged from 25% to 62% preliminarily, and were consistently ~40% on larger testing dataset
- Classifiers trained on the same data split always resulted in the same classification accuracy



https://i.ytimg.com/vi/BoGNyWW9-mE/maxresdefault.jpg

Results Summary

- Feature selection did not significantly improve accuracy
- Accuracy of NN (~74%) > SVM (~56%) > Adaboost (~40%)
- Best overall accuracy: 74% (NN)
    - 2 hidden layers with 39 neurons per layer

What we learned

- Dealing with actual giant databases
- Interpreting the available datasets and make the bridge to translate the data to array inputs
- Learned how to work with available packages: Scikit
- Temporal data is very difficult to work with using the tools covered in class
    - Recurrent networks that can operate in the time domain would give better results than MSD; future direction

~~Questions?~~ No Time For Questions.