COMS E6998 012/15099

Practical Deep Learning Systems Performance

Lecture 2 09/12/19

Logistics

- Course syllabus and slides available on courseworks
- Assignment 1 will be posted 09/15/2019; due in two weeks
- Use google colab to submit assignments
- Submit notebooks on course git

Recall from Last lecture

- Cloud and AI
- Factors contributing to AI success
- Model code is only a small part of the entire machine learning system
- Performance concerns are at each stage of ML life cycle
- Regression, Classification, Supervised, Unsupervised

Model Complexity Tradeoffs

- Simple model
  - Fail to completely capture the relationship between features
  - Introduce bias: Consistent test error across different choices of training data
  - Low variance
  - Increasing training data does not help in reducing bias
- Complex model captures nuances in training data causing Overfitting
  - Low bias
  - With different training instances, the model prediction for same test instance will be very different – High Variance

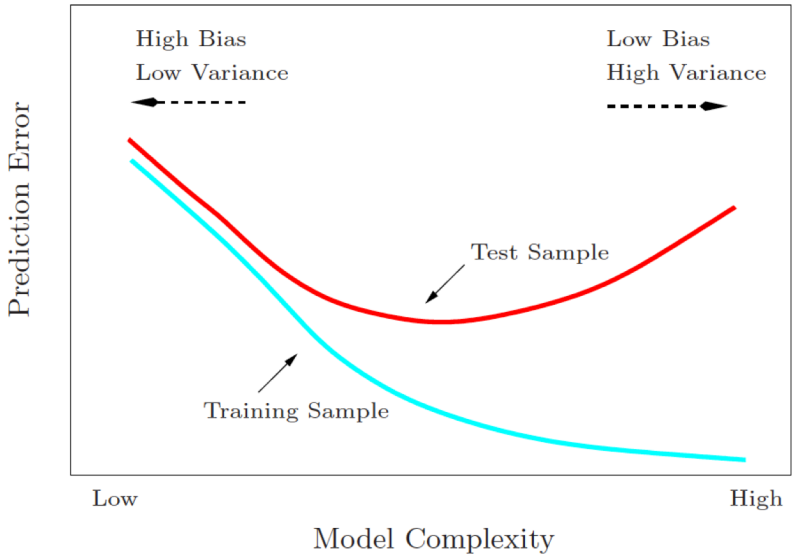
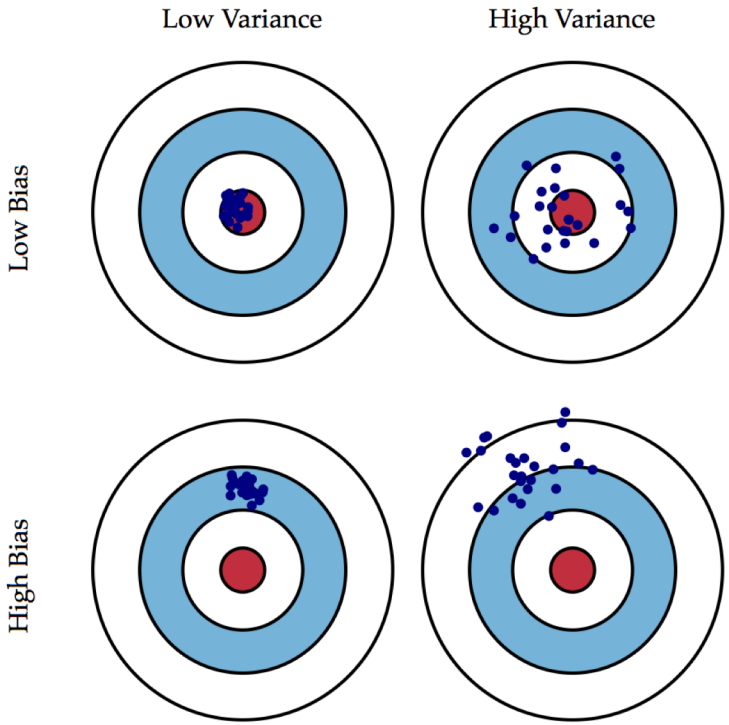


Figure from Hastie, Tibshirani, Friedman, The Elements of Statistical Learning

Bias-Variance Tradeoff



$$MSE_{test} = Bias^2 + Variance + Irreducible\ error$$

High Bias

Increase model complexity

Low Bias

High Variance

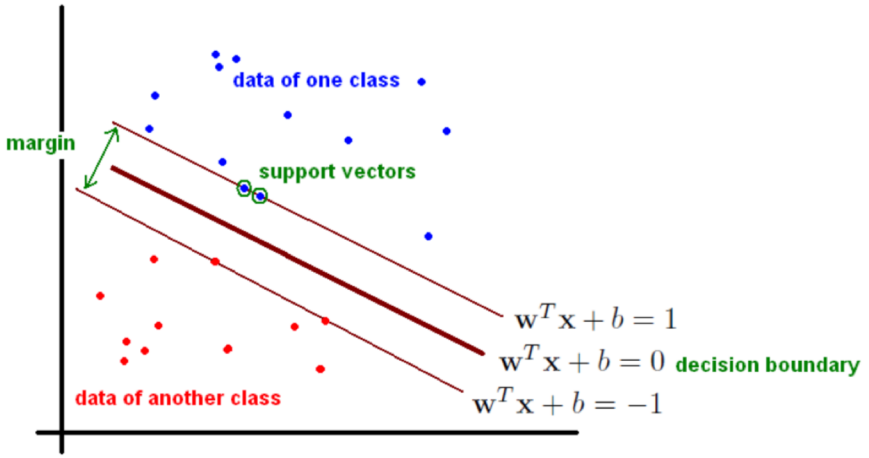
Increase training data

Low Variance

Picture from Scott Fortmann-Roe, Understanding the Bias-Variance Tradeoff, 2012

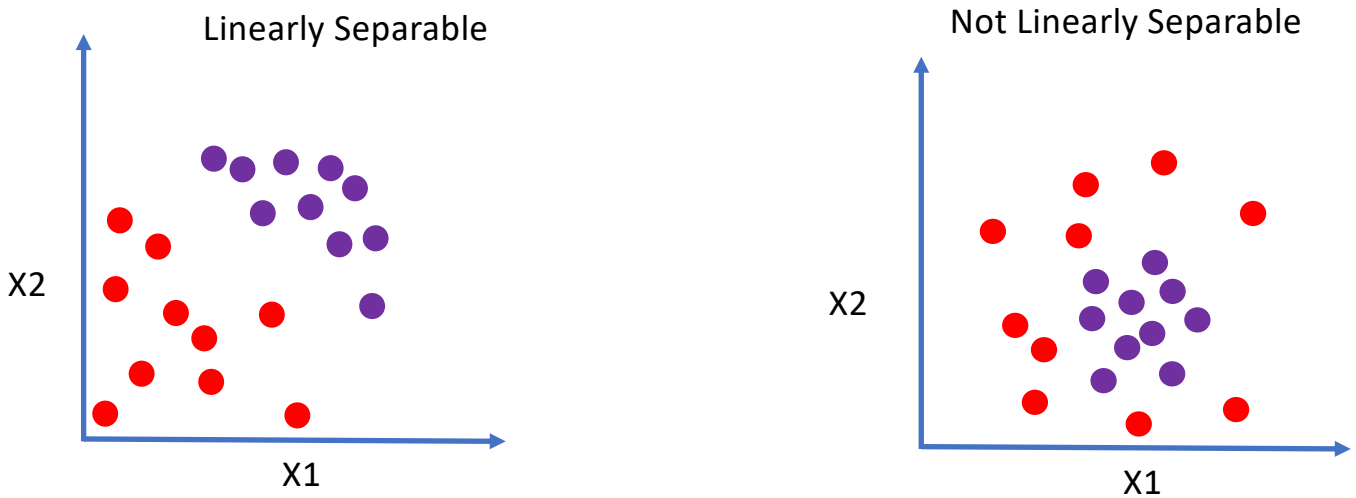
Support Vector Machines

- Support vectors are the extreme points in any dataset
- Hyperplane is the separator between the two dataset
- SVM problem
  - Find a hyperplane such that the distance between support vectors and the hyperplane is the largest
- Widest margin problem: Hard vs Soft Margin
- Good at dealing with high dimensional dataset
  - Linear separability in higher dimensional space wit appropriate transformations
- Kernel methods: Selecting the right kernel and hyperparameters can be computationally intensive
- Hyperparameters: Kernel method (linear, RBF), C value



Zoya Gavrilov http://web.mit.edu/zoya/www/SVM.pdf

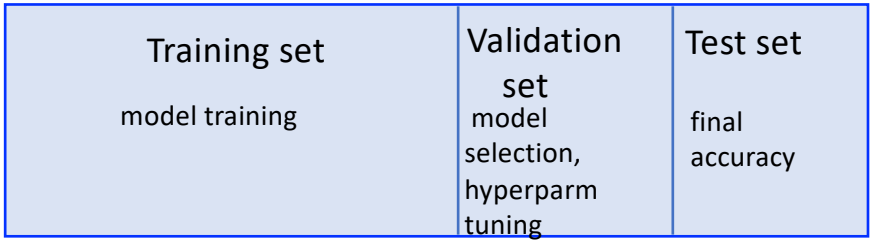
Linear Separability



- Measure of data complexity for classification problems
- For binary classification, linear separability implies the existence of a hyperplane completely separating the two classes
- Tests for Linear Separability:
  - Convex Hull: intersection of convex hulls of classes is empty
  - 100% SVM accuracy with linear kernel
  - Review example using IRIS dataset

Generalization

- Labeled dataset partitioning: Training data, Validation data, Test data
- Training data: For model training, i.e., to learn model parameters
- Validation data: For model selection and/or hyperparameter tuning
- Test data: For final accuracy of the tuned and trained model



Hold-out and Cross-Validation

- Holdout technique
  - Data divide into 2 subsets: training-set and validation set
  - Train on training-set and test on validation set
- ***K-fold*** cross-validation
  - Data divided into K subsets
  - Repeat hold K times, each time with (K-1) subsets as training set and 1 subset as validation set.
  - Model performance is average across K validation sets
  - K=number of data points, *leave-one-out cross-validation*
- Gives an idea of model generalization, identify overfitting
- Can be used for hyperparameter tuning

Regularization

- Techniques that make the model simple
- Techniques to make an algorithm perform well on test data often at expense of its performance on training data
- Avoids overfitting, increases bias, reduces variance
- Sample techniques:
  - Parameter norm penalties
    - L_2 and L_1 norm weight decay
  - Noise injection
    - Dropout

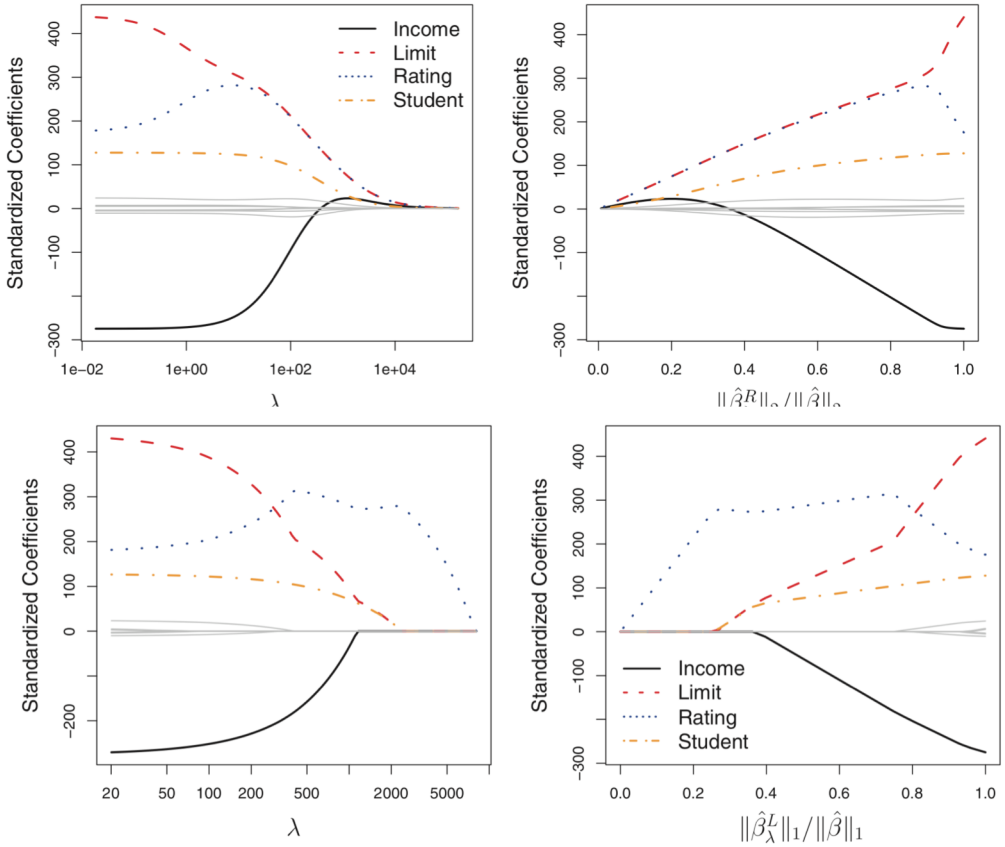
Regularization in Regression

L_2 Regularization Loss

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}\beta_j^2$$

L_1 Regularization Loss

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}|\beta_j|$$



G. James et al, Introduction to Statistical Learning Theory

Bias-Variance Tradeoff Revisited

Increase model complexity

Low Bias

Increase training data

Low Variance

Increase model complexity + Training with increased amount of data + Regularization

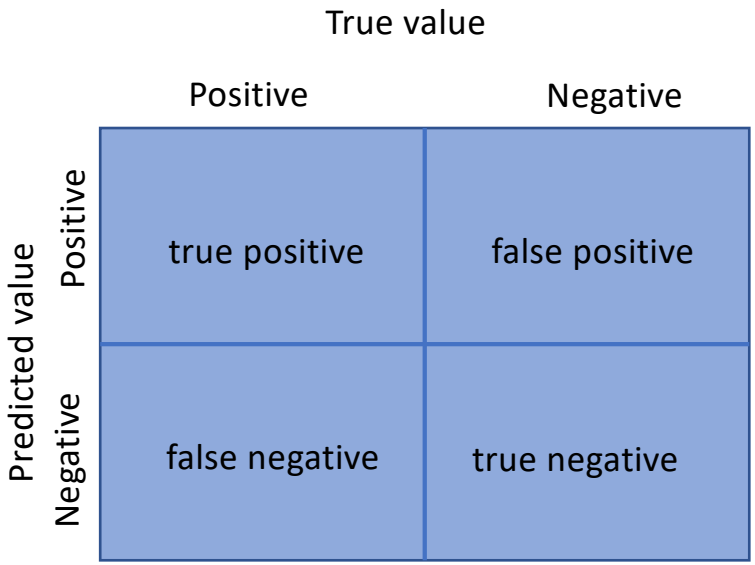
Low Bias, Low Variance

Increased data and faster compute has enabled working with deep neural networks, which with proper Regularization can achieve low bias and low variance

Performance Metrics

- Accuracy, Loss
- Precision, Recall, F score
- Confusion matrix
- Training time, inference time
- Training cost

Accuracy, Precision, Recall, Specificity



$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$$

$$\text{Precision} = \frac{tp}{tp + fp}$$

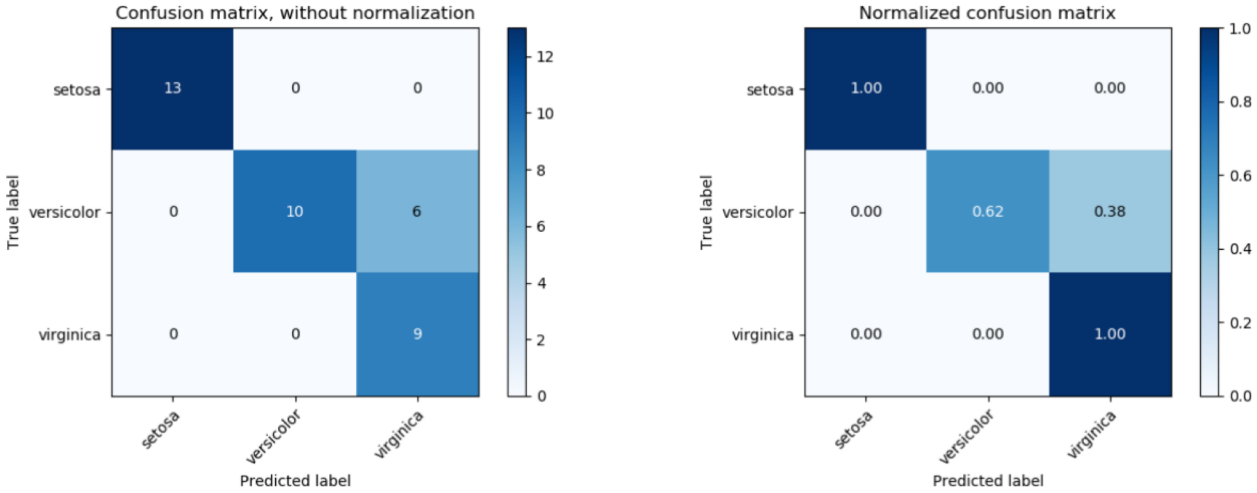
$$\text{Recall} = \frac{tp}{tp + fn} \qquad \text{True negative rate} = \frac{tn}{tn + fp}$$

Balanced accuracy = $(True\ negative\ rate + Recall)/2$

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}}$$

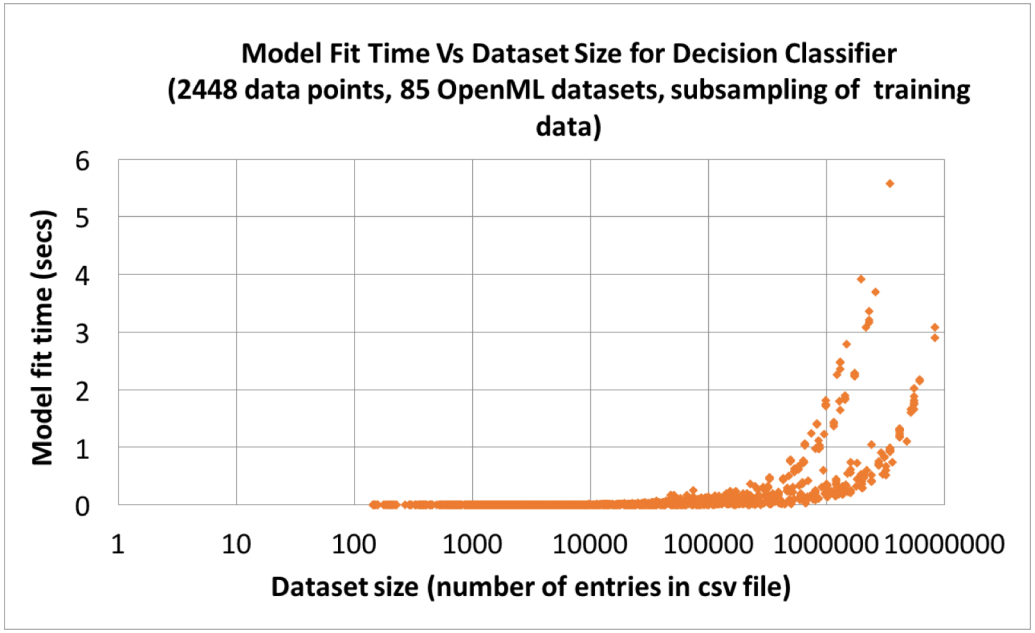
Loss: Residual sum of squares for Regression and cross-entropy for classification

Confusion matrix



- Heat map visualization
- Not symmetric: *versicolor* confused with *virginica* not vice-versa
- Prediction is most accurate for *setosa* and *virginica*
- Helps calculate Precision and Recall for multiclass classification

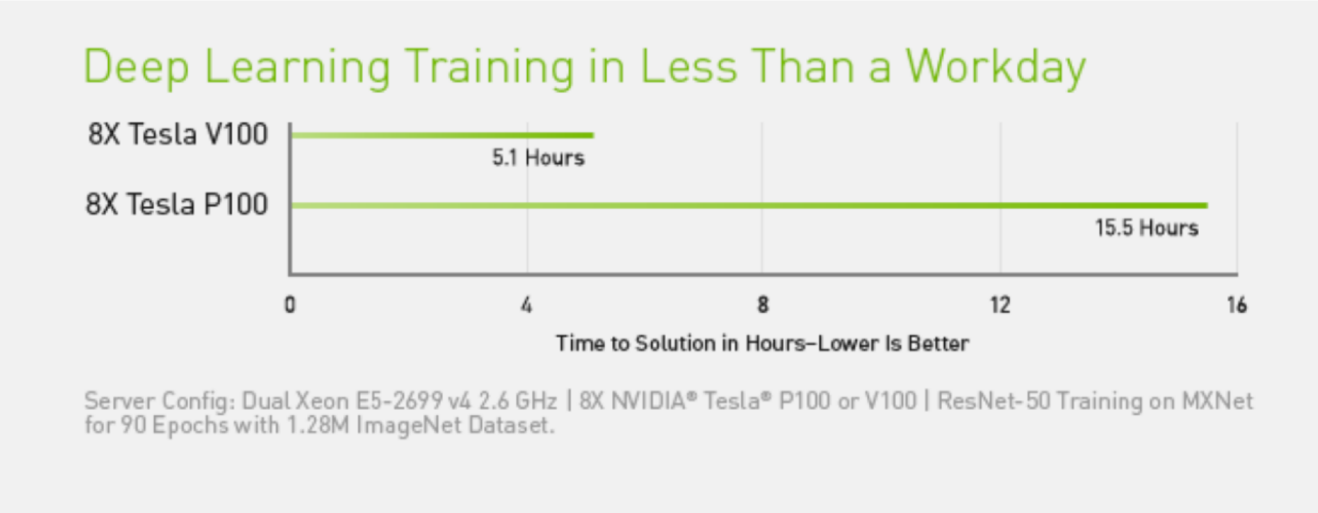
Training time: ML



- Training time of decision tree classifier is exponential in dataset size beyond a certain threshold.
- Training time also depends on how the model scales with different features of input: number of training data, linear separability, complexity (number of output classes for classification problems

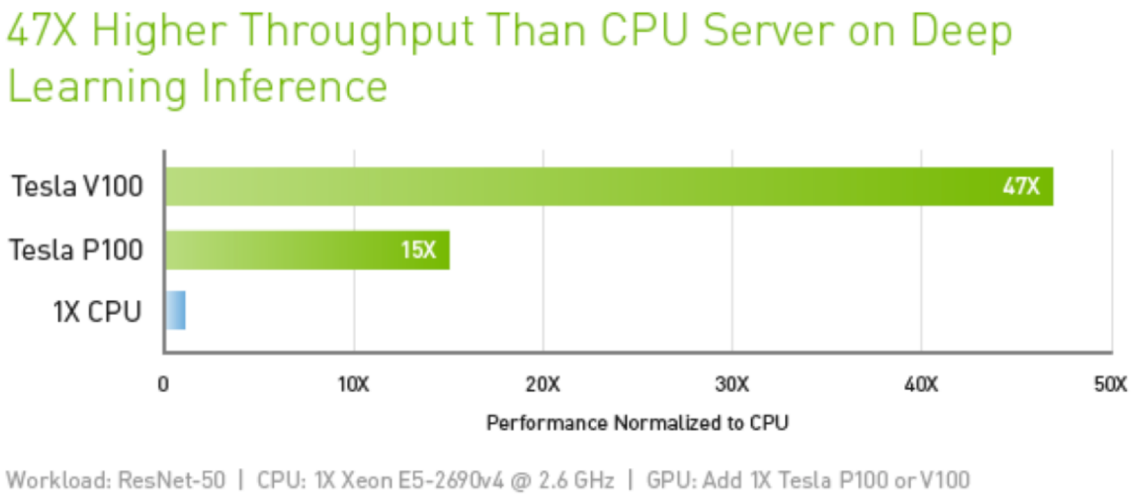
DL Training Time

- DL performance is closely tied to the hardware
- Training Time



https://www.nvidia.com/en-us/data-center/tesla-v100/

DL Inference Throughput



https://www.nvidia.com/en-us/data-center/tesla-v100/

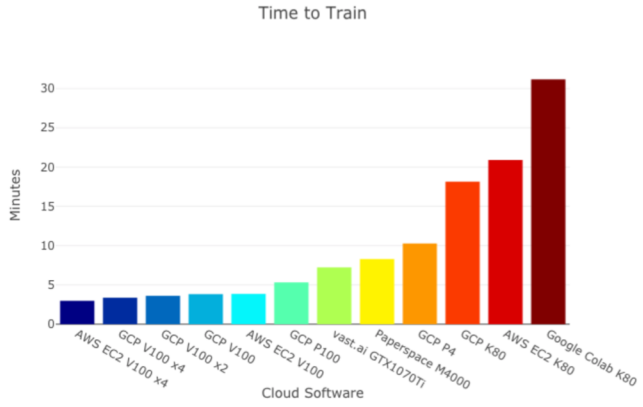
Training time and cost: An example

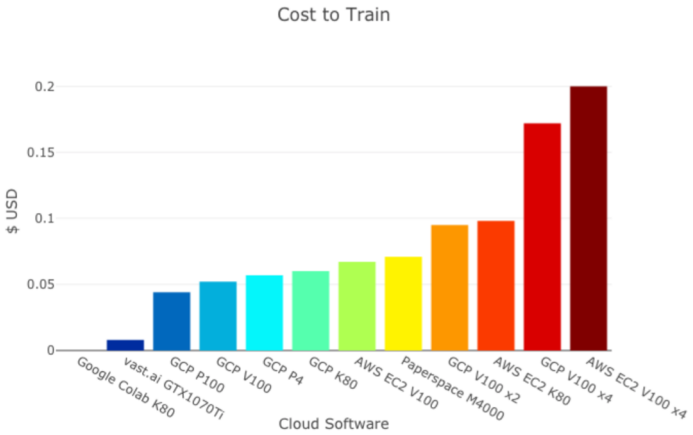
Cloud Service	NVIDIA GPU	CPUs	GPU RAM	CPU RAM	Cost Per Hour	Wall Time	Cost to Train
Google Colab	K80	1	12	13	0.00	31.17	0.000
Google Cloud Compute Engine	P100	6	16	20	0.50	5.32	0.044
Google Cloud Compute Engine	K80	6	12	17	0.20	18.13	0.060
Google Cloud Compute Engine	V100	8	16	20	0.82	3.83	0.052
Google Cloud Compute Engine	P4	4	8	26	0.33	10.28	0.057
Google Cloud Compute Engine	V100 x 2	8	32	30	1.57	3.63	0.095
Google Cloud Compute Engine	V100 x 4	8	64	30	3.05	3.38	0.172
AWS EC2	K80 (p2.xlarge)	4	12	61	0.28	20.90	0.098
AWS EC2	K80 x 8 (p2.8xlarge)	32	96	488	2.35	16.12	0.631
AWS EC2	V100 (p3.2xlarge)	8	16	61	1.05	3.85	0.067
AWS EC2	V100 x 4 (p3.8xlarge)	64	128	488	4.05	2.97	0.200
vast.ai	GTX 1070 Ti	4	8.1	16	0.06	7.23	0.008
Paperspace	Quadro M4000	8	8	30	0.51	8.30	0.071

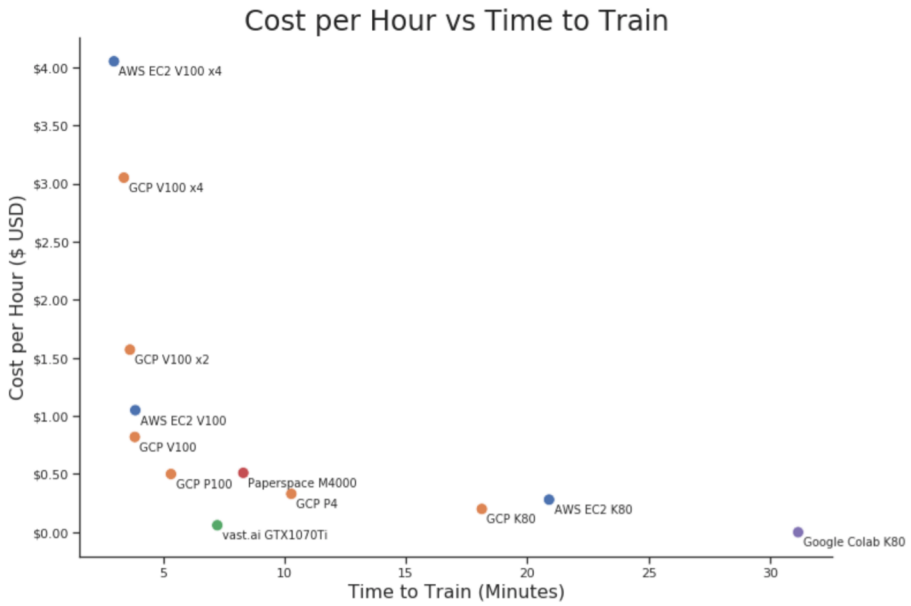
- Training time and cost are important when doing DL on cloud
- While runtime (Wall Time) is mostly governed by GPU type, different cloud platforms show differences (18.13 on GCP vs 20.90 on AWS EC2)
- GCP is cheaper than AWS EC2 for same GPU (compare cost with 1 K80 and 1 V100)
- Job does not scale linearly with increasing compute

Data collected by Jeff Dale . Included in medium article, *Best Deals in Deep Learning Cloud Providers*
Jupyter notebook available at https://www.kaggle.com/discdiver/best-values-in-deep-learning-cloud-providers/

Training time and cost tradeoffs







Data collected by Jeff Dale . Included in medium article, *Best Deals in Deep Learning Cloud Providers*
Jupyter notebook available at https://www.kaggle.com/discdiver/best-values-in-deep-learning-cloud-providers/

What Performance Metrics are Important ?

- Depends on the use case
- Model deployment on edge devices: model size is very important; model should fit into device (limited) memory
- Model deployment as a cloud service: model robustness, de-biasing, inference time
- Model development and training: training time, training cost, accuracy
- Instead of a single performance objective, performance optimization of ML models is often a *multi-objective problem*

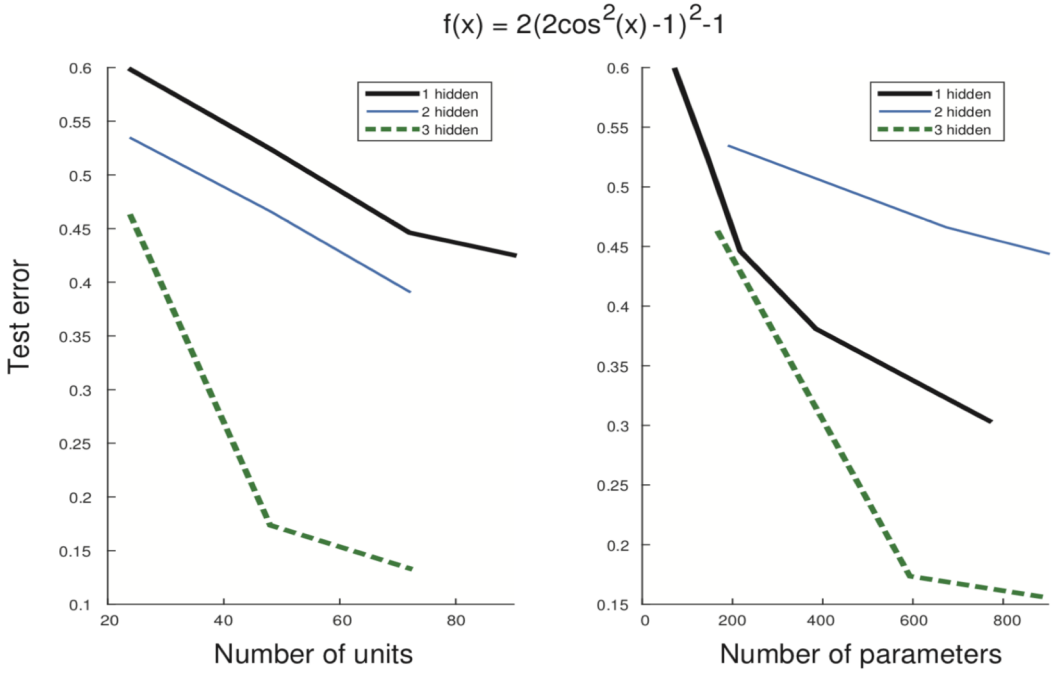
Universal Approximators Theorem

- Multilayer layer feedforward network, with as little as two layers, with arbitrary activation functions, and **sufficiently large hidden units** can approximate any arbitrary function

- Then why have Deep neural networks (more than one hidden layer) ?

- Going deep requires fewer units per layer; reduces the capacity of the network

Neural Network Size: Depth vs Width



Deep Networks require exponentially less number of parameters than shallow networks for approximating the same function

Easier to train with less data

Hrushikesh Mhaskar, Qianli Liao, Tomaso Poggio
When and Why Are Deep Networks Better than Shallow Ones? AAAI 2017

Dataset Augmentation

- Augment the training dataset by applying domain-specific transformations
- Provides more training data
- Helps in model generalization
- Augmentation techniques (images):
  - Horizontal and Vertical shift
  - Horizontal and Vertical flip
  - Rotation
  - Brightness
  - Zooming
  - Noising
- Only applied to training set, not to validation and test set

L_2 and L_1 Regularization in Neural Network

L_2 Regularization Loss

$$L = \sum_{(x,y)\in\mathcal{D}} (y - \hat{y})^2 + \underbrace{\lambda \cdot \sum_{i=0}^{d} w_i^2}_{L_2-\text{Regularization}}$$

$$w_i \Leftarrow w_i(1 - \alpha\lambda) - \alpha\frac{\partial L}{\partial w_i}$$ Weight decay

L_1 Regularization Loss

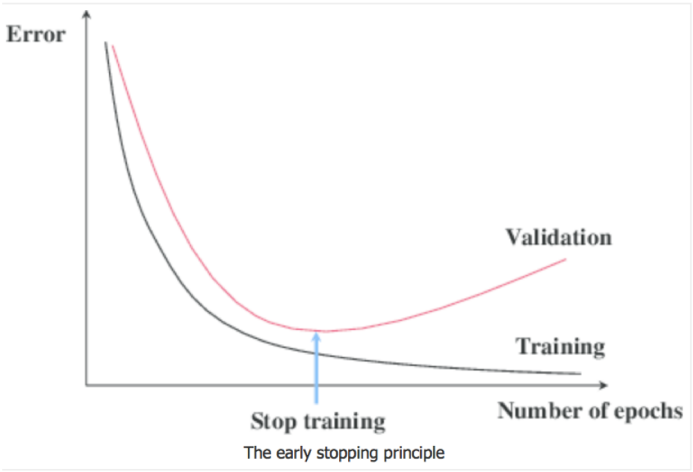
$$L = \sum_{(x,y)\in\mathcal{D}} (y - \hat{y})^2 + \lambda \cdot \sum_{i=0}^{d} |w_i|_1$$

$$w_i \Leftarrow w_i - \alpha\lambda s_i - \alpha\frac{\partial L}{\partial w_i} \qquad s_i = \begin{cases} -1 & w_i < 0 \\ +1 & w_i > 0 \end{cases}$$

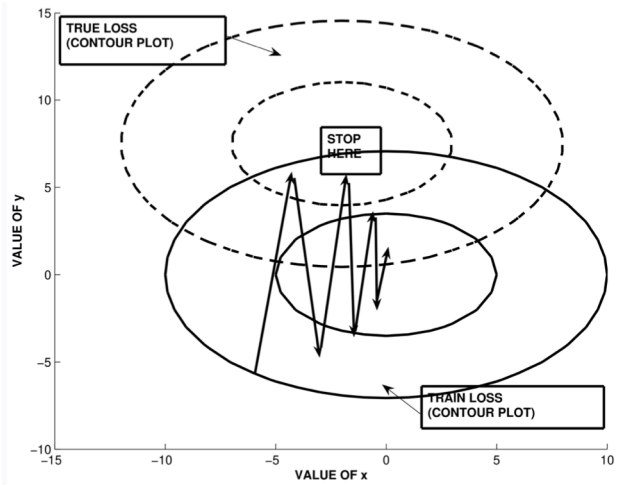
L_2 vs L_1 Regularization in Neural Network

- Value of lambda (hyperparameter) can be tuned using the validation set
- L_1 regularization leads to sparse weight matrices; Used to determine edges to prune
- Both L_2 and L_1 regularization move the weights progressively towards 0
- Multiplicative vs additive weight decay

Early Stopping



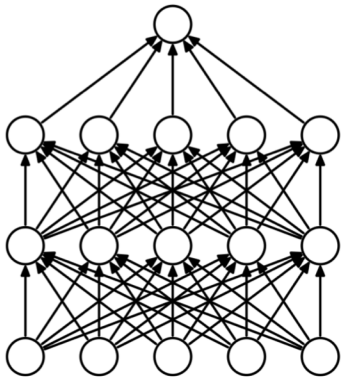
The early stopping principle



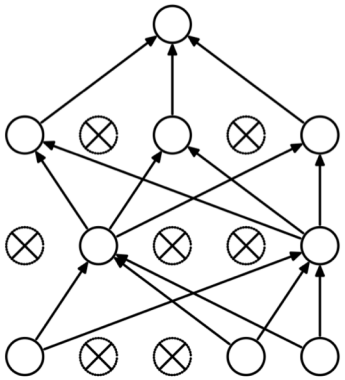
- Stop the training when validation error starts rising to prevent overfitting
- Early stopping is an implicit regularization technique
- Done in hindsight; define a performance criteria and checkpoint the latest “best” model
- May not help with large datasets with less likelihood of overfitting
- No principled approach to early stop. Can be tricky when validation error has multiple local minimas
- L_2 regularization can achieve similar or better performance than early stopping

Dropout

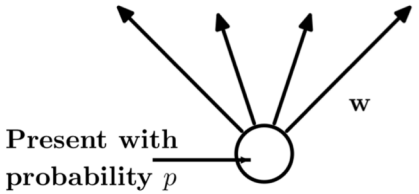
- Dropout is a regularization technique to deal with overfitting problem
- Prevents co-adaptation of activation units
- Probabilistically drop input features or activation units in hidden layers
- Layer dependent dropout probability (~0.2 for input, ~0.5 for hidden)



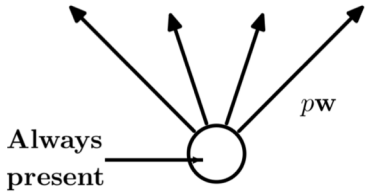
(a) Standard Neural Net



(b) After applying dropout.



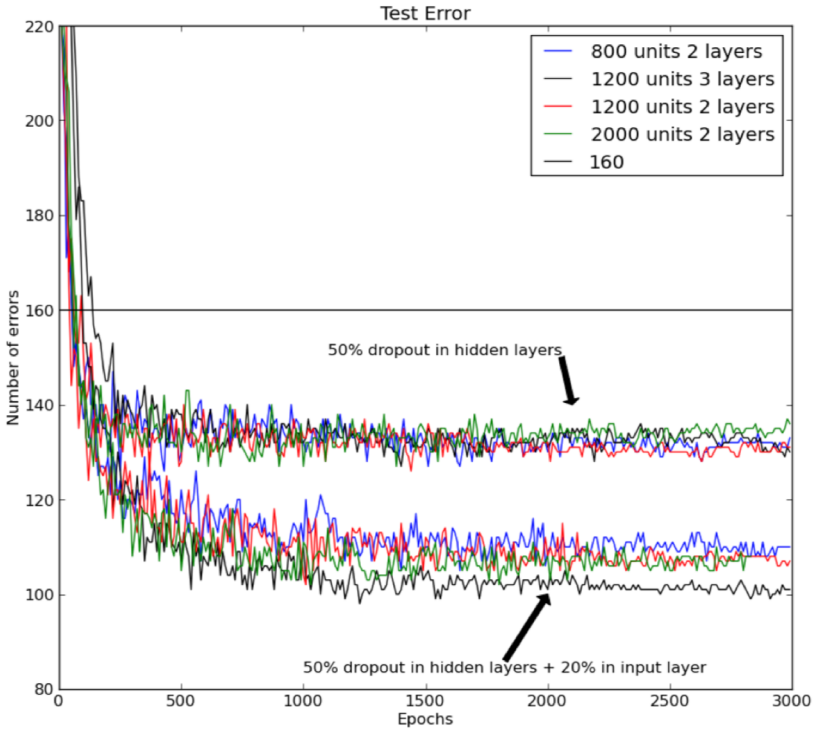
(a) At training time



(b) At test time

N. Srivastava et al, Dropout: A Simple Way to Prevent Neural Networks from Overfitting, JMLR 15 (2014) 1929-1958

Dropout with MNIST Image Classification



MNIST dataset (images of handwritten digits)
60,000 train images
10,000 test images
160 errors (without droput)
130 errors (with 50% dropout in hidden layers)
110 errors (with 50% dropout in hidden layers + 20% dropout in input layer)
Improvement seen across different networks with different number of layers and units

MNIST dataset available at http://yann.lecun.com/exdb/mnist/

G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, 2012

Prepare for Lecture 3

- Access Jupyter Notebook
- Install Tensorflow and Keras
- You can use Google colab
- Work on Homework #1, due on 09/29/2019
- Read and Review “Suggested Reading” and “Code Links”
- Start thinking about project ideas and discuss with me

Seminar Reading List

- **Precision-Recall (papers also needed for Question 4 in Assignment 1)**
  - Jesse Davis, Mark Godarich The Relationship Between Precision-Recall and ROC Curves, ICML 2006
  - Peter A. Flach, Meelis Kul, Precision-Recall-Gain Curves: PR Analysis Done Right, NIPS 2015
- **Dropout**
  - G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R. R. Salakhutdinov
    Improving neural networks by preventing co-adaptation of feature detectors
  - Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov
    Dropout: A Simple Way to Prevent Neural Networks from Overfitting
- **ML Runtime Prediction**
  - Frank Hutter, Lin Xu, Holger H. Hoos, Kevin Leyton-Brown Algorithm Runtime Prediction: Methods & Evaluation

Suggested Reading

- Jeff Dale, *Best Deals in Deep Learning Cloud Providers*, *medium article*
- Lutz Prechelt, Early Stopping --- but when ?
- David E. Rumelhart, Geoffrey E. Hinton, Ronald J. Williams, Learning Representations by back-propagating errors

Blogs/Code Links

- Methods for testing linear separability in Python
  http://www.tarekatwan.com/index.php/2017/12/methods-for-testing-linear-separability-in-python/#fnref-102-6
- Use Early Stopping to Halt the Training of Neural Networks At the Right Time
  https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/
- Jupyter notebook comparing cloud providers
  https://www.kaggle.com/discdiver/best-values-in-deep-learning-cloud-providers/
- Dataset augmentation keras examples
  https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/
  https://machinelearningmastery.com/image-augmentation-deep-learning-keras/
- Model Selection: Underfitting, Overfitting, and the Bias-Variance Tradeoff
  https://theclevermachine.wordpress.com/tag/bias-variance-decomposition/