COMS E6998 010

Practical Deep Learning Systems Performance

Lecture 2 09/17/20

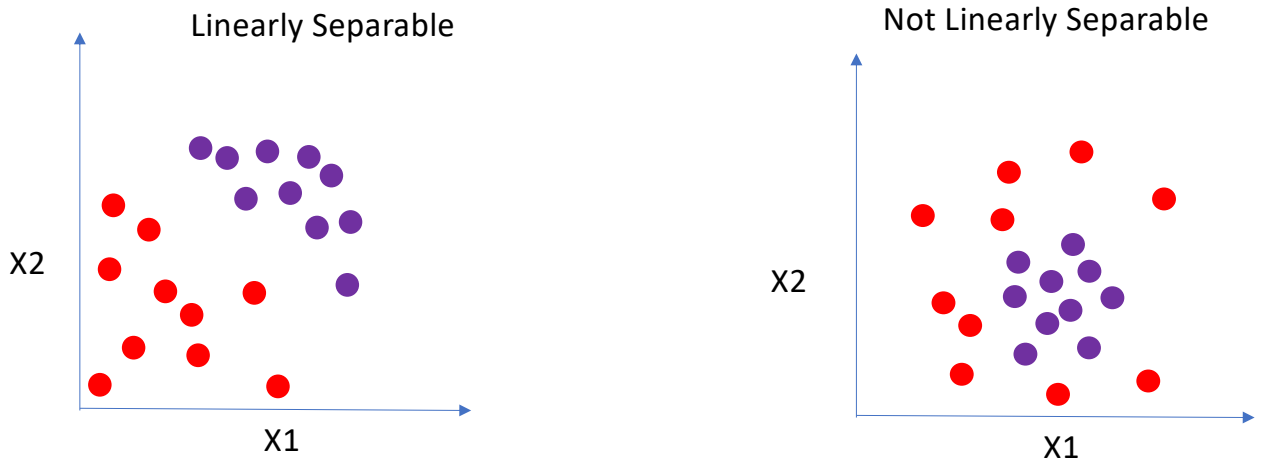
Logistics

- Assignment 1 will be posted today 09/17/2020; due in two weeks
- Use courseworks to submit assignments; pdf of jupyter notebook, pdf of written response, jupyter notebook, code
- No code, no marks

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
- Stages of ML life cycle
- Bias-variance tradeoff

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

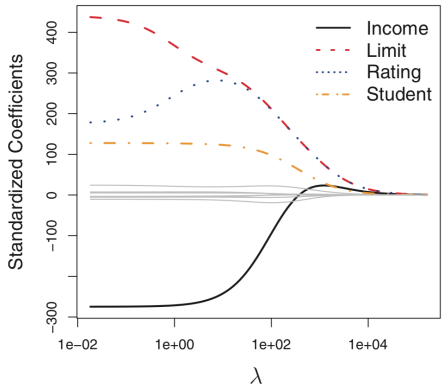
Regularization

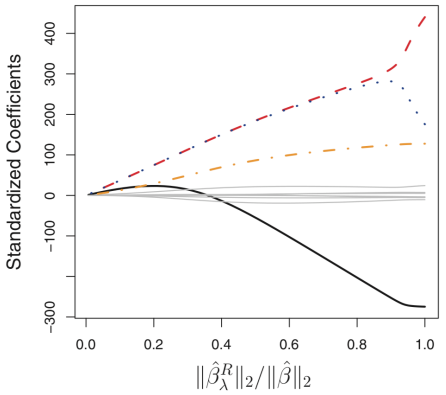
- Techniques used to improve generalization of a model by reducing its complexity
- Techniques to make a model perform well on test data often at expense of its performance on training data
- Avoid overfitting, increase bias, reduce variance
  - Relative decrease in variance is more than the increase in bias
- Sample techniques:
  - Parameter norm penalties
    - L_2 and L_1 norm weight decay
  - Noise injection
    - Dropout

Regularization in Regression

L_2 Regularization Loss (Ridge Regression)

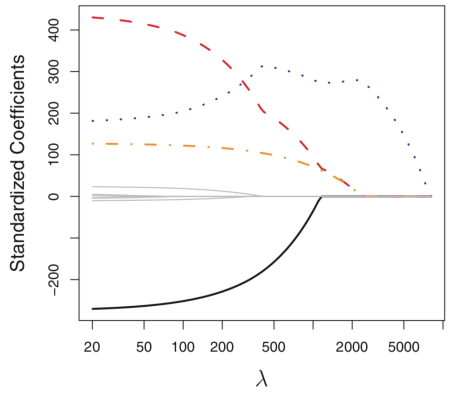
$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}\beta_j^2$$

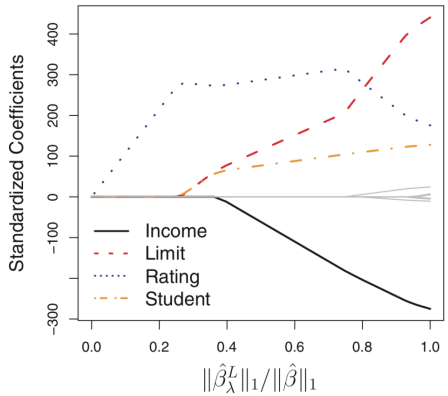




L_1 Regularization Loss (LASSO)

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



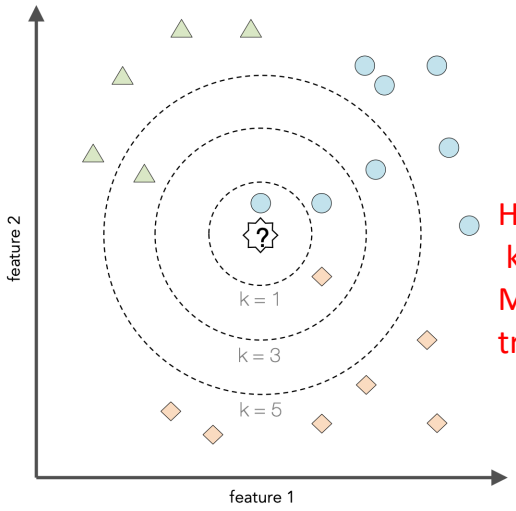


What value of lambda to choose ?

Hyperparameters vs Model parameters

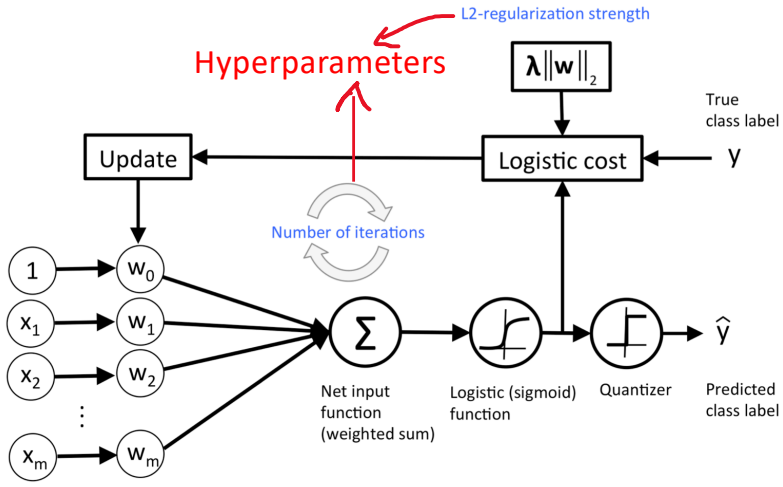
- Hyperparameters: parameters of the learning algorithm , which needs to specified a priori – before model fitting
- Model parameters: learned parameters of the model

k-nearest neighbor



Hyperparameters:
k, distance metric
Model parameters:
training instance

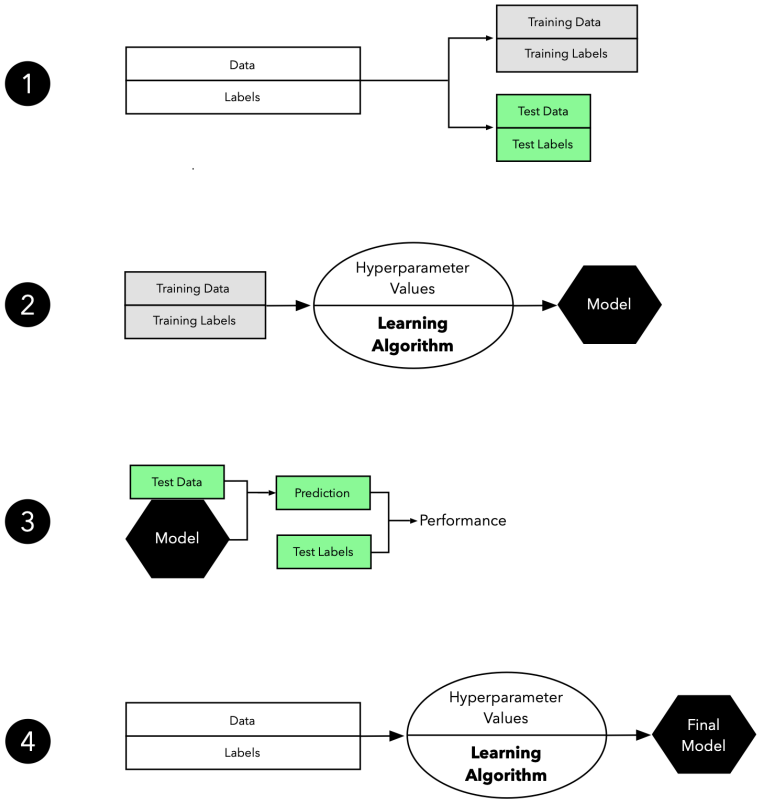
Logistic regression



Model Evaluation

- Estimate the generalization performance
  - What is the performance of my model on unseen dataset
- Improve predictive performance by tweaking the learning algorithm and selecting the best performing model (Hyperparameter Tuning and Model Selection)
  - What hyperparameter values gives the best performance
  - Computational requirements can be considered
- Identify the best machine learning algorithm (Algorithm Selection)
  - What is the best algorithm

Holdout Validation



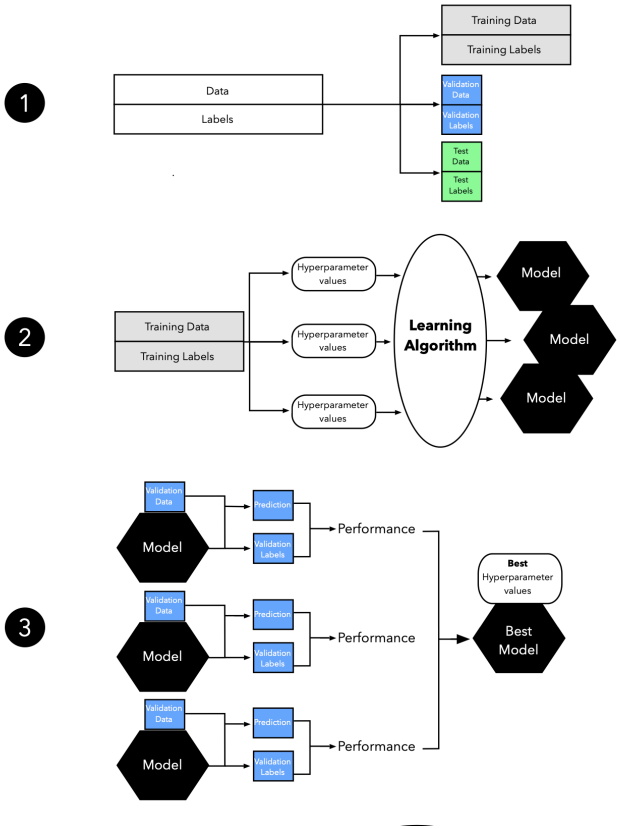
Training set	**Test set**
model training	final performance

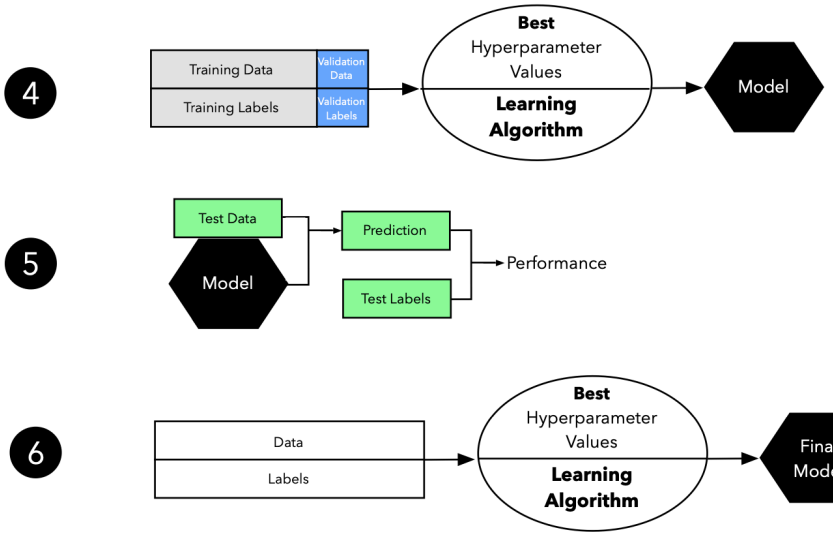
Model Evaluation using Cross-Validation

- Training data: model training, i.e., to learn model parameters
- Validation data: model evaluation for hyperparameter tuning/model selection
- Test data: final performance of the tuned and trained model
- **3-way Holdout technique**
  - Data training divide into 2 subsets: training set and validation set
  - Train on training set and evaluate model performance on validation set
  - High variance in performance

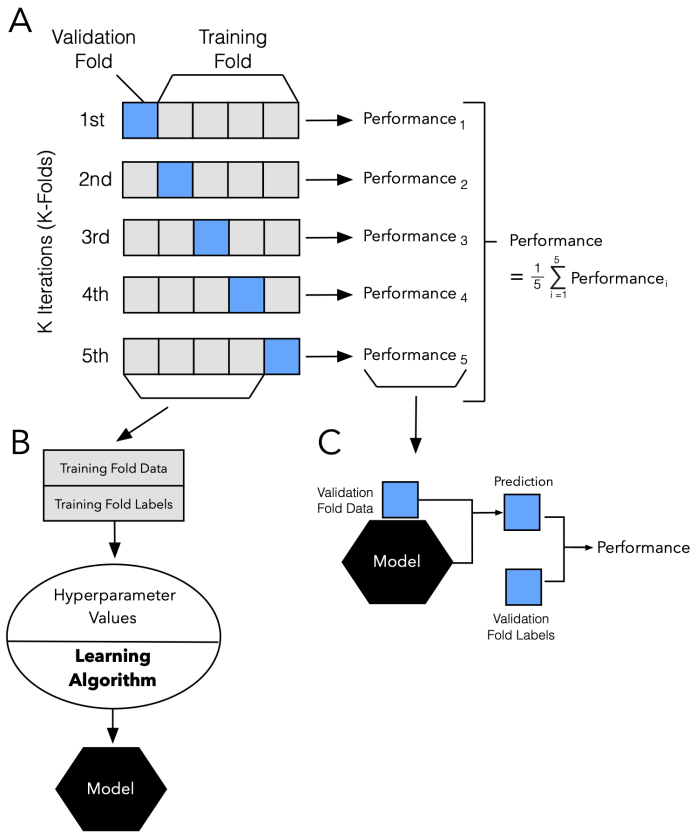
Training set	**Validation set**	**Test set**
model training	hyperparameter tuning/model selection	final performance

3-way Holdout for Model Selection



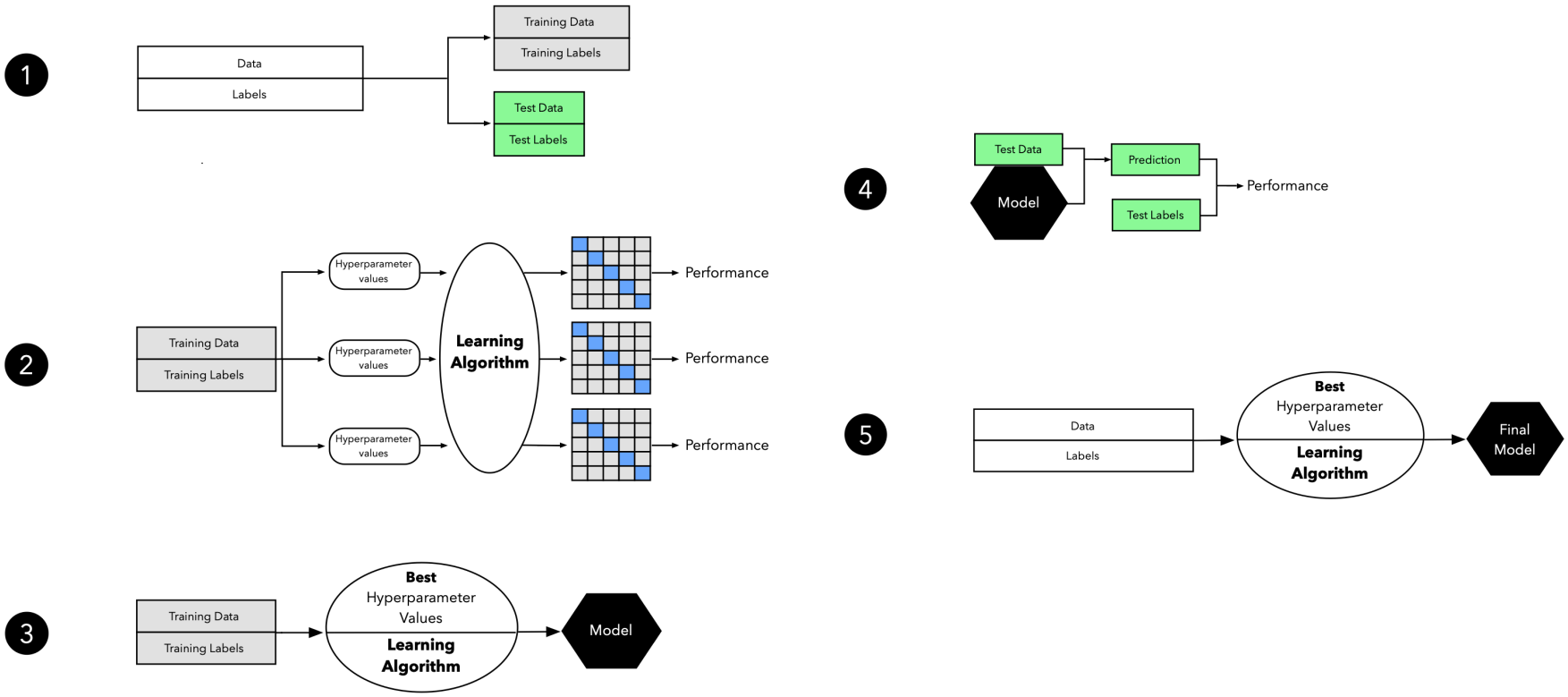


K-fold cross validation



- Data divided into K subsets
- Repeat hold K times, each time with *(K-1)* subsets as training set and 1 subset as validation set.
- Every data point gets to be in test set once
- Model performance is average across *K* validation sets
- Variance of the model performance estimate is reduced as *K* is increased.
- When K=1 its *leave-one-out cross-validation*

K-fold Cross-validation for Model Selection



Cross Validation in Deep Neural Networks

- 3-way holdout is preferred over k-fold cross validation
- Computational efficiency
- Availability of large dataset
- High variance is less of a problem in model performance

Bias-Variance Tradeoff Revisited

Increase model complexity

Low Bias

Increase training data

Low Variance

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

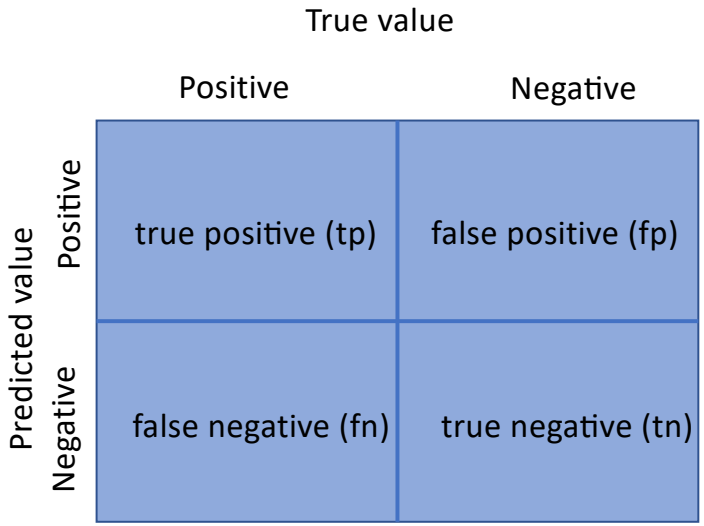
Low Bias, Low Variance

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
- Memory requirement

Accuracy, Precision, Recall, Specificity



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

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

False positive rate = 1-Precision

$$\text{Recall} = \frac{tp}{tp + fn}$$

Sensitivity, True positive rate

$$\text{True negative rate} = \frac{tn}{tn + fp}$$

Specificity

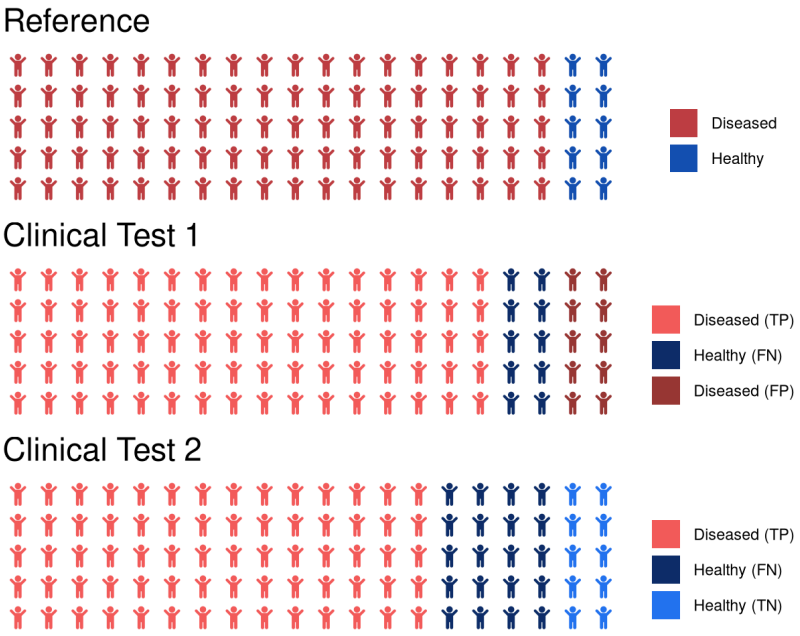
Balanced accuracy = $(Sensitivity + Specificity)/2$

Considers all entries in the confusion matrix

value between 0 (worst classifier) and 1 (best classifier)

Precision and Recall

True negatives are not accounted in Precision and Recall
Metrics work in situations where the correct identification of negative class is not important



$\beta = 1$ *is* $F1$ *score*

Harmonic mean of precision and recall

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

Confusion matrix for the first test

Prediction/Reference	Diseased	Healthy
Diseased	TP = 80	FP = 10
Healthy	FN = 10	TN = 0

Confusion matrix for the second test

Prediction/Reference	Diseased	Healthy
Diseased	TP = 70	FP = 0
Healthy	FN = 20	TN = 10

Prefer Test 1 with higher F1 score ➔
No one is healthy (Specificity = 0%)

Comparison of the two tests

Let us compare the performance of the two tests:

Measure	Test 1	Test 2
Sensitivity (Recall)	88.9%	77.7%
Specificity	0%	100%
Precision	88.9%	100%
F1 score	0.889	0.87

ROC, AUC, PRC

- A prediction model of outcome of SARS-CoV-2 pneumonia based on laboratory findings
- Beyond Accuracy: Precision and Recall

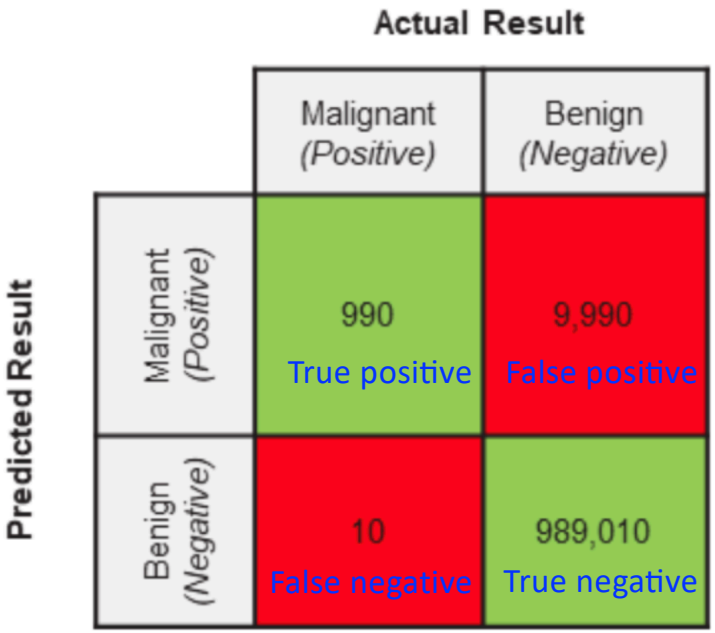
An Example AI System

- AI-assisted tumor diagnosis for cancer detection
- "The firm's deep-learning tool was able to correctly distinguish metastatic cancer 99% of the time, a greater accuracy rate than human pathologists."
- 99% sounds great. What does this actually mean ?

An Example AI System (contd.)

- Test data for performance evaluation:
  - 1,000,000 tumors in total
  - 999,000 out of 1,000,000 are *actually* benign (Actu Negatives)
  - 1,000 out of 1,000,000 are *actually* malignant (Actu Positives)
- Vendor's performance
  - Accuracy = 99%
  - Precision = 9.02%
  - Recall = 99%
- How many tumors did the system correctly classify (i.e. both True Positives or True Negatives) out of all the tumors? 99%
- What is the chance that a tumor identified b the system as malignant is *in fact* malignant"? 9%

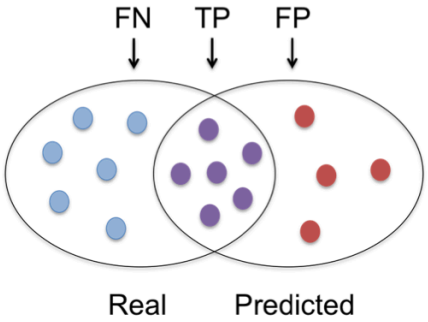
imbalanced dataset



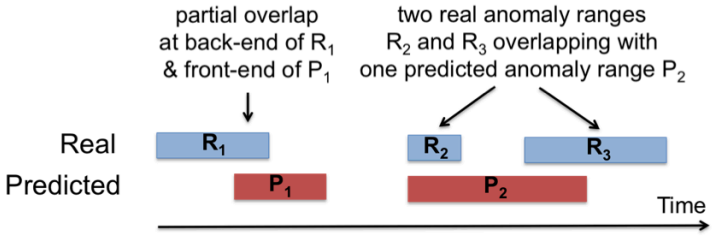
Importance of Context

- Prioritizing Precision vs Recall depends on problem context
- **Recall should be optimized over precision** when there is a **high cost associated with a False Negative, i.e. system predicts benign when tumour is in fact malignant**.
- **Precision should be optimized over recall** when there is a **high cost associated with a False Positive, i.e. spam detection.**

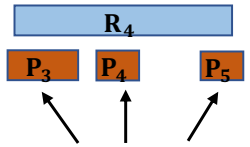
Precision and Recall in time-series data



(a) Precision = 0.6, Recall = 0.5



(b) Precision = ?, Recall = ?

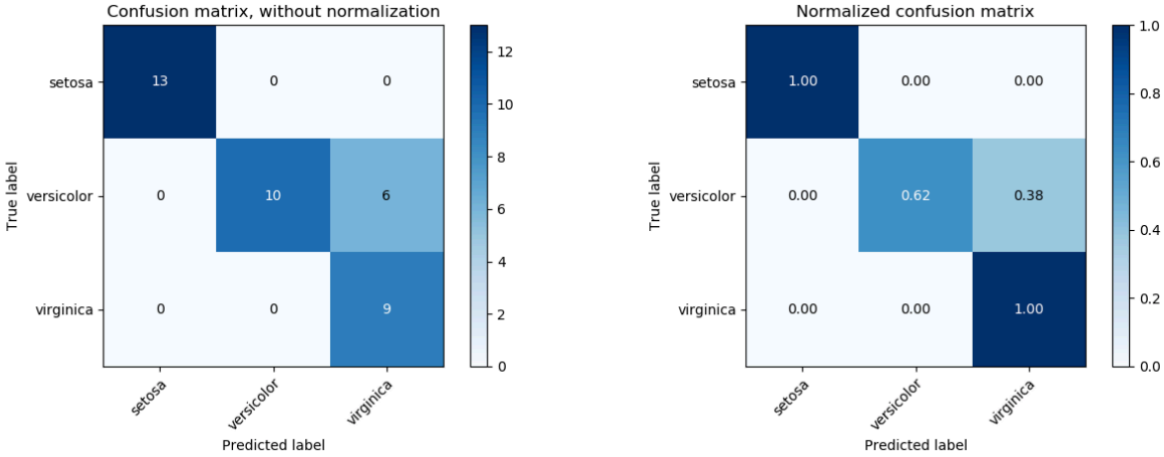


One real anomaly range $\mathbf{R_4}$ overlapping with three Predicted anomaly ranges $\mathbf{P_3}$, $\mathbf{P_4}$, and $\mathbf{P_5}$

Figure 1: Point-based vs. range-based anomalies.

- Precision and Recall for Time Series
- Range based recall and Range based prediction
- Range based metrics criteria
  - Catching existence of an anomaly is valuable
  - Size of correctly predicted portion; larger the size, higher the recall score
  - Relative position of correctly predicted portion might be important to application
  - Cardinality: detecting an anomaly range with single prediction range is more value

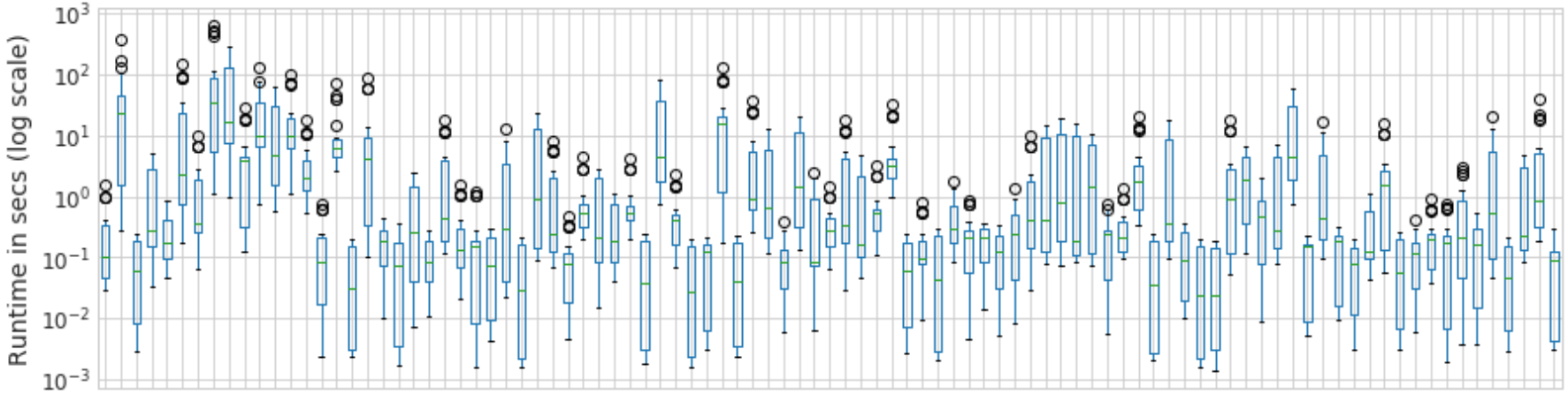
Confusion matrix



- 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

Runtime: ML

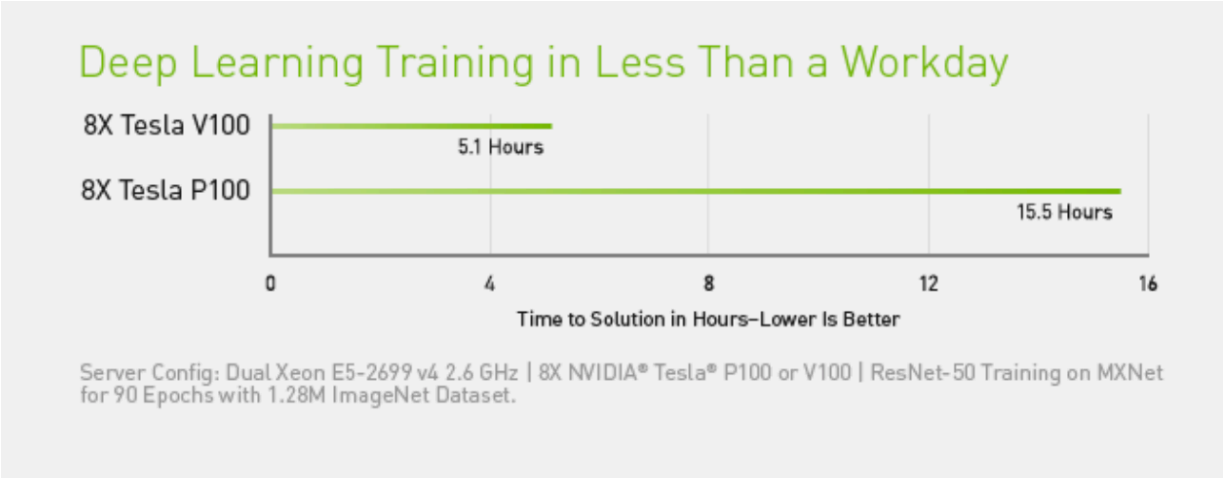
95 OpenML datasets, 7 classification algorithms, fit time



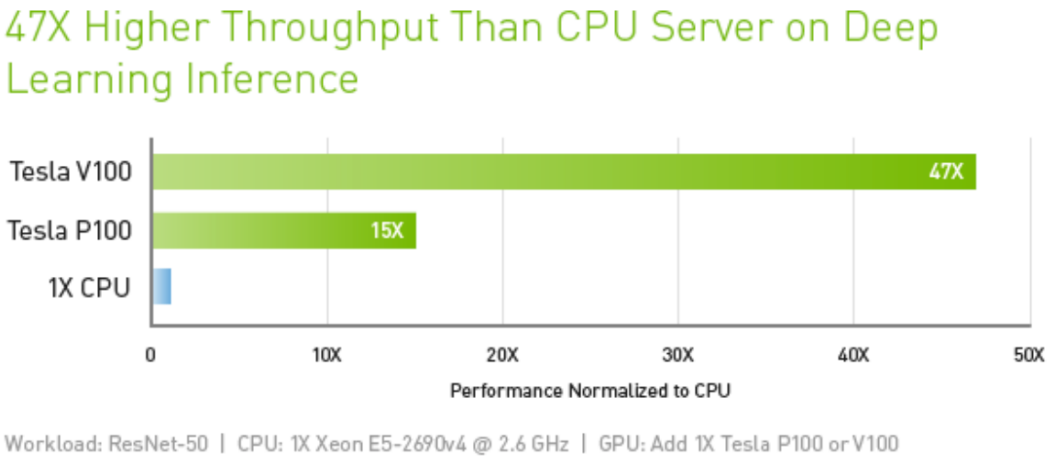
- For the same dataset, depending on the choice of classifier, the runtime can differ by orders of magnitude
- Runtime has both algorithmic, dataset, and system level dependency
- Runtime depends on how the model scales with different features of input: training data size, linear separability, complexity (number of output classes for classification problems), algorithmic hyperparameters, statistical meta-features of data (mean size per class, log number of features)

DL Training Time

- DL performance is closely tied to the hardware
  - compute power, memory, network
  - Tesla V100: 640 tensor cores (> 100 TFLOPS), 16 GB
  - NVIDIA NVLink: 300 GB/s



DL Inference Throughput

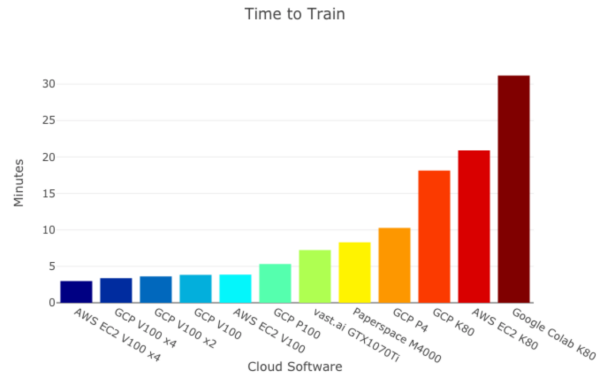


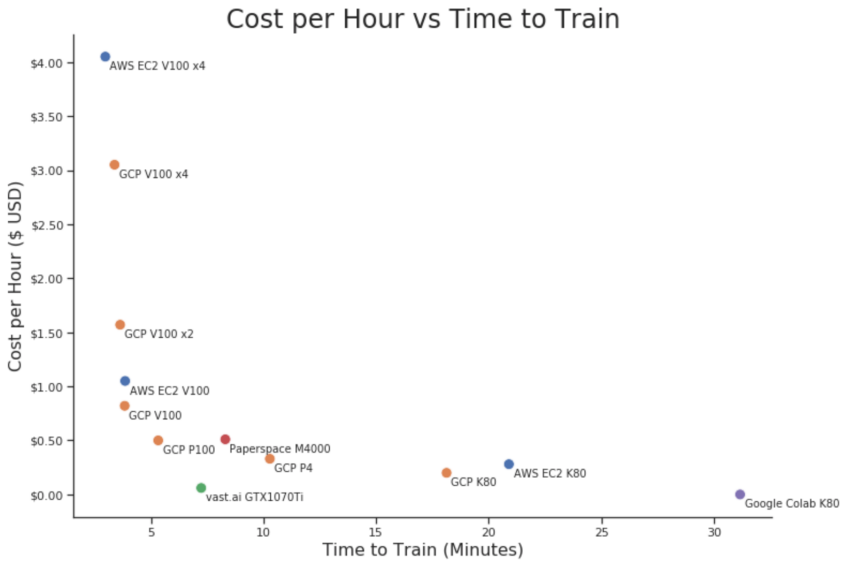
Training time and cost: An example

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

Training time and cost tradeoffs





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*

Multi-objective optimization and Pareto front

- Example: Control complexity of the model while minimizing MSE
- Control complexity:
  - Improve model Interpretability: In general, the lower the complexity, the easier it is to understand the model.
  - Prevent overfitting
- Scalarized multiobjective learning $\min f = E + \lambda\Omega$
- Pareto based multiobjective learning

$$\min\{f_1, f_2\}$$
$$f_1 = E$$
$$f_2 = \Omega.$$

$$\Omega = \sum_{i=1}^{M} w_i^2$$

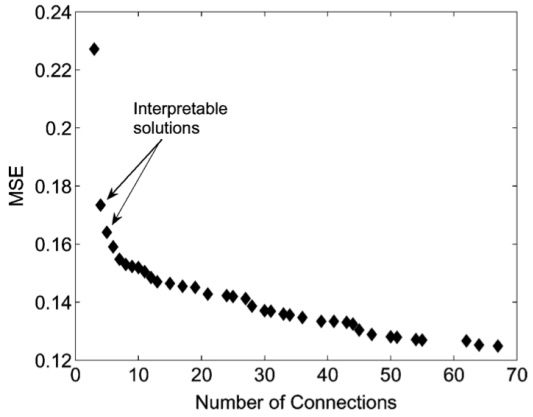


Fig. 9. Typical Pareto-front obtained for the diabetes data composed of 37 solutions.

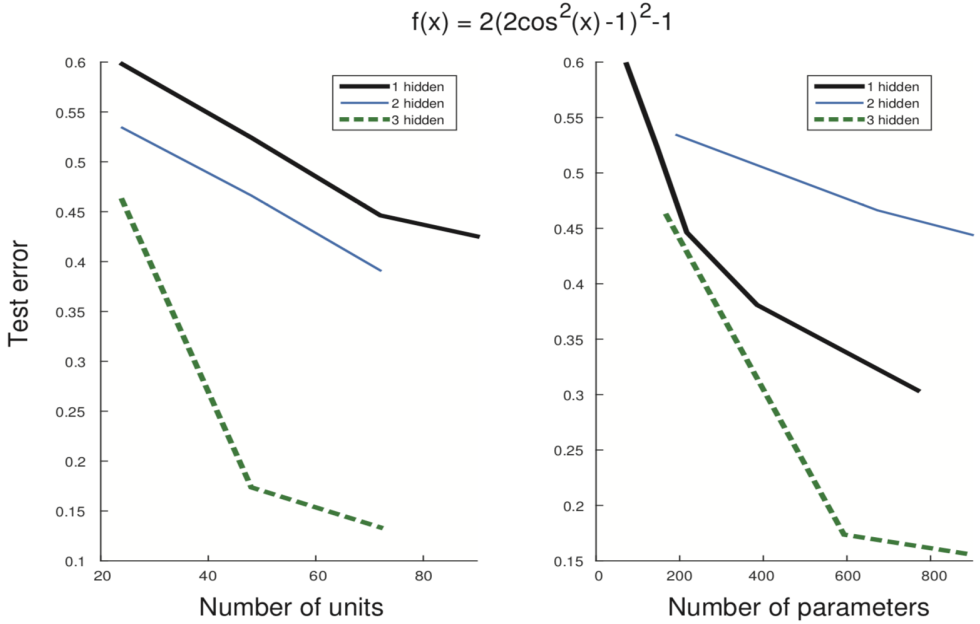
Universal Approximators Theorem

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

- Why need 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 lower number of parameters than shallow networks for approximating the same function

Easier to train with less data

Dataset Augmentation

- Artificially enlarge the training set by adding transformations/perturbations of the training data
- Domain-specific transformations
- Provides more training data
- Helps in model generalization and prevents overfitting
- 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

Dataset Augmentation Using Keras

- ImageDataGenerator class
- Keras examples: https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/

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

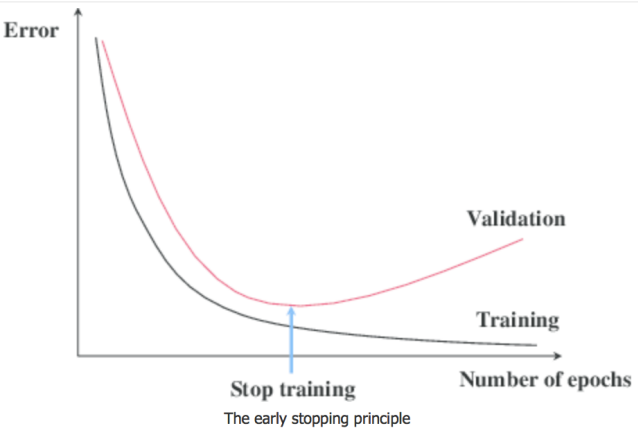
$$L = \sum_{(x,y)\in\mathcal{D}} (y - \hat{y})^2 + \lambda \cdot \sum_{i=0}^{d} |w_i|_1$$

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

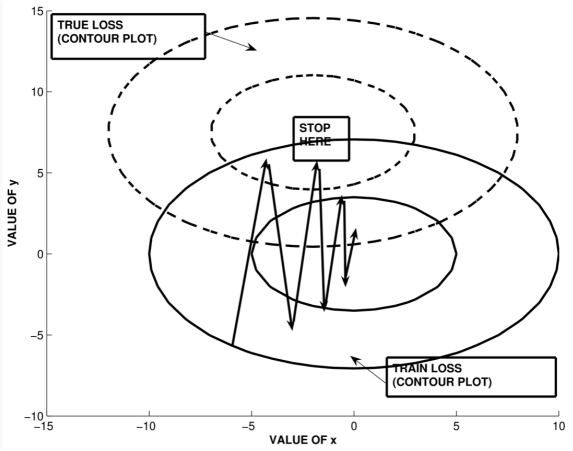
L_2 vs L_1 Regularization in Neural Network

- 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

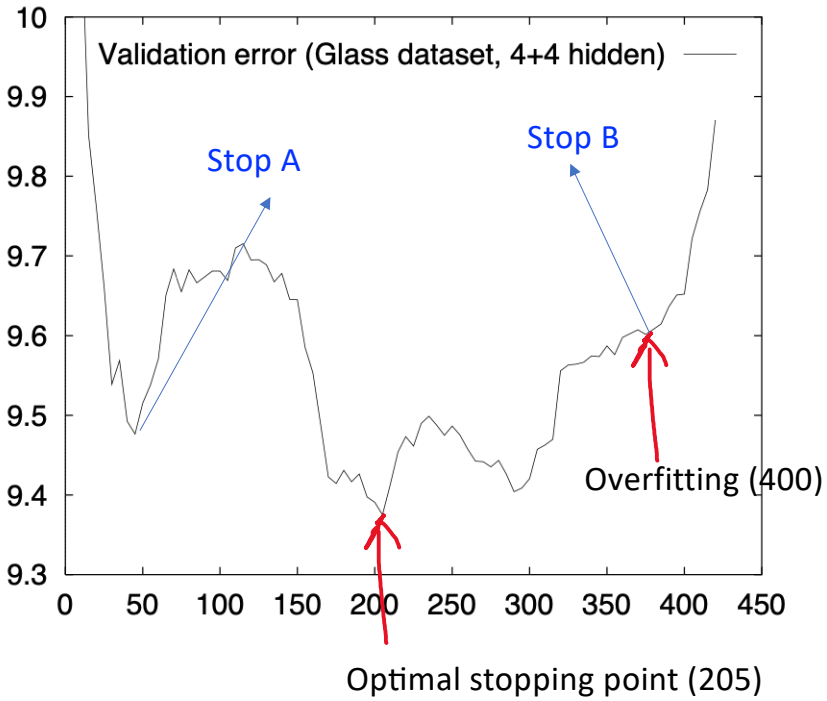


The early stopping principle



- 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
- $\boldsymbol{L_2}$ regularization can achieve similar or better performance than early stopping

Reality is Ugly



- 16 local minima before epoch 400
- 4 are global minimum up to where they occur
- Stop A: you get model with validation error > 9.5
- Stop B: you get model with validation error < 9.4
- Stop A to Stop B is ~7x increase in training time and
- 1.1% decrease in validation error

Examples Early Stopping Tests

- **Stopping criterion 1**: stop as soon as the generalization loss (GL) exceeds a certain threshold

$$GL(t) = 100 \cdot \left(\frac{E_{va}(t)}{E_{opt}(t)} - 1 \right)$$

Validation error till epoch t

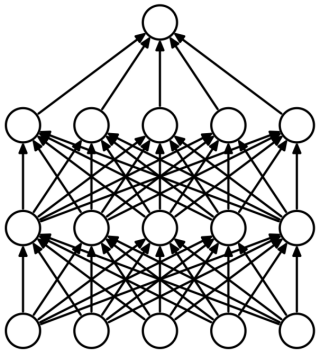
Minimum validation error till epoch t

stop after first epoch t with $GL(t) > \alpha$

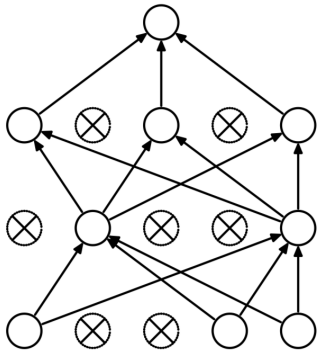
- **Stopping criterion 2**: suppress stopping if training is progressing rapidly
  - When training error decreases rapidly, generalization loss has higher chance of getting repaired
  - Overfitting usually begins when training error decreases slowly
  - Use the quotient of generalization loss and training progress
- **Stopping criterion 3**: stop when generalization error increased in s successive intervals

Dropout

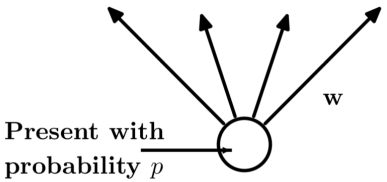
- Dropout is a regularization technique to deal with overfitting problem and improve generalization
- Prevents co-adaptation of activation units
- Probabilistically drop input features or activation units in hidden layers
- Layer dependent dropout probability (~0.2 for input, ~0.5 for hidden)



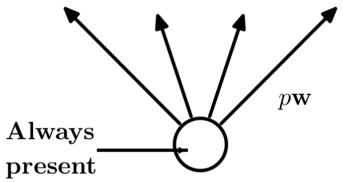
(a) Standard Neural Net



(b) After applying dropout.

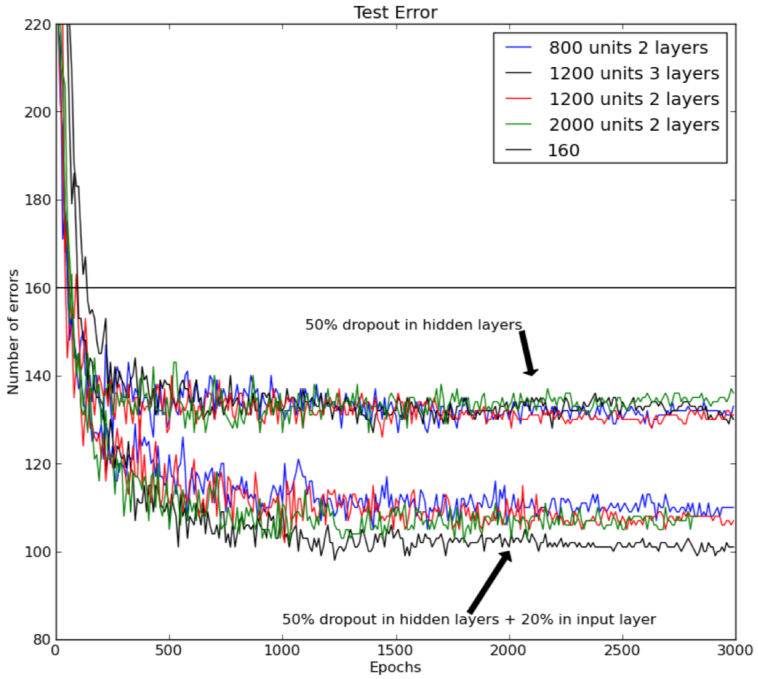


(a) At training time



(b) At test time

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/

Prepare for Lecture 3

- Access Jupyter Notebook
- Install Tensorflow and Keras and play with examples from Tensorflow
- You can use Google colab
- Work on Homework #1, due on 10/01/2020 by 11:55 PM; **NO LATE SUBMISSIONS ALLOWED**
- (optional) Read and Review “Reading List” and “Blogs/Code Links”
- Start thinking about seminar (team, topic, papers)
- Start thinking about project ideas and discuss with me

Reading List

- **Model Selection**
  - Sebastian Raschka. Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning. 2018
- **Practical ML**
  - Pedro Domingos, A few useful things to Know about machine Learning
- **Precision-Recall (papers also needed for Question 4 in Homework 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

Reading List

- 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
- Frank Hutter, Lin Xu, Holger H. Hoos, Kevin Leyton-Brown Algorithm Runtime Prediction: Methods & Evaluation
- Application Papers:
  - A prediction model of outcome of SARS-CoV-2 pneumonia based on laboratory findings

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 (Question 2 in Homework 1)

  https://dustinstansbury.github.io/theclevermachine/bias-variance-tradeoff

Blogs/Code Links

- Blog. The case against precision as a model selection criterion
- Blog. 4 things you need to know about AI: accuracy, precision, recall, and F1 scores
- Blog. Beyond Accuracy: Precision and Recall
- Early stopping in Tensorflow