Road Map

Deep Neural Networks

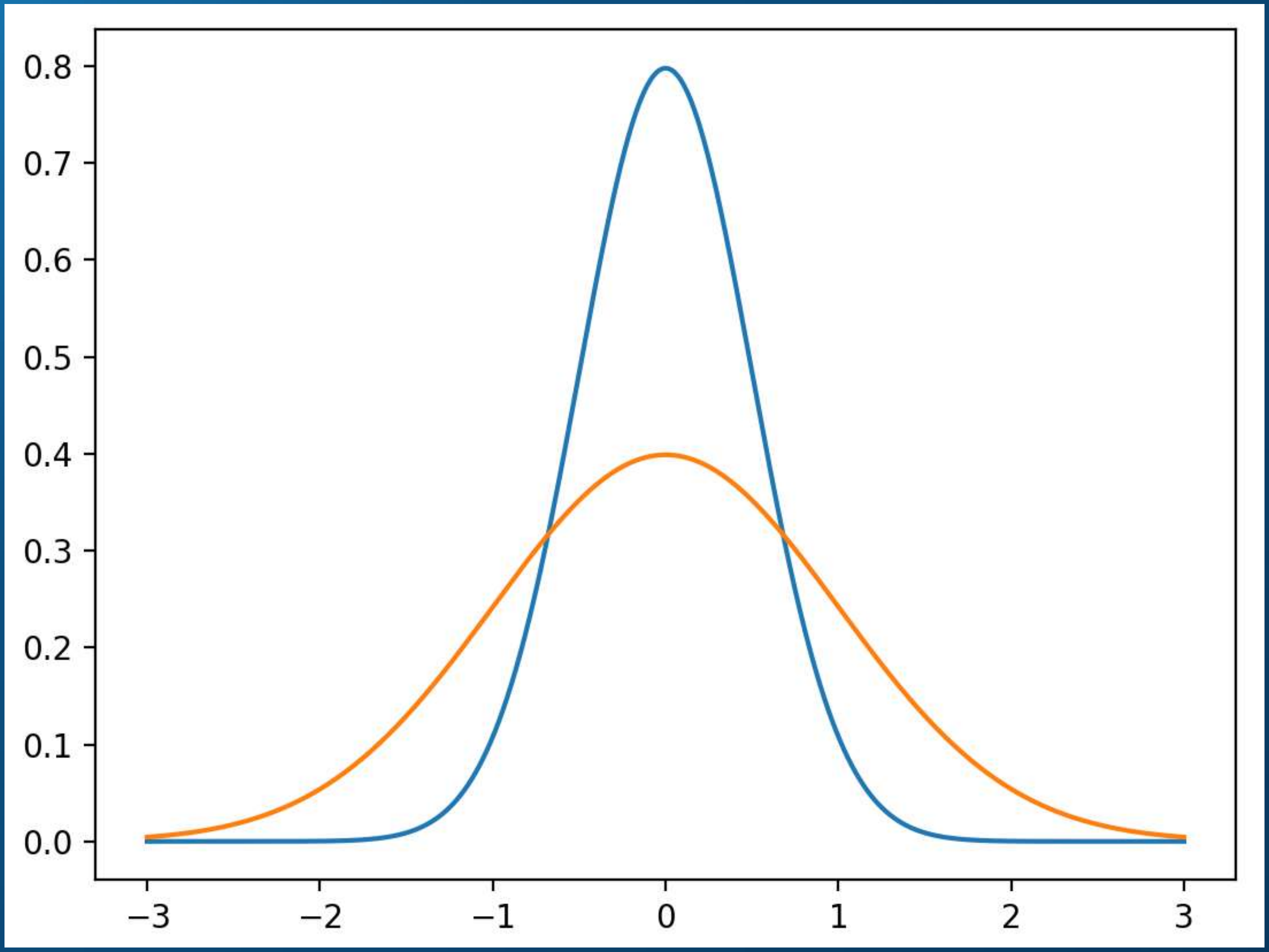
Session 12

Pramod Sharma
pramod.sharma@prasami.com

Agenda

- Standardization
- Training – Testing Split
- Bias and Variance
- Network Layout
- Parameter vs Hyperparameter

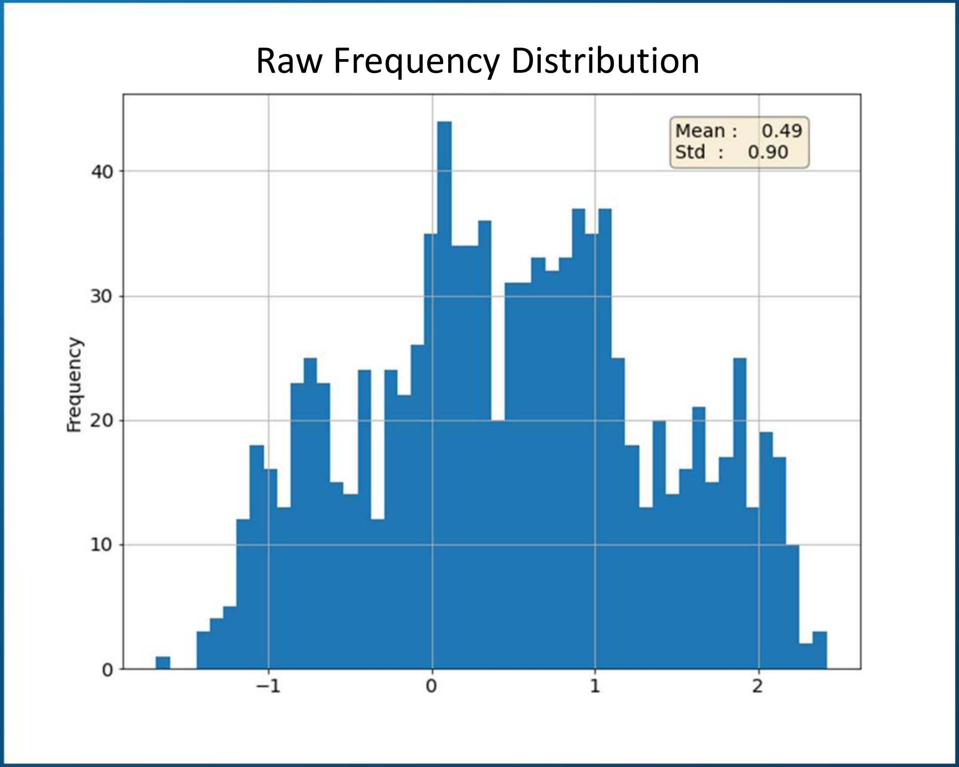
Normal Distribution

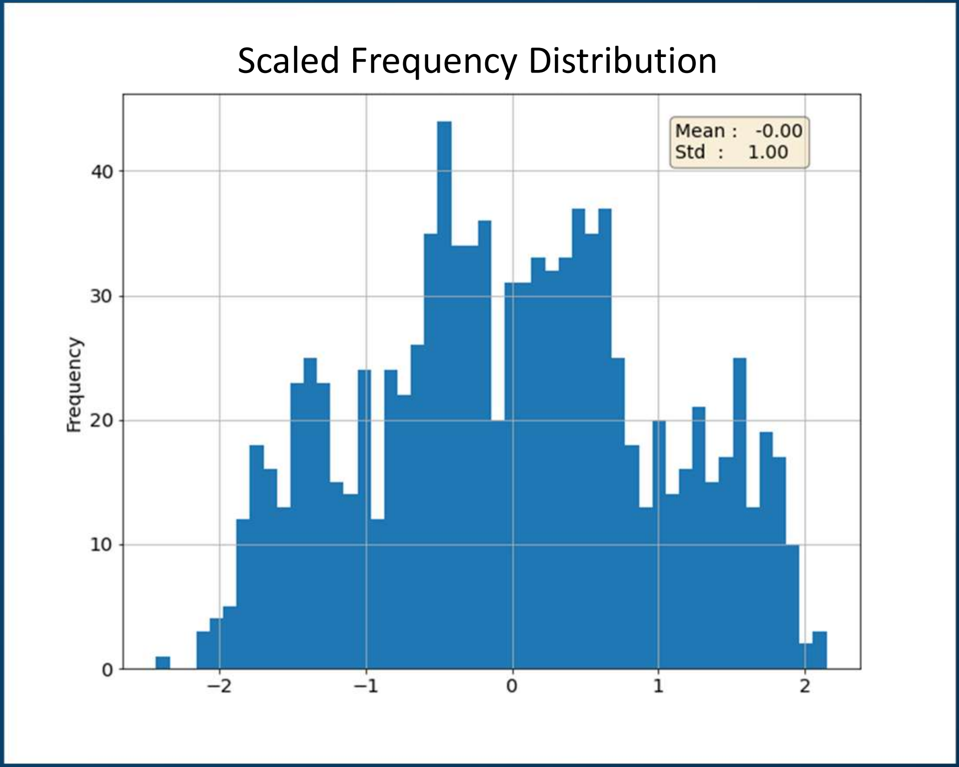


Standardization

- Standardization of datasets is a common requirement for many machine learning algorithms
- They might misbehave if the individual features do not, more or less, look like normally distributed data:
  - Gaussian with zero mean and unit variance
  - Sometimes also referred as “whitening”
- It is particularly needed where order of magnitude is different
- We can use libraries to standardize data
- At the very least, transform the data to center it by removing the mean value of each feature
- If possible give due importance to the shape of the distribution
- Many learning algorithms (such as l1 and l2 regularizers) assume that all features are centered around zero and have variance in the same order.
  - If a feature has a variance that has orders of magnitude larger than others, it might dominate the objective function

Standard Scaler



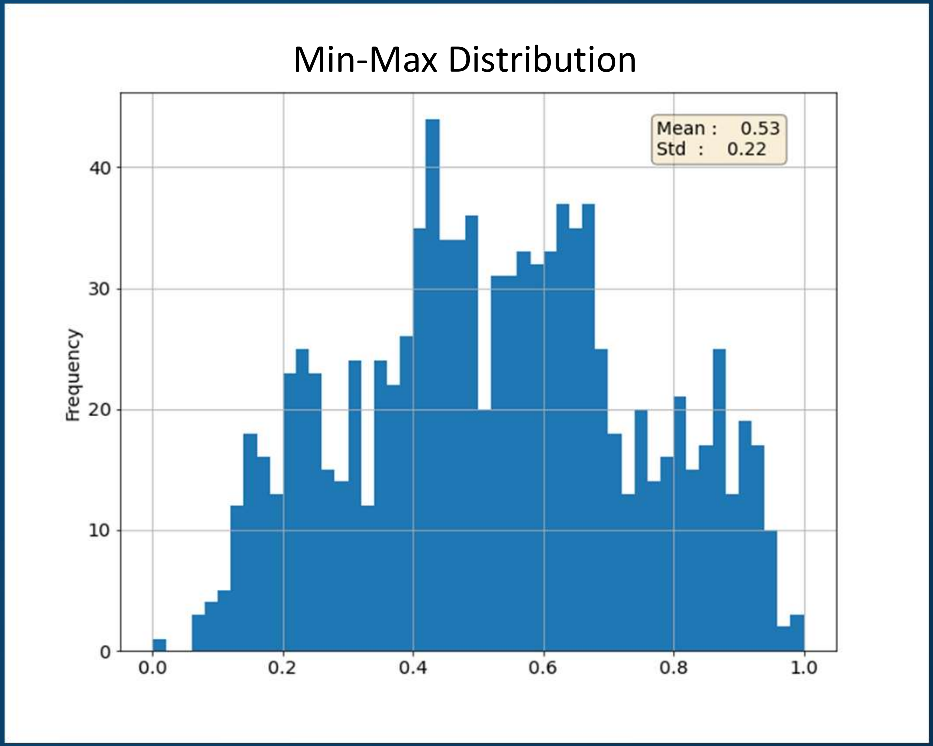


Min Max Scaler & Special Cases

- ❑ Scale to some known min and max values is also possible

- ❑ Sparse Data:
  - ❖ Not recommended for sparse data
  - ❖ Destroys the sparseness structure in the data
  - ❖ Think twice before applying standardization

- ❑ Outliers:
  - ❖ Data may contain outliers,
  - ❖ Not to confuse with boundary conditions
  - ❖ Some specialized scalers are available to handle data with outliers.
  - ❖ May need further parameter tuning.



Power Transformer

- Power transformers are a family of parametric, monotonic transformations
  - Aim to map data from any distribution to as close as possible to a Gaussian distribution
  - In order to stabilize variance and minimize skewness.

- The log transform is a specific example of a family of transformations known as power transforms. I
  - n statistical terms, these are variance-stabilizing transformations

- Two such power transformations:
  - Yeo-Johnson transform
  - Box-Cox transform

Power Transformer - Box-Cox

- Original thought was to apply strictly positive data

- A hyper-parameter, lambda (λ) is used to control the nature of the transformation
  - lambda (λ) varies from 5 to -5
    - lambda = -1 is a reciprocal transform.
    - lambda = -0.5 is a reciprocal square root transform.
    - lambda = 0.0 is a log transform.
    - lambda = 0.5 is a square root transform.
    - lambda = 1.0 is no transform.

$$y(\lambda) = \begin{cases} \frac{y^\lambda - 1}{\lambda}, & \text{if } \lambda \neq 0; \\ \log y, & \text{if } \lambda = 0. \end{cases}$$

- Box and Cox did propose a second formula that can be used for negative y-values:

$$y(\boldsymbol{\lambda}) = \begin{cases} \frac{(y+\lambda_2)^{\lambda_1} - 1}{\lambda_1}, & \text{if } \lambda_1 \neq 0; \\ \log(y + \lambda_2), & \text{if } \lambda_1 = 0. \end{cases}$$

Power Transformer - Yeo-Johnson transformation

- Yeo-Johnson propose the following transformation:

$$\psi(y,\lambda) = \begin{cases} \dfrac{(y+1)^\lambda - 1}{\lambda} & y \geq 0 \text{ and } \lambda \neq 0, \\ \log(y+1) & y \geq 0 \text{ and } \lambda = 0, \\ -\dfrac{(-y+1)^{2-\lambda} - 1}{2-\lambda} & y < 0 \text{ and } \lambda \neq 2, \\ -\log(-y+1) & y < 0, \lambda = 2. \end{cases}$$

  - ψ is concave in y for λ<1 and convex for λ> 1.
  - The constant shift of +1 makes is such that the transformed value will always have the same sign as the original value.
  - The constant shift of + 1 also allows to become the identity transformation when λ = 1.

- The new transformations on the positive line are equivalent to the Box-Cox transformation for (after accounting for the constant shift),
  - Yeo-Johnson transformation can be viewed as a generalization of the Box-Cox transformation

Training – Testing Split

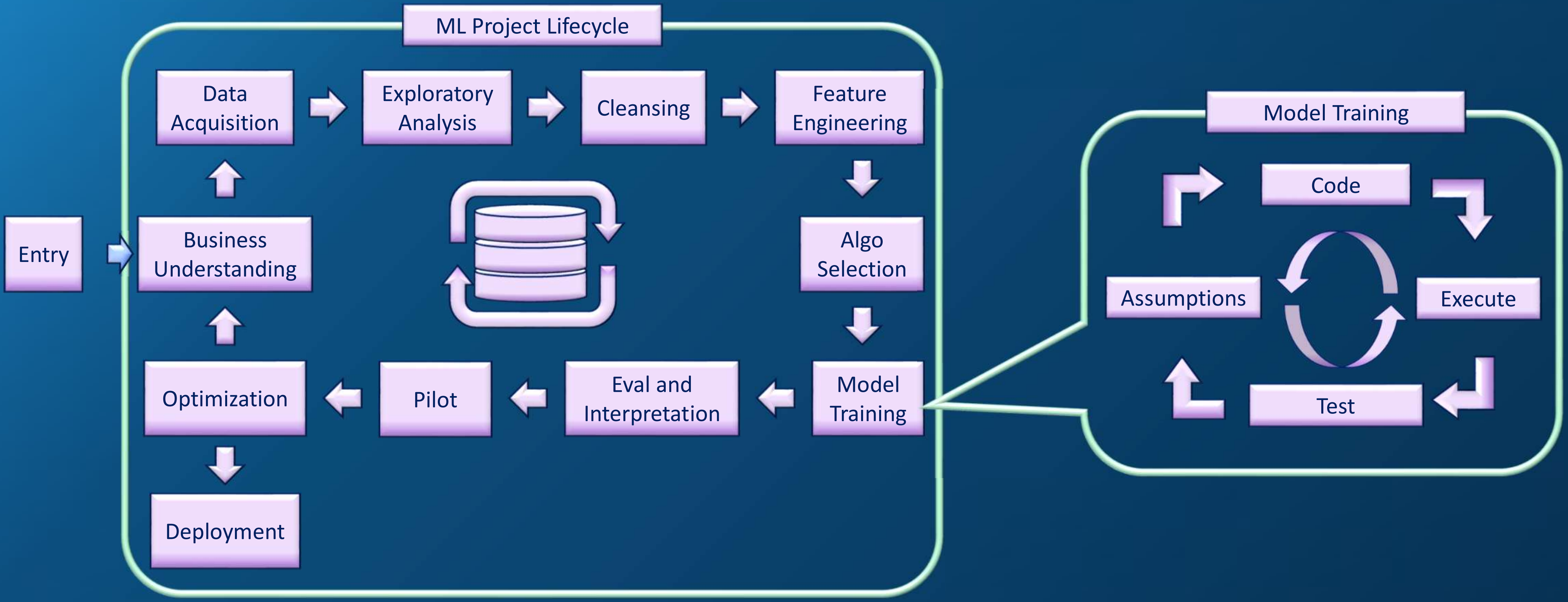
Data – Train – Test – Validate

- Researchers and Engineers are constantly working on making the models work better

- It becomes highly iterative process to find right mix of hyper parameter
  - Number of hidden layers
  - Number of hidden units in different layers
  - Learning rate – α
  - Momentum – β
  - Adam's hyper-parameter – β_1, β_2, ε
  - Learning rate decay
  - Mini-batch size and many more

There is no silver bullet!

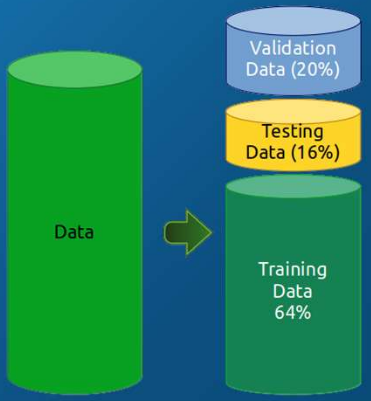
Project Lifecycle



Training Data and Test Data

- Same data cannot be used for Training and Testing
- Models give better results on seen data,
- Given huge number of epoch, it may even give perfect score
  - But would fail to predict anything useful on yet-unseen data
- In supervised machine learning hold out part of the available data as a test set and another part as validation set
- Train and tune the model on train + test set,
- Once tuning is complete, train the model on train + test data and validate on validation data
  - Remove random seed!
- Typical cross validation and grid search techniques are also clubbed in this process.
  - Be very careful not to leak information from test set to train set specially in cross validation
- Time series: No peeping into future

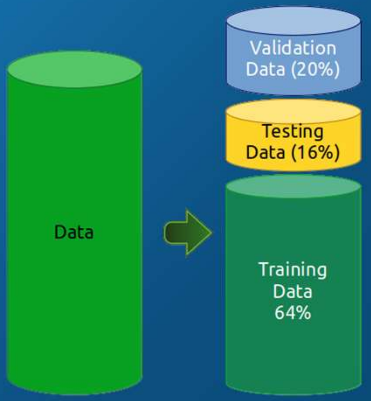
Data Split



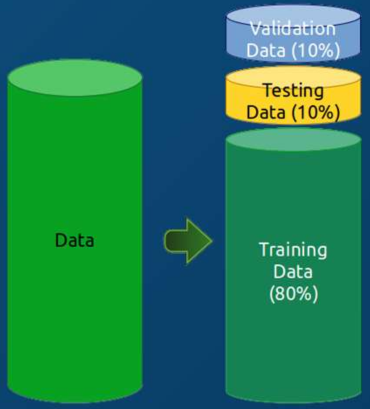
Up to 10,000 records

pra-sâmi

Data Split

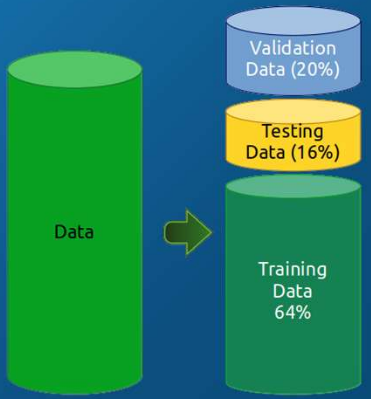


Up to 10,000 records

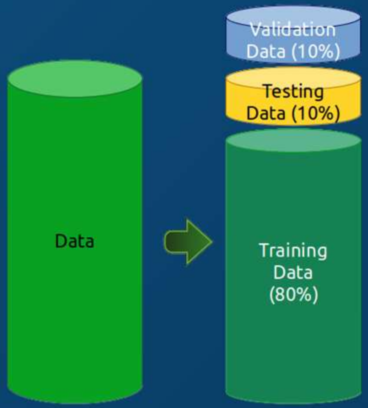


From 10,000 to 1 Million

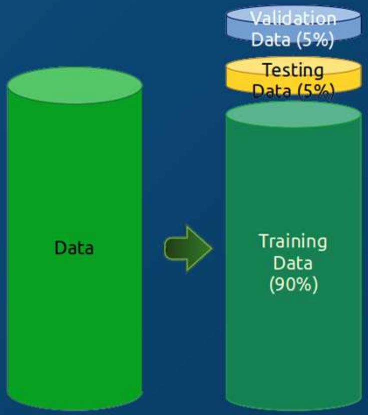
Data Split



Up to 10,000 records



From 10,000 to 1 Million



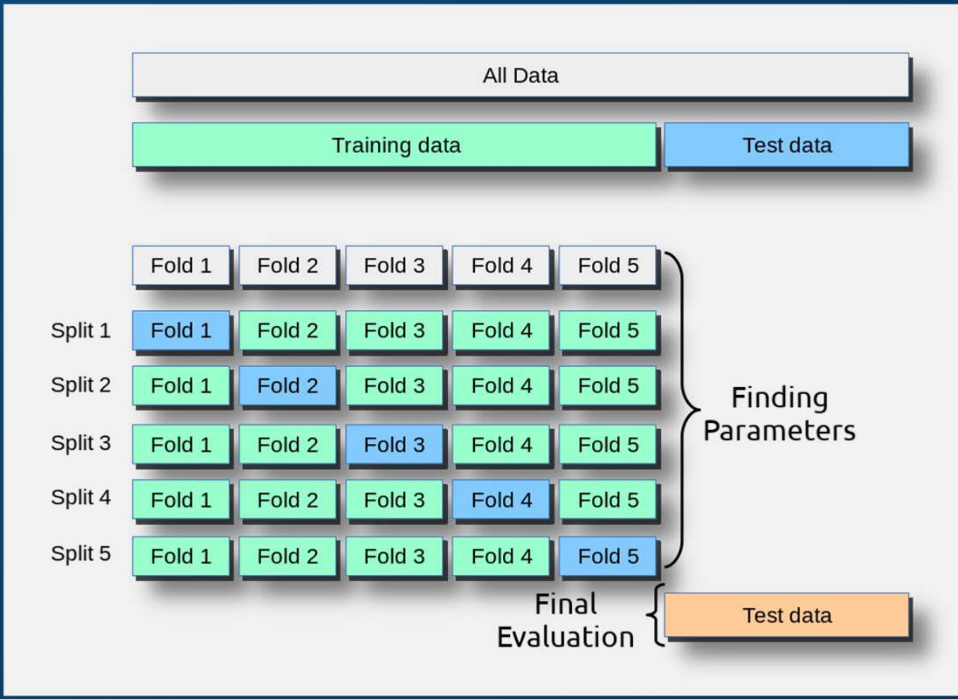
Above 1 Million

Matching Distribution of Test and Train Data Sets

- Data Scientists are always looking for data
- There are plenty of curated datasets available
- Curated data sets are clean with sufficient features to explain a concept,
  - May not be suitable for real life application
  - Real data is dirty
- Example:
  - For object identification or image analytics, it is very common to perform web search and download picture
  - These pictures are of good resolution, taken under favorable lighting condition
  - After rollout, pictures taken by actual user may not be of similar quality
  - Resulting in significant drop of accuracy on real data

Cross-validation : Source Sklearn

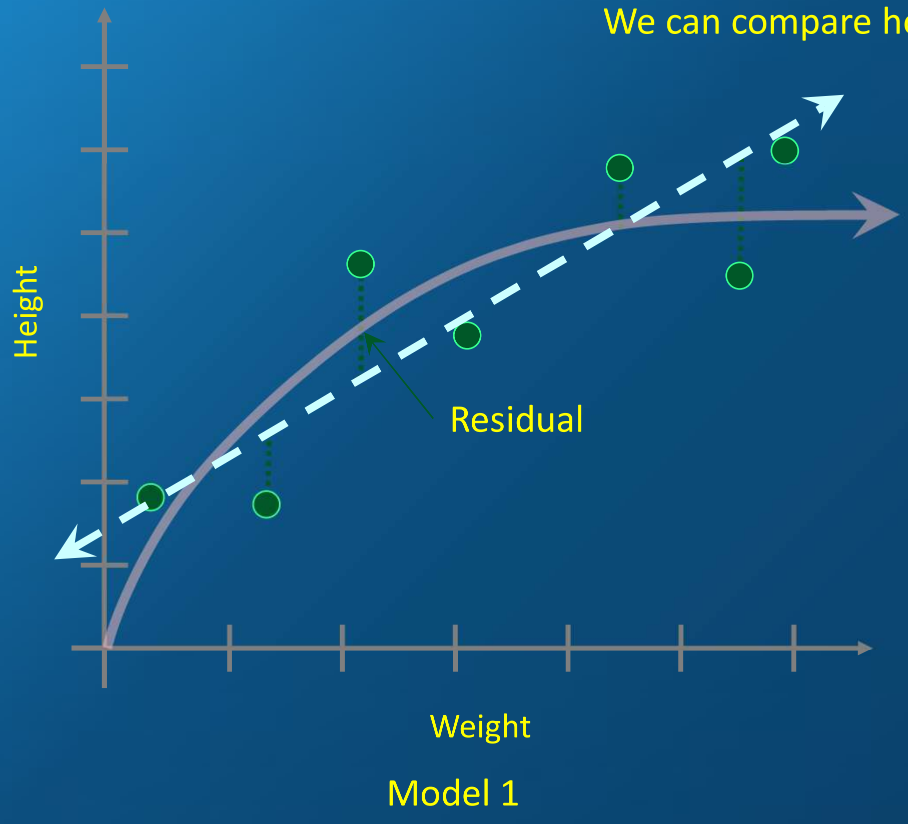
- Partitioning the available data into three sets, drastically reduce the number of samples in each bucket
- Use cross validation to solve it.
  - The training set is split into k smaller sets
- Model is trained using multiple folds as training data;
  - By rotation one fold is kept out
  - Performance measure is average of the values in the loop
- Approach is computationally expensive
  - Recommended where data is extremely limited
  - No data augmentation can be applied

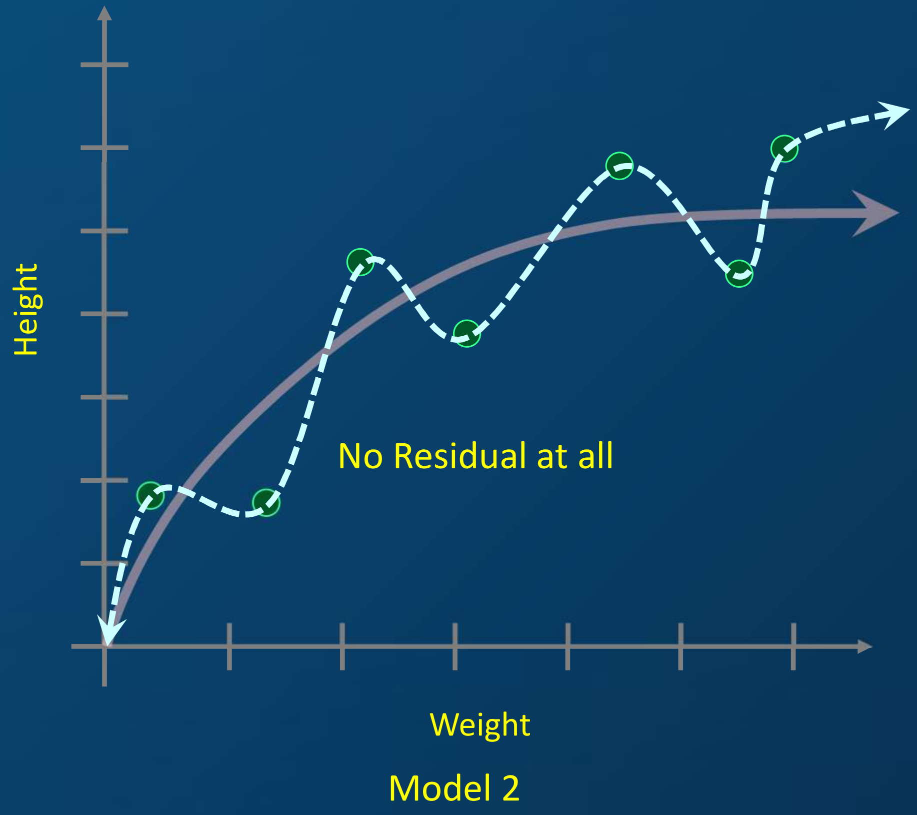


Bias and Variance

Comparison

We can compare how well two models are fitting **Training Data**

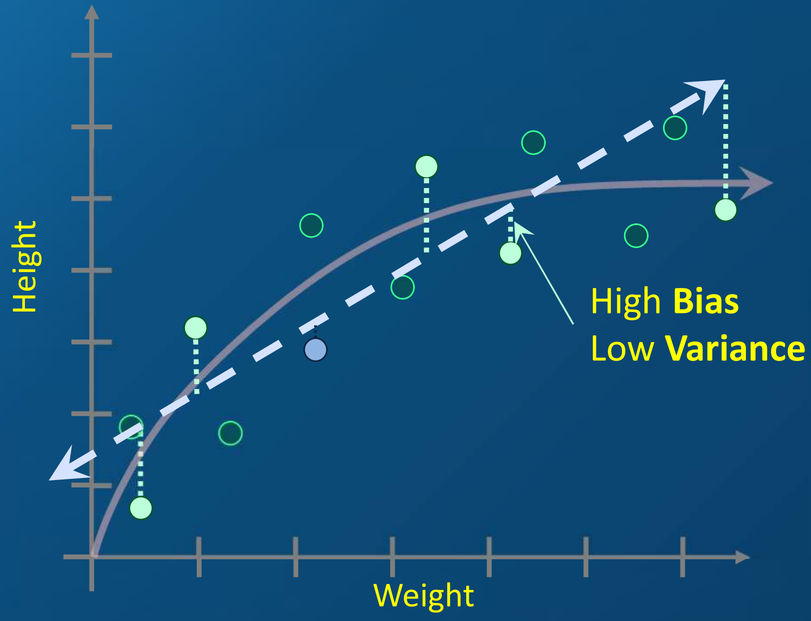




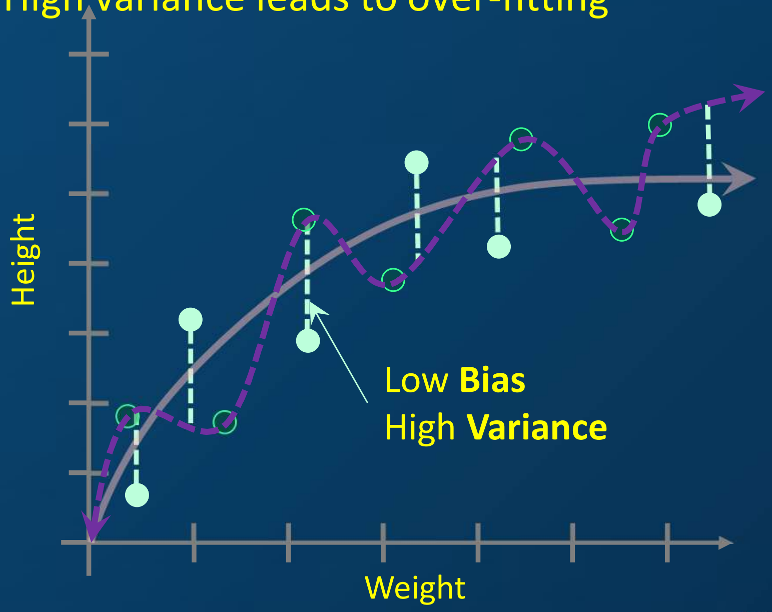
Complex model does a good job at Training!

Bias vs Variance

- "Bias" : "Expected" difference between estimated value and its true value
- Low bias models that are accurate on average, but inconsistent
- High bias leads to under-fitting



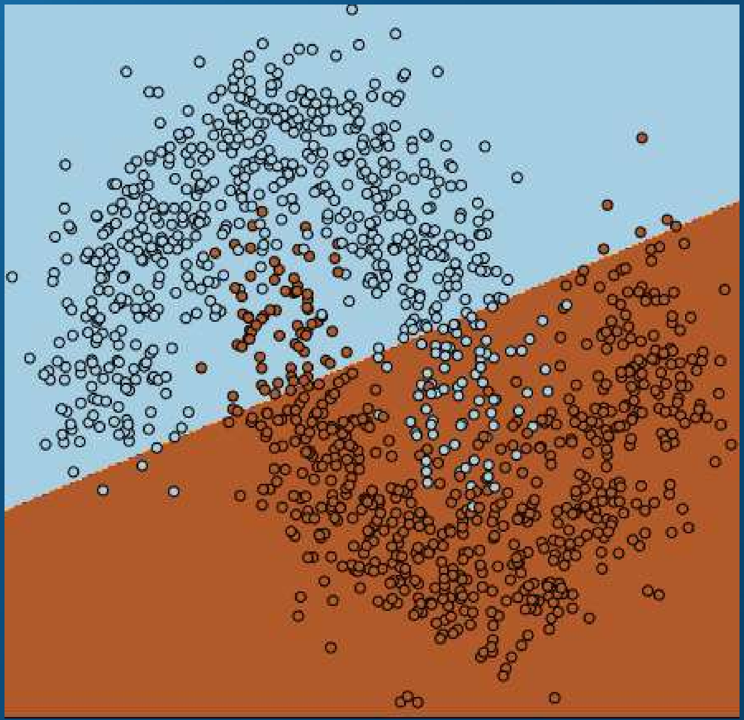
- "Variance" : "Expected" value of the squared difference between the estimate of a model and the "expected" value of the estimate
- Low Variance Models are consistent, but inaccurate on average
- High variance leads to over-fitting



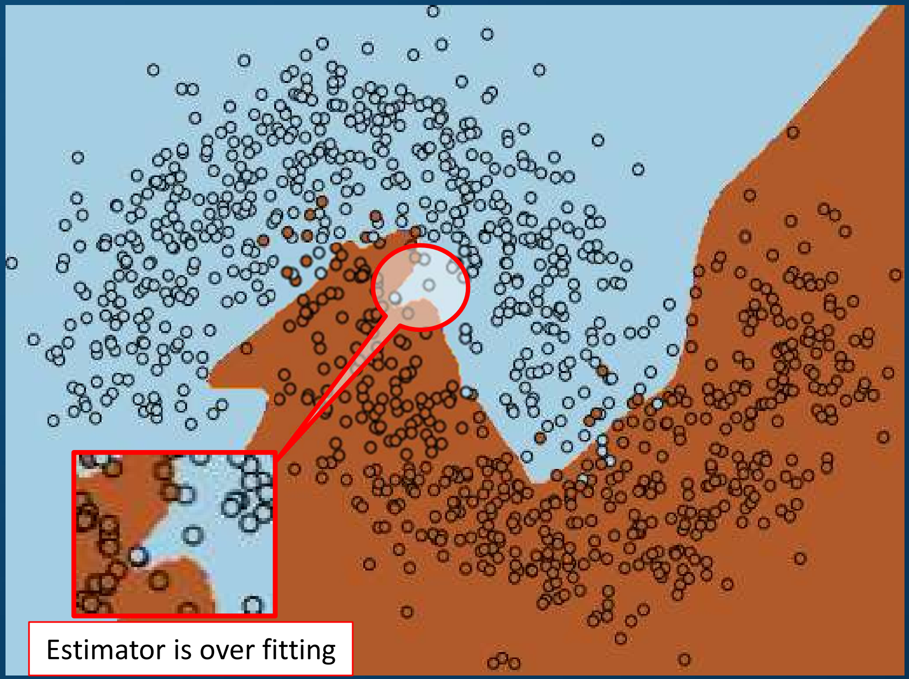
pra-sâmi

Bias Vs. Variance

High Bias : solution is too generic!



High Variance : solution is too specific!



The Bias-Variance Trade-off

- Bias and Variance were complimentary
  - Old school… not strictly applicable for deep networks
- Errors could be separated in Bias and Variance
- Model with high Variance are expected to have low Bias and vice versa
- Less complex algorithms give low Variance
  - Regression, Naïve Bayes,
- Low Bias algorithms are more complex
  - Decision trees, Nearest Neighbors, Random Forest
- Within each of the family there is a tradeoff
- Deep learning has changed it…
  - It is possible to address bias and variance separately…

Network Layout

Deep Networks

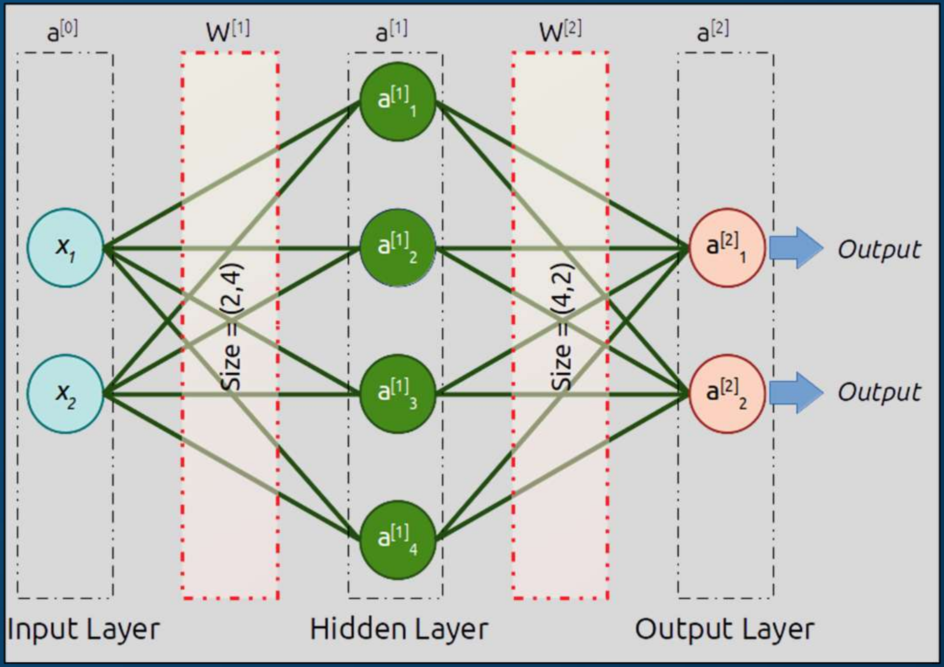
High Bias	High Variance
❑ Assemble a bigger network	❑ Prepare separate testing and validation sets apart from training set
❑ Increase number of cycles (epochs)	❑ Add regularization, dropouts or a combination thereof
❑ Try different architecture	❑ And try different architecture
	❑ Add more data

pra-sâmi

Network Layout

- The Input Layer
  - Every NN has exactly one of them--no exceptions
  - Number of neurons is equal to the number of features input features
  - Some NN configurations add one additional node for a bias term (Very few and far in between)

- The Output Layer
  - Like the Input layer, every NN has exactly one output layer
  - For regressor - the output layer has a single node
  - For classifier - has a single node unless softmax is used in which case the output layer has one node per class label
  - When estimating bounding boxes, one node per bounding box.



Network Layout

- The Hidden Layers
  - Single hidden layer gives reasonably accurate results in most cases
  - Performance improves with additional layer
  - Generally, upto five hidden layers are enough for ANN.
  - Specialized cases need special networks
  - Read http://www.faqs.org/faqs/ai-faq/neural-nets/part1/preamble.html

- Number of Neurons in Hidden layer
  - Too few results in under-fitting and too many lead to over fitting
  - Start with same numbers as input layer
  - Taper off to a number same as output layer
  - Try both liners (10/8/6/4) as well as geometric (32/16/8/4)

No silver bullet solution…
Read, iterate, experiment…

Number of Neurons – Rule of Thumb

- ❑ $N_h = \frac{N_s}{(\alpha*(Ni+N\ \ ))}$

This formula is obsolete — it comes from 1980s/1990s MLP heuristic

  - ❖ N_i = number of input neurons.
  - ❖ N_o= number of output neurons.
  - ❖ N_s = number of samples in training data set.
  - ❖ α = an arbitrary scaling factor usually 2-10.

- ❑ Capacity is now decided through architecture design, regularization, and compute limits

- ❑ Deep stacks (ResNet, Transformers) with sizes not related to input/output

Selection is iterative process.
These thumb rules will give you good starting point.
Monitor loss and optimize for best option

Number of Neurons – Modern Version

- No fixed formula
- Choose neurons based on problem complexity and architecture patterns, not fixed rules.
  - Start with widths like 64–512 for MLPs and tune based on validation performance.
- Larger networks often generalize better when combined with:
  - Dropout
  - Batch normalization
  - Regularization
  - Early stopping
- Use validation data to determine if the model is underfitting (too few neurons) or overfitting (too many without regularization).

Parameter or Hyperparameter

Parameters or Hyper -parameters

- Very confusing!
- Why Two separate names?
- Both parameters as well as hyper-parameters… we find the right value (?)...

Parameters or Hyper -parameters

- Are they parameters or hyper-parameters
  - Weights
  - Biases
  - Learning rates
  - Epoch
  - Number of hidden layers
  - Number of hidden units
  - Activation functions
  - Gradient descent function
- Some of the literature and research papers use them interchangeably
- We **train** Parameters and **tune** Hyper-parameters

pra-sâmi

Parameters or Hyper-parameters

- Hyper-parameter is a parameter whose value is used to control the learning process
- Weights and biases are derived via training
- Hyper-parameters cannot be inferred while fitting
- Hyper-parameters, in principle, have no influence on the performance of the model
  - But affect the speed and quality of the learning process

pra-sâmi

Parameters or Hyper-parameters

- Parameters:
  - Weights, biases : $W^{[1]}$, $b^{[1]}$, $W^{[2]}$, $b^{[2]}$, $W^{[3]}$, $b^{[3]}$…

- Hyper-parameters:
  - Everything else can be considered as hyper-parameters
  - Learning rate α
  - Epoch,
  - Number of hidden layers
  - Number of hidden units
  - Activation functions
  - Gradient descent functions
  - Momentum
  - Mini batch size
  - Regularizations

Parameters or Hyper-parameters

- Broad definition: hyper-parameters control values of Weights and biases

- What is a Model Parameter?
  - A model parameter is a configuration variable that is internal to the model and whose value can be estimated from data
  - Are required by the model when making predictions
  - Their values define the skill of the model on the problem
  - They are estimated or learned from data
  - They are often not set manually by the practitioner
  - They are often saved as part of the learned model

Very iterative process.
For every problem, try various hyper- parameters and see what works.
Even if one parameter has worked, over period it will change

Broad Guidelines

- Learning rate – α Try various values
- Activations
- Mini batch size
- Hidden units
- Number of layers can also make lot of difference
- Optimizer related parameters
- These are guidelines and not rules
  - Different domains may have different order
- Some parameters are more important than others

Grid Search vs Random Search!

1	0	0	1	0	0	1	0	0	1	0	0
0	1	0	0	1	0	0	1	0	0	1	0
0	0	1	0	0	1	0	0	1	0	0	1
1	0	0	1	0	0	1	0	0	1	0	0
0	1	0	0	1	0	0	1	0	0	1	0
0	0	1	0	0	1	0	0	1	0	0	1

1	0	0	1	0	0	1	0	0	1	0	0
0	1	0	0	1	0	0	1	0	0	1	0
0	0	1	0	0	1	0	0	1	0	0	1
1	0	0	1	0	0	1	0	0	1	0	0
0	1	0	0	1	0	0	1	0	0	1	0
0	0	1	0	0	1	0	0	1	0	0	1

Don’t Let The Regularization Overwhelm The Data

- Many losses are composed of two terms:
  - $L = L_{data} + \lambda L_{reg}$
  - where L_{reg} often L2 weight decay.

- If the regularization strength λ is too large, the gradient from the regularization term can dominate the gradient from the data loss.

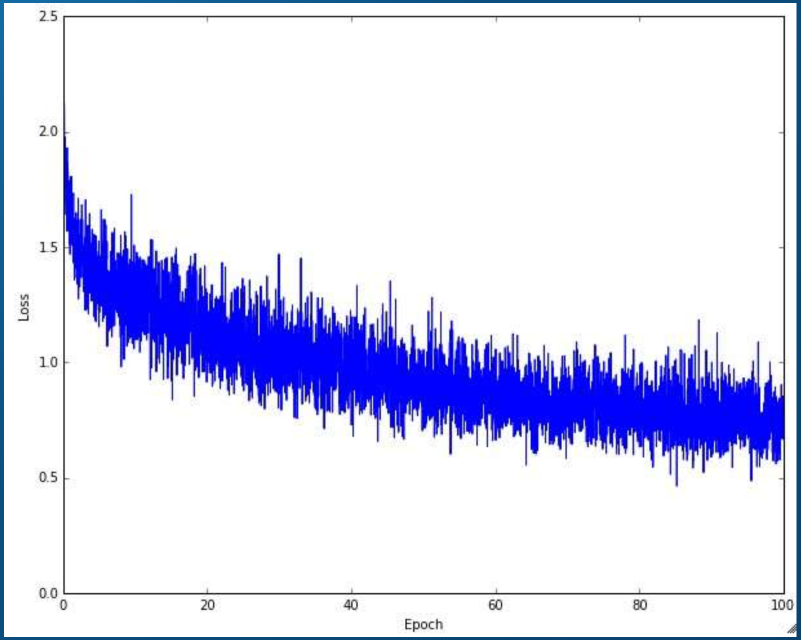
- Since regularization gradients (e.g., λw) are simple and predictable, they can mask bugs in the implementation of the data-loss gradient during gradient checking.

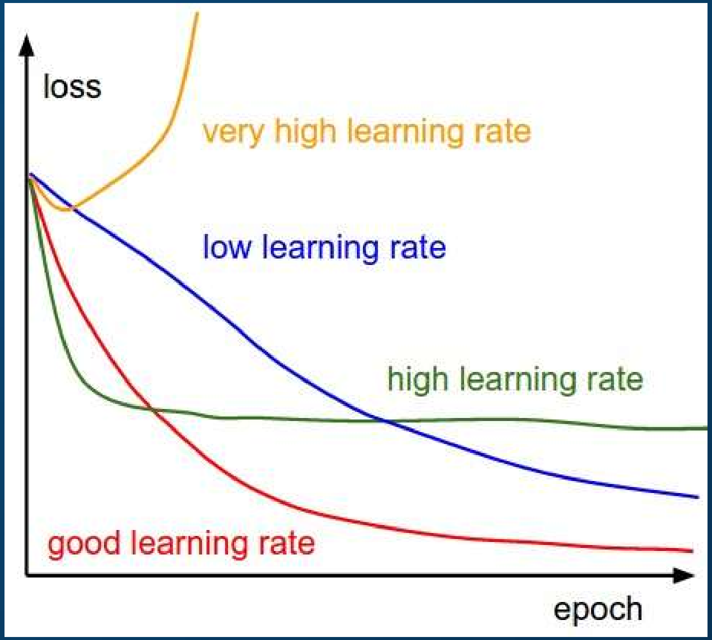
- Best Practices
  - Turn off regularization (λ = 0) and verify the data-loss gradient alone.
  - Test the regularization term separately to ensure its gradient is correct.
  - During gradient checking:
    - Set λ = 0 to isolate data loss, or
    - Set λ very large temporarily to ensure the regularization gradient is noticeable.

- Reason
  - Incorrect data-loss gradients can go unnoticed if regularization dominates
  - Proper checking ensures both terms are implemented correctly before training

Monitor Loss Function

- Evaluated on the individual batches during the forward pass.
- Following diagram showing the loss over time, and especially what the shape might tell you about the learning rate:





Reflect…

- What does the term "learning rate" refer to?
  - A. The speed at which the network processes input data
  - B. The step size used to update model parameters during training
  - C. The number of layers in the neural network
  - D. The size of the input data

- Ans B - The step size used to update model parameters during training

- What is the purpose of regularization techniques?
  - A. To increase model complexity
  - B. To reduce overfitting
  - C. To speed up training time
  - D. To eliminate bias in the model

- Ans B - To reduce overfitting

- Which activation function is commonly used in the hidden layers of deep neural networks to address the vanishing gradient problem?
  - A. Sigmoid
  - B. ReLU (Rectified Linear Unit)
  - C. Tanh (Hyperbolic Tangent)
  - D. Softmax

- Ans B - ReLU (Rectified Linear Unit)

- What is the role of batch normalization?
  - A. It normalizes the output of each layer to improve convergence.
  - B. It reduces the size of the input data to speed up training.
  - C. It initializes the weights of the network.
  - D. It regularizes the model by adding noise to the input.

- Ans A - It normalizes the output of each layer to improve convergence.

Reflect…

- What is dropout used for?
  A. To drop layers with low relevance
  B. To randomly drop neurons during training to prevent overfitting
  C. To speed up the convergence of the model
  D. To increase the capacity of the model

- B - To randomly drop neurons during training to prevent overfitting

- What is the purpose of hyperparameter tuning in DNN?
  A. To adjust the learning rate during training
  B. To optimize the architecture and configuration of the neural network
  C. To select the appropriate activation function
  D. To preprocess the input data

- B - To optimize the architecture and configuration of the neural network

- What is the vanishing gradient problem and how does it affect training?
  A. It causes the model to converge too quickly.
  B. It slows down the convergence of the model.
  C. It leads to exploding gradients.
  D. It prevents early layers from learning due to gradients becoming extremely small.

- D - It prevents early layers from learning due to gradients becoming extremely small.

- What is the role of the Adam optimization algorithm?
  A. It initializes the weights of the network.
  B. It regularizes the model by adding noise to the input.
  C. It adjusts the learning rate during training.
  D. It adapts the learning rates of individual parameters.

- D - It adapts the learning rates of individual parameters.

Reflect…

- What is the purpose of data augmentation?
  A. To increase the size of the training dataset
  B. To reduce the dimensionality of the input data
  C. To normalize the input data
  D. To add noise to the output of each layer

- A - To increase the size of the training dataset

- In transfer learning, what is the role of fine-tuning?
  A. Fine-tuning adjusts the learning rate during training.
  B. Fine-tuning unfreezes some or all pre-trained layers and continues training them on new data..
  C. Fine-tuning initializes the weights of the network.
  D. Fine-tuning adds noise to the input data.

- B - Fine-tuning unfreezes some or all pre-trained layers and continues training them on new data.

- What is the concept of early stopping in the training of DNN?
  A. It stops the training process if the learning rate is too high.
  B. It stops the training process once the model achieves 100% accuracy.
  C. It stops the training process if the validation loss stops decreasing.
  D. It stops the training process after a fixed number of epochs.

- C - It stops the training process if the validation loss stops decreasing.

- How does the choice of weight initialization impact the training of DNN?
  A. It has no effect on training.
  B. It affects the convergence of the model.
  C. It determines the learning rate.
  D. It controls the activation function used in the network.

- B - It affects the convergence of the model.

Next session… Batch Normalization

THANK YOU

ADDITIONAL MATERIAL

What’s Next

pra-sâmi

Debugging a learning algorithm

- Suppose we have built a model and its Cost is J (W, b)

- However, when you test your hypothesis on a new set of data, you find that it makes unacceptably large errors in its predictions. What should you try next?
  - Get more training examples
  - Try smaller sets of features
  - Try getting additional features
  - Try adding derived features (square terms)
  - Try decreasing λ
  - Try increasing λ

Fails to generalize to new examples not in training set….

Size	Price	Set
2104	400	Train
1600	330	
2400	369	
1416	232	
3000	540	
1985	300	
1534	315	
1427	199	Test
1380	212	
1494	243	

Under fitting / Just Right / Over fitting

Training/testing procedure

- Keep an eye on the overall cost… BIG Time

- Always keep 3 separate buckets
  - Training (64 %)
  - Testing (16 %)
  - Validation (20 %)

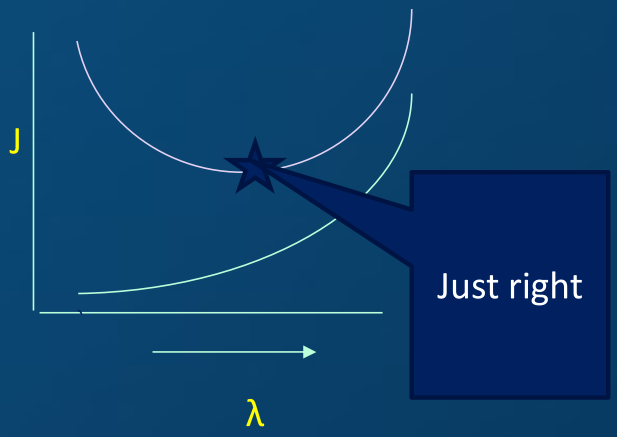
Reach overfitting stage and regularize

- ❑ First train the model so that it starts overfitting
- ❑ Training performance far better than test

- ❑ Then Regularize => Dropout, L2, early stopping…

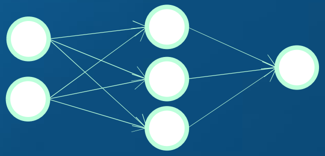
Choosing the regularization parameter λ

- ❑ λ = 0
- ❑ λ = 0.01
- ❑ λ = 0.02
- ❑ λ = 0.03
- ❑ λ = 0.04
- ❑ λ = 0.05
- ❑ ….
- ❑ λ = 10   (! why not….)



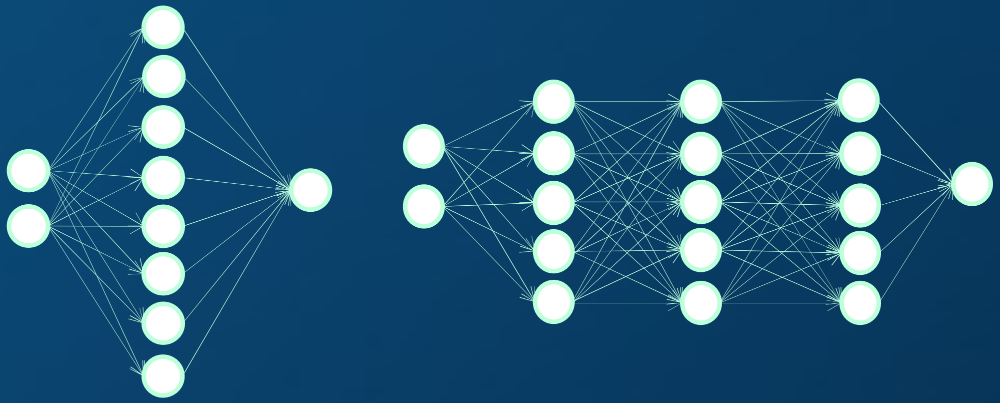
Neural networks and overfitting

- "Small" neural network (fewer parameters; more prone to underfitting)



- Computationally cheaper

- Large" neural network (more parameters; more prone to overfitting)



- Computationally more expensive.
- Use regularization (λ) to address overfitting.

Start small… something very simple…

Approach with caution

- ❑ Start with a simple algorithm that you can implement quickly. Implement it and test it on your cross-validation data.
- ❑ Plot learning curves to decide if more data, more features, etc. are likely to help.
- ❑ Error analysis: Manually examine the examples (in cross validation set) that your algorithm made errors on. See if you spot any systematic trend in what type of examples it is making errors on.

Manually examine

- Fashion Data
  - Which class was mis classified… Class 7
- Look for probable reasons… then investigate how to remove them.
- Error analysis may not be helpful for deciding if this is likely to improve performance. Only solution is to try it and see if it works.
- Need numerical evaluation (e.g., cross validation error) of algorithm’s performance with and without reclassification of “Sneakers”.

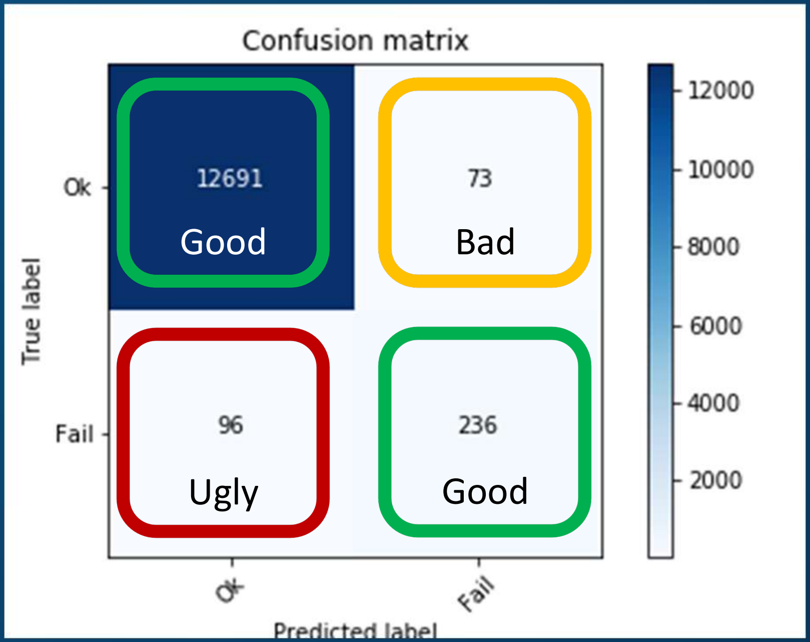
Error metrics for skewed classes

- ❑ What happens when predicting rare event or class
- ❑ Yay! excellent performance… it worked 90 % time…
- ❑ BUT… those 10% matter more….
- ❑ Example : Credit Card Frauds…

Look at Confusion Matrix

- Good:
  - 12691 cases accurately predicted as “ok”.
  - 236 cases accurately predicted as “Fail”.
- Bad:
  - 73 cases which were “ok” but predicted as “Fail”.
- Ugly:
  - 96 cases which were “Fail” but predicted as “Ok”

- Precision
  - Of all machines where we predicted fail, what fraction actually failed
- Recall
  - Of all machines that actually failed, what fraction did we correctly detect as failed…



Legend:
Fail – the unit is expected to fail in next 30 cycles.
Ok – the unit is not expected to fail in next 30 cycles

Provide Business Justification…

Without PdM

- ❑ Fail cases: 236 + 96 = 332
- ❑ Cost of Failure repair:
  - ❖ 332 x 4 = 1328

Percentage Saving :

(1328 – 693)/1328 = 49%

Assuming failure cost is 4* times the maintenance cost, PdM gave you **49 %**** cost saving!

With PdM

❑	Fail cases	: 96
❑	Failure predicted by PdM	: 236
❑	Incorrectly marked as fail	: 73

- ❑ Cost :

#	Description		Cost
1	Fail cases	96 * 4	384
2	Failure predicted by PdM	236 * 1	236
3	Incorrectly marked as fail	73 * 1	73
Total			693

* - Assumed value
** - Actual saving may vary

pra-sâmi

Trading off precision and recall

- Sigmoid $0 < pred < 1$
- Predict 1 if pred ≥ ~~0.5~~ ~~0.4~~ 0.3
- Predict 0 if pred < ~~0.5~~ ~~0.4~~ 0.3

Set the Threshold

- What is more important in the domain….. Precision Or Recall… May be both….

F1 Score (F score)

- How to compare precision/recall numbers?

	Precision(P)	Recall (R)	Average	F_1 Score
Algorithm 1	0.5	0.4	0.45	0.444
Algorithm 2	0.7	0.1	0.4	0.175
Algorithm 3	0.02	1.0	0.51	0.0392

“It’s not who has the best algorithm that wins. It’s who has the most data.” - [Banko and Brill, 2001]

Think and Think Hard

Features : Can you manually predict with this information???

Want more data….

- ❑ Use a learning algorithm with many parameters (e.g. neural network with many hidden units).
- ❑ Use a very large training set (unlikely to overfit)