Open Elective Course [OE]

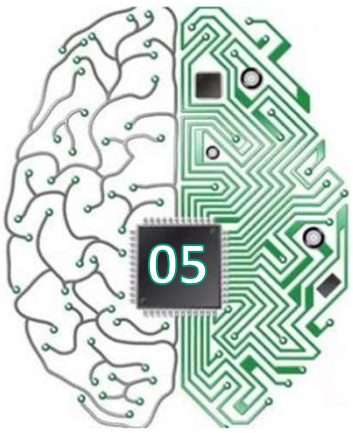
Course Code: CSO507

Winter 2023-24

Lecture#

Deep Learning

Unit-1: ML concept from Information Theory
Machine Learning (ML) Basics [Part-I]



Course Instructor:

Dr. Monidipa Das

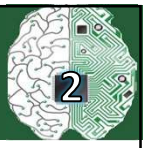
Assistant Professor

Department of Computer Science and Engineering

Indian Institute of Technology (Indian School of Mines) Dhanbad, Jharkhand 826004, India

[Topic continued from previous lecture]

Cross-entropy



- ***Cross-entropy*** is closely related to the KL divergence, and it is defined as the summation of the entropy $H(P)$ and KL divergence $D_{KL}(P||Q)$

$$CE(P,Q) = H(P) + D_{KL}(P||Q)$$

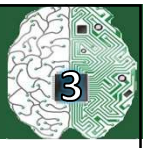
- Alternatively, the cross-entropy can be written as

$$CE(P,Q) = -\mathbb{E}_{X\sim P}\left[\log Q(X)\right]$$

- In machine learning, let's assume a classification problem based on a set of data examples $\{x_1, x_2, \dots, x_n\}$, that need to be classified into k classes
  - For each data example x_i we have a class label y_i
  - The goal is to train a classifier (e.g., a NN) parameterized by θ, that outputs a predicted class label $\hat{y}_i$ for each data example x_i
  - The cross-entropy loss between the true distribution P and the estimated distribution Q is calculated as: $CE(\mathbf{y}, \hat{\mathbf{y}}) = -\mathbb{E}_{X\sim P}\left[\log Q(X)\right] = -\sum_X P(X)\log Q(X) = -\sum_i y_i \log \hat{y}_i$

- Minimizing the cross-entropy loss is the same as **Maximizing the likelihood**

Maximum Likelihood



- In ML, we want to find a model with parameters θ such that $\text{argmax}_\theta\, P(\text{model} \mid \text{data})$
- From Bayes' theorem, $\text{argmax}\, P(\text{model} \mid \text{data})$ is proportional to $\text{argmax}\, P(\text{data} \mid \text{model})$

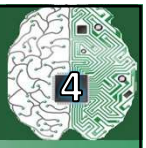
$$P(\theta|x_1, x_2, \dots, x_n) = \frac{P(x_1, x_2, \dots, x_n|\theta)\, P(\theta)}{P(x_1, x_2, \dots, x_n)}$$

- Therefore, the maximum likelihood estimate of θ is based on solving

$$\arg\max_\theta P(x_1, x_2, \dots, x_n|\theta)$$

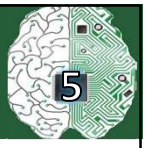
  - Assuming that the data examples are independent, the likelihood of the data given the model parameters θ can be written as $\mathcal{P}(x_1, x_2, \dots, x_n|\theta) = \mathcal{P}(x_1|\theta) \cdots \mathcal{P}(x_n|\theta) = \prod_j \hat{y}_{1j}^{\,y_{1j}} \cdot \prod_j \hat{y}_{2j}^{\,y_{2j}} \cdots \prod_j \hat{y}_{nj}^{\,y_{nj}} = \prod_i \prod_j \hat{y}_{ij}^{\,y_{ij}}$
  - $\log \mathcal{P}(x_1, x_2, \dots, x_n|\theta) = \log\left(\prod_i \prod_j \hat{y}_{ij}^{\,y_{ij}}\right) = \sum_i \sum_j y_{ij} \log \hat{y}_{ij}$
  - A negative of the log-likelihood allows us to use minimization approaches, i.e.,

$$-\log \mathcal{P}(x_1, x_2, \dots, x_n|\theta) = -\sum_i \sum_j y_{ij} \log \hat{y}_{ij}$$ ← ***cross-entropy***

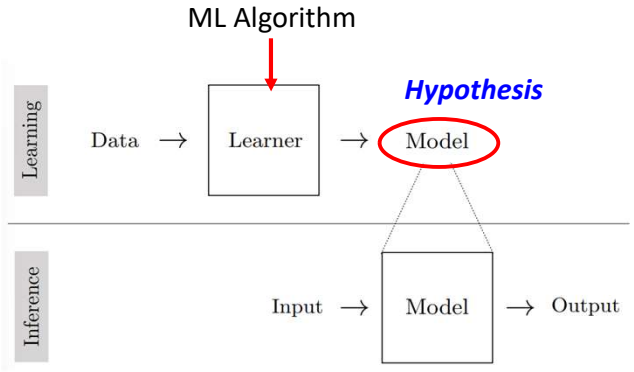


Machine Learning Basics

What do we mean by learning?



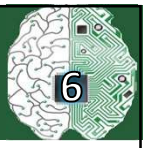
- Definition (well-posed learning problem):
  - A computer program is said to learn from experience **E**
  - with respect to some class of **tasks T** and performance measure **P**,
  - if its performance at task **T**, as measured by **P**, improves with experience **E**
- One can imagine a wide variety of experiences **E**, tasks **T**, and performance measures **P**



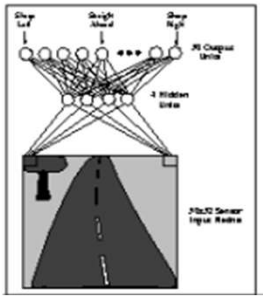
Hypothesis: a function f that reads in low level properties (which are referred to as features) of a data point and delivers the prediction for the same. Usually denoted as h_θ

Hypothesis class: set of possible such functions

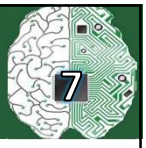
Example



- Autonomous Vehicle Navigation
  - Task T : driving on public highway using vision sensors
  - Training experience E : sequence of images and steering commands recorded observing a human driver
  - Performance measure P : average distance traveled before an error (as judged by human overseer)



The Task, T



- Machine Learning (ML) enables tackling tasks too difficult to solve with fixed programs, written and designed by human beings
- Process of learning itself is not the task
  - Learning is our means of attaining ability to perform the task

customer	height	weight	*shirt size*
C_1	158	58	M
C_2	158	59	M
C_3	158	63	M
C_4	160	59	M
C_5	160	60	M
C_6	163	60	M
C_7	163	61	M
C_8	160	64	L
C_9	163	64	L
C_{10}	165	61	L
C_{11}	165	62	L
C_{12}	165	65	L
C_{13}	168	62	L
C_{14}	168	63	L
C_{15}	168	66	L

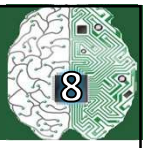
Shirt Size Prediction

Face Recognition

Real World Occluded Faces

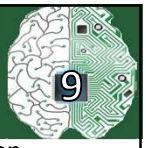
Upper-Face Occlusion Lower-Face Occlusion

Machine Learning Task Description

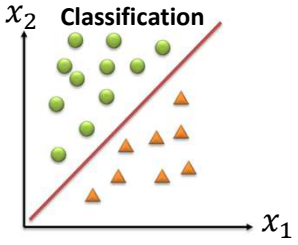


- Usually described in terms of how the machine learning system should process an example
- An example is a collection of features that have been quantitatively measured for some object/event that we want the ML system to process
- Typically represent an example as a vector $x \in \mathbb{R}^n$ where each entry x_i of the vector is a feature

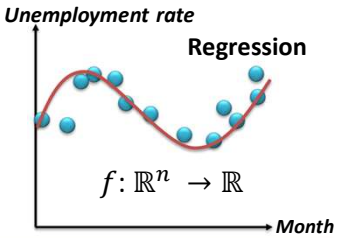
Kinds of Tasks solved using ML



1. **Classification**
2. Classification with missing inputs
3. **Regression**
4. Transcription
5. Machine Translation
6. Structured Output
7. Anomaly Detection
8. Synthesis and Sampling
9. Imputation of Missing Values
10. Denoising
11. Density Estimation

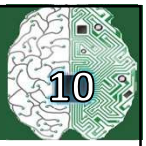


- $f: \mathbb{R}^n \rightarrow \{1, .., k\}$, where n =no of input variables k =no of classes
- f outputs a probability distribution over classes



	Regression	Classification
Outcome	Continuous	Class

The Performance Measure, P



- Need measure of performance specific to task T
  - For classification, classification with missing inputs and transcription:
    - **Accuracy:** Proportion of samples for which correct output is produced
  - For density estimation
    - **Average log probability** assigned to some examples
- Usually on data not seen before, a test set
  - 1. Separate the data into training set and test set
  - 2. Train the model with training set
  - 3. Measure the model's performance with test set
- Often difficult to choose a good measure

The Experience, E

- Most algorithms **experience a dataset**
  - A dataset is a collection of many examples (also called data points) **[Example: Next Slide]**

- **Supervised learning algorithms:**
  - Experiences a dataset associated with labels
  - Learns to predict the labels from the data

  $Training\ data$: $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$

  - Used for prediction of data labels
  - Classification or regression
  - May directly learn decision boundary: Discriminative
  - Or learns probability distributions of the data: Generative

- **Unsupervised learning algorithms:**
  - Experiences a dataset containing many features
  - Learns useful properties of the structure of the dataset

  $Training\ data$: $\{\mathbf{x}_i\}_{i=1}^{N}$

  - Usually learns the entire probability distribution that generated this data set
  - Others perform a role such as clustering

- **Reinforcement learning algorithms:**
  - Not just experience with a fixed dataset, but interact with an environment
  - Learns actions to maximize cumulative rewards

  - No supervised output but delayed reward
    - ✓ Credit assignment

Size or length of the input X_i is commonly known as data/input dimensionality or feature dimensionality

Describing a data set

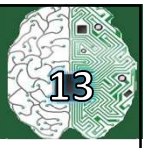
- Common way of describing a data set is with a design matrix
- Different examples in each row
- Each column corresponds to a different feature
- Iris dataset contains 150 examples with four features for each example
- Data set is a design matrix $X \in \mathbb{R}^{150\times 4}$

Anderson's Iris data (oldest set in stat/ML)

Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	*I. setosa*
4.9	3.0	1.4	0.2	*I. setosa*
4.7	3.2	1.3	0.2	*I. setosa*
7.0	3.2	4.7	1.4	*I. versicolor*
6.4	3.2	4.5	1.5	*I. versicolor*
6.9	3.1	4.9	1.5	*I. versicolor*
6.3	3.3	6.0	2.5	*I. virginica*
5.8	2.7	5.1	1.9	*I. virginica*
7.1	3.0	5.9	2.1	*I. virginica*

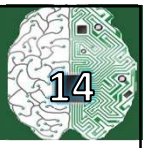
- $X_{i,1}$ is sepal length of plant i, $X_{i,2}$, is sepal width of plant i

Dealing with varying sizes



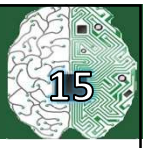
- For a design matrix, each example is a vector of same size
  - But photos of varying size contain different nos. of pixels
- Rather than describing the matrix with m rows we describe it as a set of m elements $\{x^1, x^2, .., x^m\}$. It does not imply that vectors $x^{\{i\}}$ and $x^{\{j\}}$ have the same size
  - Instead of multiplying by a weight matrix of fixed size, convolution with a kernel is applied different no. of times depending on size of input

Types of Features and Types of Outputs



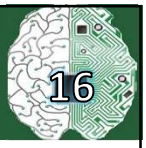
- Features as well as outputs can be real-valued, binary, categorical, ordinal, etc.
- Real-valued: Pixel intensity, house area, house price, rainfall amount, temperature, etc
- Binary: Male/female, adult/non-adult, or any yes/no or present/absent type value
- Categorical/Discrete: Zipcode, blood-group, or any “one from a finite many choices“ value
- Ordinal: Grade (A/B/C etc.) in a course, or any other type where relative values matter
- Often, the features can be of mixed types (some real, some categorical, some ordinal, etc.)

Data and Features

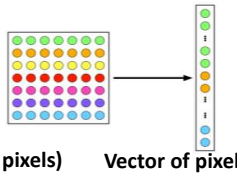


- ML algos require a numeric feature representation of the inputs
- Features can be obtained using one of the two approaches
  - Approach 1: Extracting/constructing features manually from raw inputs
  - Approach 2: Learning the features from raw inputs
- Approach 1 is what we will focus on primarily for now
- Approach 2 is what is followed in Deep Learning algorithms (will see later)
- Approach 1 is not as powerful as Approach 2 but still used widely

Example: Feature Extraction for Image Data



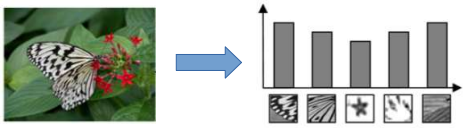
- A very simple feature extraction approach for image data is flattening



7x7 image (49 pixels) Vector of pixel intensities

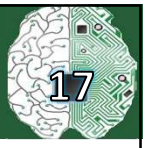
Flattening and histogram based methods destroy the spatial information in the image but often still work reasonably well

- Histogram of visual patterns is another popular feature extraction method for images



- Many other manual feature extraction techniques developed in computer vision and image processing communities (SIFT, HoG, and others)

Example: Feature Extraction for Text Data



- Consider some text data consisting of the following sentences:
  - John likes to watch movies
  - Mary likes movies too
  - John also likes football

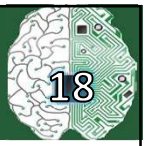
BoW is just one of the many ways of doing feature extraction for text data. Not the most optimal one, and has various flaws, but often works reasonably well

- Want to construct a feature representation for these sentences
- Here is a "bag-of-words" (BoW) feature representation of these sentences

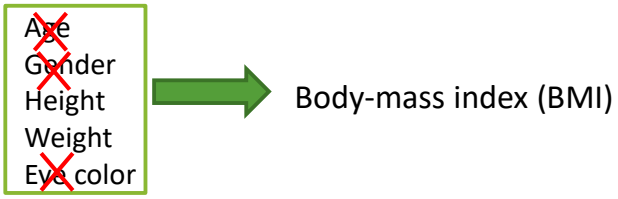
	John	likes	to	watch	movies	Mary	too	also	football
Sentence 1	1	1	1	1	1	0	0	0	0
Sentence 2	0	1	0	0	1	1	1	0	0
Sentence 3	1	1	0	0	0	0	0	1	1

- Each sentence is now represented as a binary vector (each feature is a binary value, denoting presence or absence of a word). BoW is also called "unigram" representation

Feature Selection



- Not all the extracted features may be relevant for learning the model (some may even confuse the learner)
- Feature selection (a step after feature extraction) can be used to identify the features that matter, and discard the others, for more effective learning



- Many techniques exist – some based on intuition, some based on algorithmic principles
- More common in supervised learning but can also be done for unsupervised learning

Some More Postprocessing: Feature Scaling

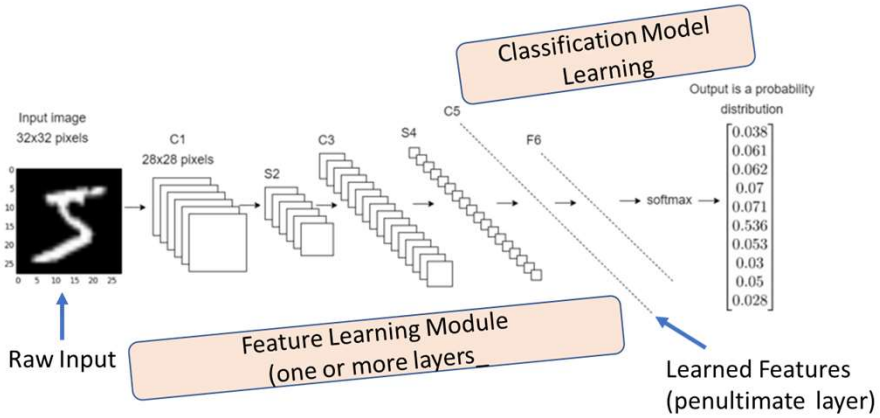
19

- Even after feature selection, the features may not be on the same scale
- This can be problematic when comparing two inputs – features that have larger scales may dominate the result of such comparisons
- Therefore helpful to standardize the features (e.g., by bringing all of them on the same scale such as between 0 to 1)
- Also helpful for stabilizing the optimization techniques used in ML algos

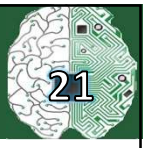
Deep Learning: An End-to-End Approach to ML

20

- Deep Learning = ML with automated feature learning from the raw inputs
- Feature extraction part is automated via the feature learning module



A Simple ML Algorithm: Linear Regression



Task, T: to predict y from x by outputting $\hat{y} = w^{\top} x$

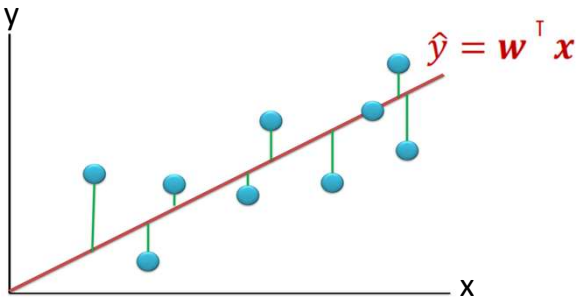
$x \in \mathbb{R}^n$: input data

$y \in \mathbb{R}$: output value ($\hat{y}$: predicted by the model)

$w \in \mathbb{R}^n$: parameters (or weights)

Experience, E: training set (X^{train}, y^{train})

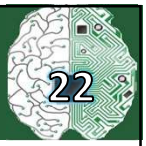
Performance measure, P:
mean squared error (MSE) on (X^{test}, y^{test})



$$MSE_{test} = \frac{1}{m} \sum_i (\hat{y}^{test} - y^{test})_i^2$$

m: Number of data points in the test set

Find w by Minimizing MSE_{train}



$$\nabla_w \text{MSE}_{\text{train}} = 0$$

$$\Rightarrow \nabla_w \frac{1}{m} ||\hat{y}^{(\text{train})} - y^{(\text{train})}||_2^2 = 0$$

$$\Rightarrow \frac{1}{m} \nabla_w ||X^{(\text{train})} w - y^{(\text{train})}||_2^2 = 0$$

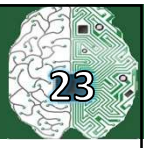
$$\Rightarrow \nabla_w \left(X^{(\text{train})} w - y^{(\text{train})} \right)^{\top} \left(X^{(\text{train})} w - y^{(\text{train})} \right) = 0$$

$$\Rightarrow \nabla_w \left(w^{\top} X^{(\text{train})\top} X^{(\text{train})} w - 2 w^{\top} X^{(\text{train})\top} y^{(\text{train})} + y^{(\text{train})\top} y^{(\text{train})} \right) = 0$$

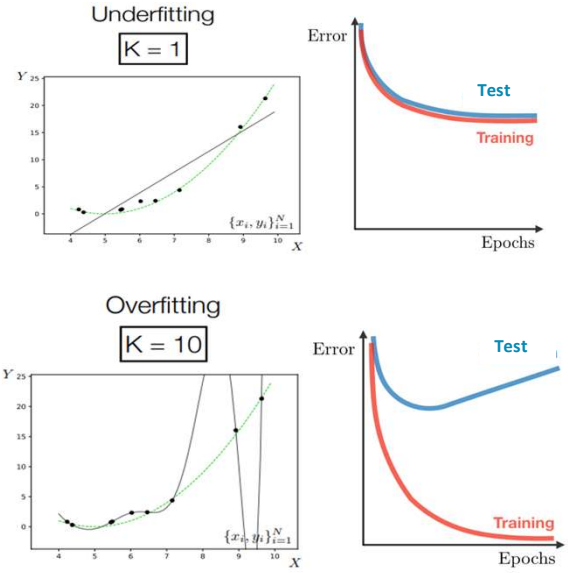
$$\Rightarrow 2 X^{(\text{train})\top} X^{(\text{train})} w - 2 X^{(\text{train})\top} y^{(\text{train})} = 0$$

$$\Rightarrow w = \left(X^{(\text{train})\top} X^{(\text{train})} \right)^{-1} X^{(\text{train})\top} y^{(\text{train})}$$

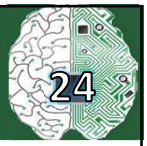
Generalization, Underfitting and Overfitting



- In ML, **generalization** is the ability to perform well on previously unobserved inputs
- **Factors determining how well an ML algorithm will perform** are its ability to
  - Make the training error small
  - Make gap between training and test errors small
- They correspond to two ML challenges
  - **Underfitting:** Inability to obtain low enough error rate on the training set
  - **Overfitting:** Gap between training error and testing error is too large
- We can **control whether a model is more likely to overfit or underfit by altering its capacity**

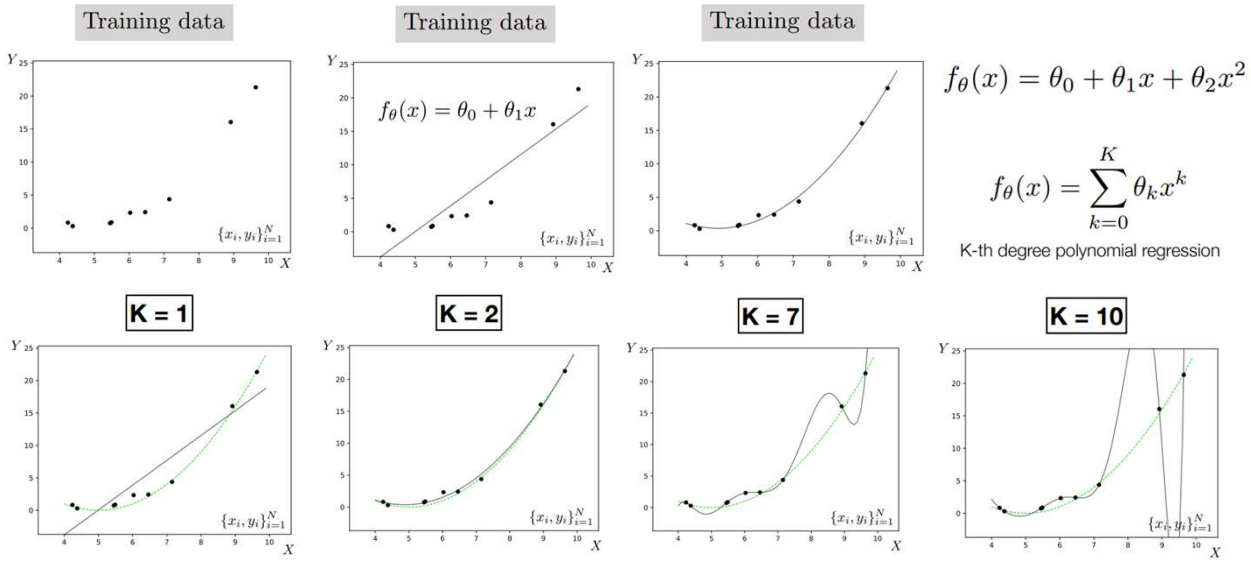


Capacity of a model



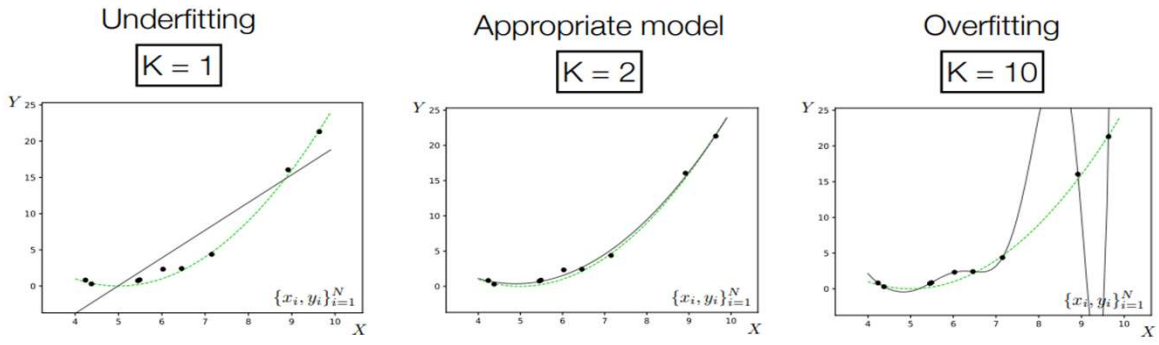
- **Model capacity is ability to fit variety of functions**
  - E.g. linear regression models cannot fit a sine function
  - **Low capacity leads to under-fitting**….struggles to fit training set.
  - **A High capacity model can overfit** by memorizing properties of training set not useful on test set
    - When model has higher capacity, it overfits
  - One way to control capacity of a learning algorithm is by choosing the hypothesis space
    - i.e., set of functions that the learning algorithm is allowed to select as being the solution

Example

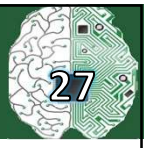


Appropriate Capacity

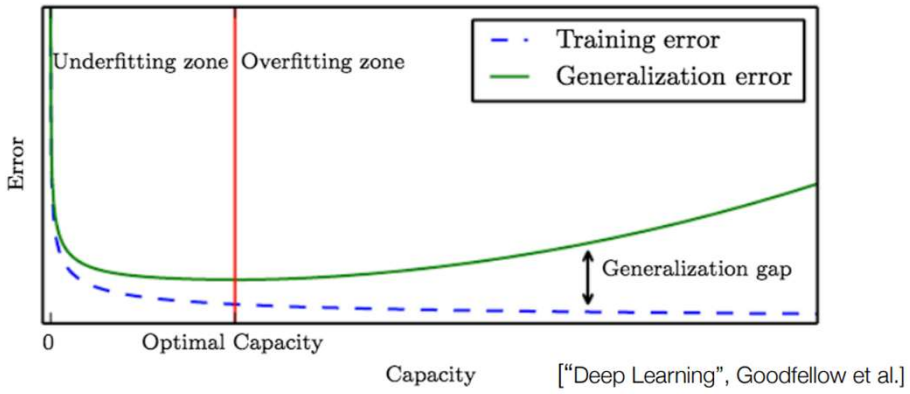
- **Machine Learning algorithms will perform well when their capacity is appropriate** for the true complexity of the task that they need to perform and the amount of training data they are provided with



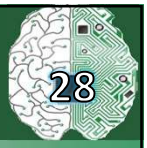
Generalization Error and Capacity



- Relationship between capacity and error
- Typically generalization error has a U-shaped curve



["Deep Learning", Goodfellow et al.]



Questions?