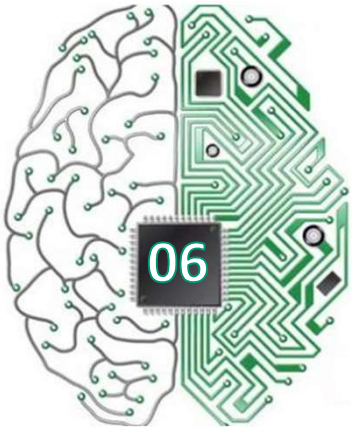
Open Elective Course [OE]

Course Code: CSO507

Winter 2023-24

Lecture#



Deep Learning

Unit-1: Machine Learning Basics [Part-II]

Course Instructor:

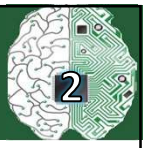
Dr. Monidipa Das

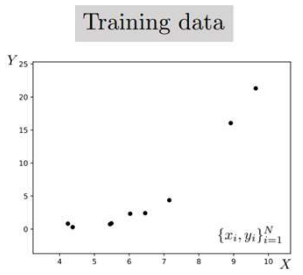
Assistant Professor

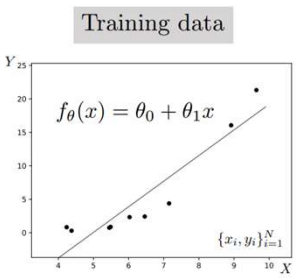
Department of Computer Science and Engineering

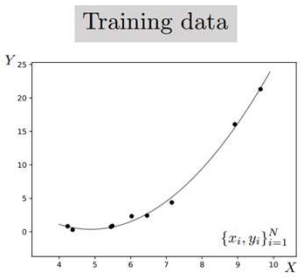
Indian Institute of Technology (Indian School of Mines) Dhanbad, Jharkhand 826004, India

Example (revisited from previous lecture)





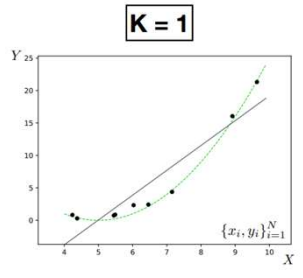


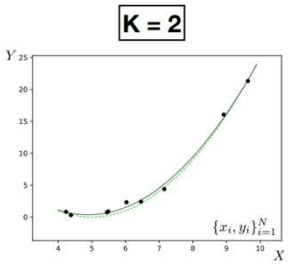


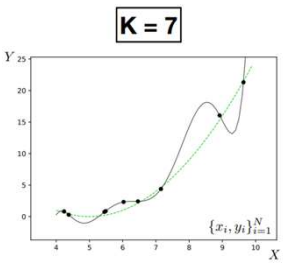
$$f_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$

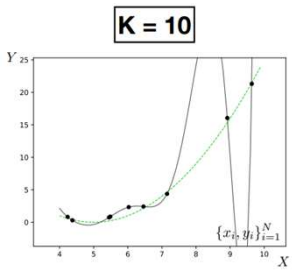
$$f_\theta(x) = \sum_{k=0}^{K} \theta_k x^k$$

K-th degree polynomial regression

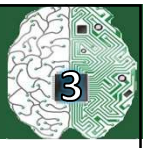






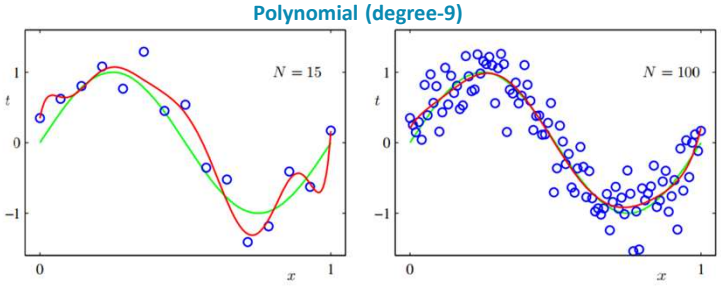


Fitting a model



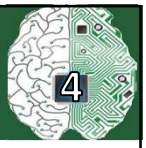
- **Underfitting?**
  - Complexify model
    - Add more parameters (more features, more layers, etc.)
  - Train longer
- **Overfitting?**
  - Perform regularization
  - Get more data

Polynomial (degree-9)



Occam's razor: Among competing hypotheses that explain known observations equally well, choose the simplest one

Regularization



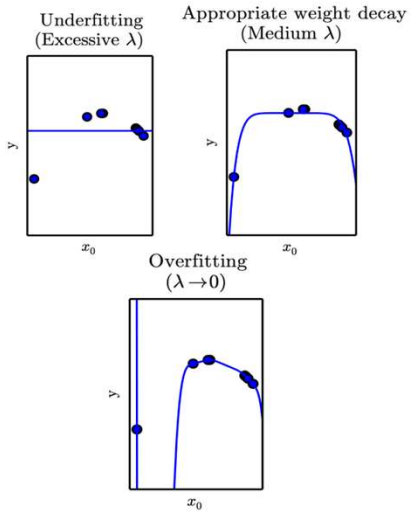
- Procedure aims to avoid the model to overfit the data
- Regularization is able to control the performance of an algorithm
  - Intended to reduce test error but not training error
- **Example: weight decay** for regression problem

$$J(\boldsymbol{w}) = MSE_{train} + \lambda \boldsymbol{w}^{\top} \boldsymbol{w}$$

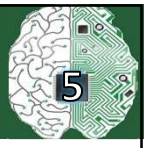
  - $J(\cdot)$: cost function **to be minimized** on training
  - λ: control factor of the preference for smaller weight ($\lambda \geq 0$)

➢ Trades-off between fitting the training data and being small $\boldsymbol{w}$

Effect of regularization on models with degree 9



Hyperparameters vs. parameters



Hyperparameters are higher-level properties for a model

- Decides model's capacity (or complexity)
- Not be learned during training

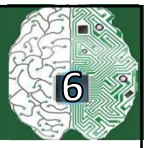
e.g., degree of regression model, weight decay

Parameters are properties of the training data

- Learned during training by a ML model

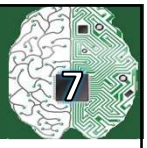
e.g., weights of regression model

Setting Hyperparameters with Validation Set

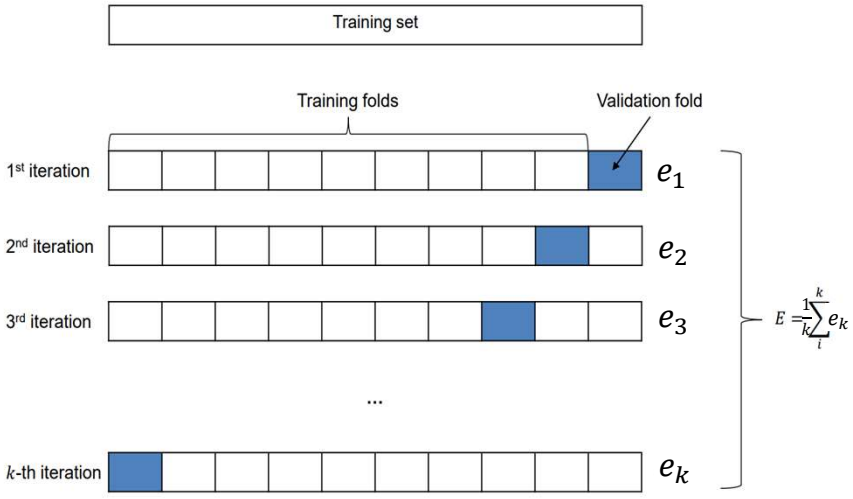


- **Setting hyperparameters in training step is not appropriate**
  - Hyperparameters will be set to yield overfitting
  - (e.g., higher degree of regression model, $\lambda \rightarrow 0$)
- **Test set will not be seen for training nor model choosing** (hyperparameter setting)
- So, we need **validation set** that the training algorithm does not observe
  - 1. Split validation set from training data
  - 2. Train a model with training data (not including validation set)
  - 3. Validate a model with validation set, update hyperparameters

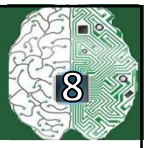
k-fold Cross Validation



- k - fold cross-validation
  - Partition the training data into k non-overlapping subsets
  - On trial i, i th subset of data is used as the test set
  - Rest of the data is used as the training set

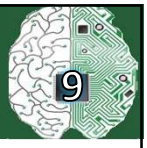


k -fold cross validation algorithm



Define KFoldXV($\mathbb{D}, A, L, k$):
Require: $\mathbb{D}$, the given dataset, with elements $\boldsymbol{z}^{(i)}$
Require: A, the learning algorithm, seen as a function that takes a dataset as input and outputs a learned function
Require: L, the loss function, seen as a function from a learned function f and an example $\boldsymbol{z}^{(i)} \in \mathbb{D}$ to a scalar $\in \mathbb{R}$
Require: k, the number of folds
Split $\mathbb{D}$ into k mutually exclusive subsets $\mathbb{D}_i$, whose union is $\mathbb{D}$.
for i from 1 to k **do**
  $f_i = A(\mathbb{D} \backslash \mathbb{D}_i)$ — Train A on dataset without D_i
  for $\boldsymbol{z}^{(j)}$ in $\mathbb{D}_i$ **do**
    $e_j = L(f_i, \boldsymbol{z}^{(j)})$ — Determine errors for samples in D_i
  end for
end for
Return $\boldsymbol{e}$ — Return vector of errors $\boldsymbol{e}$ for samples in D

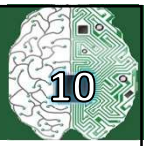
The No-Free Lunch Theorem (NFLT)



- No free lunch theorem states:
  - Averaged over all distributions, every algorithm has same error classifying unobserved points
    - i.e., no ML algorithm universally better than any other
      - Most sophisticated algorithm has same error rate that merely predicts that every point belongs to same class

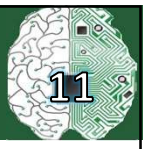
- *We don't seek a universal learning algorithm because there isn't one*

Generalization Error



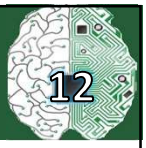
- Central challenge of ML is that ***the algorithm must perform well on new, previously unseen inputs***
  - Not just those on which our model has been trained
- Ability to perform well on previously unobserved inputs is called **generalization**
- **Generalization error (also called Test Error) definition**
  - Expected value of the error on a new input
    - Expected value is computed as average over inputs taken from distribution encountered in practice
- **How can we affect performance when we observe only the training set?**

Statistics provides tools for ML



- The field of statistics provides many tools to achieve the ML goal of solving a task not only on the training set but also to generalize
- Foundational concepts such as
  - Parameter estimation
  - Bias
  - Variance
- They characterize notions of generalization, over- and under-fitting

Point Estimation and Function Estimation



- **Point Estimation:** attempt to provide the single best prediction of some quantity of interest

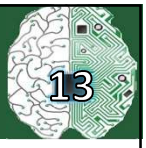
Point estimator

$$\widehat{\boldsymbol{\theta}}_{\boldsymbol{m}} = g(\boldsymbol{x^1}, \dots, \boldsymbol{x^m})$$

$\widehat{\boldsymbol{\theta}}$: point estimator for the property of a model (e.g., expectation)
m: number of data elements
$\{\boldsymbol{x^1}, \dots, \boldsymbol{x^m}\}$: independent and identically distributed (i.i.d.) data points
$g(\cdot)$: any estimation function for the given data points

- **Function Estimation:** Point estimation that refers to estimation of relationship between input and target variables
- Most commonly studied properties of point estimators: **Bias** and **Variance**

Bias and Variance of an estimator

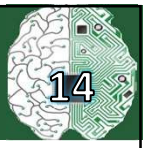


- The **bias of an estimator** $\hat{\theta}_m = g(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)})$ for parameter θ is defined as

$$\text{bias}\left(\hat{\theta}_m\right) = E\left[\hat{\theta}_m\right] - \theta$$

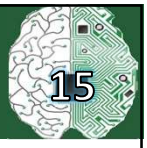
- The estimator is unbiased if bias($\hat{\theta}_m$)=0
- The **variance of an estimator** Var($\hat{\theta}$) is how much we expect the estimator to vary as a function of the data sample $Var(\hat{\theta}_m) = E[(\hat{\theta}_m - E[\hat{\theta}_m])^2]$
- The square root of the variance is called the standard error, denoted $SE(\hat{\theta}_m)$
- Bias and Variance measure **two different sources of error of an estimator**

Bias vs Variance Tradeoffs



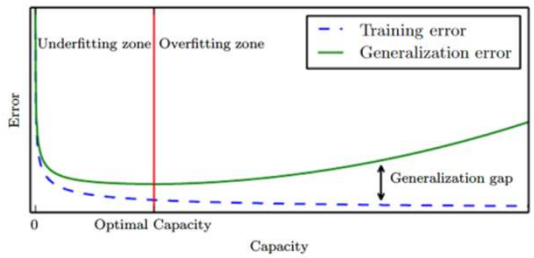
- Bias error is an error from erroneous assumptions in the learning algorithm.
  - High bias can cause an algorithm to miss the relevant relations between features and target outputs (underfitting).
- Variance is an error from sensitivity to small fluctuations in the training set.
  - High variance can cause an algorithm to model the random noise in the training data, rather than the intended outputs (overfitting).

Underfit-Overfit : Bias-Variance

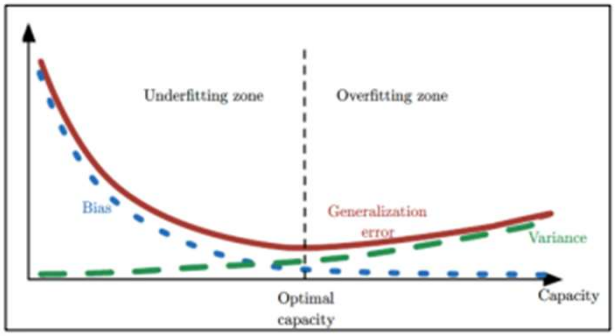


- Relationship of bias-variance to capacity is similar to underfitting and overfitting relationship to capacity

Model complexity to capacity

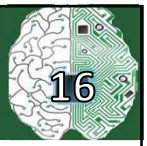


Bias-Variance to capacity



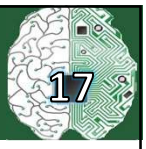
- Both have a U-shaped curve of generalization Error as a function of capacity

Negotiating between bias -variance trade-off



- How to choose between two algorithms, one with a large bias and another with a large variance?
  - Most common approach is to
    - ***use cross-validation***
    - Alternatively we can ***minimize Mean Squared Error*** which incorporates both bias and variance

Mean Squared Error



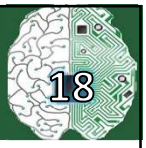
- Mean Squared Error of an estimate is

$$\text{MSE} = E\left[\left(\hat{\theta}_m - \theta\right)^2\right]$$
$$= \text{Bias}\left(\hat{\theta}_m\right)^2 + \text{Var}\left(\hat{\theta}_m\right)$$

- Minimizing the MSE keeps both bias and variance in check

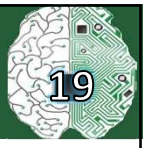
$$E[(\hat{\theta} - \theta)^2]$$
$$= E[(\hat{\theta} - E[\hat{\theta}] + E[\hat{\theta}] - \theta)^2]$$
$$= E[\hat{\theta}^2 - \hat{\theta}E[\hat{\theta}] + \hat{\theta}E[\hat{\theta}] - \hat{\theta}\theta - \hat{\theta}E[\hat{\theta}] + E[\hat{\theta}]^2 - E[\hat{\theta}]^2 + E[\hat{\theta}]\theta + \hat{\theta}E[\hat{\theta}] - E[\hat{\theta}]^2 + E[\hat{\theta}]^2 - E[\hat{\theta}]\theta - \hat{\theta}\theta + E[\hat{\theta}]\theta - E[\hat{\theta}]\theta + \theta^2]$$
$$= E[(\hat{\theta} - E[\hat{\theta}])^2] + (E[\hat{\theta}] - \theta)^2 + 2(E[\hat{\theta}E[\hat{\theta}] - \hat{\theta}\theta - E[\hat{\theta}]^2 + E[\hat{\theta}]\theta])$$
$$= \underbrace{E[(\hat{\theta} - E[\hat{\theta}])^2]}_{Var(\hat{\theta})} + \underbrace{(E[\hat{\theta}] - \theta)^2}_{bias(\hat{\theta})^2}$$

Bayesian statistics



- **Bayesian perspective**
  - Uses probability to reflect degrees of certainty of states of knowledge
  - The dataset is directly observed and so is not random
  - Parameter $\boldsymbol{\theta}$ is represented as random variable
- **The prior**
  - We represent our knowledge of $\boldsymbol{\theta}$ using the prior probability distribution, notation with $\boldsymbol{p}(\boldsymbol{\theta})$, before observing data
  - Select broad priori distribution (with high degree of uncertainty), such as finite range of volume, with a uniform distribution, or Gaussian.

Mathematical description



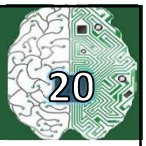
- Set of data samples $\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(m)}\}$
- The dataset is directly observed and so is not random
- Parameter $\boldsymbol{\theta}$ is represented as random variable
- Combine the data likelihood with the prior via Bayes' rule:

Likelihood *Prior*

$$p\left(\boldsymbol{\theta}|\boldsymbol{x}^{(1)}, \cdots, \boldsymbol{x}^{(m)}\right) = \frac{p\left(\boldsymbol{x}^{(1)}, \cdots, \boldsymbol{x}^{(m)}|\boldsymbol{\theta}\right)p(\boldsymbol{\theta})}{p(\boldsymbol{x}^{(1)}, \cdots, \boldsymbol{x}^{(m)})}$$

Bayesian inference

Relative to Maximum Likelihood Estimate



Make prediction using a full distribution over θ

After observing m samples, predict distribution over the next data sample, $x^{(m+1)}$, is given by:

$$p\left(x^{(m+1)}|x^{(1)}, \cdots, x^{(m)}\right) = \int p\left(x^{(m+1)}|\theta\right)p\left(\theta|x^{(1)}, \cdots, x^{(m)}\right)d\theta$$

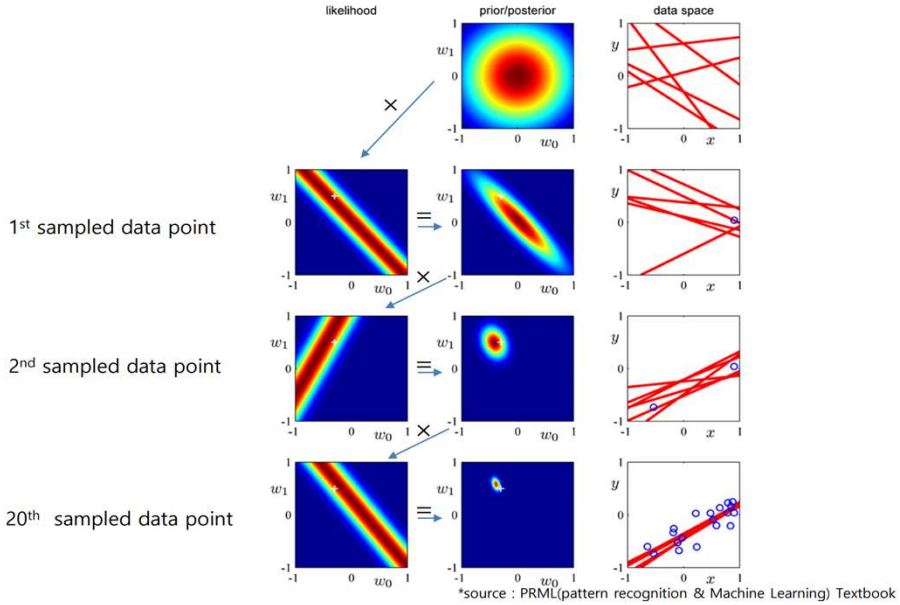
Prior distribution has influence by shifting probability toward the parameter space

Bayesian method typically generalize much better

But high computational cost

Example: Bayesian statistics

21



*source : PRML(pattern recognition & Machine Learning) Textbook

Maximum A Posterior (MAP) Estimation

22

Chose the point of maximal posterior probability

$$\theta_{MAP} = \underbrace{\arg\max_{\theta} p(\theta|x)}_{\text{posterior}} = \underbrace{\arg\max_{\theta} \log p(x|\theta)}_{\text{likelihood}} + \underbrace{\log p(\theta)}_{\text{prior}}$$

Similar with weight decay term

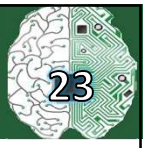
Has the advantage of leveraging information that is brought by the prior

Additional information helps the variance of MAP estimation

But it increase bias

Regularized estimation strategies can be interpreted as making the MAP approximation

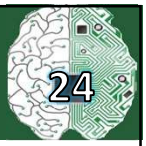
Exercise (1)



1. How can we prevent underfitting?
   a) Increase the number of data samples
   b) Increase the number of features
   c) Decrease the number of features
   d) Decrease the number of data samples

2. If your ML model (say Neural Network) seems to have high variance, which of the following would be promising to try?
   a) Increasing the number of parameters/features
   b) Get more training data
   c) Get more test data
   d) Add regularization

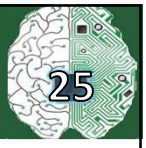
Exercise (2)



3. What happens when you increase the regularization (hyper)parameter λ?
   a) The weights are pushed towards becoming smaller (closer to 0)
   b) The weights are pushed towards becoming larger (further from 0)
   c) The total number of weights increased
   d) None of the above, because there would be no impact on weights

4. You are working on building a classifier model to distinguish apples, bananas, and oranges. Suppose your classifier obtains a training set error of 0.3%, and validation set error of 8%. Which of the following would you try?
   a) Increase regularization (hyper)parameter lambda
   b) Decrease regularization (hyper)parameter lambda
   c) Increase the number of training data samples
   d) Increase the number of parameters/features

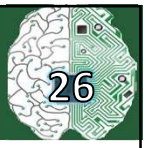
Recipe for Machine Learning



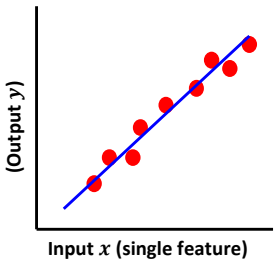
- All Machine Learning (ML) is an instance of a recipe:
  - Specification and representation of a dataset
  - Math model of the ML architecture
  - Evaluation (training/generalization) with a cost function
  - Optimization procedure

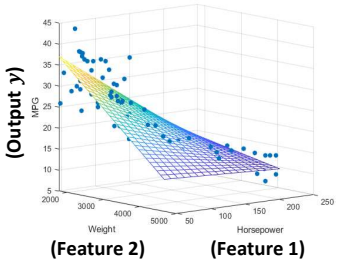
- Example of building an **ML model for linear regression** is shown next

Example: Linear Regression



- Linear regression is like fitting a line or (hyper)plane to a set of points
- Given: Training data with N input-output pairs $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$, $\boldsymbol{x_n} \in \mathbb{R}^D$, $y_n \in \mathbb{R}$
  Goal: Learn a model to predict the output for new test inputs
- **Model:** A function that approximates the I/O relationship to be a linear model
  $y_n \approx f(\boldsymbol{x}_n) = \boldsymbol{w}^\top \boldsymbol{x}_n \quad (n = 1,2,\dots,N)$
- **Cost Function:** the total error or "loss" of this model over the training data:
  $L(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N} \ell(y_n, \boldsymbol{w}^\top \boldsymbol{x}_n)$

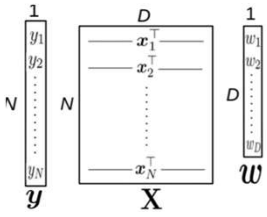




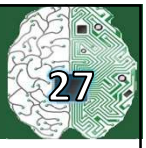
Linear Regression with Squared Loss

$L(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N}(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2$

- **Optimization:** find the $\boldsymbol{w}$ that optimizes (minimizes) the above squared loss



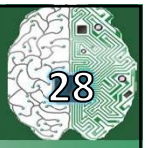
Cost Function



- Typically the cost function can be written as an average over a training set

$$J(\theta) = E_{(\boldsymbol{x},y)\sim\hat{p}_{\text{data}}}\left(L(f(\boldsymbol{x};\theta),y)\right)$$

  - Where
    - L is the per-example loss function
    - $f(\boldsymbol{x};\boldsymbol{\theta})$ is the predicted output when the input is $\boldsymbol{x}$
    - $\hat{p}_{data}$ is the empirical distribution
  - In supervised learning y is target output
- Trivial to extend to cases:
  - Where parameters $\boldsymbol{\theta}$ and input $\boldsymbol{x}$ are arguments or
  - Exclude output y as argument
  - For regularization or unsupervised learning
- **Typical examples of cost functions will be discussed along with the ML tasks in subsequent lectures**



Questions?