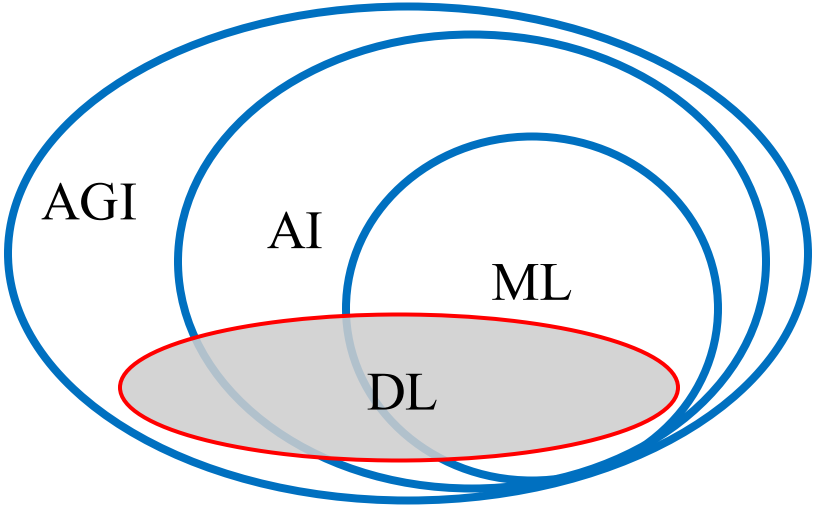
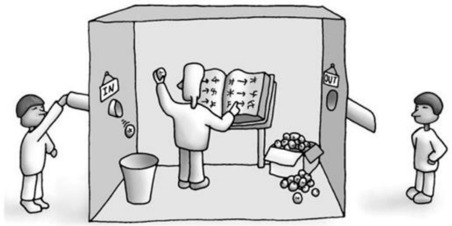
What is Machine Learning?

- AI versus ML versus DL
- ML as an inverse problem
- ML Inference

AI versus ML versus DL





The Chinese room

Source: Wikicomms

- Artificial Intelligence (AI):
  - Definitions:
    - Narrow AI: Task specific AI such as Siri
    - Artificial general intelligence (AGI): Data on Star Trek
    - Strong (sentience or consciousness) vs Weak AI (performs task):
    - Artificial Super Intelligence (AGI): More intelligent than any human being and able to design new versions of itself.
  - Chinese room (Searle 1980):
    - “Does the machine literally "understand" Chinese? Or is it merely simulating the ability to understand Chinese?”
  - Tests of AI:
    - Turing Test (Turing): A machine can fool a human in a conversation.
    - Robot College Student Test (Goertzel): A machine can pass a class
    - Employment Test (Nilsson): A machine can successfully perform a job.
    - The Ikea test (Marcus): A machine can assemble furniture correctly.
    - Coffee Test (Wozniak): A machine can make coffee.
- Machine Learning (ML): Train an algorithm to reproduce answers from data
- Deep Learning (DL): A particularly successful ML method based on deep sequences of neural networks

State of AI

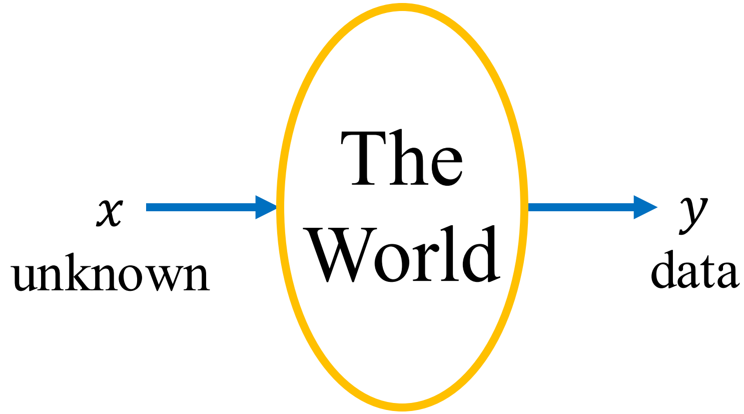
- Turing Test has been passed!
  - GPT-4.5 was judged to be the human 73% of the time.
  - Significantly more often than interrogators selected the real human participant. ☹
- Measuring intelligence
  - GPT5.1 may have passed the turning test, but there is clearly still something missing.
- When will AGI happen?
  - Do we just need more computers, data, and money?
  - Or have we plateaued?
- How will AGI/ASI change the world?
  - AI 2027: https://ai-2027.com
    - Not good.
  - "Genesis", Kissinger, et al.
    - Many different ASI will compete and lead to a meanable world

[1] Cameron R Jones and Benjamin K Bergen, "Large Language Models Pass the Turing Test," arXiv preprint arXiv:2503.23674, 2025, https://arxiv.org/pdf/2503.23674 .

[2] Henry A Kissinger and Eric Schmidt and Craig Mundie and Daniel Huttenlocher, "Genesis: Artificial Intelligence, Hope, and the Human Spirit," Little, Brown and Company, 2024, https://www.hachettebookgroup.com/titles/henry-a-kissinger/genesis/9780316574552/ .

Inverse Problems

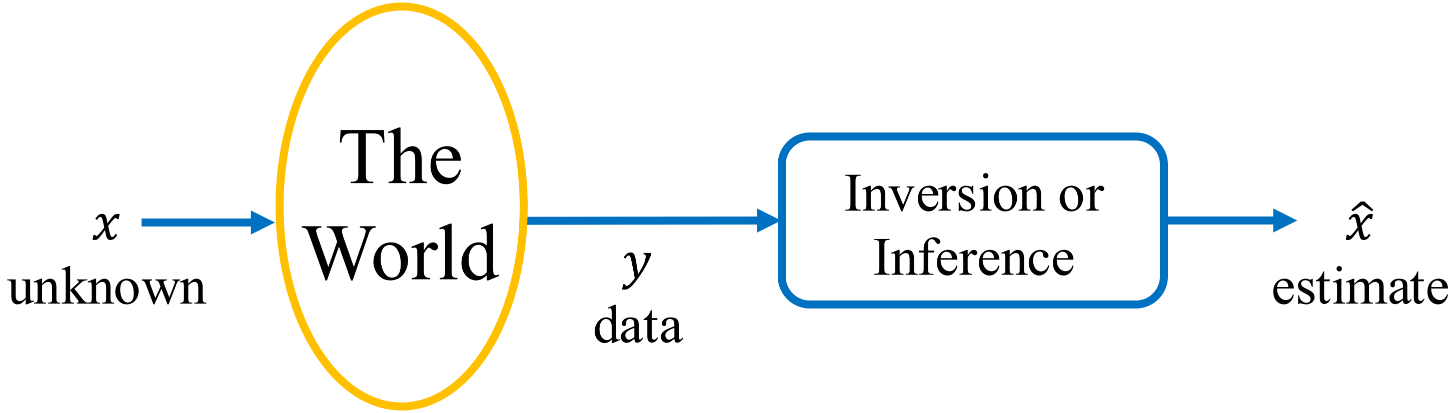
- Determine some unknown quantity from available data.



  - y what we can measure or observe
  - x what we would like to know

- This is an inverse problem:
  - Computer vision, sensing, demodulation, speech recognition, etc.
  - Business analytics: What movie will the customer like best?
  - Science: What is the structure of this particle?

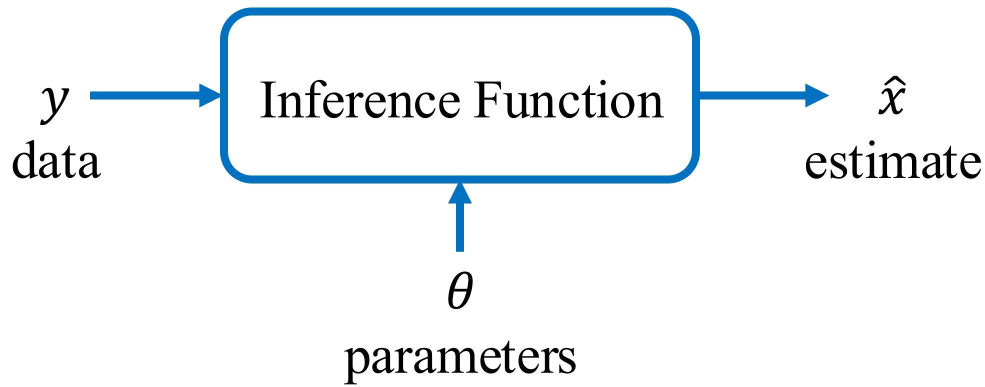
Solving an Inverse Problems



- Goal of Machine Learning (ML): Solve this inverse problem
- Observations:
  - The answer, $\hat{x}$, is usually not equal to the unknown, x.
  - But hopefully, $\hat{x}$ is close to x.
- Questions:
  - How do we compute the inverse?
  - Is there a best inverse?
- Mick Jagger's Theorem:
  - You can't always get what you want. But if you try sometimes, you might fine, you get what you need.

Machine Learning: Inference

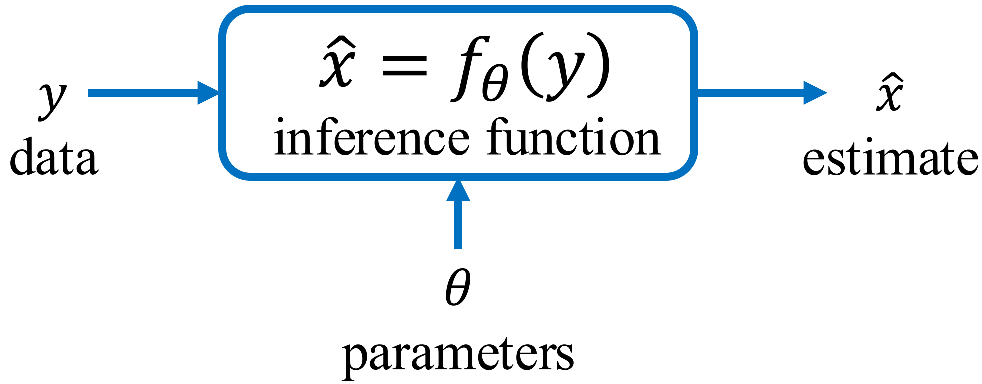
- Machine Learning approach



- Comments
  - Adjust parameter θ to achieve the "best" or at least a "good" answer
  - $\hat{x}$ has a "hat" because it is an estimate (i.e., a guess) of the unknown x.
  - y, $\hat{x}$, and θ are usually finite dimensional vectors.

ML: Mathematical Inference

- Mathematical representation of ML inference

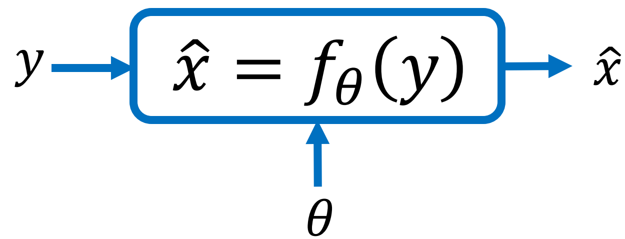


- Questions:
  - What family of functions do we choose for $f_\theta(\cdot)$?
  - What is the dimension of $\theta \in \Re^P$?
  - What is our goal?
  - How do we determine the best value of θ?

Single Layer Neural Networks

- Mathematical representation
- Graphical representation
- One hot encoding
- Activation functions

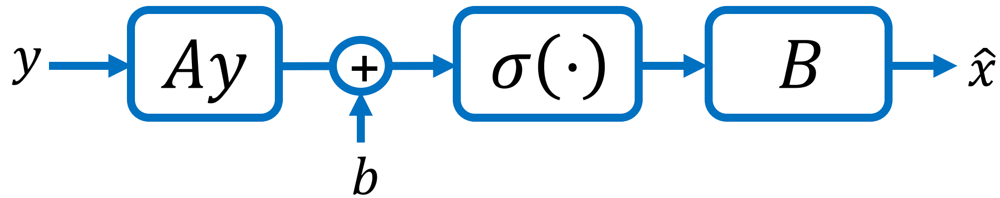
What ML Function should we use?



- Possible choices
  - Support vector machines (SVM)
  - Radial basis functions (RBF)
  - Gaussian mixture functions
- Neural Networks
  - Very high capacity/model order
  - Easy to train with modern analytical/computational tools.
  - Shallow neural networks: Use one easy-to-train layer.
  - Deep neural networks: Train a hierarchical stack of layers.

Single Layer Dense NN

- Single layer NN, graphically



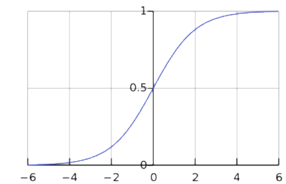
  – Mathematically, $\hat{x} = B\sigma(Ay + b)$ where
    - $A \in \Re^{N_1 \times N_y}$ is a matrix of multiplicative weights.
    - $b \in \Re^{N_1}$ is a column vector of additive offsets.
    - $\sigma: \Re^{N_1} \to \Re^{N_1}$ is a point-wise activation function.
    - $B \in \Re^{N_x \times N_1}$ is a matrix of multiplicative weights.
    - Typical activation function: Logistic sigmoid

$$\sigma_i(z) = \frac{1}{1 + e^{-z_i}}$$

Point-Wise Activation Functions

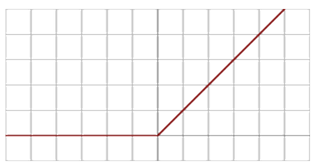
- Logistic sigmoid function

$$\sigma_i(z) = \frac{1}{1 + e^{-z_i}}$$



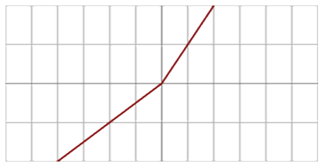
- Rectified linear unit (ReLU)

$$\sigma_i(z) = \begin{cases} 0 \; if \; z_i \leq 0 \\ z_i \;\; if \; z_i > 0 \end{cases}$$



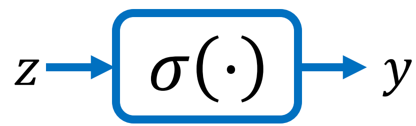
- Leaky ReLU

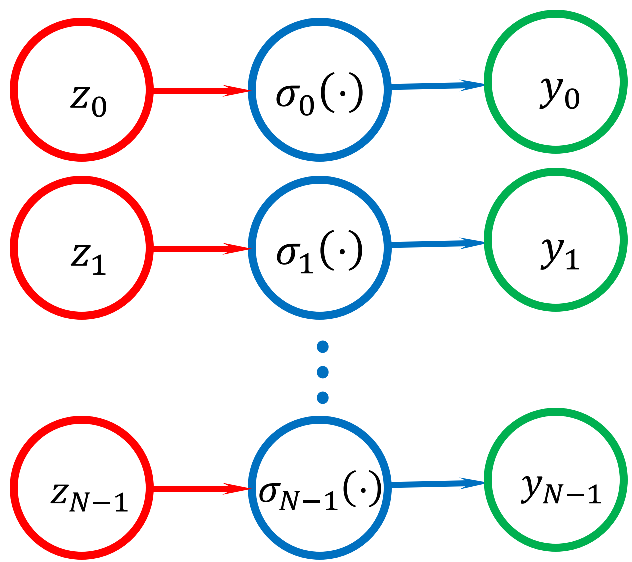
$$\sigma_i(z) = \begin{cases} \alpha z_i \; if \; z_i \leq 0 \\ z_i \;\; if \; z_i > 0 \end{cases}$$



Point-Wise Activation Function

- Point-wise activation function, $\sigma: \Re^N \rightarrow \Re^N$





Gradient of Point-Wise Activation Function

- Gradient Matrix is:
  - Diagonal
  - Sparse (most entries are zero)
  - Fast to compute and apply

j index

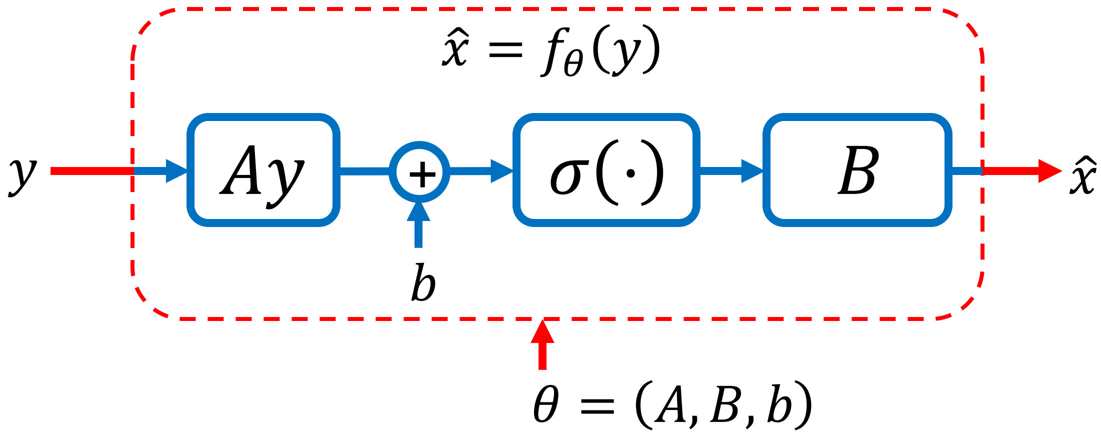
gradient matrix

$$\nabla\sigma(z) : \begin{array}{cc} & \text{j index} \\ \text{i index} & \left[\dfrac{\partial \sigma_i(z)}{\partial z_j}\right] \end{array} = \begin{bmatrix} d_0 & & & 0 \\ & d_1 & & \\ & & \ddots & \\ 0 & & & d_N \end{bmatrix}$$

$$d_i = \frac{\partial \sigma(z_i)}{\partial z_i}$$

Single Layer NN: Abstract Form

- Single layer NN,



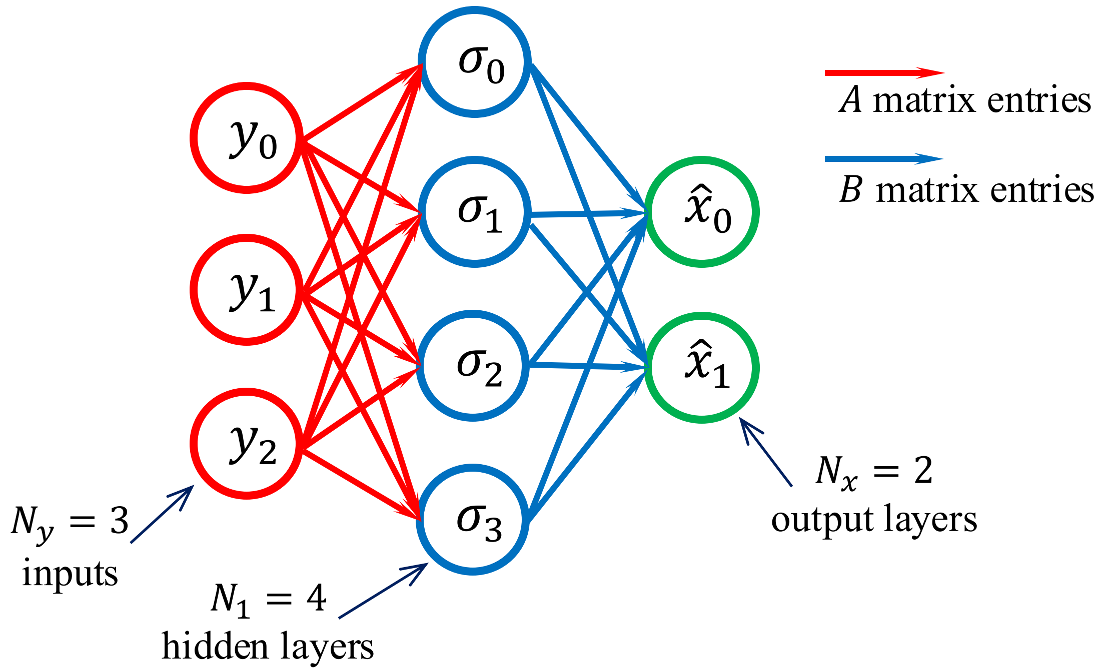
- Mathematical representation is

$$f_\theta(y) = B\sigma(Ay + b)$$

where $\theta = (A, B, b)$ is the set of all NN parameters.

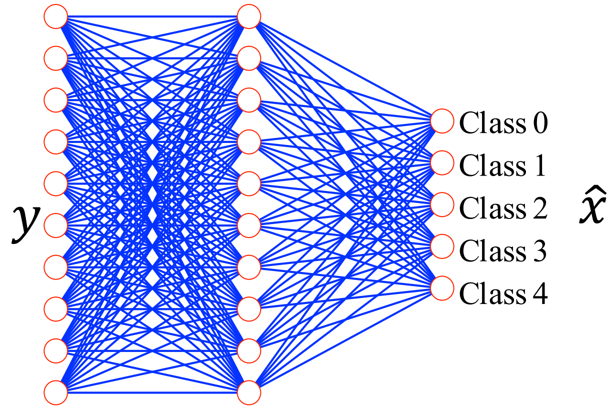
Single Layer NN Flow Diagram

- Example for $N_y = 3$, $N_1 = 4$, and $N_x = 2$.



- Approximation theorem:
  - Cybenko 1989, "*Approximation by Superpositions of Sigmoidal Functions*" $\Rightarrow$ Any function can be approximated by a single layer neural network!
  - But number of hidden layers might be huge!!

One-Hot Encoding for Classification

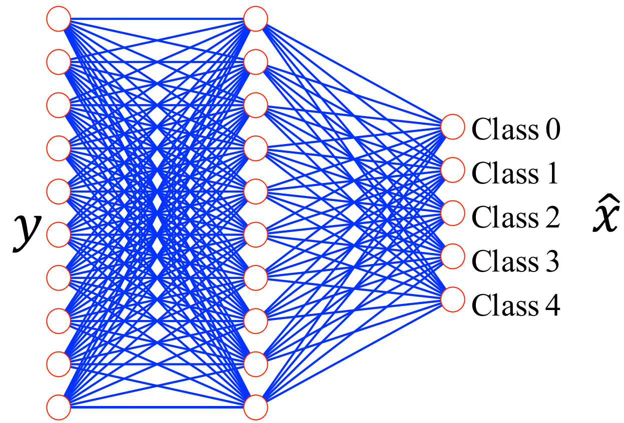


- A method to encode the class of an object
  - A vector $y \in \Re^{10}$ needs to be classified into one of M possible classes.
- Standard encoding:
  - $\hat{x} \in \{0, \cdots, M-1\}$ each value represents a different class
- One-hot encoding:
  - $\hat{x} \in \Re^{M}$ s.t.

$$\hat{x}_i = \begin{cases} 1 & if\ class = i \\ 0 & if\ class \neq i \end{cases}$$

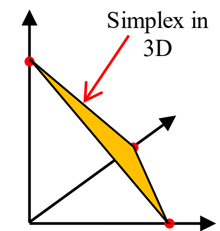
- Example: For $M = 5$, and class=3, then $\hat{x} = [0,0,1,0,0]$

Continuous Encoding for Classification



- Define an M-dimensional simplex as

$$\mathcal{S}^M = \left\{ x \in \Re^M : \forall i, x_i \geq 0 \text{ and } 1 = \sum_{i=0}^{M-1} x_i \right\}$$



  - Then $\hat{x} \in \mathcal{S}^5 \subset \Re^5$
  - Like a probability density for each class

- Advantage:
  - Continuous function on a convex set
  - Makes optimization easier
  - Allows for representation of probability densities

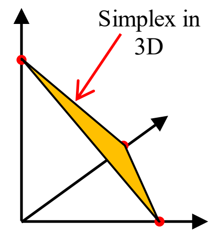
- Example: For $M = 5$, and class=3, then $\hat{x} = [p_0, p_1, p_2, p_3, p_4]$ where $p_i \in \mathcal{S}$

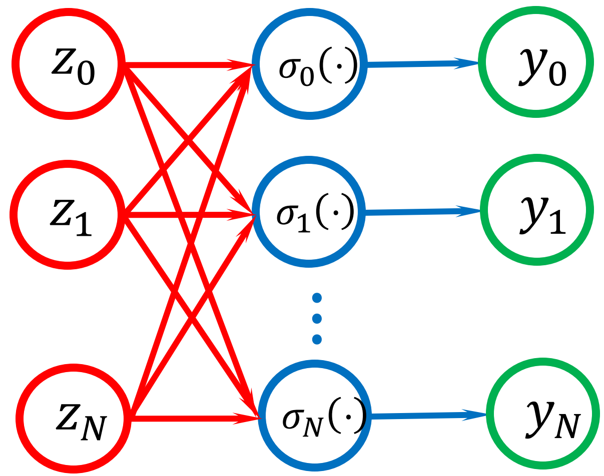
Softmax Activation Functions

- Softmax

$$\sigma_i(z) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

  - Joint activation function
  - Notice that $\sigma_i(z) \in \mathcal{S}^N \subset \Re^N$.
  - It can be interpreted as a probability density.





Gradient of Softmax Function

- Gradient Matrix is:
  - Dense matrix (most or all entries are non-zero)
  - Usually slow to compute, but some tricks in this case

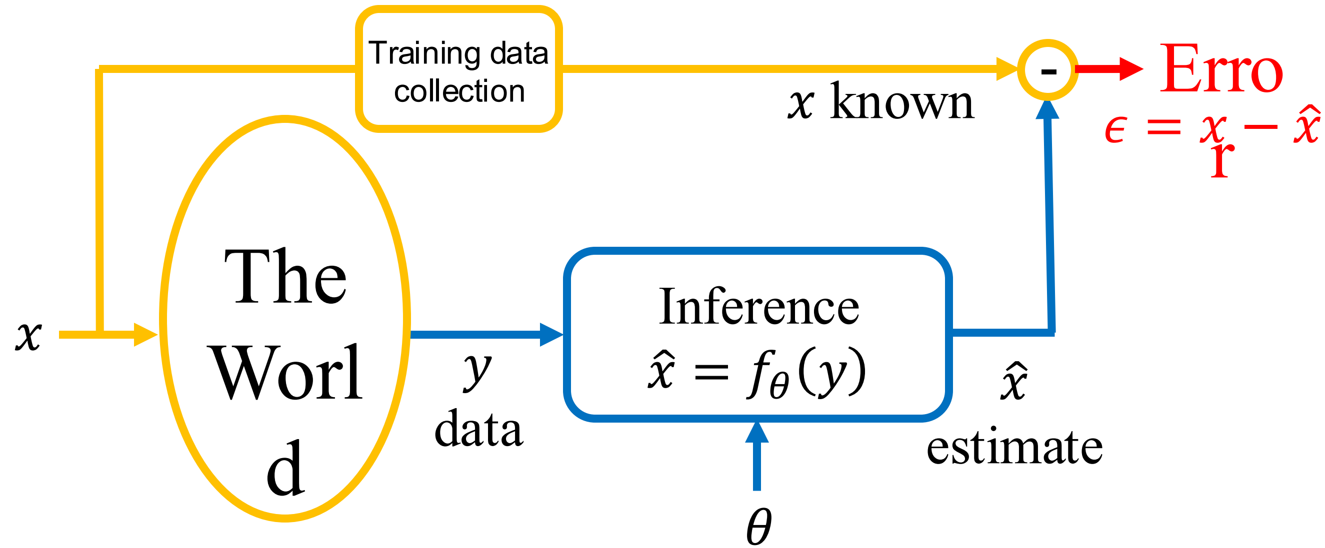
$$[\nabla\sigma(z)]_{i,j} = \frac{1}{\sum_k e^{z_k}}\left(e^{z_i}\delta_{i-j} - \frac{e^{z_i}e^{z_j}}{\sum_k e^{z_k}}\right)$$

$$\nabla\sigma(z) = \frac{1}{\sum_k e^{z_k}} \boxed{\begin{matrix} e^{z_0} & & & 0 \\ & e^{z_1} & & \\ & & \ddots & \\ 0 & & & e^{z_N} \end{matrix}} - \frac{1}{\left(\sum_k e^{z_k}\right)^2} \boxed{\begin{matrix} e^{z_0} \\ e^{z_1} \\ \vdots \\ e^{z_N} \end{matrix}} \boxed{\begin{matrix} e^{z_0} & e^{z_1} & \cdots & e^{z_N} \end{matrix}}$$

The Loss Function

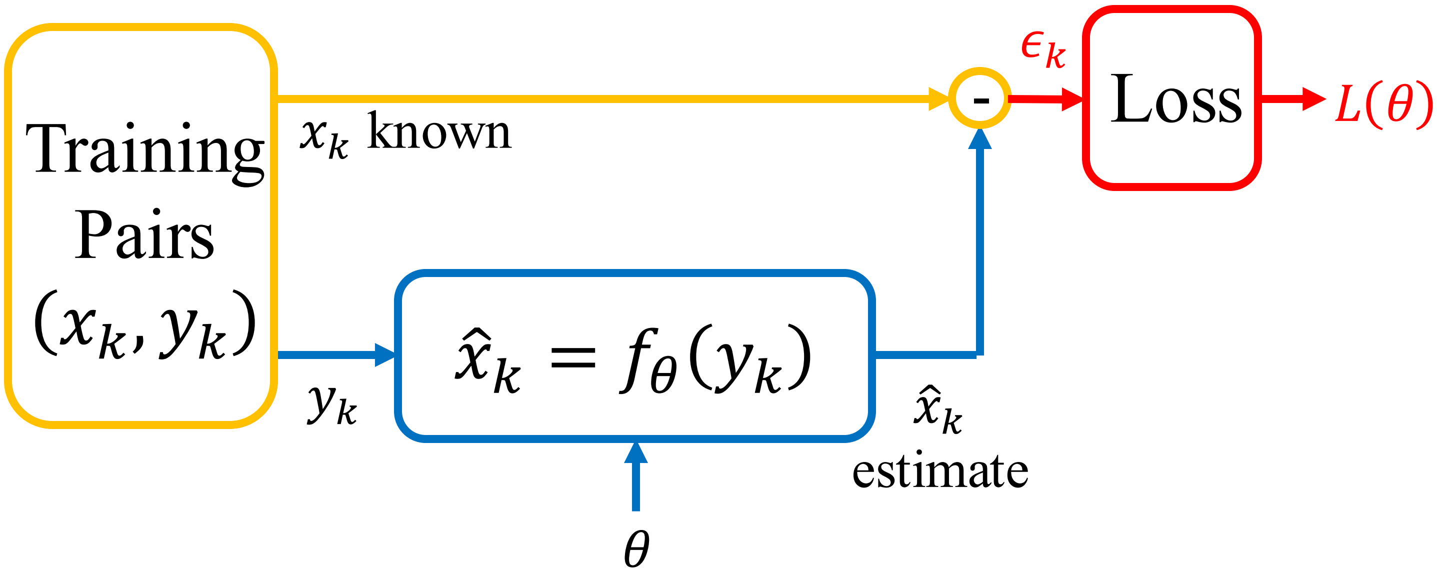
- Measuring supervised training error
- Mathematical representation
- Parameter estimation through loss minimization
- Inference and training as inverse problems

Machine Learning: Supervised Training



- Training:
  - Measure x too! This produces training data.
    - Collecting training data is application specific
    - It can be difficult, expensive, or even impossible
  - Select θ so that $\epsilon = x - \hat{x}$ is small
  - How do we measure “small”?

ML: Supervised Training



- Find lots of training data
  - (x_k, y_k) for $k = 0, \cdots, K-1$
- Define a loss function $L(\theta)$
- Pick θ to minimize $L(\theta)$

The ML Loss Function

- What is a loss function?
  - The loss function is a measure of training error
  - So for example, $x_k \in \Re^{N_x}$

$$Loss = \underbrace{L(\theta) = \frac{1}{K}\sum_{k=0}^{K-1} \|x_k - \hat{x}_k\|^2}_{\text{But wait! Gobbligood Alert!}} = \frac{1}{K}\sum_{k=0}^{K-1}\sum_{i=0}^{N_y-1}\left(x_{k,i} - \hat{x}_{k,i}\right)^2$$

  - What I really mean is

$$Loss = L(\theta) = \frac{1}{K}\sum_{k=0}^{K-1} \|x_k - f_\theta(y_k)\|^2$$

Loss Function Properties

$$L_{MSE}(\theta) = \frac{1}{K}\sum_{k=0}^{K-1} \|x_k - f_\theta(y_k)\|^2$$

- Facts:
  - Usually called mean squared error (MSE). But technically, MSE is really defined as
    $$\text{MSE} = E[\|x_k - f_\theta(y_k)\|^2] \approx L_{MSE}(\theta)$$
    So this should really be called Total Squared Error (TSE), but what are you going to do…
  - When $L(\theta) = 0$, then for all k, $x_k = f_\theta(y_k)$.

Parameter Estimation using Loss Minimization

$$\theta^* = \arg\min_{\theta}\{L_{MSE}(\theta)\} = \arg\min_{\theta}\left\{\frac{1}{K}\sum_{k=0}^{K-1}\|x_k - f_\theta(y_k)\|^2\right\}$$

- Estimate θ^* by minimizing loss

What does this mean?

$$\theta^* = \arg\min_{\theta}\{l_{MSE}(\theta)\}$$

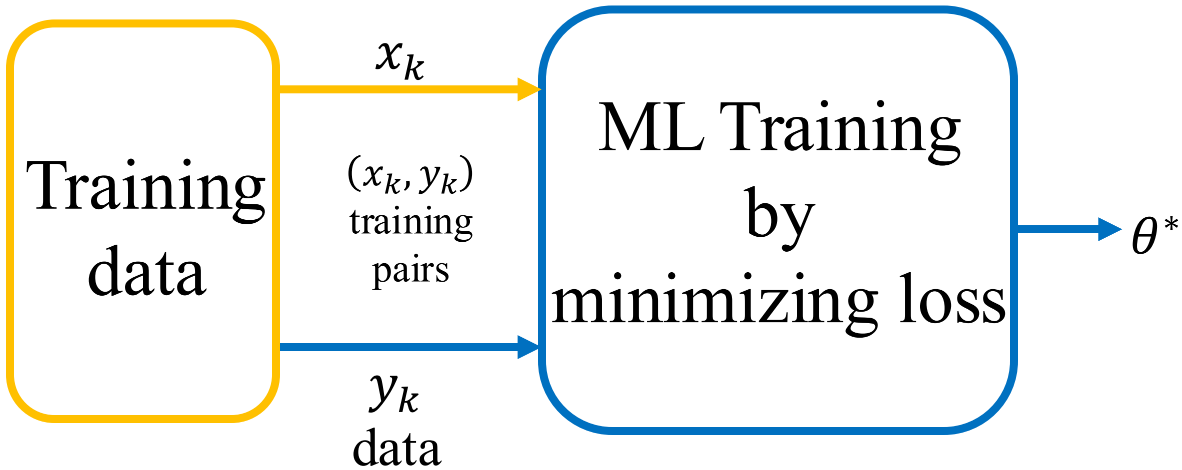
Question: What is the lowest point in the USA?

$$\begin{matrix}\text{Death Valley}\\ \text{Badwater Basin}\end{matrix} = \arg\min_{\theta\in USA}[Altitude(\theta)]$$

$$-282\text{ feet} = \min_{\theta\in USA}[Altitude(\theta)]$$

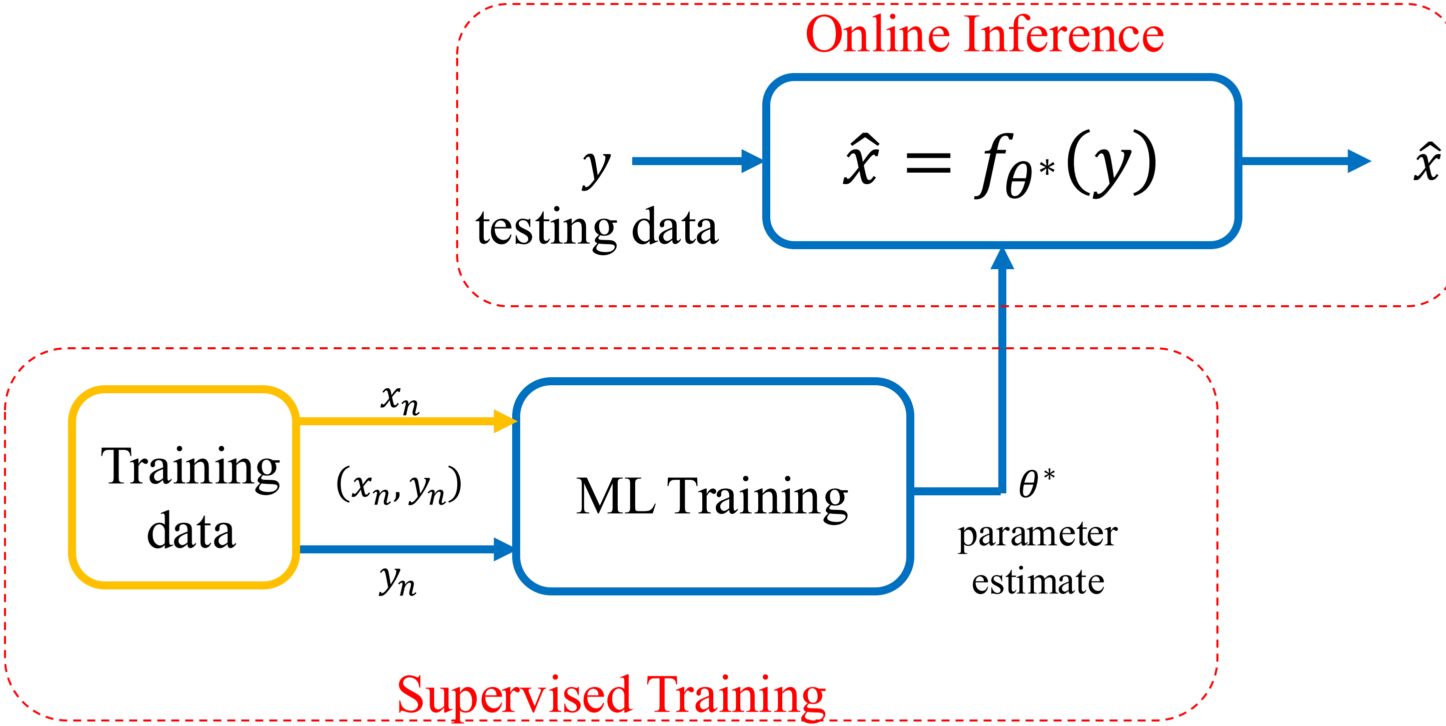
- "*min*" returns the minimum value
- "*arg min*" returns the parameter that minimizes the value

ML Supervised Training



- Generate training data:
  - Collect K training pairs, (x_0, y_0) , (x_1, y_1) , ⋯, (x_{K-1}, y_{K-1})
- Estimate parameter:
  - $\theta^* = \arg\min_{\theta}\{L_{MSE}(\theta)\}$
- This is also an inverse problem!
  - Trying to find the "hidden" or "latent" value of θ.

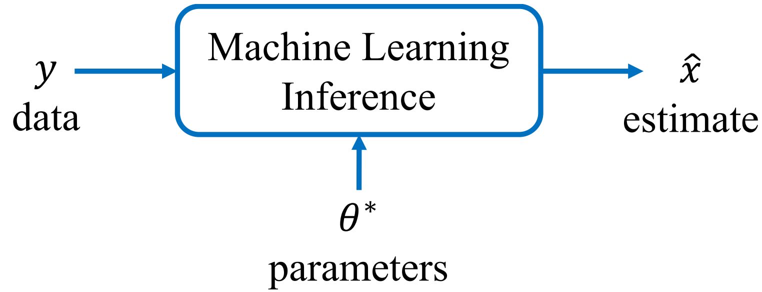
Complete ML System



- Supervised training:
  - Off-line and often slow
  - Requires expensive training data
- Online inference:
  - On-line and usually needs to be fast

Two Inverse Problems in ML

- Inference inverse problem:
  - Estimate the unknown, $\hat{x}$, from the available measurements, y.



- Training inverse problem:
  - Estimate the unknown parameter, θ^*, from the training pairs, $(x_n, y_n)|_{n=0}^{N-1}$