CS 190I
Deep Learning
Logistic Regression

Lei Li (leili@cs)
UCSB

Acknowledgement: Slides borrowed from Bhiksha Raj's 11485 and Mu Li & Alex Smola's 157 courses on Deep Learning, with modification

Reminder

- Course website:
  - https://sites.cs.ucsb.edu/~leili/course/dl23w/
- Homework 1 due 11am Jan 25 (till midnight),
  - Please prepare your solution PDF using LaTeX or Word or GoogleDoc (to make it clear and rigorous)
  - Handwritten and scanned image will not be accepted.
  - Submit to Gradescope. (please let me know immediately if you do not have access)
- Everyone enrolled should submit answer for in-class quiz.
  - Class quiz counts 5%. Due at 11pm of the class day.
  - DSP students are allowed extra time for quiz.
  - Participation: answering questions on Edstem counts 5%

Recap

- Machine learning is the study of machines that can improve their performance with more experience
- Linear Regression Model
  – Output is linearly dependent on the input variables
  – Minimize squared loss

Linear Regression

- Add bias into weights by

$$\mathbf{X} \leftarrow [\mathbf{X}, \mathbf{1}] \quad \mathbf{w} \leftarrow \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$$

$$\ell(\mathbf{X}, \mathbf{y}, \mathbf{w}) = \frac{1}{n} \| \mathbf{y} - \mathbf{X}\mathbf{w} \|^2$$

- Loss is convex, so the optimal solutions satisfies

$$\frac{\partial}{\partial \mathbf{w}} \ell(\mathbf{X}, \mathbf{y}, \mathbf{w}) = 0$$

$$\Leftrightarrow \mathbf{w}^* = \left(\mathbf{X}^T \mathbf{X}\right)^{-1} \mathbf{X}\mathbf{y}$$

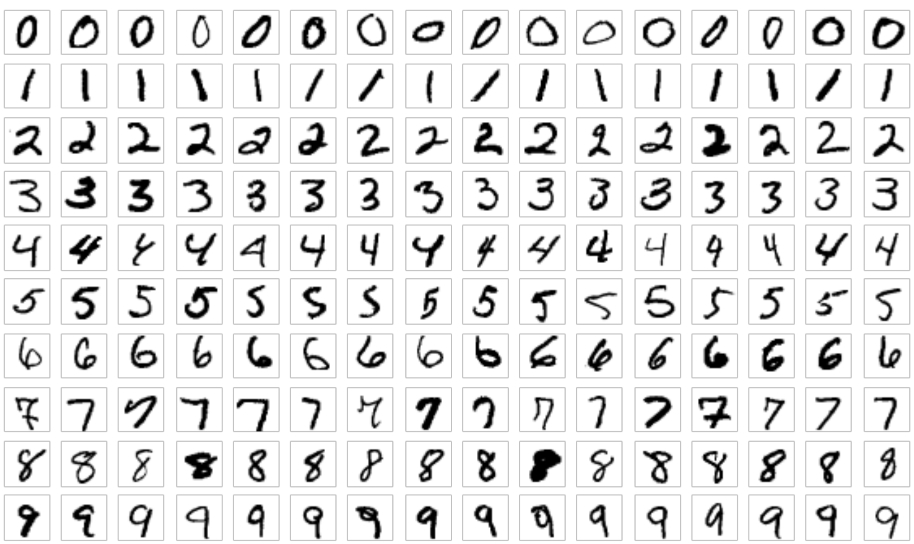
Quiz-3.1

- https://edstem.org/us/courses/31035/lessons/53467/

Regression vs. Classification

- Regression estimates a continuous value
- Classification predicts a discrete category

MNIST: classify hand-written digits (10 classes)



ImageNet: classify nature objects (1000 classes)

Cat

Dog

Handwriting Recognition

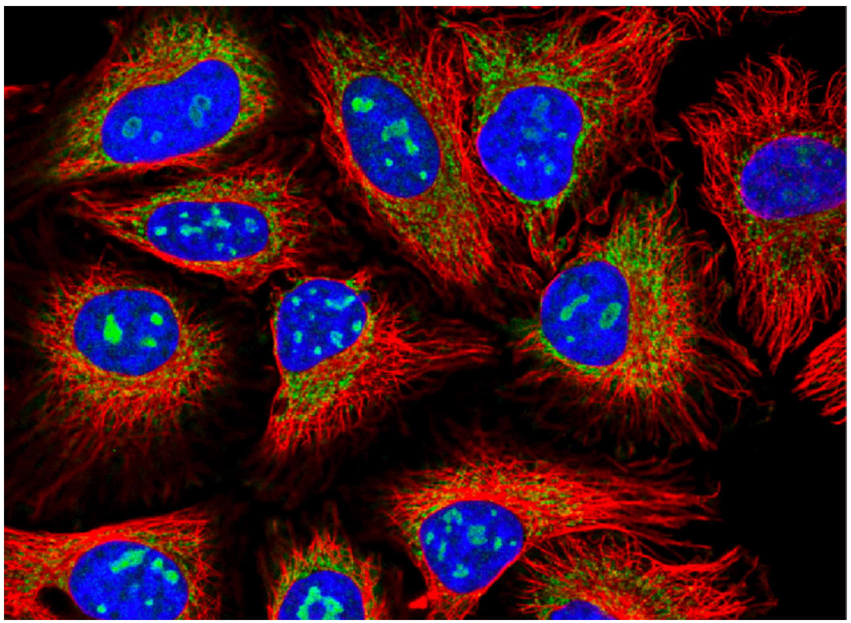
Optical Character Recognition (OCR)

?

0
1
2
3
4
5
6
7
8
9

Classifying Protein

Classify human protein microscope images into 28 categories



0. Nucleoplasm
1. Nuclear membrane
2. Nucleoli
3. Nucleoli fibrillar
4. Nuclear speckles
5. Nuclear bodies
6. Endoplasmic reticu
7. Golgi apparatus
8. Peroxisomes
9. Endosomes
10. Lysosomes
11. Intermediate fila
12. Actin filaments
13. Focal adhesion si
14. Microtubules
15. Microtubule ends
16. Cytokinetic bridg

https://www.kaggle.com/c/human-protein-atlas-image-classification

Text Classification

Classifying the sentiment of online movie reviews. (Positive, negative, neutral)

Spider-Man is an almost-perfect extension of the experience of reading comic-book adventures. 👍

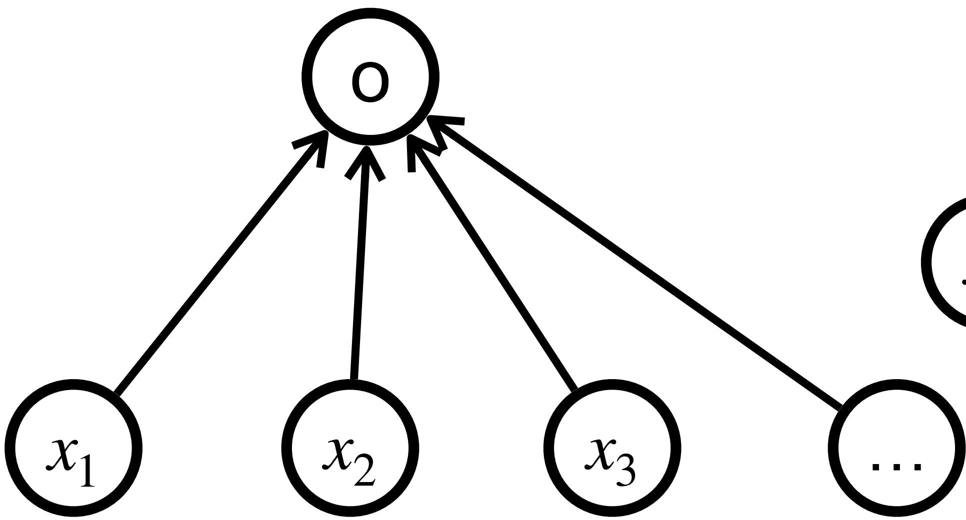
The acting is decent, casting is good. 👍

It was a boring! It was a waste of a movie to even be made. It should have been called a family reunion. 👎

From Regression to Multi-class Classification

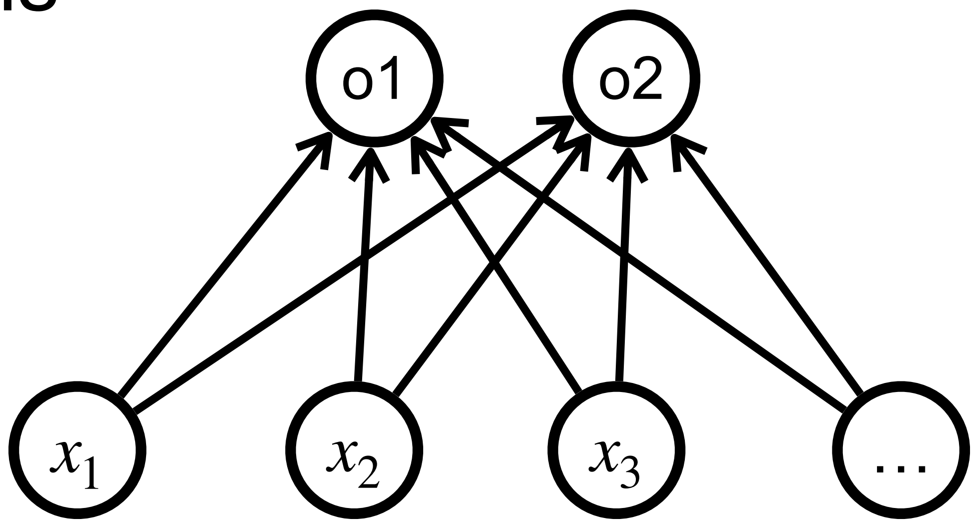
Regression

- Single continuous output
- Natural scale in
- Loss given e.g. in terms of difference



Classification

- Discrete output
- Score *should* reflect confidence/uncertainty …



From Regression to Multi-class Classification

Square Loss

- One hot encoding per class

$$\mathbf{y} = [y_1, y_2, \ldots, y_n]^\top$$

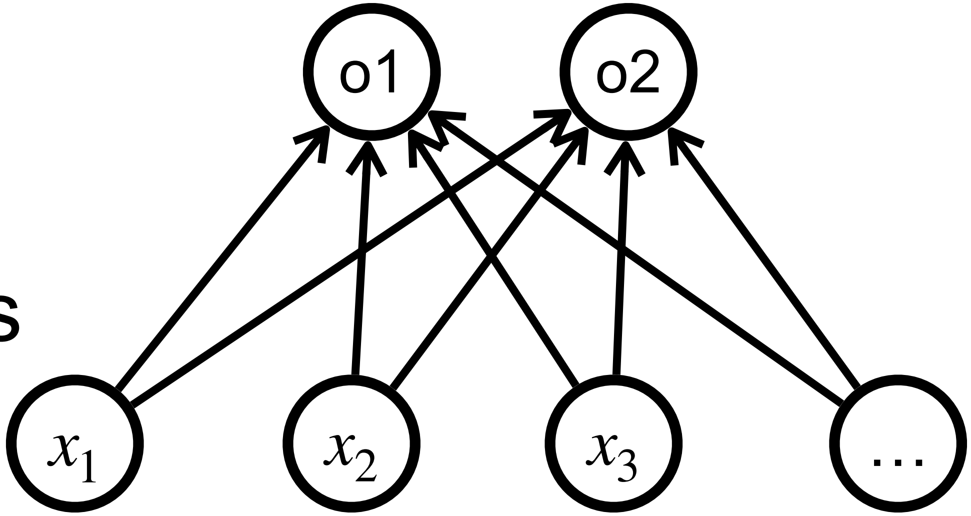
$$y_i = \begin{cases} 1 \text{ if } i = y \\ 0 \text{ otherwise} \end{cases}$$

- Train with squared loss
- Largest output wins

$$\hat{y} = \underset{i}{\operatorname{argmax}}\, o_i$$

Classification

- Discrete output
- Score *should* reflect confidence/uncertainty …



But, is there better way to model?

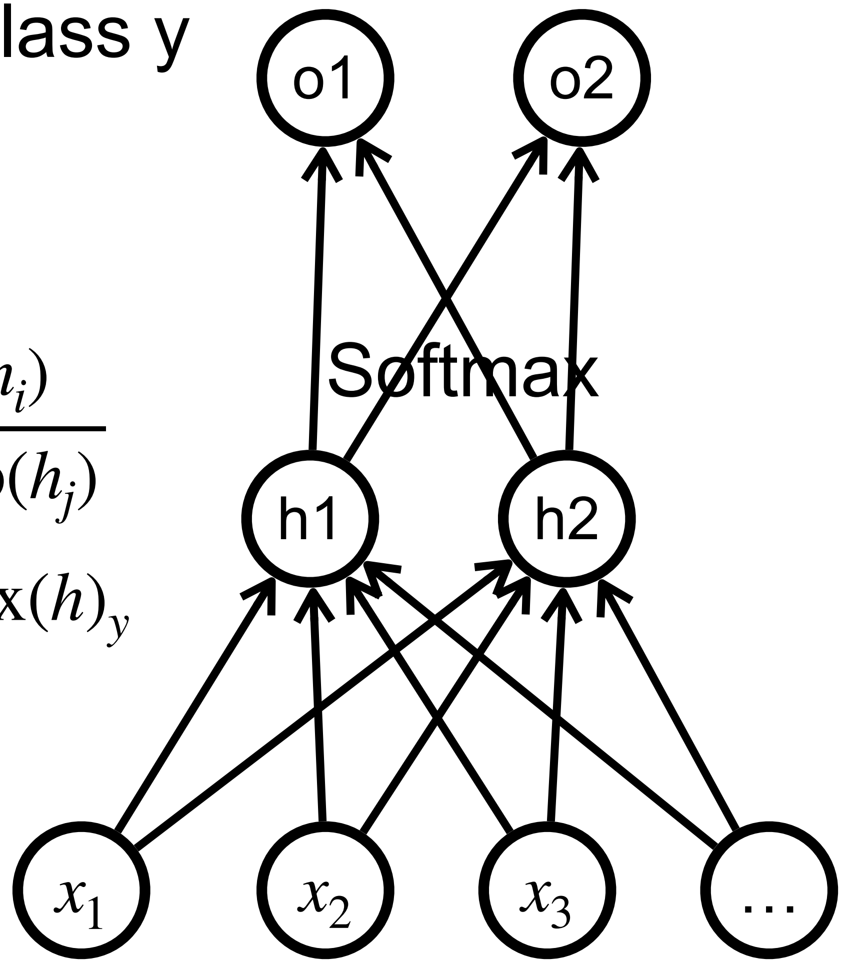
Logistic Regression

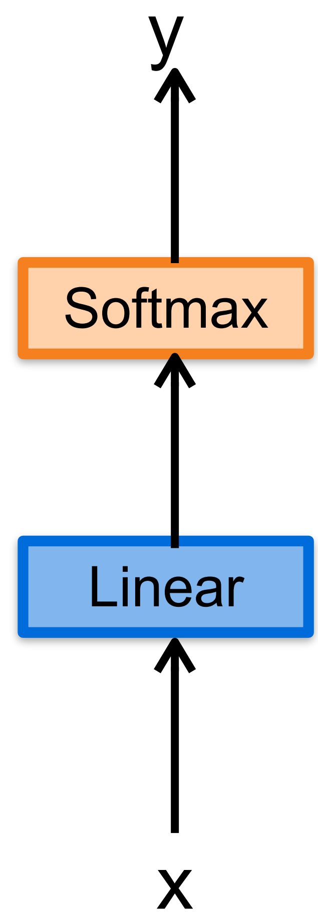
output: prob. of class y

$$h = \mathbf{W} \cdot \mathbf{x}$$

$$\mathrm{softmax}(h)_i = \frac{\exp(h_i)}{\sum_j \exp(h_j)}$$

$$p(y \mid h) = \mathrm{softmax}(h)_y$$





Logistic Regression in Pytorch

```
class LogisticRegression(torch.nn.Module):
    def __init__(self, input_dim, output_dim):
        super(LogisticRegression, self).__init__()
        self.linear = torch.nn.Linear(input_dim, output_dim)

    def forward(self, x):
        outputs = torch.sigmoid(self.linear(x))
        return outputs
```

Maximum Likelihood Estimation

$$\hat{\theta} = \arg\max \mathscr{L}(\theta; D)$$

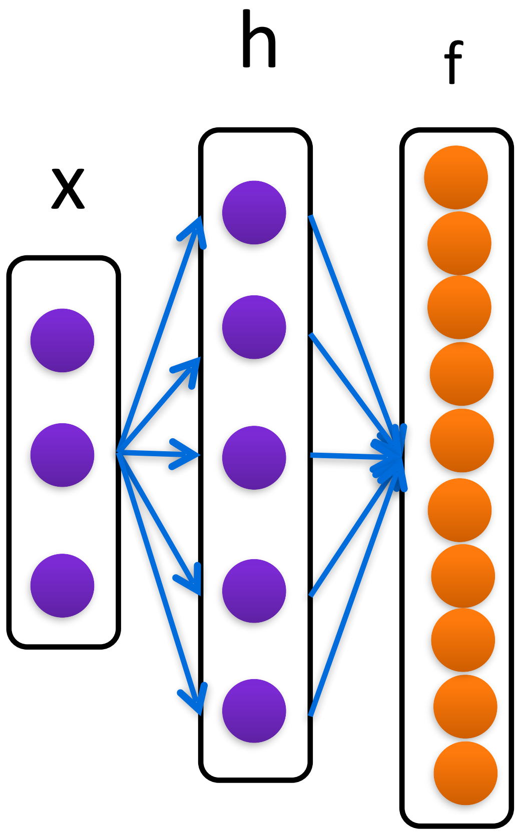
$\mathscr{L}$ is the log-likelihood function

$$\mathscr{L}(\theta; D) = \frac{1}{N} \sum_{n=1}^{N} \log p(y_n \mid x_n; \theta)$$

Or. equivalent to minimize negative log-likelihood

$$\hat{\theta} = \arg\min \ell(\theta; D) = -\frac{1}{N} \sum_{n=1}^{N} \log p(y_n \mid x_n; \theta)$$

Loss for Classification: Cross-Entropy



$$\mathscr{L}(\theta) = \frac{1}{N}\sum_{n=1}^{N} \ell(y_n, f(x_n); \theta)$$

$$\ell(y_n, f(x_n)) = H(y_n, f(x_n)) = -\log f(x_n)_{y_n}$$

$f(x_n)$ is a vector (e.g. $\in R^{10}$), representing predicted distribution

y_n is the ground-truth label, can be represented as an one-hot "distribution" [0,…,0, 1, 0,…,0]

Cross-entropy

$$H(p, q) = -\sum_k p_k \log q_k$$

Maximum Likelihood and Cross-Entropy

MLE

$$\boxed{\max \frac{1}{N}\sum_{n=1}^{N}\log p(y_n \mid x_n;\theta) = \frac{1}{N}\sum_{n=1}^{N}\sum_{k} y_{n,k} f(x_n)_k}$$

Or equivalently, minimize CE loss

$$\boxed{\min \mathscr{L}(\theta) = \frac{1}{N}\sum_{n=1}^{N} H(y_n, f(x_n)) = \frac{1}{N}\sum_{n=1}^{N} -\log f(x_n)_{y_n}}$$

Cross-Entropy Loss with Softmax

- Negative log-likelihood (for given label y)

$$-\log p(y \mid h) = \log \sum_i \exp(h_i) - h_y$$

- Cross-Entropy Loss (the true label y is an one-hot vector)

$$\ell(y, h) = \log \sum_i \exp(h_i) - y^\top h$$

- **Gradient**

$$\partial_h \ell(y, h) = \frac{\exp(h)}{\sum_i \exp(h_i)} - y$$

Difference between true and estimated

Quiz-3

https://edstem.org/us/courses/16390/lessons/27551/slides/156087

Compute the cross-entropy loss for the prediction prob.

Cat	0.6	0.2	0.4
Dog	0.1	0.8	0.05
Tiger	0.3	0	0.55

Training and Evaluation

Training

- The procedure to obtain optimal parameters using a set of data so that the error on the data is minimum
- Training data:
  - a set of pairs <x, y>
- Objective (training loss):
  - Cross-entropy for logistic regression

A simple algorithm for Logistic Regression

1. Randomly initialize parameters w
2. Repeat until convergence
    1) g = 0
    2) for each data point x_n, y_n
        (1) calculate h_n = w * x_n
        (2) calculate p_n = Softmax(h_n)
        (3) calculate err_n = p_n - y_n
        (4) multiply g_n = err_n * x_n^T
        (5) g += g_n
    3) update w = w - g
3. output w

Inference

- After training a model, given an input data x, to compute the prediction for output y
- For regression:
  - just model output
- For classification: output the class w/ max prob.
  - $\hat{y} = \arg\max_i f(x)_i$
- Need to do inference for validation and testing

Inference Example

Cat	0.6	0.2	0.4
Dog	0.1	0.8	0.05
Tiger	0.3	0	0.55

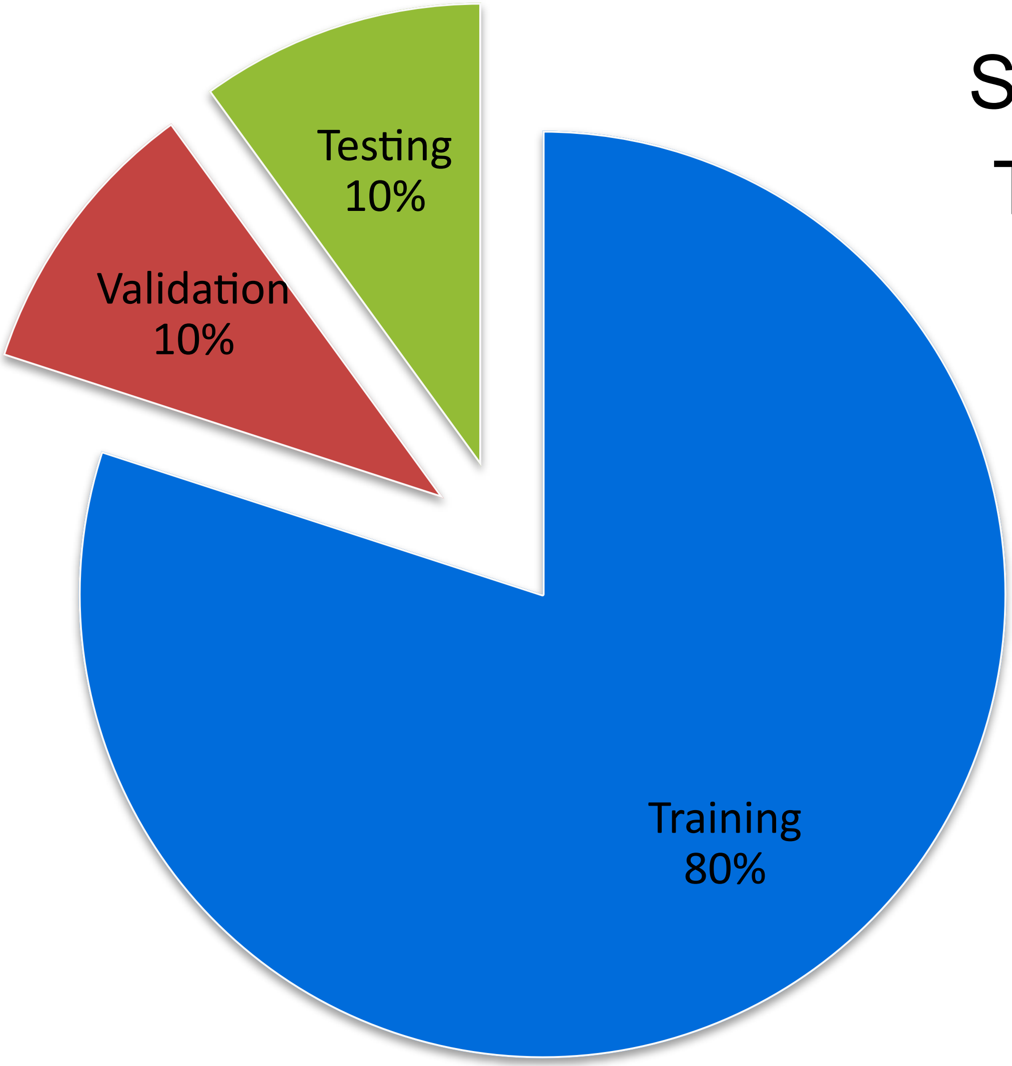
Training and Generalization

- Training error (=empirical risk, next lecture): model prediction error on the training data
- Generalization error (= expected risk, next lecture): model error on new unseen data over full population
- Example: practice a GRE exam with past exams
  - Doing well on past exams (training error) doesn't guarantee a good score on the future exam (generalization error)
  - Student A gets 0 error on past exams by rote learning
  - Student B understands the reasons for given answers

Validation Dataset and Test Dataset

- Validation dataset: a dataset used to evaluate the model performance
  - E.g. Take out 50% of the training data
  - Should not be mixed with the training data (#1 mistake)
- Test dataset: a dataset can be used once, e.g.
  - A future exam
  - The house sale price I bided
  - Dataset used in private leaderboard in Kaggle

Training/Validation/Testing



Similar to learning a course

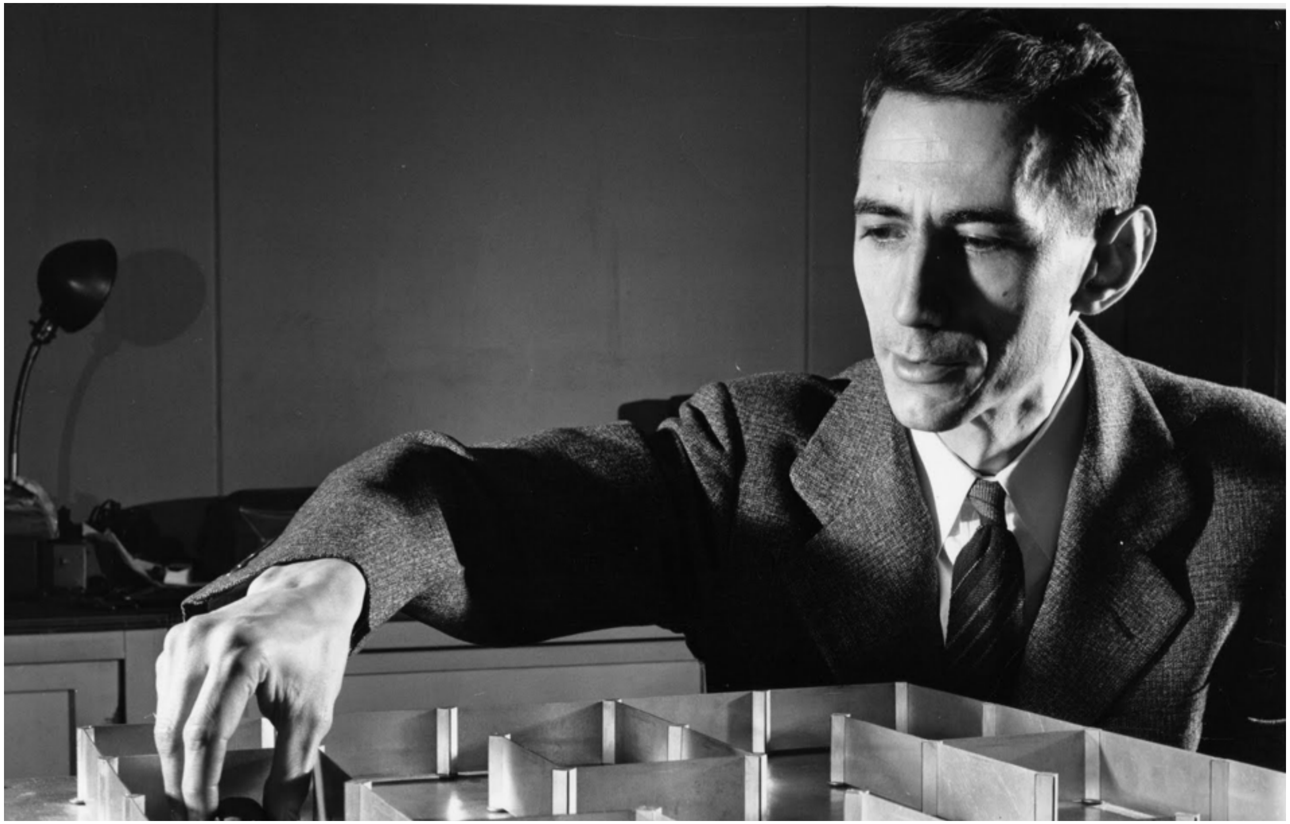
Training: doing homework

Validation: mock-exam

Testing: real final exam

Never use testing data in training!

Information Theory



Claude Shannon

Entropy

- Data source producing observations $x_1 \ldots x_n$
- **How much 'information' is in this source?**
  - Tossing a fair coin - at each step the surprise is whether it's heads or tails
  - Rolling a fair dice - we have 1 out of 6 outcomes. This should be *more* surprising than the coin
  - Picture of a white wall vs. picture of a football stadium (the football stadium should have more information)
- **Measure is minimum number of bits needed**

Entropy

- Data source producing data $x_1 \ldots x_n$ with probability $p(x)$
- **Definition** $H[p] = -\sum_j p_j \log p_j$
- **Coding theorem**
  Entropy is lower bound on bits (or rather nats - base e) $2^a = e^b$ hence $a \log 2 = b$ hence $\text{bits} = \frac{H[p]}{\log 2}$

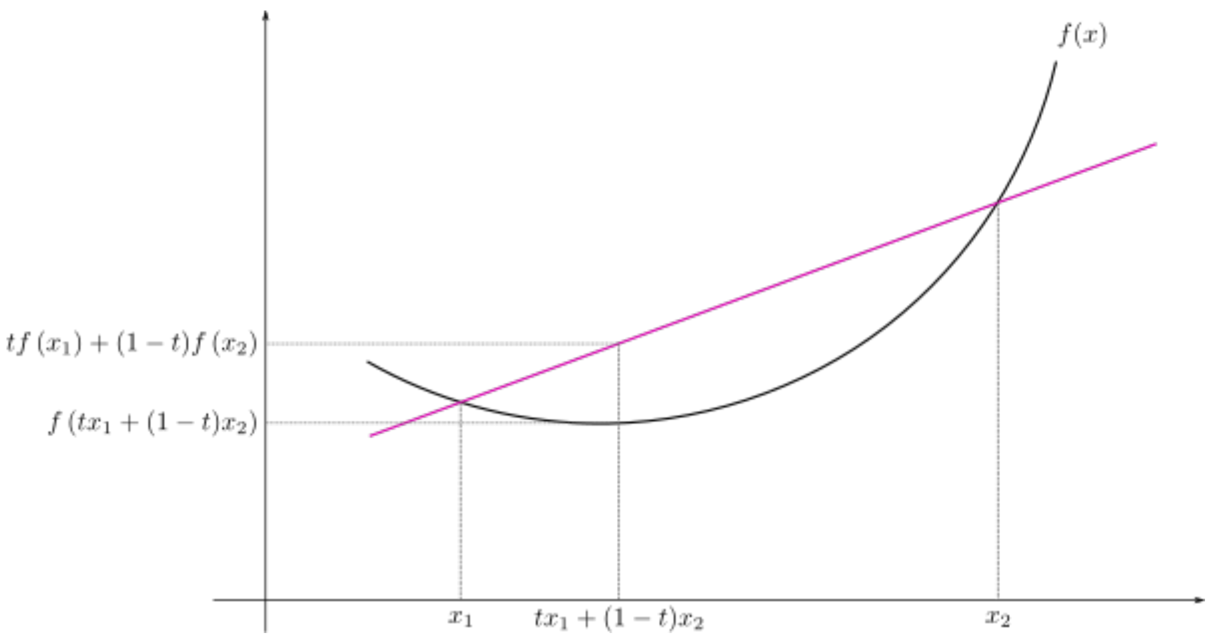
$$H[\lambda p + (1-\lambda)q] \geq \lambda H[p] + (1-\lambda)H[q]$$

- Entropy is concave

Convex Function

f is convex iff

for all $0 < t < 1$, and all $x_1 \neq x_2$
$$tf(x_1) + (1-t)f(x_2) \geq f\left(tx_1 + (1-t)x_2\right)$$



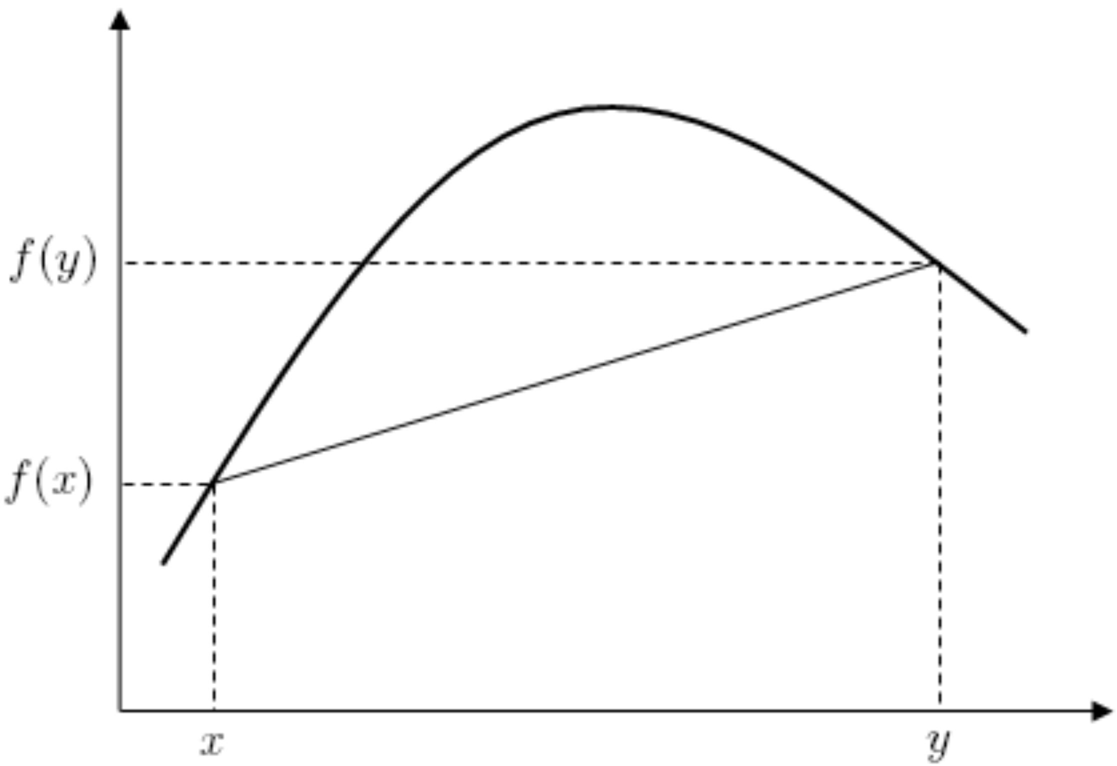
Convex function is very useful in optimization.

Concave Function

f is concave iff

for all $0 < t < 1$, and all $x_1 \neq x_2$

$$tf(x_1) + (1-t)f(x_2) \leq f\left(tx_1 + (1-t)x_2\right)$$



Entropy (binary form)

- Fair coin (p = 0.5)

$$H[p] = -0.5 \cdot \log_2 0.5 - 0.5 \cdot \log_2 0.5 = 1 \text{ bit}$$

- Biased coin (p = 0.9)

$$H[p] = -0.9 \cdot \log_2 0.9 - 0.1 \cdot \log_2 0.1 = 0.47 \text{ bit}$$

- Dungeons and Dragons (20-sided dice)

$$H[p] = -\log_2 \frac{1}{20} = 4.32 \text{ bit}$$

Kullback-Leibler Divergence

- Distance between distributions (e.g. truth & estimate)

Number of extra bits when using the wrong code

$$D[p\|q] = \int dp(x) \log \frac{p(x)}{q(x)} = \int dp(x) \underbrace{\left[-\log q(x)\right]}_{\text{Inefficient bits}} - \underbrace{\left[-\log p(x)\right]}_{\text{Optimal bits}}$$

- Nonnegativity of KL Divergence

Jensen Inequality
log is concave

$$D[p\|p] = \int dp(x) \log \frac{p(x)}{p(x)} = 0$$

$$D[p\|q] = -\int dp(x) \log \frac{q(x)}{p(x)} \geq -\log \int dp(x) \frac{q(x)}{p(x)} = 0$$

Minimizing Cross-Entropy is equivalent to Minimizing the KL divergence!

- Cross entropy loss

$$\ell(y, x) = H(y, f(x)) = -\log f(x)_y$$

- Cross entropy loss for softmax

$$\ell(y, h(x)) = \log \sum_i \exp(h(x)_i) - y^\top h(x)$$

- Kullback Leiber divergence

$$\begin{aligned} D(q \| p(\hat{y} \mid x)) &= D(q \| \mathrm{softmax}(h(x))) \\ &= \sum_i q_i \log q_i - q_i \log \mathrm{softmax}(h(x))_i \\ &= -H[q] + \log \sum_i \exp(h(x)_i) - \sum_i q_i h(x)_i \end{aligned}$$

Independent of h()x

Recap

- The smallest number of bits to encode message is lower-bounded by entropy
- Minimizing cross entropy is equivalent to minimizing Kullback-Leibler Divergence

Next Up

- More powerful model: Multilayer Perceptron / Feedforward Network
- How to train general neural network from data
- MP1 is out today: start early!
- Friday recitation:
  - More Linear algebra and gradient calculation
  - Next Friday: mini-tutorial on pytorch and training on servers.