165B
Machine Learning
Linear Models

Lei Li (leili@cs)
UCSB

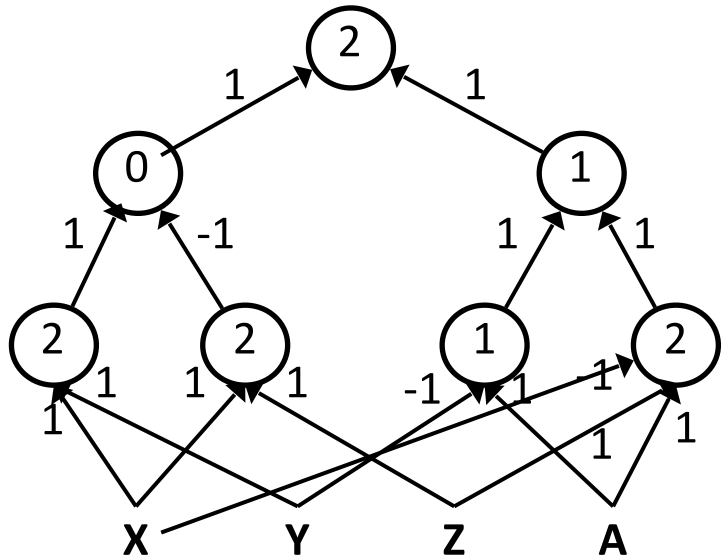
Acknowledgement: Slides borrowed from Bhiksha Raj's 11485 and Mu Li & Alex Smola's 157 courses on Deep Learning, with modification

Recap

- Neural networks began as computational models of the brain
- Neural network models are connectionist machines
  - The comprise networks of neural units
- Neural Network can model Boolean functions
  - McCullough and Pitt model: Neurons as Boolean threshold units
  - Hebb's learning rule: Neurons that fire together wire together
  - Rosenblatt's perceptron : A variant of the McCulloch and Pitt neuron with a provably convergent learning rule
    - But individual perceptrons are limited in their capacity (Minsky and Papert)
  - Multi-layer perceptrons can model arbitrarily complex Boolean functions

A model for boolean function

$$((A\bar{X}Z) \mid (A\bar{Y}))((X\;\;Y) \mid (\bar{X}Z))$$



Neural Network

- A network is a function
  - Given an input, it computes the function layer wise to predict an output
    - More generally, given one or more inputs, predicts one or more outputs
- Given a labeled dataset $\{(x_n, y_n)\}$, how to train a model that maps from x —> y
- Idea: develop a complex model using massive basic simple units

What is Deep Learning

- Deep learning is a particular kind of machine learning
- that achieves great power and flexibility by representing the world as a nested hierarchy of concepts,
- with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones.

Ian Goodfellow and Yoshua Bengio and Aaron Courville. Deep Learning, 2016

What is Machine Learning?

- 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 tasks in T, as measured by P, improves with experience E”
  - [Tom Mitchell, Machine Learning, 1997]

How to build a Machine Learning system

- Task T:
  - What is input and output?
- Experience E:
  - What is training data? How to get them easily?
- Performance Measure P
  - How to measure success
- Model:
  - What is the computational architecture?
- Training:
  - How to improve with experience?
  - What is the loss?

Task T

- To find a function f: x -> y
  - Classification: label y is categorical
  - Regression: label y is continuous numerical
- Example:
  - Image classification
    - Input space: x in $R^{h \times h \times 3}$ is h x h pixels (rgb), so it is a tensor of h x h x 3.
    - Output space: y is {1..10} in Cifar-10, or {1..1000} in ImageNet.
  - Text-to-Image generation
    - Input: x is a sentence in V^L, V is vocabulary, L is length
    - Output: y is $R^{h \times h \times 3}$

Neural Networks that map input to output

我很高兴 → Machine Translation → I am very happy

→ Automatic Speech Recognition → Hi, Siri, please turn on the light

→ Image Recognition → Cat

Experience E

- **Supervised Learning**: if pairs of (x, y) are given
- **Unsupervised Learning**: if only x are given, but not y
- **Semi-supervised Learning**: both paired data and raw data
- **Self-supervised Learning**:
  - use raw data but construct supervision signals from the data itself
  - e.g. to predict neighboring pixel values for an image
  - e.g. to predict neighboring words for a sentence

How Experience is Collected?

- Offline/batch Learning:
  - All data are available at training time
  - At inference time: fix the model and predict
- Online Learning:
  - Experience data is collected one (or one mini-batch) at a time (can be either labeled or unlabeled)
  - Incrementally train and update the model, and make predictions on the fly with current and changing model
  - e.g. predicting ads click on search engine
- Reinforcement Learning:
  - A system (agent) is interacting with an environment (or other agents) by making an action
  - Experience data (reward) is collected from environment.
  - The system learns to maximize the total accumulative rewards.
  - e.g. Train a system to play chess

Learning w/ various Number of Tasks

- Multi-task learning
  - one system/model to learn multiple tasks simultaneously, with shared or separate Experience, with different performance measures
  - e.g. training a model that can detect human face and cat face at the same time
- Pre-training & Fine-tuning
  - Pre-training stage: A system is trained with one task, usually with very large easily available data
  - Fine-tuning stage: it is trained on another task of interest, with different (often smaller) data
  - e.g. training an image classification model on ImageNet, then finetune on object detection dataset.

Machine Translation as a Machine Learning Task

- Input (Source)
  - discrete sequence in source language, V_s
- Output (Target)
  - discrete sequence in target langauge, V_t
- Experience E
  - Supervised: parallel corpus, e.g. English-Chinese parallel pairs
  - Unsupervised: monolingual corpus, e.g. to learn MT with only Tamil text and English text, but no Eng-Tamil pairs
  - Semi-supervised: both
- Number of languages involved
  - Bilingual versus Multilingual MT
  - Notice: it can be multilingual parallel data, or multilingual monolingual data
- Measure P
  - Human evaluation metric, or Automatic Metric (e.g. BLEU), see previous lecture

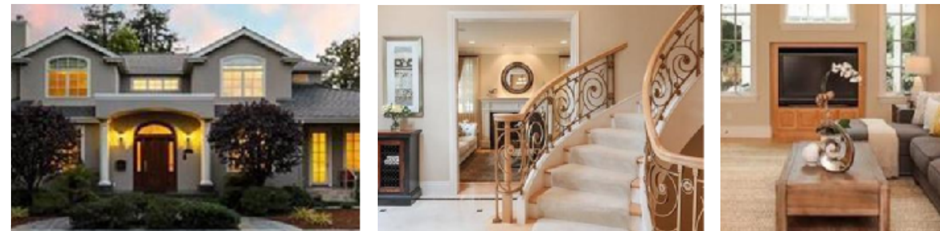
Story so far

- **Machine learning is the study of machines that can improve their performance with more experience**

Linear Models

House Buying

- Pick a house, take a tour, and read facts
- Estimate its price, bid



Listing

$5,498,000	7	5	4,865 Sq. Ft.
Price	Beds	Baths	$1130 / Sq. Ft.

Redfin Estimate: $5,390,037 On Redfin: 15 days

Predicte

Virtual Tour
- Branded Virtual Tour
- Virtual Tour (External Link)

Parking Information
- Garage (Minimum): 2
- Garage (Maximum): 2
- Parking Description: Attached Garage, On Street
- Garage Spaces: 2

Multi-Unit Information
- # of Stories: 2

School Information
- Elementary School: El C
- Elementary School Dist
- Middle School: Jane Lat
- High School: Palo Alto H
- High School District: Pal

Interior Features

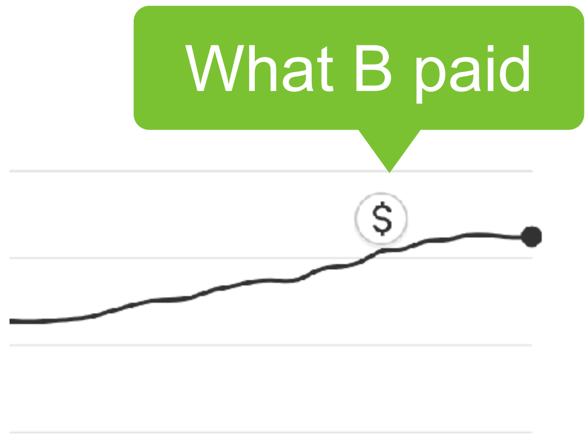
Bedroom Information
- # of Bedrooms (Minimum): 7

- Kitchen Description: Co
Dishwasher, Garbage Di
Island with Sink, Microw

House Price Prediction

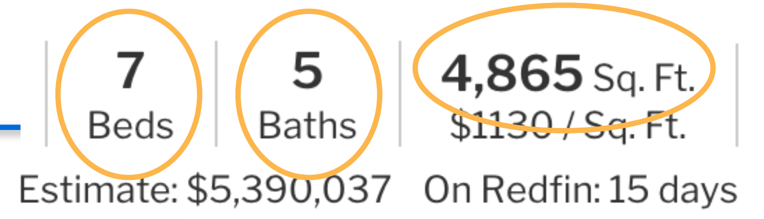
Very important, that's real money…

$100K+



Redfin overestimated the price, and B believed it

A Simplified Model



- Assumption 1
  The key factors impacting the prices are #Beds, #Baths, Living Sqft, denoted by x_1, x_2, x_3
- Assumption 2
  The sale price is a weighted sum over the key factors $y = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$

  Weights and bias are determined later

Linear Model (Linear Regression)

- Given *n*-dimensional inputs
  $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$
- Linear model has a *n*-dimensional weight and a bias $\quad \mathbf{w} = [w_1, w_2, \ldots, w_n]^T, \quad b$

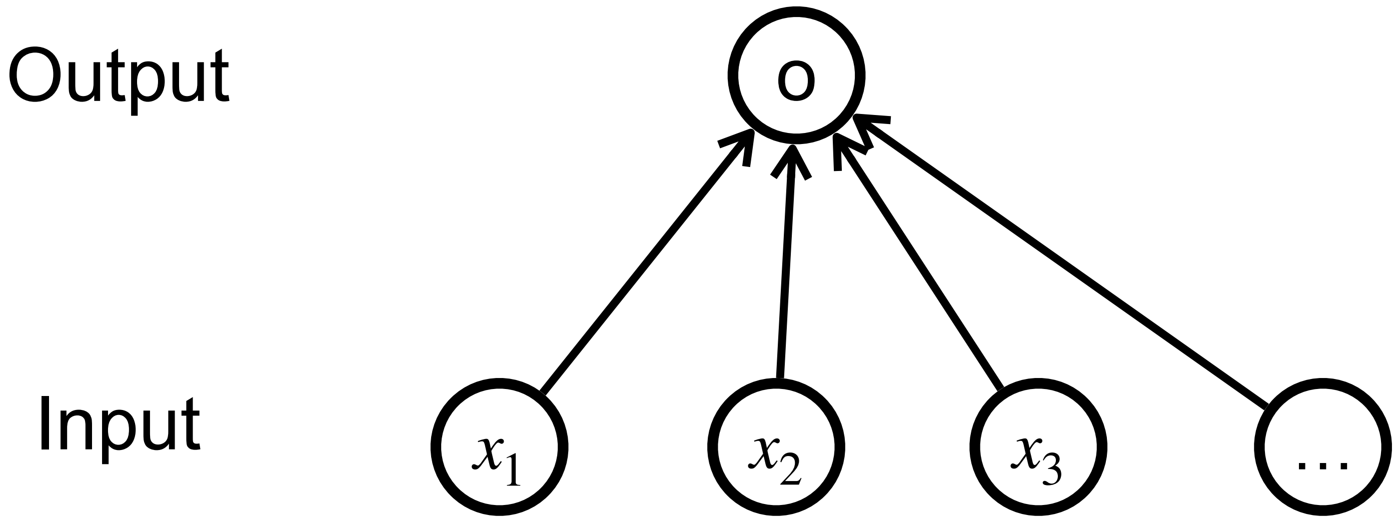
$$y = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b$$

- The output is a weighted sum of the inputs

$$y = \langle \mathbf{w}, \mathbf{x} \rangle + b$$

Vectorized version

Linear Model as a Single-layer Neural Network



Measure Estimation Quality

- Compare the true value vs the estimated value
  Real sale price vs estimated house price
- Let y the true value, and $\hat{y}$ the estimated value, we can compare the loss

$$\ell(y, \hat{y}) = \left(y - \hat{y}\right)^2$$

  It is called squared loss

Training Data

- Collect multiple data points to fit parameters
  Houses sold in the last 6 months
- It is called the training data
- The more the better
- Assume n examples $D = \{\langle x_n, y_n \rangle\}$

$$\mathbf{X} = \left[\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n\right]^T$$

$$\mathbf{y} = \left[y_0, y_1, \ldots, y_n\right]^T$$

Training Objective

- Training loss

$$\ell(\mathbf{X}, \mathbf{y}, \mathbf{w}, b) = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \langle \mathbf{x}_i, \mathbf{w} \rangle - b\right)^2 = \frac{1}{n} \parallel \mathbf{y} - \mathbf{X}\mathbf{w} - b \parallel^2$$

- Minimize loss to learn parameters

$$\mathbf{w}^*, \mathbf{b}^* = \arg\min_{\mathbf{w}, b} \ell(\mathbf{X}, \mathbf{y}, \mathbf{w}, b)$$

Norm

- A "distance" metric
- l1 norm
  - $\| x \|_1 = |x_1| + |x_2| + \cdots$
- l2 norm
  - $\| x \| = \sqrt{x_1^2 + x_2^2 + \cdots}$
- lp norm
  - $\| x \|_p = (x_1^p + x_2^p + \cdots)^{\frac{1}{p}}$

Closed-form Solution

- Add bias into weights by

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

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

- Loss is convex, so the optimal solutions satisfies

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

$$\Leftrightarrow \frac{2}{n} \left(\mathbf{y} - \mathbf{X}\mathbf{w}\right)^T \mathbf{X} = 0$$

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

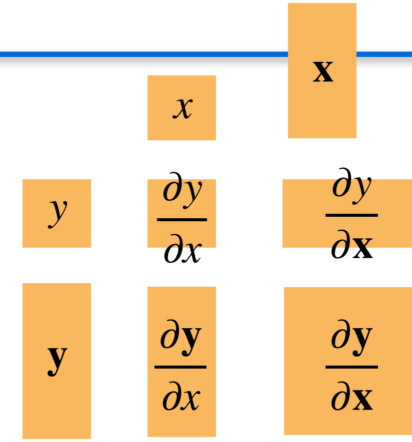
Matrix Calculus

Gradients

- Generalize derivatives into vectors

	Scalar x	Vector $\mathbf{x}$
Scalar y	$\frac{\partial y}{\partial x}$	$\frac{\partial y}{\partial \mathbf{x}}$
Vector $\mathbf{y}$	$\frac{\partial \mathbf{y}}{\partial x}$	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$

Gradients of vector functions

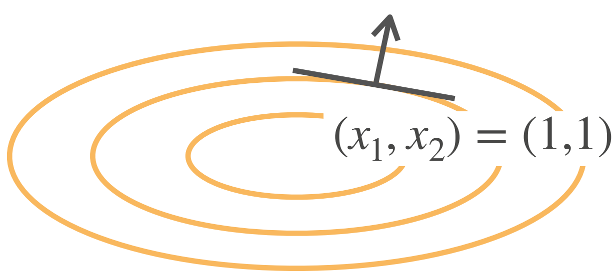


$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad \frac{\partial y}{\partial \mathbf{x}} = \left[\frac{\partial y}{\partial x_1}, \frac{\partial y}{\partial x_2}, \ldots, \frac{\partial y}{\partial x_n}\right]$$

$$y = x_1^2 + 2x_2^2$$

$$\frac{\partial}{\partial \mathbf{x}} x_1^2 + 2x_2^2 = [2x_1, 4x_2]$$

Direction (2, 4), perpendicular to the contour lines



Examples

y	a	au	$\text{sum}(\mathbf{x})$	$\|\mathbf{x}\|^2$
$\frac{\partial y}{\partial \mathbf{x}}$	$\mathbf{0}^T$	$a\frac{\partial u}{\partial \mathbf{x}}$	$\mathbf{1}^T$	$2\mathbf{x}^T$

a is not a function of $\mathbf{x}$

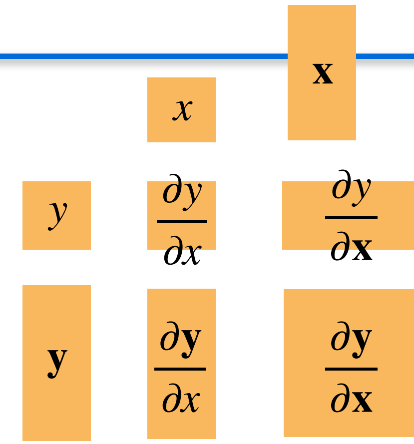
$\mathbf{0}$ and $\mathbf{1}$ are vectors

y	$u+v$	uv	$\langle \mathbf{u}, \mathbf{v} \rangle$
$\frac{\partial y}{\partial \mathbf{x}}$	$\frac{\partial u}{\partial \mathbf{x}} + \frac{\partial v}{\partial \mathbf{x}}$	$\frac{\partial u}{\partial \mathbf{x}}v + \frac{\partial v}{\partial \mathbf{x}}u$	$\mathbf{u}^T\frac{\partial \mathbf{v}}{\partial \mathbf{x}} + \mathbf{v}^T\frac{\partial \mathbf{u}}{\partial \mathbf{x}}$

Gradients of vector functions

$\partial \mathbf{y} / \partial x$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \qquad \frac{\partial \mathbf{y}}{\partial x} = \begin{bmatrix} \frac{\partial y_1}{\partial x} \\ \frac{\partial y_2}{\partial x} \\ \vdots \\ \frac{\partial y_m}{\partial x} \end{bmatrix}$$



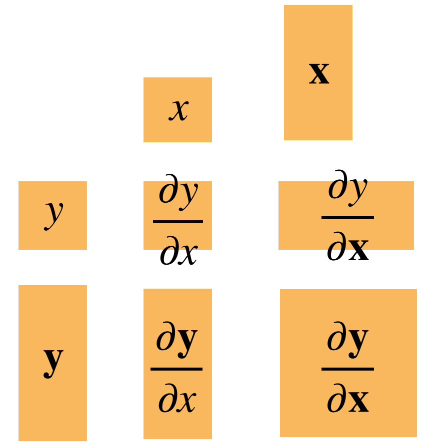
$\partial y / \partial \mathbf{x}$ is a row vector, while $\partial \mathbf{y} / \partial x$ is a column vector

It is called numerator-layout notation. The reversed version is called denominator-layout notation

$$\mathbf{x} \in \mathbb{R}^n, \quad \mathbf{y} \in \mathbb{R}^m, \quad \frac{\partial \mathbf{y}}{\partial \mathbf{x}} \in \mathbb{R}^{m \times n}$$

$\partial \mathbf{y} / \partial \mathbf{x}$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$



$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial \mathbf{x}} \\ \frac{\partial y_2}{\partial \mathbf{x}} \\ \vdots \\ \frac{\partial y_m}{\partial \mathbf{x}} \end{bmatrix} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1}, \frac{\partial y_1}{\partial x_2}, \ldots, \frac{\partial y_1}{\partial x_n} \\ \frac{\partial y_2}{\partial x_1}, \frac{\partial y_2}{\partial x_2}, \ldots, \frac{\partial y_2}{\partial x_n} \\ \vdots \\ \frac{\partial y_m}{\partial x_1}, \frac{\partial y_m}{\partial x_2}, \ldots, \frac{\partial y_m}{\partial x_n} \end{bmatrix}$$

Examples

$\mathbf{y}$	$\mathbf{a}$	$\mathbf{x}$	$\mathbf{A}\mathbf{x}$	$\mathbf{x}^T\mathbf{A}$
$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$	$\mathbf{0}$	$\mathbf{I}$	$\mathbf{A}$	$\mathbf{A}^T$

$\mathbf{y}$	$a\mathbf{u}$	$\mathbf{A}\mathbf{u}$	$\mathbf{u} + \mathbf{v}$
$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$	$a\frac{\partial \mathbf{u}}{\partial \mathbf{x}}$	$\mathbf{A}\frac{\partial \mathbf{u}}{\partial \mathbf{x}}$	$\frac{\partial \mathbf{u}}{\partial \mathbf{x}} + \frac{\partial \mathbf{v}}{\partial \mathbf{x}}$

$$\mathbf{x} \in \mathbb{R}^n, \quad \mathbf{y} \in \mathbb{R}^m, \quad \frac{\partial \mathbf{y}}{\partial \mathbf{x}} \in \mathbb{R}^{m \times n}$$

a, $\mathbf{a}$ and $\mathbf{A}$ are not functions of $\mathbf{x}$

$\mathbf{0}$ and $\mathbf{I}$ are matrices

Generalize to Matrices

		Scalar	Vector	Matrix
		x $(1,)$	$\mathbf{x}$ $(n,1)$	$\mathbf{X}$ (n,k)
Scalar	y $(1,)$	$\frac{\partial y}{\partial x}$ $(1,)$	$\frac{\partial y}{\partial \mathbf{x}}$ $(1,n)$	$\frac{\partial y}{\partial \mathbf{X}}$ (k,n)
Vector	$\mathbf{y}$ $(m,1)$	$\frac{\partial \mathbf{y}}{\partial x}$ $(m,1)$	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$ (m,n)	$\frac{\partial \mathbf{y}}{\partial \mathbf{X}}$ (m,k,n)
Matrix	$\mathbf{Y}$ (m,l)	$\frac{\partial \mathbf{Y}}{\partial x}$ (m,l)	$\frac{\partial \mathbf{Y}}{\partial \mathbf{x}}$ (m,l,n)	$\frac{\partial \mathbf{Y}}{\partial \mathbf{X}}$ (m,l,k,n)

Generalize to Vectors

$$y = f(u),\ u = g(x) \qquad \frac{\partial y}{\partial x} = \frac{\partial y}{\partial u}\frac{\partial u}{\partial x}$$

$$\underset{(1,n)}{\frac{\partial y}{\partial \mathbf{x}}} = \underset{(1,)}{\frac{\partial y}{\partial u}}\underset{(1,n)}{\frac{\partial u}{\partial \mathbf{x}}} \qquad \underset{(1,n)}{\frac{\partial y}{\partial \mathbf{x}}} = \underset{(1,k)}{\frac{\partial y}{\partial \mathbf{u}}}\underset{(k,n)}{\frac{\partial \mathbf{u}}{\partial \mathbf{x}}} \qquad \underset{(m,n)}{\frac{\partial \mathbf{y}}{\partial \mathbf{x}}} = \underset{(m,k)}{\frac{\partial \mathbf{y}}{\partial \mathbf{u}}}\underset{(k,n)}{\frac{\partial \mathbf{u}}{\partial \mathbf{x}}}$$

Example 1

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \frac{\partial \mathbf{y}}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial \mathbf{x}}$$

Assume $\mathbf{x}, \mathbf{w} \in \mathbb{R}^n, \quad y \in \mathbb{R}$

$$z = \left(\langle \mathbf{x}, \mathbf{w} \rangle - y\right)^2$$

Compute $\frac{\partial z}{\partial \mathbf{w}}$

Decompose

$$a = \langle \mathbf{x}, \mathbf{w} \rangle$$
$$b = a - y$$
$$z = b^2$$

$$\begin{aligned}
\frac{\partial z}{\partial \mathbf{w}} &= \frac{\partial z}{\partial b} \frac{\partial b}{\partial a} \frac{\partial a}{\partial \mathbf{w}} \\
&= \frac{\partial b^2}{\partial b} \frac{\partial a - y}{\partial a} \frac{\partial \langle \mathbf{x}, \mathbf{w} \rangle}{\partial \mathbf{w}} \\
&= 2b \cdot 1 \cdot \mathbf{x}^T \\
&= 2\left(\langle \mathbf{x}, \mathbf{w} \rangle - y\right) \mathbf{x}^T
\end{aligned}$$

Solving Linear Model

$$\hat{w} = \arg\min \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2$$

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \frac{\partial \mathbf{y}}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial \mathbf{x}}$$

Assume $\mathbf{X} \in \mathbb{R}^{m \times n}, \quad \mathbf{w} \in \mathbb{R}^{n}, \quad \mathbf{y} \in \mathbb{R}^{m}$

$$z = \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2$$

Compute $\dfrac{\partial z}{\partial \mathbf{w}} = 0$

$$\begin{aligned}\frac{\partial z}{\partial \mathbf{w}} &= \frac{\partial z}{\partial \mathbf{b}} \frac{\partial \mathbf{b}}{\partial \mathbf{a}} \frac{\partial \mathbf{a}}{\partial \mathbf{w}} \\ &= \frac{\partial \|\mathbf{b}\|^2}{\partial \mathbf{b}} \frac{\partial \mathbf{a} - \mathbf{y}}{\partial \mathbf{a}} \frac{\partial \mathbf{X}\mathbf{w}}{\partial \mathbf{w}} \\ &= 2\mathbf{b}^T \times \mathbf{I} \times \mathbf{X} \\ &= 2\left(\mathbf{X}\mathbf{w} - \mathbf{y}\right)^T \mathbf{X}\end{aligned}$$

Decompose

$$\mathbf{a} = \mathbf{X}\mathbf{w}$$

$$\mathbf{b} = \mathbf{a} - \mathbf{y}$$

$$z = \|\mathbf{b}\|^2$$

Let

$$2\left(\mathbf{X}\mathbf{w} - \mathbf{y}\right)^T \mathbf{X} = 0$$

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

More about matrix calculus

- Matrix cookbook
- http://www2.imm.dtu.dk/pubdb/edoc/imm3274.pdf

Linear model in PyTorch

```
import torch
from torch.autograd import Variable

class linearRegression(torch.nn.Module):
    def __init__(self, inputSize, outputSize):
        super(linearRegression, self).__init__()
        self.linear = torch.nn.Linear(inputSize, 
outputSize)

    def forward(self, x):
        out = self.linear(x)
        return out
```

Next Up

- Multilayer Perceptron
- More on neural networks as universal approximators
  - And the issue of depth in networks
  - How to train neural network from data