Neural Networks and Deep Learning

CS5242

Wei WANG

National University of Singapore

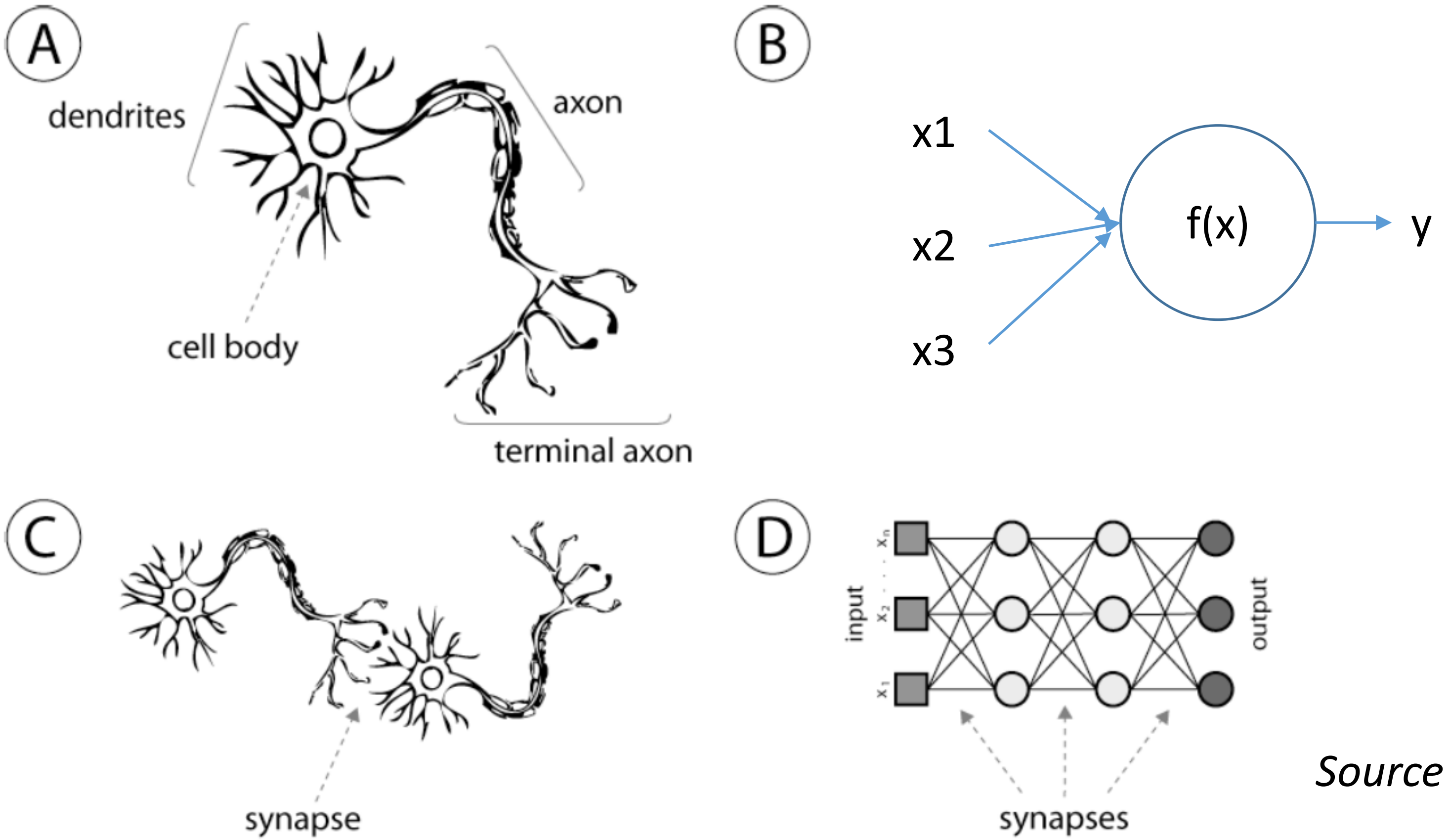
cs5242@comp.nus.edu.sg

Concepts

- **AI**
  - 1950’s
  - “Human intelligence exhibited by machines”
  - Narrow AI: image recognition, machine translation
- **Machine learning**
  - 1980’s
  - “An approach to achieve AI through systems that can learn from experience to find patterns in that data”
- **Neural networks and Deep learning**
  - 2010’s
  - A class of machine learning algorithm that use a cascade of multiple layers of nonlinear processing units for feature extraction and transformation---Wikipedia

Neural networks (NN)

- Artificial neural networks



Source from Hwee Kuan Lee

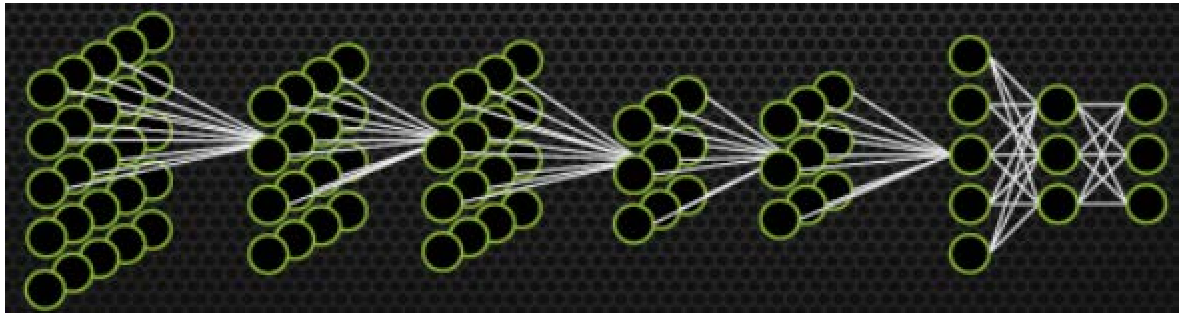
Neural networks (NN)

- Shallow to deep

Source from Hwee Kuan Lee

Neural networks (NN)

- Multi-layer perceptron (MLP)
- Convolutional neural network (CNN)
- Recurrent neural networks (RNN)
- (Restricted) Boltzmann machine
- Deep belief network
- Spike neural network
- Radial basis function neural network
- Hopfield networks



Source from [3]

Deep Learning

Rebrand/resurgence of neural networks

- Big data (new oil/electricity)
- New models and training algorithms
- Powerful machines, e.g. GPU

Feature learning

- Not only neural networks
- Distributed representation of words, e.g. word

History [2]

1958 First perceptron, Frank Rosenblatt

1974 Backpropagation, Paul Werbos

1980 Neocogitron (inspired CNN), Kunihiko Fukushima

1986 Restricted Boltzmann Machine, Paul Smolensky

1986 Recurrent Neural Network (RNN), Michael I. Jordan

1990 LeNet, Yann LeCun

1998 Hochreiter & Schmidhube, LSTM

2006 Deep Belief Networks, Geoffrey Hinton

2012 AlexNet, Alex Krizhevsky & Geoffrey Hinton

2014 GoogleNet & VGG

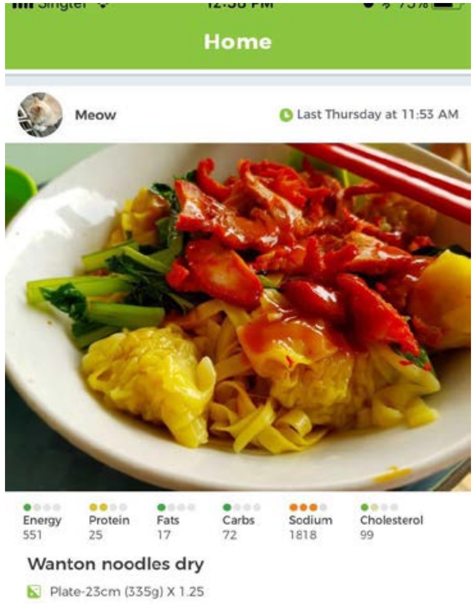
2014 Generative adversarial network, Ian Goodfellow

2015 ResNet, Kaimin He

2015 AlphaGo, DeepMind

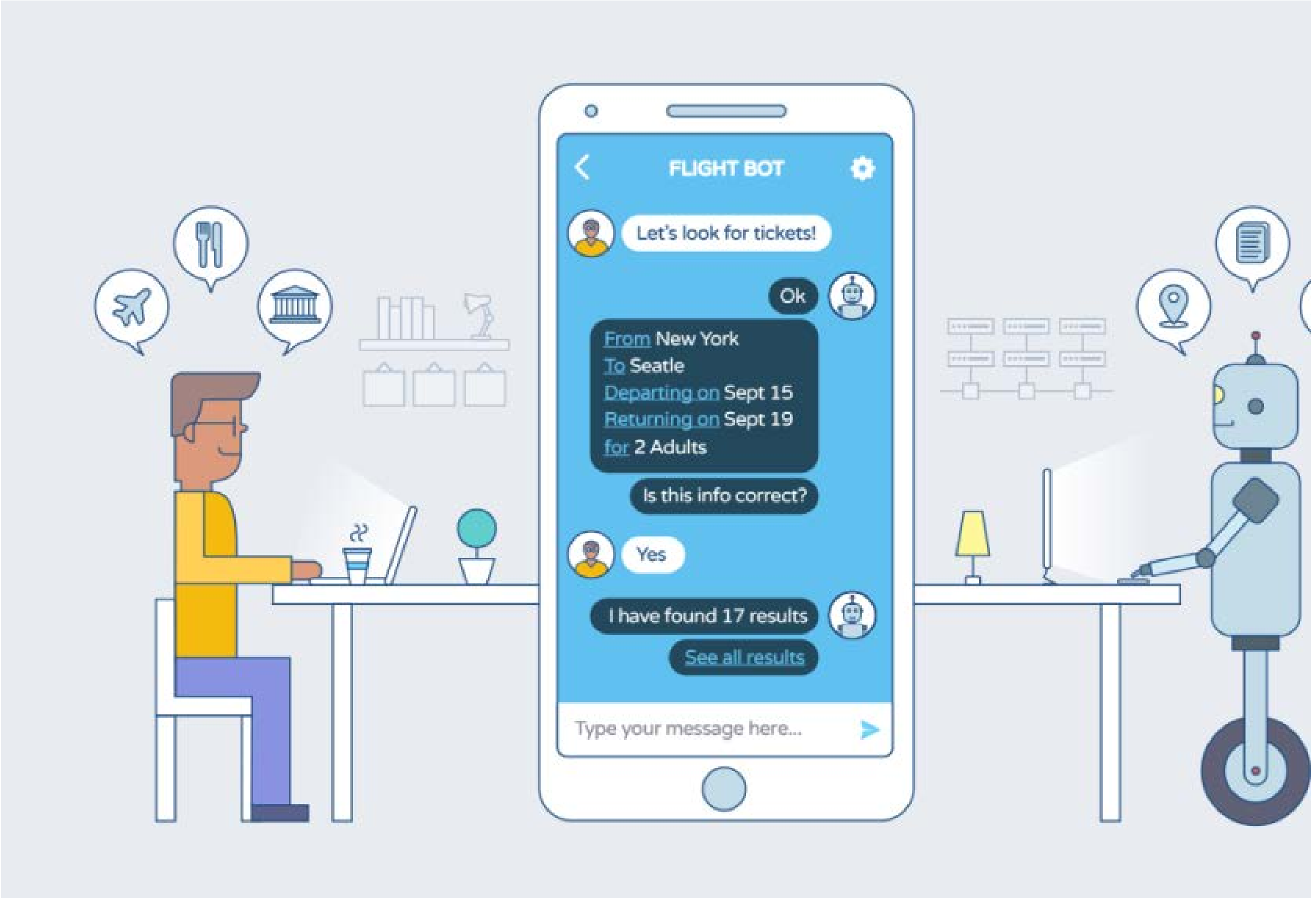
Applications

- [Vision]()
    - Image classification
    - Object detection and tracking
    - Scene text recognition



Applications

- Natural language processing
  - Question answering
  - Machine translation

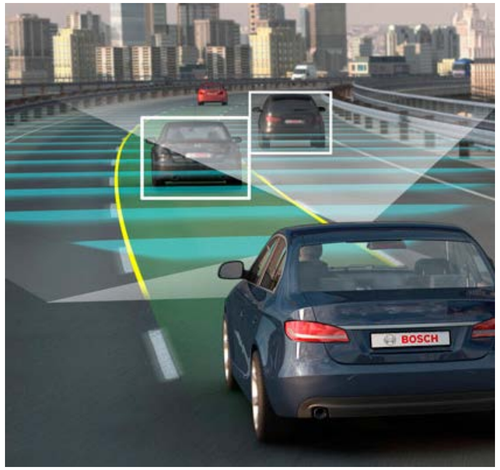


Applications

- Speech
  - Speech recognition, e.g. [Amazon Echo]

Applications

- And more?
  - Health-care
  - Agriculture
  - Environment
  - Fin-tech
  - Autonomous vehicle
  - Manufacture



Similar online courses

Neural Networks and Deep Learning

- Andrew Ng
- https://www.coursera.org/learn/neural-networks-deep-learning

CS231n: Convolutional Neural Networks for Visual Recognition

- Fei-Fei Li, Justin Johnson, Serena Yeung
- http://cs231n.stanford.edu/

CS224d: Deep Learning for Natural Language Processing

- Richard Socher
- http://cs224d.stanford.edu/

CSC 321: Intro to Neural Networks and Machine Learning

- Roger Grosse
- http://www.cs.toronto.edu/~rgrosse/courses/csc321_2017/

Neural Networks for Machine Learning

- Geoffrey Hinton
- https://www.coursera.org/learn/neural-networks

Schedule (Check IVLE for latest plan)

- Part I: Basics of training neural networks
    - Week 1: Introduction
        - History, applications and administrative
        - Linear regression
    - Week 2: Training tricks
        - Data normalization
        - Bias and variance
        - Regularization
        - SGD
    - Week 3: Deep neural networks
        - Loss function, BP
        - Initialization
        - Activation
        - Dropout
        - Batch-Normalization

Schedule (Check IVLE for latest plan)

- Part II: Convolutional neural networks
  - Week 4: Layers
    - Convolution, pooling
  - Week 5: Architecture comparison
    - AlexNet, VGG, ResNet, InceptionNet
  - Week 6: Applications
    - Image classification, object detection, image segmentation
  - Week 7: Practice session
- Part III: Recurrent neural networks
  - Week 8: Layer and architecture
    - Vanilla RNN, LSTM, GRU, Bi-directional RNN
  - Week 9: Training and prediction & **Quiz**
    - Greedy search VS Beam search
  - Week 10: Applications
    - Machine translation, question answering
  - Week 11: Practice session
- Part IV: Advanced topics (TBD)
  - Week 12: Generative adversarial network
  - Week 13: Distributed deep learning

Intended learning outcomes

1

Explain the logics (intuitions) of the operations of different layers and training algorithms

2

Compare different neural network architectures in terms of their characteristics

3

Implement popular neural networks

4

Solve real problems using neural networks and deep learning techniques

Pre-requisite

Course

- Machine Learning (CS3244)
  - Or https://www.coursera.org/learn/machine-learning
- Linear Algebra (MA1101R)
  - Or https://ocw.mit.edu/courses/mathematics/18-06-linear-algebra-spring-2010/
- Calculus (MA1521)
- Probability (ST2334)

Coding

- Python (ONLY, version 3.x is recommended)
  - Numpy
  - **Keras**, TensorFlow, PyTorch, MxNet, Caffe, SINGA
- Jupyter notebook

Grading policy

Weightage:

- Assignment 1 20%
- Assignment 2 20%
- Quiz 30%
  - Closed book with one page cheat sheet
- Project 30%
  - Register the group information on IVLE

Late submission

- 15% off per day late (17:01 is the start of one day)

Collaboration

- Every assignment is an individual task (no discussion)
- The project is a group-based task (should discuss)
- Avoid academic offence (cheating, plagiarism)

Contacts

- IVLE forum
  - For all issues
- Email: cs5242@comp.nus.edu.sg
  - For private/personal issues
- Consultation (make appointments on IVLE)
  - Teacher
    - Wei WANG, wangwei@comp.nus.edu.sg
    - Wed, 11:00-12:00 (Except 24 Jan), COM2-04-09
  - Teaching Assistants
    - Yaqi Xie, e0205023@u.nus.edu
      - Friday 11:00-12:00, Week 1-6 (Except 23 Feb)
    - Xindi SHANG, shangxin@comp.nus.edu.sg
      - Fridy 11:00-12:00, Week 7-13
    - Yao SHU, shuyao95@u.nus.edu

Machine learning basics

Model/task categorization

- Supervised learning
  - Input --- Output
  - Learning the pattern from labeled training data
  - Image classification
- Unsupervised learning
  - Learning to differentiate/cluster unlabeled data
  - Clustering of images
- Reinforcement learning
  - Learning from trail-and-error through rewards
  - CS4246

House price prediction

- Problem definition
  - Given the **information** of a house, predict its **price**.
- Data
  - Input
    - A house: size, number of floors, distance to CBD, etc
  - Output
    - Price

Data preparation

- Collect data
  - From websites, e.g. kaggle, property guru
  - From data sellers
- Clean data
  - Extract useful fields, size, distance to CBD, height, etc.
  - Handle missing values
- Partition data
  - Training
  - Validation
  - Testing

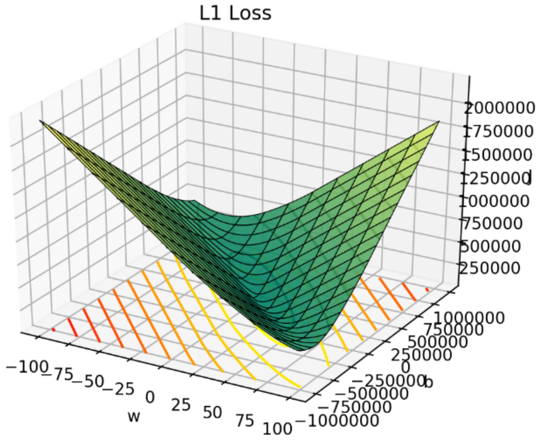
Modeling with a single feature

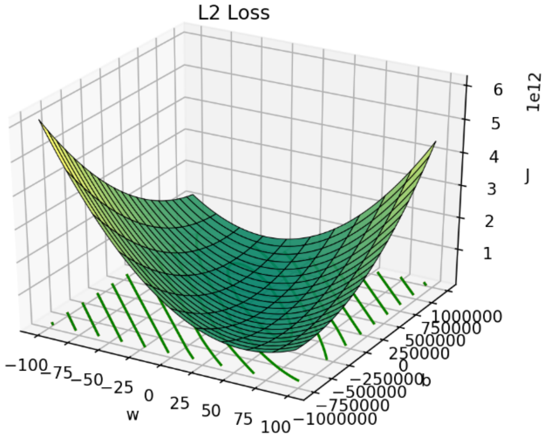
- Model
  - How to represent the input samples, i.e. features/attributes?
  - How to map the input to the output?
- Formalize the input and output
  - Represent the house by its size, denoted as $x \in R$
    - x is called the **feature** of **a data sample** (i.e. one house)
  - Denote the price by $\mathrm{y} \in R^{+}$, called the **ground truth (label)**
- Map from input to output by **linear regression**
  - $\tilde{y} = xw + b, w \in R \quad , b \in R$
  - $\tilde{y}$ is called the **prediction**

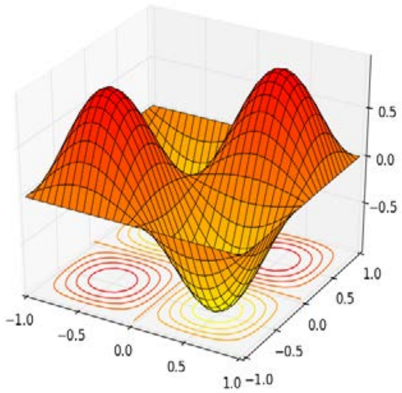
Loss function

- Define the training objective, i.e. loss function
  - $J(w, b) = \sum_{<x,y>\in S_{train}} L(x, y|w, b) /|S_{train}|$
    - $L_1(x, y|w, b) = |\tilde{y} - y|$
      - Not smooth. No gradient at some positions
    - $L_2(x, y|w, b) = |\tilde{y} - y|^2$
      - Smooth. Has gradient at all positions.

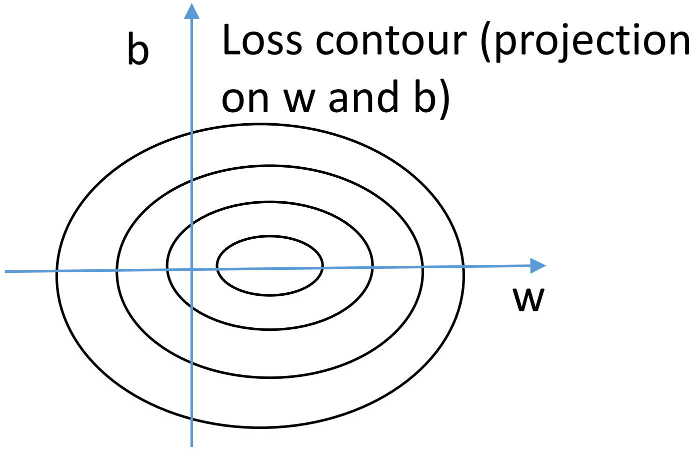
Loss contour

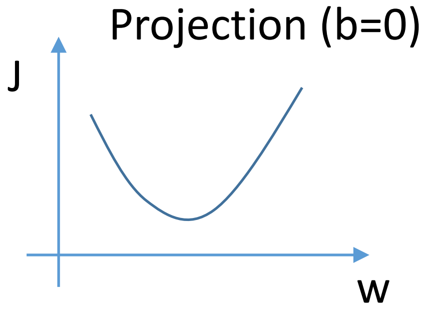






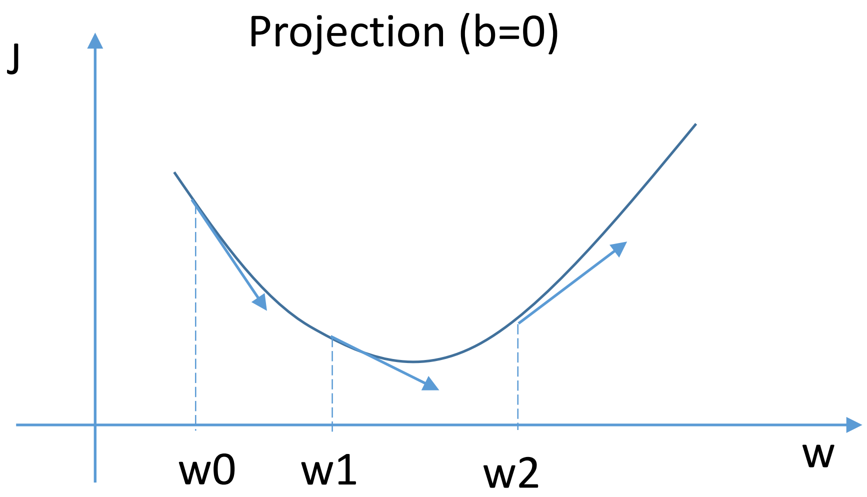
Source from
https://goo.gl/ULkt2Y





Training by gradient descent

- Gradient descent (GD) algorithm for optimization



α is called the learning rate, which controls the moving step length. It important for convergence. If it is large, w would oscillate around the optimal position. If it is small, it would take many iterations to reach the optimal position.

Initialize w as w0

Compute $\frac{\partial J}{\partial w0}$, negative;

Move w from w0 to the right by

$\mathrm{w1} = \mathrm{w0} - \alpha \frac{\partial J}{\partial w0}$

Compute $\frac{\partial J}{\partial w1}$, negative;

Move w from w1 to the right by

$\mathrm{w2} = \mathrm{w1} - \alpha \frac{\partial J}{\partial w1}$

Compute $\frac{\partial J}{\partial w2}$, positive

Move w from w2 to the left by

$\mathrm{w3} = \mathrm{w2} - \alpha \frac{\partial J}{\partial w2}$

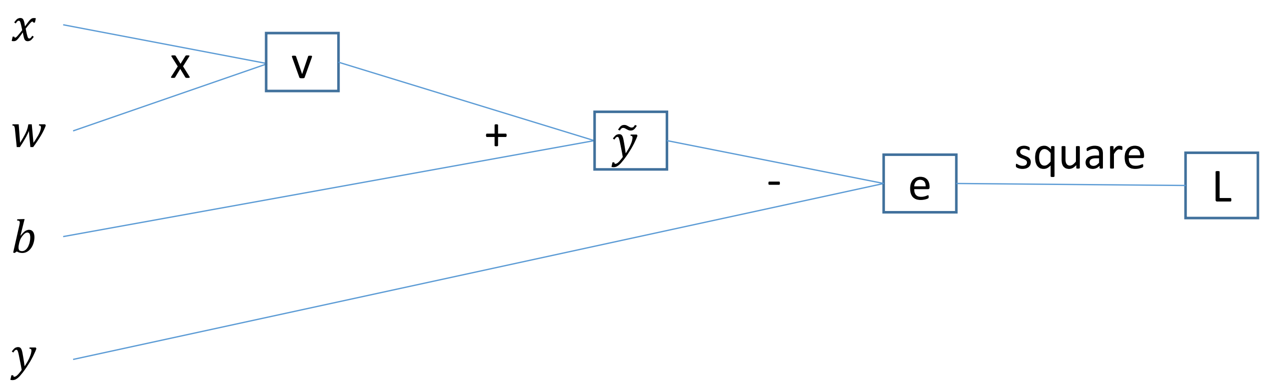
Gradually decrease J and move w to the optimal position

Training by gradient descent

- Gradient descent algorithm for optimization
- Set w = 1, b = 0
- Repeat
  - For each data sample, compute $\tilde{y} = xw + b$
  - Compute the average loss, $\sum_{<x,y>\in S_{train}} L(x, y|w, b) /|S_{train}|$
  - Compute partial derivative $\frac{\partial J}{\partial w}, \frac{\partial J}{\partial b}$
  - Update $\mathrm{w} = \mathrm{w} - \alpha \frac{\partial J}{\partial w}, \mathrm{b} = \mathrm{b} - \alpha \frac{\partial J}{\partial b}$

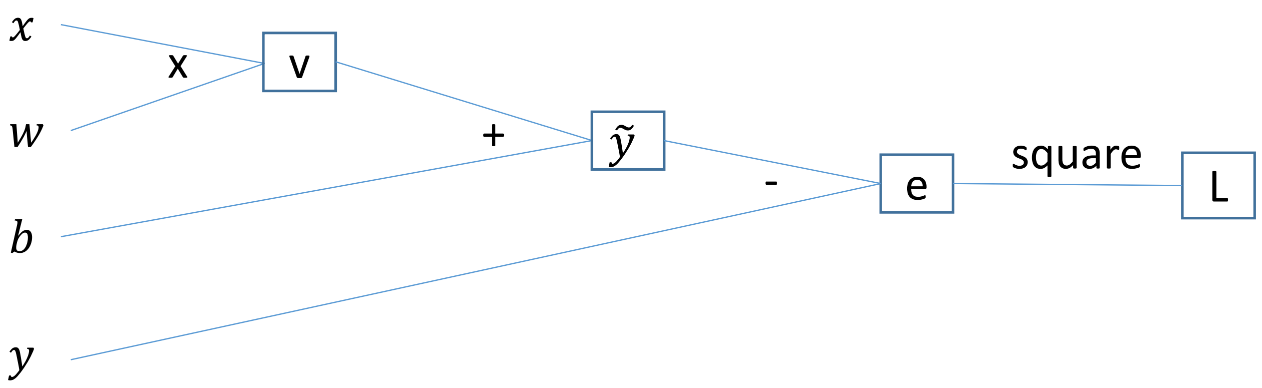
Back-propagation (BP) for computing derivatives

- $\tilde{y} = xw + b$
- $L \quad (x, y|w, b) = |\tilde{y} - y|^2$
- $\frac{\partial J}{\partial w} = \sum_{<x,y>\in S_{train}} \frac{\partial L(x,y|w,b)}{\partial w}$
- Computation graph



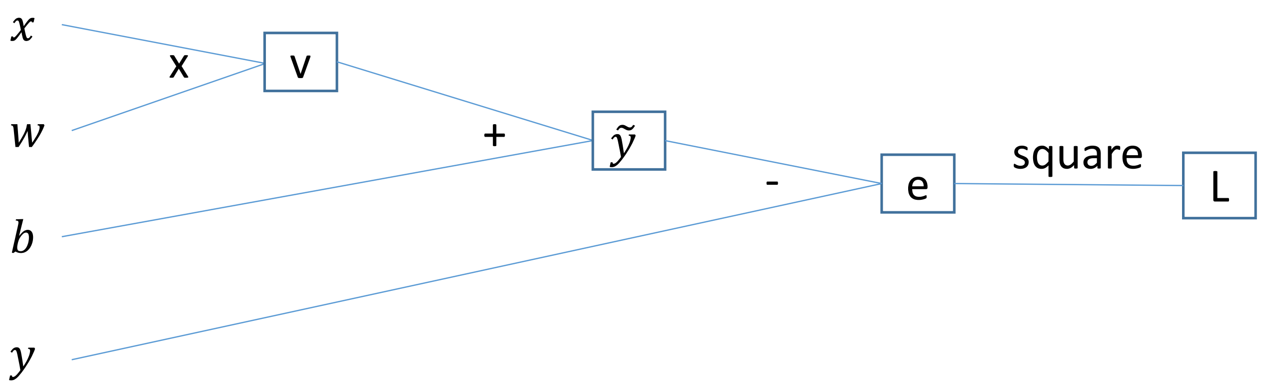
Back-propagation for a single sample

- Forward (w=1, b=0)
  - Given x=1.5, y=2
  - v=x*w=1.5x1=1.5
  - $\tilde{y}$=v + b=1.5+0=1.5
  - e= $\tilde{y}$-y=1.5-2=-0.5
  - L=(-0.5)x(-0.5)=0.25

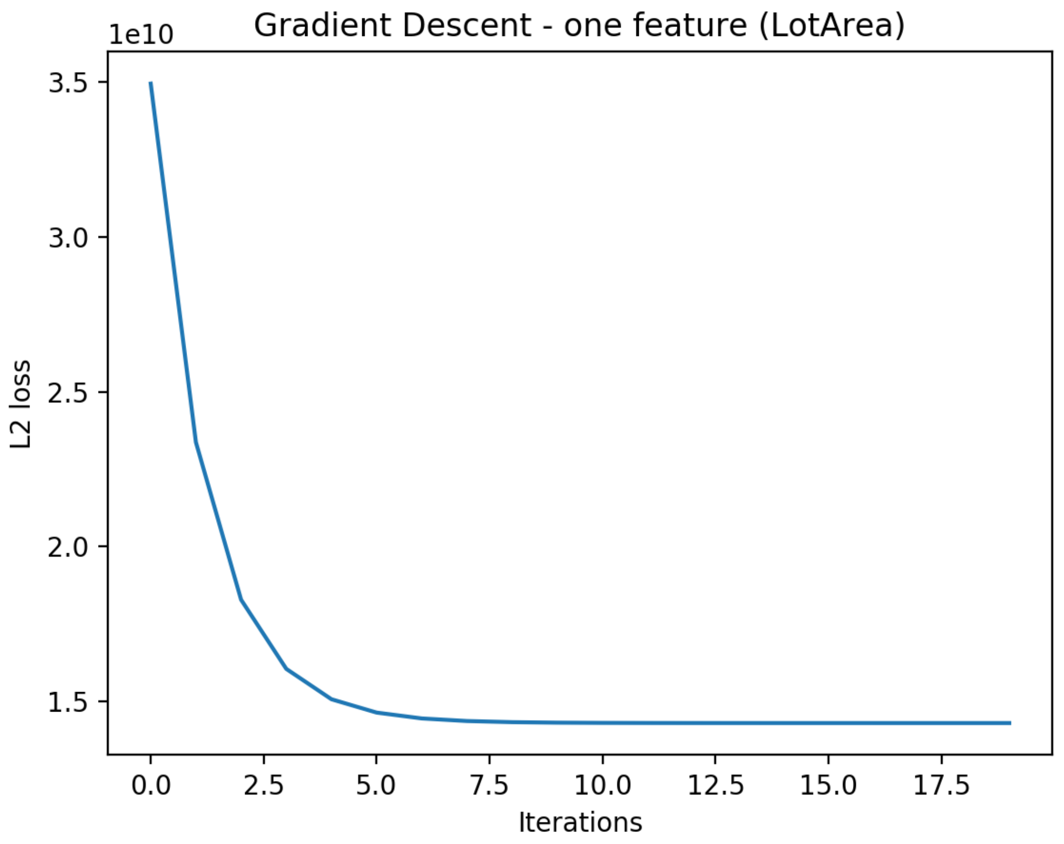


Back-propagation

- Forward (w=1, b=0)
  - x=1.5, y=2
  - v=x*w=1.5x1=1.5
  - $\tilde{y}$=v + b=1.5+0=1.5
  - e= $\tilde{y}$-y=1.5-2=-0.5
  - L=(-0.5)x(-0.5)=0.25
- Backward
  - $\frac{\partial L}{\partial e} = 2e = 2 \times (-0.5) = -1$
  - $\frac{\partial L}{\partial \tilde{y}} = \frac{\partial L}{\partial e} \times \frac{\partial e}{\partial \tilde{y}} = -1 \times 1 = -1$
  - $\frac{\partial L}{\partial b} = \frac{\partial L}{\partial \tilde{y}} \times \frac{\partial \tilde{y}}{\partial b} = -1 \times 1 = -1$
  - $\frac{\partial L}{\partial v} = \frac{\partial L}{\partial \tilde{y}} \times \frac{\partial \tilde{y}}{\partial v} = -1 \times 1 = -1$
  - $\frac{\partial L}{\partial w} = \frac{\partial L}{\partial v} \times \frac{\partial v}{\partial w} = -1 \times x = -1.5$



Training loss

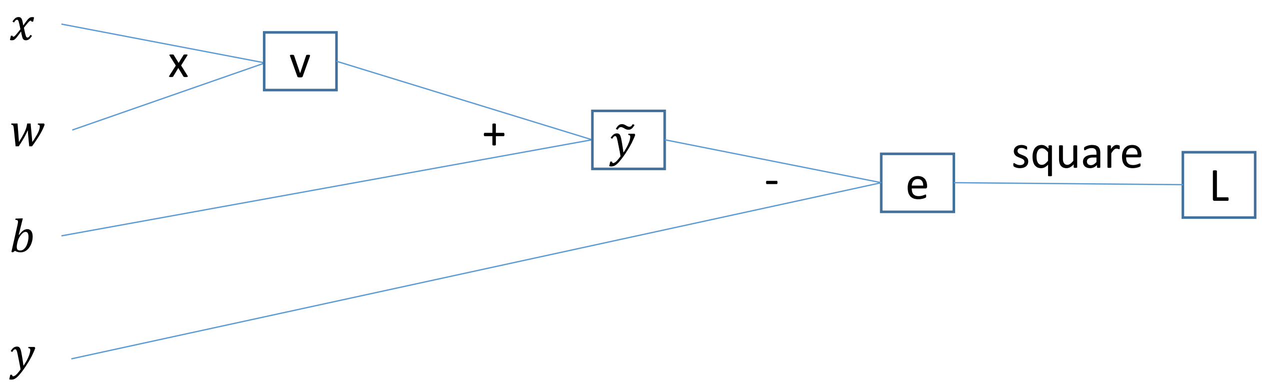


J

Modeling with multiple features

- Consider more features
  - Distance to CBD, number of floors, etc.
  - $x \in R^m, y \in R^+$, x_i for the i-th feature (attribute)
- Map from input to output
  - $\tilde{y} = w^T x + b, w \in R^m, b \in R$, w_i for the i-th co-efficient
    $= \sum_{i=1}^{d} x_i \times w_i + b$
- $L(x, y|w, b) = |\tilde{y} - y|^2$
- $J(w, b) = \frac{\sum_{<x,y> \in S_{train}} L(x, y|w, b)}{|S_{train}|}$

Back-propagation

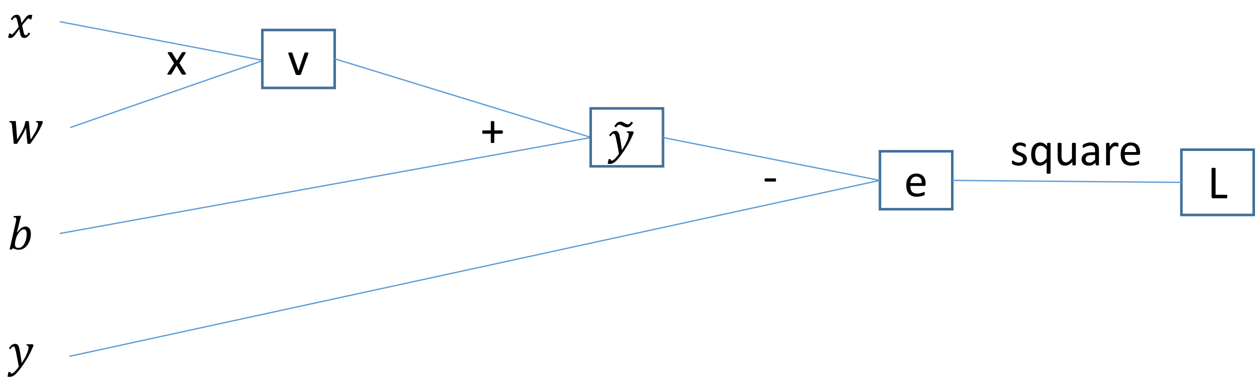


- Forward
  - Given
    - w=$[1,1,1]^T$, b=0
    - x=$[0.5,1,1.5]^T$
    - y=2
  - v= $w^T x$ =1x0.5+1x1+1x1.5 = 3
  - $\tilde{y}$= 3+0 = 3
  - e=3-2 = 1
  - L=1x1 = 1

Back-propagation

- Backward
  - $\frac{\partial L}{\partial e} = 2e = 2 \times 1 = 2$
  - $\frac{\partial L}{\partial \tilde{y}} = \frac{\partial L}{\partial e} \times \frac{\partial e}{\partial \tilde{y}} = 2 \times 1 = 2$
  - $\frac{\partial L}{\partial b} = \frac{\partial L}{\partial \tilde{y}} \times \frac{\partial \tilde{y}}{\partial b} = 2 \times 1 = 2$
  - $\frac{\partial L}{\partial v} = \frac{\partial L}{\partial \tilde{y}} \times \frac{\partial \tilde{y}}{\partial v} = 2 \times 1 = 2$
  - $\frac{\partial L}{\partial w} = \frac{\partial L}{\partial v} \times \frac{\partial v}{\partial w} = 2 \times x^T = 2 \times [0.5,1,1.5]^T = [1,2,3]^T$

  - Shape check: w and $\frac{\partial L}{\partial w}$, b and $\frac{\partial L}{\partial b}$



Vectorization

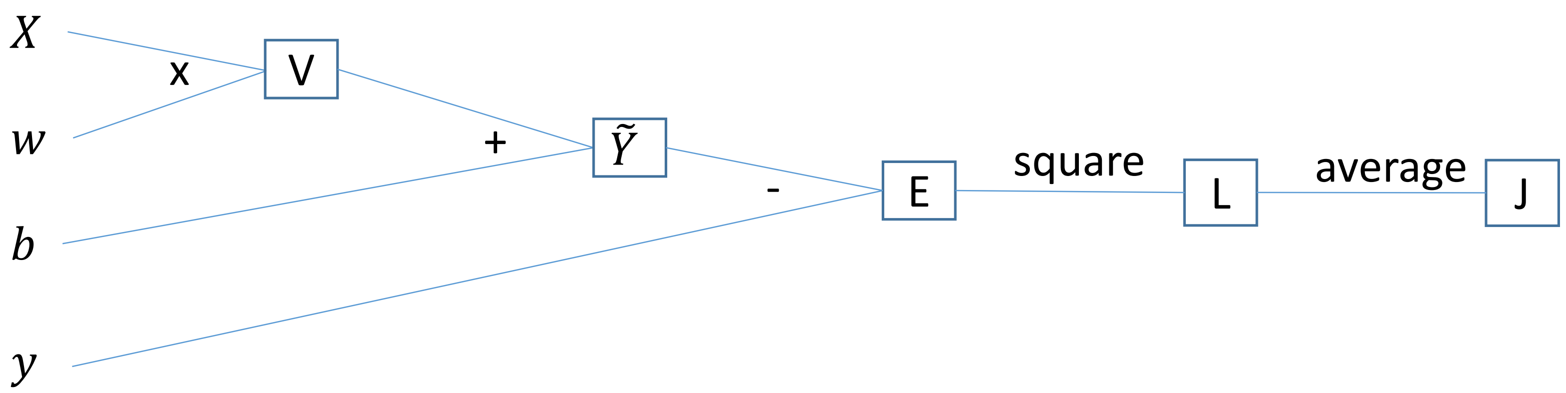
The notation m and n are different to those introduced in the lecture. We will use m for the number of features (attributes) and n for the number of samples for the next lectures.

- Back-propagation over all samples
  - $J(w,b) = \frac{\sum_{<x,y>\in S_{train}} L(x,y|w,b)}{|S_{train}|} = \frac{\sum_{i=1}^{n} L(x^{(i)},y^{(i)}|w,b)}{n}$
  - n=$|\ S_{train}\ |$, $x^{(i)} \in R^m$ and $y^{(i)} \in R^+$ for the i-th sample (i.e. i-th house)
- Put all sample features into a matrix (one column per sample)
  - $X = [x^{(0)}, x^{(1)}, \ldots, x^{(n)}]\ ,\ \in R^{m*n}$
- Put all labels (price) into a matrix
  - $Y = [y^{(0)}, y^{(1)}, \ldots, y^{(n)}]\ ,\ \in R^{1*n}$

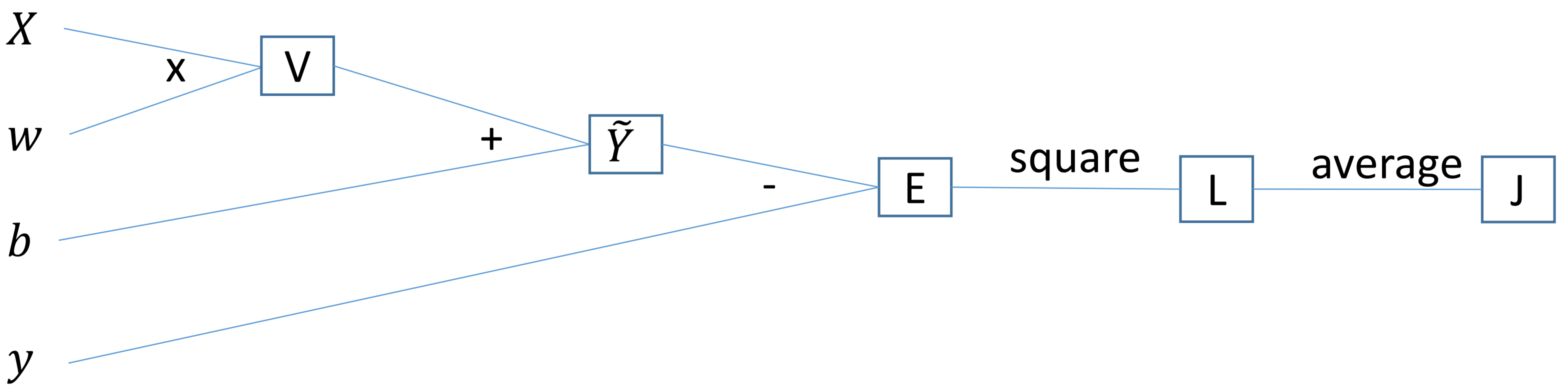
Vectorization

Correction: The shapes of the variables are wrong in the handwriting notes. Please ignore the handwriting notes and use this slide.

- Forward
- $V = w^T X, \in R^{1*n}$
- $\tilde{Y} = \mathrm{V} + \mathrm{b}, \in R^{1*n}$
- $E = \tilde{Y} - Y, \in R^{1*n}$
- $L = E^2, \in R^{1*n}$
- $J = numpy.average(L) \in R^{+}$



Vectorization



- Backward
  - $\frac{\partial J}{\partial L} = \left[\frac{1}{n}, \frac{1}{n}, \frac{1}{n}, \ldots\right] \quad , \in R^{1*n}$
  - $\frac{\partial J}{\partial E} = \frac{\partial J}{\partial L} \times \frac{\partial L}{\partial E} = \frac{\partial J}{\partial L} \times 2E = 2E/n, \in R^{1*n}$
  - $\frac{\partial J}{\partial \tilde{Y}} = \frac{\partial J}{\partial E} \times \frac{\partial E}{\partial \tilde{Y}} = \frac{\partial J}{\partial E} \times [1,1,1,\ldots] = 2E/n\ , \in R^{1*n}$
  - $\frac{\partial J}{\partial b} = \frac{\partial J}{\partial \tilde{Y}} \cdot \frac{\partial \tilde{Y}}{\partial b} = \frac{\partial J}{\partial \tilde{Y}} \cdot [1,1,1,\ldots] \quad , \in R$ (dot product)
  - $\frac{\partial J}{\partial V} = \frac{\partial J}{\partial \tilde{Y}} \times \frac{\partial \tilde{Y}}{\partial V} = \frac{\partial J}{\partial \tilde{Y}} \times [1,1,1,\ldots] = 2E/n, \in R^{1*n}$
  - $\frac{\partial J}{\partial w} = \left(\frac{\partial J}{\partial V}\frac{\partial V}{\partial w}\right)^T = X\left(\frac{\partial J}{\partial V}\right)^T, \in R^m$ (matrix-matrix product)

$\times$: element-wise multiplication

Element-wise multiplication?
dot product?
matrix product?
transpose ?

Shape check: for every node in the graph, its shape should be the same during forward and backward.

Assignment 0 (0 marks) --- GPU machines

- NSCC (National Super-Computing Center)
  - Free for NUS students
  - 128 GPU nodes, each with a NVIDIA Tesla K40 (12 GB)
  - Follow the manual and test program on IVLE Assignments/assignment0 to get yourself familiar with it.
  - **Check the versions of the software**
- SoC GPU Cluster
  - https://docs.comp.nus.edu.sg/node/1814
- Google Cloud Platform
  - 300 USD credit for new register
  - NVIDIA Tesla K80 (2x12GB) is available for Region Taiwan
- Amazon EC2 (g2.8xlarge)
  - 100 USD credit for students
  - NVIDIA Grid GPU, 1536 CUDA cores, 4GB memory
  - Available for Singapore region
- Alibaba cloud
- Tips:
  - STOP/TERMINATE the instance immediately after your program terminates
  - Check the usage status frequently to avoid high outstanding bills
  - Use Amazon or Google cloud platform for debugging and use NSCC for actual training and tuning

References

- [1] Goodfellow Ian, Bengio Yoshua, Courville Aaron. Deep learning. MIT Press. http://www.deeplearningbook.org
- [2] Haohan Wang, Bhiksha Raj. On the Origin of Deep Learning. 2017 https://arxiv.org/abs/1702.07800
- [3] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. “Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations.” In ICML 2009
- http://neuralnetworksanddeeplearning.com/chap1.html
- https://www.analyticsvidhya.com/blog/2017/06/a-comprehensive-guide-for-linear-ridge-and-lasso-regression/