1. Deep Layer Neural Network
   1. Notation
      * Số lượng Layer:
      * Số lượng Units ở layer thứ :
      * Activation của layer thứ :
   2. Chiều của các Ma trận
2. Setting up you Machine Learning Application
   1. Train/Dev/Test sets
      * Chia data thành 2-3 phần:
        + Train/Dev: 70/30
        + Train/Dev/Test: 60/20/10
      * Train dùng để train model
      * Dev dùng để tuning hyperparameter
      * Test dùng để kiểm tra đầu ra
   2. Bias/Variance
      * Mô tả:
        + High Bias:  
          A picture containing text

          Description automatically generated
        + High Variance:  
          A picture containing text

          Description automatically generated
        + “Just Right”:  
          A picture containing text

          Description automatically generated
        + High Bias & High Variance:  
          A picture containing text

          Description automatically generated
      * Trong 1 model classification nếu Human Error Optimal (Bayes) Error thì:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | TH1 | TH2 | TH3 | TH4 |
| Train Set Error | 1% | 15% | 15% | 0.5% |
| Dev Set Error | 11% | 16% | 16% | 1% |
|  | High Variance | High Bias | High Bias High Variance | Low Bias Low Variance |

* + - * Dùng phương pháp này khi mà input có nhiều hơn 2 chiều và không thể vẽ đồ thị.
      * Nếu Optimal Error thì TH2 có thể là Low Bias & Variance
  1. Basic Recipe for Machine Learning (“Công thức cơ bản” để xử lý High Bias với High Variance)
     + A picture containing qr code

       Description automatically generated

1. Regularizing your Neural Network
   1. Regularization
      * Logistic Regression:
        + - regularization
          - : regularization parameter
      * Neural Network
        + - Frobenius Norm (đây vẫn là Regularization với tên gọi và kí hiệu khác)
   2. Cách Regularization giảm Overfitting
      * nằm trong khoảng nhỏ Activation func sẽ trở thành gần với Linear func khiến Network trở nên đơn giản giảm Overfitting
   3. Dropout Regularization
      * Ý tưởng: làm giảm độ phức tạp của Network bằng cách loại bỏ ngẫu nhiên 1 số node ở các layer.
      * Implementing Dropout (“Inverted Dropout”):
        + Với layer :
          - (mỗi node có 20% bị loại bỏ. Và biểu thức trên gán giá trị 0/false cho node đó)
          - (cách để loại bỏ các node được gán 0: nhân với 0)
          - (đảm bảo giá trị đầu ra không thay đổi)
      * Tại sao nó lại hiệu quả:
        + Khiến 1 node không thể phụ thuộc nhiều vào 1 node ở layer trước (khiến node đó có weight lớn) vì node đó có thể bị loại bỏ 1 cách ngẫu nhiên
        + Khiến weight của layer trước phải dàn trải ra 🡪 weight trở nên bé đi 🡪 Network trở nên đơn giản hơn
      * : xác xuất giữ lại 1 node trong layer (HyperParameter)
        + Mỗi layer có thể có giá trị khác nhau
        + Layer nào cảm thấy có thể bị Overfitting thì nên có giá trị nhỏ hơn
        + Layer nào không thể bị Overfitting thì
      * Dropout thường được dùng trong Computer Vision vì hầu như không có đủ Data, dễ bị Overfitting
      * Điểm yếu:
        + Vì bị lược bỏ một số node nên hàm không còn giống như định nghĩa ban đầu   
          🡪 Khiến việc vẽ đồ thị của hàm mỗi iteration trở nên không chính xác
        + Khắc phục: Trước khi áp dụng Dropout, vẽ đồ thị của hàm trước để kiểm tra rồi mới áp dụng Dropout.
   4. Một số cách để giảm Overfitting khác
      * Kiếm nhiều Data hơn với **Data Augmentation**
        + Ý tưởng: lấy 1 example (giả sử là hình ảnh) và lật nó theo chiều ngang để sinh ra 1 example mới
      * **Early Stopping**
        + Ý tưởng: dừng train model khi mà weight còn đang chưa lớn, gần 0
        + Ngữ cảnh: hàm giảm đều khi train, nhưng khi áp dụng với dev set thì sau 1 thời gian giảm đều giống ban đầu thì lại tăng lên chứ không giảm. 🡪 Dừng train ở lúc trước khi tăng lên
2. Setting up your Optimization Problem
   1. Normalizing Input
      * Mục đích: Khiến thuật toán chạy nhanh hơn
      * Ý tưởng: giả sử input nằm trong khoảng trong khi input nằm trong khoảng , thì có sự chênh lệnh lớn giữa 2 input 🡪 khiến GD chạy chậm hơn 🡪 cần phải đưa về cùng khoảng
      * Phép tính:
        + Subtract mean:
        + Normalize variance:
        + Áp dụng cùng trên cả test set
   2. Vanishing / Exploding Gradients
      * Là 1 vấn đề khi train các deep neural network
      * Khi mà các giá trị weight tăng hoặc giảm theo hàm số mũ khiến weight trở nên cực lớn hoặc cực nhỏ (cả 2 đều không tốt) dẫn đến Vanishing (cực nhỏ) hoặc Exploding (cực lớn) khi chạy thuật toán Gradients Descent
   3. Weight Initialization for Deep Networks
      * Những cách sau đây nhằm mục đích giải quyết vấn đề của Vanishing / Exploding Gradients
      * Với input càng nhiều 🡪 càng phải nhỏ  
        + Với hàm ReLU thì biểu thức sau hoạt động tốt hơn
        + Với hàm Tanh thì biểu thức ban đầu là phù hợp
   4. Numerical Approximation of Gradients
   5. Gradient Checking
   6. Gradient Checking Implementation Notes
3. Optimization Algorithms (thuật toán tối ưu)
   1. Mini-batch Gradient Descent
      * Ý tưởng:
        + Thay vì thực hiện Gradient Descent với trong 1 lần, thì ta thực hiện Gradient Descent với nhiều lần hơn, nhưng ít traning examples hơn ở mỗi lần.
        + Tức ta chia nhỏ (tương tự trên )
        + Ví dụ: chia ra thành
        + Notation: mini-batch với
      * Pesudo-Code:
      * Batch vs Mini-batch (đồ thị)  
        Chart, line chart, scatter chart

        Description automatically generated
      * Cách chọn size của mini-batch
        + Batch Gradient Descent, , “Too long per iteration”
        + Stochastic Gradient Descent, , “Khó tìm thấy hội tụ”
        + của mini-batch nên nằm giữa 1 và
        + Hình ảnh minh hoạ của 3 giá trị trên: (tím – Stochastic, xanh – Batch, lá – Mini-batch)  
          A picture containing diagram

          Description automatically generated
        + Nếu : dùng Batch Gradient Descent
        + tiêu biểu của mini-batch:
   2. Exponentially Weighted Averages
   3. Gradient Descent with Momentum
   4. RMSprop
   5. Adem Optimization Algorithm
   6. Learning Rate Decay
   7. The Problem of Local Optima