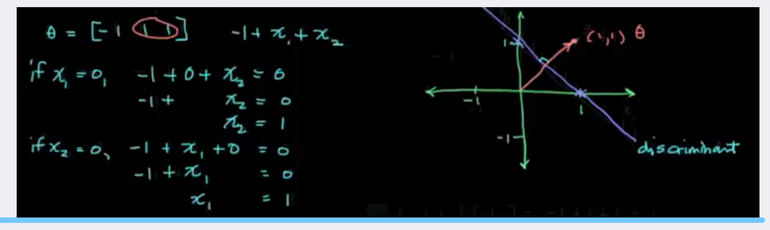
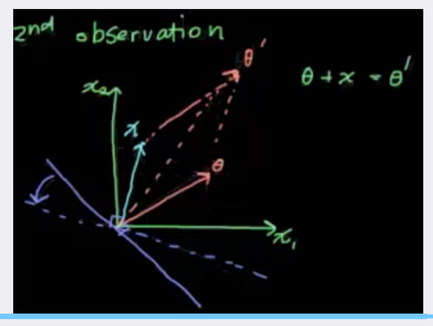
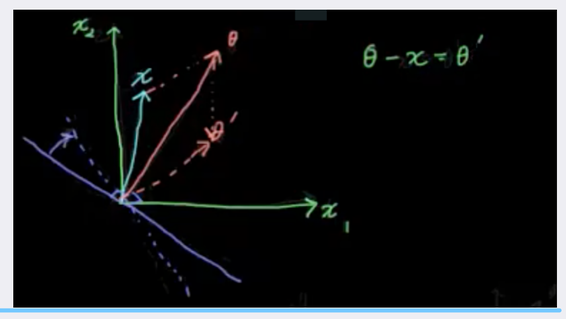
- W13
  - Perceptron (13A)
    - Connectionist: Simulate Brian instead of high level symbols and logic
    - Perceptron
      - Typical use 1 and -1
      - Threshold function
  - Perceptron Learning Rule (13B)
    - The decision boundary is orthogonal to the weights

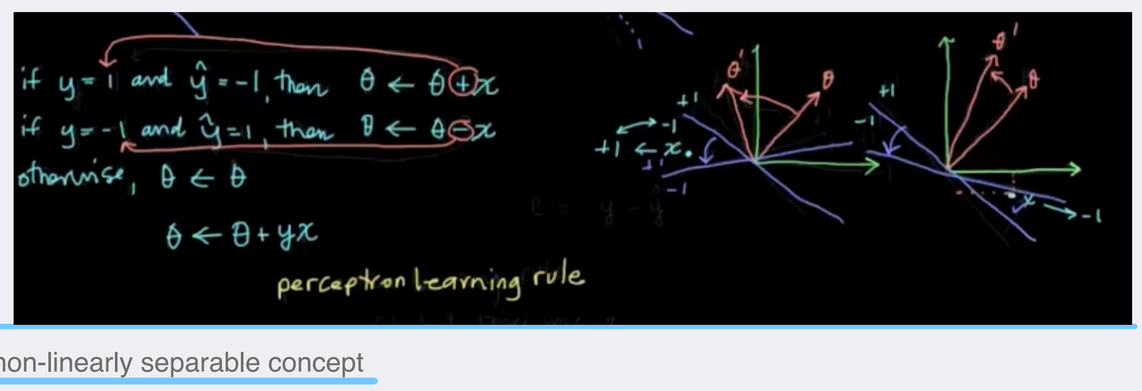


    - Basic vector arithmetic

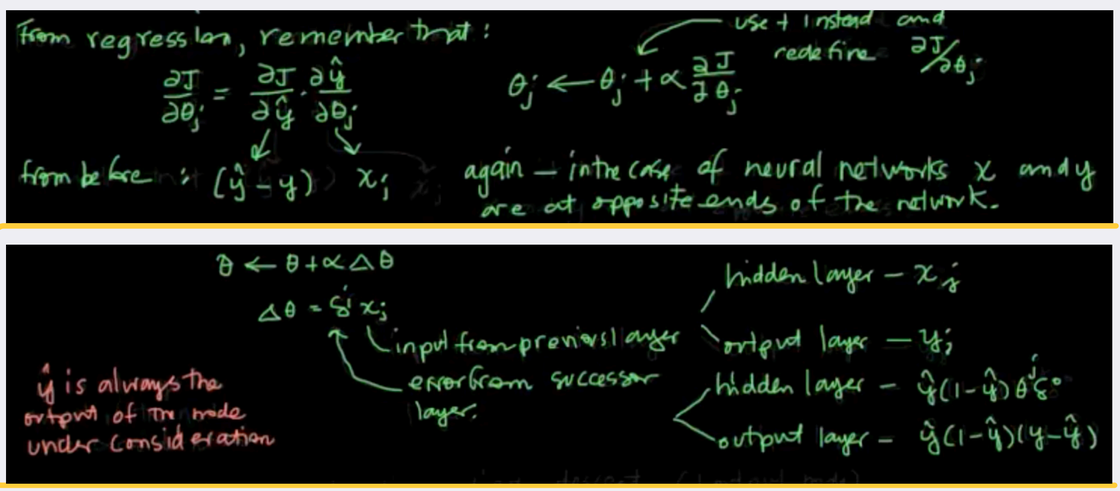




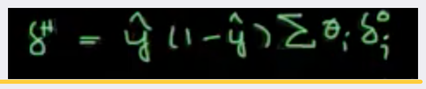
    - Perceptron learning rule



    - A single line can't represent a non-linearly separable concept
  - Multi-Layer Perceptrons (13C)
    - Progress
      - MLP
        - Adaline
          - NN version of regression
          - Stochastic gradient descent
          - Transfer function is linear, rather than threshold
          - Noise cancellation
        - Clamps to 0-1 range (or -1 to 1), for nice derivative
        - Stack them and run in parallel
          - Input, hidden, output
          - 1 hidden layer, can approximate any continuous function
          - 2 hidden layers, can approximate any function
      - Kernel trick — Based on a dot product, voted perceptron
  - Backpropagati on Algorithm (13D)
    - Feed forward NN — Main problem: training data only have input, not hidden
    - Back propagation algorithm



      - Signal go forward, error go backward
      - Multiply output



      - Stop when changing of error is small
  - Tips for Using Multi-Layer Perceptrons (13E)
    - Initialize all weights to small random values, 0-1
    - Scale all input — Mean scaling
    - Stuck in local minima — Random restarts
    - Optimize techniques
      - Genetic algoirthm
      - Conjugate gradient descenet
      - comibnations
    - Pick the number of hidden layers + number of nodes in each layer
      - Start simple
      - Use cross validation
    - Very easy overfitting — Prone, use regulation
    - Multi-class problem — Binary encoding — 00, 01, 10, 11
    - Categorical input