Lecture 4: Neural Networks. This is the final.

- A neural network is a multilayered perceptron.
- It consists of an input layer, one or more hidden layers, and an output layer.
- Each layer is a set of nodes (neurons) that compute a weighted sum of their inputs and pass it through an activation function.

**Shallow vs. Deep Networks**

- Shallow: 1 or 2 hidden layers.
- Deep: 3 or more hidden layers.

**Why do we use deep learning?**

1. **Redundancy**: Avoid overfitting.
2. **Dimensionality reduction**: Lower the number of features.
3. **Non-linearity**: Better approximation capabilities.
4. **Hierarchical representation**: Learn features at different scales.

**Deep Learning Models**

- **Convolutional Neural Networks (CNN)**: Used for image classification.
- **Recurrent Neural Networks (RNN)**: Used for sequence data (text, speech).
- **Generative Adversarial Networks (GAN)**: Used for generating realistic images.

**Key Concepts**

- **Activation Functions**: ReLU, sigmoid, tanh.
- **Loss Functions**: Cross-entropy, mean squared error.
- **Optimization Algorithms**: Gradient descent, stochastic gradient descent (SGD).

**Key Equations**

- Forward propagation:
  
  \[ y = \sigma\left( \mathbf{W}_2 \sigma\left( \mathbf{W}_1 \mathbf{x} + \mathbf{b}_1 \right) + \mathbf{b}_2 \right) \]

- Backward propagation:
  
  \[ \delta = \sigma'(z) \cdot (y - y^*) \]

  \[ \Delta \mathbf{W}_1 = \alpha \cdot \mathbf{W}_1 \mathbf{W}_1 + \lambda \mathbf{W}_1 \]

  \[ \Delta \mathbf{b}_1 = \alpha \cdot \mathbf{W}_1 \delta \hat{\mathbf{W}}_2 \]

  \[ \Delta \mathbf{W}_2 = \alpha \cdot \mathbf{W}_2 \mathbf{W}_2 + \lambda \mathbf{W}_2 \]

  \[ \Delta \mathbf{b}_2 = \alpha \cdot \mathbf{W}_2 \delta \]
Function space view

all functions \( \mathbf{R}^n \rightarrow \mathbf{R} \)

all networks of some fixed architecture, but varying weights.

Networks reach G0 in reasonable time

Some cs 0-optimal cases.

\[ F(\tilde{W}) := \sigma_1(W_c \cdots \sigma_l(W, x) \cdots ) \]

\[ G(V_1 \cdots V_n) := \{ x \mapsto F(x; W_c \cdots W_l) \} \]

\[ \| W_i - V_i \|_p \leq r_i \]

\[ \| W_c - V_c \|_p \leq r_c \]
Error decomposition

Supervised: binary classification

Training set \( \{(x_i, y_i)\}_{i=1}^{n} \) i.i.d.

Loss \( \ell(y, \hat{y}) \) or \( \ell(z) \)

Empirical risk \( \hat{R}(\hat{f}) = \frac{1}{n} \sum_{i=1}^{n} \ell(y, \hat{f}(x_i)) \)

True risk \( R(\hat{f}) = \mathbb{E}_Y \ell(-\hat{f}(x), y) \)

Goal: small \( \hat{R}(\hat{f}) \) small output of \( \phi \)

\( \mathcal{F} \) is good "reference" solution

\( R(\hat{f}) = \left\{ \begin{array}{ll} R(\hat{f}) - \hat{R}(\hat{f}) \\ \hat{R}(\hat{f}) - \hat{R}(\bar{f}) \\ \hat{R}(\bar{f}) - R(\bar{f}) \\ R(\bar{f}) \end{array} \right\} \)

"generalization" "optimization" "generalization" "approximation"

Concern (see notes in notes):

Making some small blows up others.

Solution:

Fix: make \( \frac{1}{n} \) small & adapted to data.

\( P_{x \sim \mathcal{X}, y \sim \mathcal{Y}} [Y = +1 | X = x] \geq 0.13 \)
"Least squares loss"

\[ \frac{1}{l} \sum (y_i - \hat{y}_i)^2 \]

\[ = \frac{1}{l} \sum (y_i^2)(1 - \hat{y}_i^2) \]

\[ = \frac{1}{l} \sum (1 - \hat{y}_i^2) \]

can sometimes go to zero

"Logistic loss"

\[ (y_i, \hat{y}_i) \mapsto \ln(1 + \exp(-y_i \hat{y})) \]

\[ y \in \{-1, 1\} \Rightarrow \hat{y} \rightarrow y \cdot \infty \]

?
\[ R(\hat{F}) - \hat{R}(\hat{F}) \]

\[ R(\hat{F}) - \hat{R}(\hat{F}) \]

\( \hat{F} \) can be determined from distribution on \( x, y \), and does not depend on \( (x, y)_{ii} \).

\[ \mathbb{E} \left[ l(-\hat{F}(x); y) \right] = \frac{1}{n} \sum_{i=1}^{n} l(-\hat{F}(x); y) \]

\( Z_i := l(-\hat{F}(x); y) \)

Then \( Z_1, \ldots, Z_n \) are iid.

\[ \frac{1}{n} \sum_{i=1}^{n} \left( \mathbb{E} Z_i - Z_i \right) \]

\[ \Rightarrow \text{bound via Chernoff's McDiarmid's} \]

Claim: \( R(\hat{F}) - \hat{R}(\hat{F}) \) boils down to a fancy law of large numbers applied to iid \( (Z_1, \ldots, Z_n) \).

\[ R(\hat{F}) - \hat{R}(\hat{F}) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{E} \left[ l(-\hat{F}(x); y) \right] - l(-\hat{F}(x); y) \]

\[ \mathbb{E} Z_i = Z_i \]

Issue: \( Z_1, \ldots, Z_n \) are not guaranteed iid because \( \hat{F} \) trained on \( (x, y)_{ii} \)