CS 66: Machine Learning

Prof. Sara Mathieson

Spring 2019
Outline for April 1

• Handout 12 followup
• Relating Lagrangian and geometry
• Begin: Neural Networks
  – Introduction
  – Notation and diagrams
  – Backpropogation

• Lab 6 due Friday
• Lab 7 released Friday
• Lab 6 check-in Wednesday
• Office hours today 12:30-2pm
Outline for April 1

• Handout 12 followup
• Relating Lagrangian and geometry
• Begin: Neural Networks
  – Introduction
  – Notation and diagrams
  – Backpropagation
Meta-optimization: example

\[ K = 4 \]
Round 1:
* $S = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$
* Support vectors are: $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_4$
* Alpha 0: $\mathbf{x}_3$
* Hyperplane:
Round 1:
* $S = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_4, \mathbf{x}_5\}$
* Support vectors are: $\mathbf{x}_4, \mathbf{x}_5$
* Alpha 0: $\mathbf{x}_1, \mathbf{x}_2$
* Hyperplane:  

\[
\mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \mathbf{x}_4 \\ \mathbf{x}_5
\]
Round 3:
* $S = \{x_4, x_5, x_6, x_7\}$
* Support vectors are: $x_4, x_5, x_7$
* Alpha 0: $x_6$
* Hyperplane:
Round 4:
* $S = \{x_4, x_5, x_7, x_8\}$
* Support vectors are: $x_4, x_5, x_7$
* Alpha 0: $x_8$
* Hyperplane: $\ldots$
Round 5:
* \( S = \{ x_4, x_5, x_7, x_9 \} \)
* Support vectors are: \( x_4, x_7, x_9 \)
* Alpha 0: \( x_5 \)
* Hyperplane: 

\[
\begin{align*}
  x_2 \\
  \downarrow \\
  6 \\
  \downarrow \\
  5 \\
  \downarrow \\
  4 \\
  \downarrow \\
  3 \\
  \downarrow \\
  2 \\
  \downarrow \\
  1 \\
  \downarrow \\
  0 \\
  \downarrow \\
  x_1 
\end{align*}
\]
Handout 12, Final Solution
Outline for April 1

• Handout 12 followup

• Relating Lagrangian and geometry

• Begin: Neural Networks
  – Introduction
  – Notation and diagrams
  – Backpropogation
\[ \vec{x}_1 \cdot \vec{x}_1 = \begin{bmatrix} -2 \end{bmatrix} \begin{bmatrix} -2 \end{bmatrix} = 5 \quad y_1 y_1 = 1 \]
\[ \vec{x}_2 \cdot \vec{x}_2 = \begin{bmatrix} 2 \end{bmatrix} \begin{bmatrix} 2 \end{bmatrix} = 5 \quad y_2 y_2 = 1 \]
\[ \vec{x}_3 \cdot \vec{x}_3 = \begin{bmatrix} 1 \end{bmatrix} \begin{bmatrix} 1 \end{bmatrix} = 5 \quad y_3 y_3 = 1 \]
\[ \vec{x}_1 \cdot \vec{x}_2 = \begin{bmatrix} -1 \end{bmatrix} \begin{bmatrix} -2 \end{bmatrix} = 4 \quad y_1 y_2 = -1 \]
\[ \vec{x}_1 \cdot \vec{x}_3 = \begin{bmatrix} 1 \end{bmatrix} \begin{bmatrix} -2 \end{bmatrix} = -4 \quad y_1 y_3 = -1 \]
\[ \vec{x}_2 \cdot \vec{x}_3 = \begin{bmatrix} 1 \end{bmatrix} \begin{bmatrix} 1 \end{bmatrix} = -5 \quad y_2 y_3 = 1 \]
\[ W(\alpha) = \frac{n}{2} \alpha_i^2 - \frac{1}{2} \sum_{i=1}^{n} \alpha_i y_i x_i \sum_{i=1}^{n} x_i^2 \]

\[ \sum_{i=1}^{n} \alpha_i y_i = 0 \]

Exercise: \( y_i = 1 \), solve for \( \alpha_2 \) and \( \alpha_3 \)
\[ \sum \alpha_i y_i = 0 \]

\[ \Rightarrow \alpha_1 = \alpha_2 = \alpha \]

Derivative!

\[ W'(\alpha) = 2 - 2 \alpha \]

\[ \alpha^* = 1 \]

\[ \mathbf{b} = \sum \alpha_i y_i \bar{x}_i \]

\[ \mathbf{b} = \begin{bmatrix} 2 \\ 1 \\ -1 \end{bmatrix} + \begin{bmatrix} -2 \\ 1 \\ 2 \end{bmatrix} \]

\[ \mathbf{b} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} \]
Outline for April 1

- Handout 12 followup
- Relating Lagrangian and geometry

- Begin: Neural Networks
  - Introduction
  - Notation and diagrams
  - Backpropogation
Disadvantages of SVMs

- Difficult to choose a kernel function
- Does not naturally take into account the correlations between features
- Hard to understand and interpret what the model has learned
Biological Inspiration

Figure: Stanford CS231n http://cs231n.github.io/neural-networks-1/
Goal: learn from complicated inputs

\[ X_1, X_2, X_3, X_4, X_5, X_6 \]

\[ Y_1, Y_2, Y_3 \]

parameters

(glasses)

(smiling)

(eye size)
Idea: transform data into lower dimension

input data

hidden layer

parameters

$X_1$, $X_2$, $X_3$, $X_4$, $X_5$, $X_6$

$Y_1$ (glasses), $Y_2$ (smiling), $Y_3$ (eye size)
Multi-layer networks = “deep learning”

input data

hidden layer 1

hidden layer 2

Y₁ (glasses)
Y₂ (smiling)
Y₃ (eye size)

parameters
History of Neural Networks

• Perceptron can be interpreted as a simple neural network
• Misconceptions about the weaknesses of perceptrons contributed to declining funding for NN research
• Difficulty of training multi-layer NNs contributed to second setback
• Mid 2000’s: breakthroughs in NN training contribute to rise of “deep learning”
Number of papers that mention “deep learning” over time

Backpropagation

- **High-level goal:** we want to know how the output depends on the input
Backpropagation

• *High-level goal:* we want to know how the output depends on the input

• *Issue:* network is very complicated and overall gradient may be difficult to compute
Backpropagation

• **High-level goal:** we want to know how the output depends on the input

• **Issue:** network is very complicated and overall gradient may be difficult to compute

• **Idea:** use the chain rule to compute local gradients throughout the network
Backpropagation

- **High-level goal:** we want to know how the output depends on the input
- **Issue:** network is very complicated and overall gradient may be difficult to compute
- **Idea:** use the chain rule to compute local gradients throughout the network
- **Takeaway:** nodes can know about their value and local gradient without knowing about the network they are imbedded in
\[ f(x, y, z) = (x + y)^2 \]

Want to maximize \( f \)

\[ \frac{d}{dx} f = (1)(x + y)^1 = 1 \]
\[ \frac{d}{dy} f = (1)(x + y)^1 = 1 \]

\[ x = -2 \]
\[ y = 5 \]
\[ z = -4 \]

\[ -4 \cdot -4 = 16 \]
\[ 1 \cdot -4 = -4 \]
\[ 3 \cdot -4 = -12 \]

\[ f = -12 \]
\[ f(x, y, z) = (x + y)^2 \]

want to maximize \( f \)

\[ f(x, y, z) = (x + y)^2 \]

\[ x = -2 \]
\[ 1 \cdot -4 = -4 \]
\[ y = 5 \]
\[ z = -4 \]
\[ 3 \]

\[ \frac{df}{dx} = 2(x + y) \]
\[ \frac{df}{dy} = 2(x + y) \]

\[ \frac{df}{dz} = 0 \]

\[ g = x + y \]
\[ f = g^2 \]

Chain rule:
\[ \frac{df}{dx} = \frac{df}{dg} \cdot \frac{dg}{dx} \]

\[ \frac{df}{dg} = 2g \]
\[ \frac{dg}{dx} = 1 \]

\[ \frac{df}{dx} = 2g \cdot 1 = 2(x + y) \]

\[ \frac{df}{dz} = 0 \]