Week 5: Extensions and Variations of Perceptron, and Practical Issues

Professor Liang Huang

some slides from A. Zisserman (Oxford)
Trivia: Grace Hopper and the first bug

- Edison coined the term “bug” around 1878 and since then it had been widely used in engineering
- Hopper was associated with the discovery of the first computer bug in 1947 which was a moth stuck in a relay

Smithsonian National Museum of American History
Week 5: Perceptron in Practice

• Problems with Perceptron
  • doesn’t converge with inseparable data
  • update might often be too “bold”
  • doesn’t optimize margin
  • result is sensitive to the order of examples

• Ways to alleviate these problems (without SVM/kernels)
  • Part II: voted perceptron and average perceptron
  • Part III: MIRA (margin-infused relaxation algorithm)
  • Part IV: Practical Issues and HW I
  • Part V: “Soft” Perceptron: Logistic Regression

“A ship in port is safe, but that is not what ships are for.”

– Grace Hopper (1906-1992)
Recap of Week 4

**input:** training data $D$

**output:** weights $w$

initialize $w \leftarrow 0$

while not converged

for $(x, y) \in D$

if $y(w \cdot x) \leq 0$

$w \leftarrow w + yx$

“idealized” ML

```
Input x → Training → Model w
Output y →
```

“actual” ML

```
Input x → Training → Model w
Output y →
```

feature map $\phi$

deep learning $\approx$ representation learning

```
Input x → Training → feature map $\phi$
Output y → Training → Model w
```
$ python perc_demo.py  (requires numpy and matplotlib)
Part II: Voted and Averaged Perceptron
**Brief History of Perceptron**

- **1959**: Rosenblatt invention
- **1962**: Novikoff proof
- **1969**: Minsky/Papert book killed it
- **1997**: Cortes/Vapnik SVM
- **1999**: Freund/Schapire voted/avg: revived
- **2002**: Collins structured
- **2003**: Crammer/Singer MIRA
- **2005**: McDonald/Crammer/Pereira structured MIRA
- **2006**: Singer group aggressive
- **2007--2010**: Singer group Pegasos
- **2007--2010**: Pegasos subgradient descent

- **DEAD**: conservative updates, online approx., max margin, +kernels, +max margin, +soft-margin

---

*mentioned in lectures but optional (others papers all covered in detail)

AT&T Research

- **1997**: Cortes/Vapnik SVM
- **1999**: Freund/Schapire voted/avg: revived
- **2002**: Collins structured
- **2005**: McDonald/Crammer/Pereira structured MIRA
- **2006**: Singer group aggressive
- **2007--2010**: Singer group Pegasos

ex-AT&T and students

---

**DEAD**: conservative updates, online approx., max margin, +kernels, +max margin, +soft-margin

---

AT&T Research

- **1997**: Cortes/Vapnik SVM
- **1999**: Freund/Schapire voted/avg: revived
- **2002**: Collins structured
- **2005**: McDonald/Crammer/Pereira structured MIRA
- **2006**: Singer group aggressive
- **2007--2010**: Singer group Pegasos

ex-AT&T and students
Voted/Avged Perceptron

• problem: later examples dominate earlier examples
• solution: voted perceptron (Freund and Schapire, 1999)
  • record the weight vector after each example in $D$
    • not just after each update!
  • and vote on a new example using $|D|$ models
• shown to have better generalization power
• averaged perceptron (from the same paper)
  • an approximation of voted perceptron
  • just use the average of all weight vectors
• can be implemented efficiently
Voted Perceptron

Input: a labeled training set \(\{(x_1, y_1), \ldots, (x_m, y_m)\}\)  
number of epochs \(T\)

Output: a list of weighted perceptrons \(\{(v_1, c_1), \ldots, (v_k, c_k)\}\)

\begin{itemize}
  \item Initialize: \(k := 0, v_1 := 0, c_1 := 0\).
  \item Repeat \(T\) times:
    \begin{itemize}
      \item For \(i = 1, \ldots, m\):
        \begin{itemize}
          \item Compute prediction: \(\hat{y} := \text{sign}(v_k \cdot x_i)\)
          \item If \(\hat{y} = y\) then \(c_k := c_k + 1\).
            \else \(v_{k+1} := v_k + y_i x_i;\)
            \??? \(c_{k+1} := 1;\)
            \??? \(k := k + 1.\)
        \end{itemize}
    \end{itemize}
\end{itemize}

Large Margin Classification Using the Perceptron Algorithm

YOAV FREUND  
AT&T Labs, Shannon Laboratory, 180 Park Avenue, Room A205, Florham Park, NJ 07932-0971

ROBERT E. SCHAPIRE  
AT&T Labs, Shannon Laboratory, 180 Park Avenue, Room A279, Florham Park, NJ 07932-0971

Prediction

Given: the list of weighted perceptrons: \(\{(v_1, c_1), \ldots, (v_k, c_k)\}\)  
an unlabeled instance: \(x\)
compute a predicted label \(\hat{y}\) as follows:

\[
  s = \sum_{i=1}^{k} c_i \text{sign}(v_i \cdot x); \quad \hat{y} = \text{sign}(s).
\]

our notation: \((x^{(1)}, y^{(1)})\)
\(v\) is weight,  
\(c\) is its \# of votes

if correct, increase the current model’s \# of votes;  
otherwise create a new model with 1 vote
Experiments

![Graph showing error rate over epochs for different perceptron methods]
Averaged Perceptron

- voted perceptron is not scalable
- and does not output a single model
- avg perceptron is an approximation of voted perceptron
- actually, summing all weight vectors is enough; no need to divide

\[
\begin{align*}
\text{initialize } \mathbf{w} &\leftarrow 0; \quad \mathbf{w}_s \leftarrow 0 \\
\text{while } \text{not converged} &
\quad \text{for } (\mathbf{x}, y) \in D \\
&\quad \text{if } y(\mathbf{w} \cdot \mathbf{x}) \leq 0 \\
&\quad \quad \mathbf{w} \leftarrow \mathbf{w} + y\mathbf{x} \\
&\quad \mathbf{w}_s \leftarrow \mathbf{w}_s + \mathbf{w} \\
\text{output: summed weights } \mathbf{w}_s
\end{align*}
\]
Efficient Implementation of Averaging

- naive implementation (running sum $w_s$) doesn’t scale either
- OK for low dim. (HW1); too slow for high-dim. (HW3)
- very clever trick from Hal Daumé (2006, PhD thesis)

initialize $w \leftarrow 0$; $w_a \leftarrow 0$; $c \leftarrow 0$
while not converged
for $(x, y) \in D$
if $y(w \cdot x) \leq 0$
    $w \leftarrow w + yx$
    $w_a \leftarrow w_a + cyx$
$c \leftarrow c + 1$
output: $c w - w_a$

after each update, not after each example!

$c \cdot w - w_a = 4 \cdot \Delta w^{(1)} + 3 \cdot \Delta w^{(2)} + 2 \cdot \Delta w^{(3)} + 1 \cdot \Delta w^{(4)} = w^{(1)} + w^{(2)} + w^{(3)} + w^{(4)}$
Part III: MIRA

- perceptron often makes bold updates (over-correction)
- and sometimes too small updates (under-correction)
- but hard to tune learning rate
- “just enough” update to correct the mistake?

\[ w' \leftarrow w + \frac{y - w \cdot x}{\|x\|^2} x \]

easy to show:

\[ w' \cdot x = (w + \frac{y - w \cdot x}{\|x\|^2} x) \cdot x = y \]

margin-infused relaxation algorithm (MIRA)
Example: Perceptron under-correction
MIRA: just enough

\[
\begin{align*}
\min_{w'} & \|w' - w\|^2 \\
\text{s.t.} & \quad w' \cdot x \geq 1
\end{align*}
\]

minimal change to ensure functional margin of 1 (dot-product \(w' \cdot x = 1\))

MIRA \(\approx\) 1-step SVM

functional margin: \(y(w \cdot x)\)

geometric margin: \(\frac{y(w \cdot x)}{\|w\|}\)
MIRA: functional vs geom. margin

\[
\min_{\mathbf{w}'} \| \mathbf{w}' - \mathbf{w} \|^2 \\
\text{s.t. } \mathbf{w}' \cdot \mathbf{x} \geq 1
\]

minimal change to ensure functional margin of 1
(dot-product $\mathbf{w}' \cdot \mathbf{x} = 1$)

\[
\text{MIRA } \approx \text{ 1-step SVM}
\]

functional margin: $y(\mathbf{w} \cdot \mathbf{x})$

geometric margin: $\frac{y(\mathbf{w} \cdot \mathbf{x})}{\|\mathbf{w}\|}$
Optional: Aggressive MIRA

- aggressive version of MIRA
- also update if correct but not confident enough
  - i.e., functional margin \((y \mathbf{w} \cdot \mathbf{x})\) not big enough
- \(p\)-aggressive MIRA: update if \(y (\mathbf{w} \cdot \mathbf{x}) < p\) \((0 \leq p < 1)\)
- MIRA is a special case with \(p=0\): only update if misclassified!
- update equation is same as MIRA
  - i.e., after update, functional margin becomes 1
- larger \(p\) leads to a larger geometric margin but slower convergence
Demo
Part IV: Practical Issues

“\textit{A ship in port is safe, but that is not what ships are for.}”

– Grace Hopper (1906-1992)

- you will build your own linear classifiers for HW2 (same data as HW1)
- slightly different binarizations
  - for k-NN, we binarize all categorical fields but keep the two numerical ones
  - for perceptron (and most other classifiers), we binarize numerical fields as well
- why? hint: larger “age” always better? more “hours” always better?
Useful Engineering Tips:
averaging, shuffling, variable learning rate, fixing feature scale

- averaging helps significantly; MIRA helps a tiny little bit
  - perceptron < MIRA < avg. perceptron ≈ avg. MIRA ≈ SVM
- shuffling the data helps hugely if classes were ordered (HW1)
  - shuffling before each epoch helps a little bit
- variable (decaying) learning rate often helps a little
  - 1/(total#updates) or 1/(total#examples) helps
  - any requirement in order to converge?
    - how to prove convergence now?
- centering of each dimension helps (Ex1/HW1)
  - why? => smaller radius, bigger margin!
- unit variance also helps (why?) (Ex1/HW1)
  - 0-mean, 1-var => each feature ≈ a unit Gaussian
Feature Maps in Other Domains

- how to convert an image or text to a vector?

- image

28x28 grayscale image

\[ \mathbf{x} \in \mathbb{R}^{784} \]

- text

“one-hot” representation of words (all binary features)

```
<table>
<thead>
<tr>
<th>&quot;a&quot;</th>
<th>&quot;abbreviations&quot;</th>
<th>&quot;zoology&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```

in deep learning there are other feature maps
Part V: Perceptron vs. Logistic Regression

• logistic regression is another popular linear classifier

• can be viewed as “soft” or “probabilistic” perceptron

• same decision rule (sign of dot-product), but prob. output

\[ f(x) = \text{sign}(w \cdot x) \]

perceptron

\[ f(x) = \sigma(w \cdot x) = \frac{1}{1 + e^{-w \cdot x}} \]

logistic regression
Logistic vs. Linear Regression

- Linear regression is regression applied to real-valued output using linear function.
- Logistic regression is regression applied to 0-1 output using the sigmoid function.

https://florianhartl.com/logistic-regression-geometric-intuition.html
Why Logistic instead of Linear

• linear regression easily dominated by distant points

• causing misclassification

\[ \sigma(wx + b) \] fit to \( y \)

\[ wx + b \] fit to \( y \)

• fit of \( wx + b \) dominated by more distant points

• causes misclassification

• instead LR regresses the sigmoid to the class data

Why Logistic instead of Linear

- linear regression easily dominated by distant points
- causing misclassification

\[ \sigma(w_1 x_1 + w_2 x_2 + b) \text{ fit, vs } w_1 x_1 + w_2 x_2 + b \]
Why 0/1 instead of +/-1

- perc: y=+1 or -1; logistic regression: y=1 or 0
- reason: want the output to be a probability
- decision boundary is still linear: \( p(y=1 \mid x) = 0.5 \)
Logistic Regression: Large Margin

- perceptron can be viewed roughly as “step” regression
- logistic regression favors large margin; SVM: max margin
- in practice: perc. << avg. perc. ≈ logistic regression ≈ SVM