Lecture 11: Linear Models for Classification

Hsuan-Tien Lin (林軒田)
htlin@csie.ntu.edu.tw

Department of Computer Science & Information Engineering
National Taiwan University
(國立台灣大學資訊工程系)
Roadmap

1. When Can Machines Learn?
2. Why Can Machines Learn?
3. How Can Machines Learn?
4. How Can Machines Learn Better?

Lecture 10: Logistic Regression
- **gradient descent** on **cross-entropy error**
  to get good **logistic hypothesis**

Lecture 11: Linear Models for Classification
- Linear Models for Binary Classification
- Stochastic Gradient Descent
- Multiclass via Logistic Regression
- Multiclass via Binary Classification

Hsuan-Tien Lin (NTU CSIE)
Linear Models Revisited

linear scoring function: $s = w^T x$

linear classification

$h(x) = \text{sign}(s)$

- plausible err = 0/1
- discrete $E_{\text{in}}(w)$: NP-hard to solve

linear regression

$h(x) = s$

- friendly err = squared
- quadratic convex $E_{\text{in}}(w)$: closed-form solution

logistic regression

$h(x) = \theta(s)$

- plausible err = cross-entropy
- smooth convex $E_{\text{in}}(w)$: gradient descent

can linear regression or logistic regression help linear classification?
Error Functions Revisited

linear scoring function: \( s = \mathbf{w}^T \mathbf{x} \)

for binary classification \( y \in \{-1, +1\} \)

linear classification

\[
\begin{align*}
    h(\mathbf{x}) &= \text{sign}(s) \\
    \text{err}(h, \mathbf{x}, y) &= \left[ h(\mathbf{x}) \neq y \right] \\
    \text{err}_{0/1}(s, y) &= \left[ \text{sign}(s) \neq y \right] \\
    &= \left[ \text{sign}(ys) \neq 1 \right]
\end{align*}
\]

linear regression

\[
\begin{align*}
    h(\mathbf{x}) &= s \\
    \text{err}(h, \mathbf{x}, y) &= (h(\mathbf{x}) - y)^2 \\
    \text{err}_{\text{SQR}}(s, y) &= (s - y)^2 \\
    &= (ys - 1)^2
\end{align*}
\]

logistic regression

\[
\begin{align*}
    h(\mathbf{x}) &= \theta(s) \\
    \text{err}(h, \mathbf{x}, y) &= -\ln h(\mathbf{y} \mathbf{x}) \\
    \text{err}_{\text{CE}}(s, y) &= -\ln (1 + \exp(-ys))
\end{align*}
\]

\((ys):\) classification correctness score
Visualizing Error Functions

\[
\begin{align*}
0/1 \quad \text{err}_{0/1}(s, y) &= \mathbb{I}[\text{sign}(ys) \neq 1] \\
\text{sqr} \quad \text{err}_{\text{SQR}}(s, y) &= (ys - 1)^2 \\
\text{ce} \quad \text{err}_{\text{CE}}(s, y) &= \ln(1 + \exp(-ys)) \\
\text{scaled ce} \quad \text{err}_{\text{SCE}}(s, y) &= \log_2(1 + \exp(-ys))
\end{align*}
\]

- **0/1**: 1 iff \( ys \leq 0 \)
- **sqr**: large if \( ys \ll 1 \) \textbf{but} over-charge \( ys \gg 1 \)
  - small \( \text{err}_{\text{SQR}} \rightarrow \) small \( \text{err}_{0/1} \)
- **ce**: monotonic of \( ys \)
  - small \( \text{err}_{\text{CE}} \leftrightarrow \) small \( \text{err}_{0/1} \)
- **scaled ce**: a proper upper bound of \( 0/1 \)
  - small \( \text{err}_{\text{SCE}} \leftrightarrow \) small \( \text{err}_{0/1} \)

**upper bound:**
useful for designing algorithmic error \( \hat{\text{err}} \)
### Visualizing Error Functions

- **0/1**: $\text{err}_{0/1}(s, y) = \left[\text{sign}(ys) \neq 1\right]$
- **sqr**: $\text{err}_{\text{sqr}}(s, y) = (ys - 1)^2$
- **ce**: $\text{err}_{\text{ce}}(s, y) = \ln(1 + \exp(-ys))$
- **scaled ce**: $\text{err}_{\text{sce}}(s, y) = \log_2(1 + \exp(-ys))$

#### Graphical Illustration:
- **0/1**: 1 iff $ys \leq 0$
- **sqr**: large if $ys \ll 1$
  - **but** over-charge if $ys \gg 1$
  - small $\text{err}_{\text{sqr}} \rightarrow$ small $\text{err}_{0/1}$
- **ce**: monotonic of $ys$
  - small $\text{err}_{\text{ce}} \leftrightarrow$ small $\text{err}_{0/1}$
- **scaled ce**: a proper upper bound of 0/1
  - small $\text{err}_{\text{sce}} \leftrightarrow$ small $\text{err}_{0/1}$

#### Upper Bound: Useful for Designing Algorithmic Error $\hat{\text{err}}$
Linear Models for Classification

Linear Models for Binary Classification

Visualizing Error Functions

\[
\begin{align*}
0/1 \text{ err}_{0/1}(s, y) &= \lfloor \text{sign}(ys) \neq 1 \rfloor \\
sqr \text{ err}_{\text{sqr}}(s, y) &= (ys - 1)^2 \\
ce \text{ err}_{\text{ce}}(s, y) &= \ln(1 + \exp(-ys)) \\
scaled ce \text{ err}_{\text{sce}}(s, y) &= \log_2(1 + \exp(-ys))
\end{align*}
\]

- **0/1**: 1 iff \(ys \leq 0\)
- **sqr**: large if \(ys \ll 1\) but over-charge \(ys \gg 1\)
  - small \(\text{err}_{\text{sqr}}\) \(\rightarrow\) small \(\text{err}_{0/1}\)
- **ce**: monotonic of \(ys\)
  - small \(\text{err}_{\text{ce}}\) \(\leftrightarrow\) small \(\text{err}_{0/1}\)
- **scaled ce**: a proper upper bound of 0/1
  - small \(\text{err}_{\text{sce}}\) \(\leftrightarrow\) small \(\text{err}_{0/1}\)

**upper bound:**
useful for designing algorithmic error \(\hat{\text{err}}\)

Hsuan-Tien Lin (NTU CSIE)

Machine Learning Foundations
Visualizing Error Functions

\[
0/1 \quad \text{err}_{0/1}(s, y) = \left[ \text{sign}(ys) \neq 1 \right]
\]
\[
sqr \quad \text{err}_{SQR}(s, y) = (ys - 1)^2
\]
\[
ce \quad \text{err}_{CE}(s, y) = \ln(1 + \exp(-ys))
\]
\[
scaled \ ce \quad \text{err}_{SCE}(s, y) = \log_2(1 + \exp(-ys))
\]

- **0/1**: 1 iff \( ys \leq 0 \)
- **sqr**: large if \( ys \ll 1 \)
  - **but** over-charge \( ys \gg 1 \)
  - small \( \text{err}_{SQR} \rightarrow \) small \( \text{err}_{0/1} \)
- **ce**: monotonic of \( ys \)
  - small \( \text{err}_{CE} \leftrightarrow \) small \( \text{err}_{0/1} \)
- **scaled ce**: a proper upper bound of \( 0/1 \)
  - small \( \text{err}_{SCE} \leftrightarrow \) small \( \text{err}_{0/1} \)

**upper bound:**
useful for designing algorithmic error \( \hat{\text{err}} \)
Linear Models for Classification

Linear Models for Binary Classification

Theoretical Implication of Upper Bound

For any $ys$ where $s = w^T x$

$$\text{err}_{0/1}(s, y) \leq \text{err}_{SCE}(s, y) = \frac{1}{\ln 2} \text{err}_{CE}(s, y).$$

$$\implies E^{0/1}_{in}(w) \leq E^{SCE}_{in}(w) = \frac{1}{\ln 2} E^{CE}_{in}(w)$$

$$E^{0/1}_{out}(w) \leq E^{SCE}_{out}(w) = \frac{1}{\ln 2} E^{CE}_{out}(w)$$

VC on $0/1$:  

$$E^{0/1}_{out}(w) \leq E^{0/1}_{in}(w) + \Omega^{0/1}$$

$$\leq \frac{1}{\ln 2} E^{CE}_{in}(w) + \Omega^{0/1}$$

VC-Reg on CE:  

$$E^{0/1}_{out}(w) \leq \frac{1}{\ln 2} E^{CE}_{out}(w)$$

$$\leq \frac{1}{\ln 2} E^{CE}_{in}(w) + \frac{1}{\ln 2} \Omega^{CE}$$

small $E^{CE}_{in}(w)$ $\implies$ small $E^{0/1}_{out}(w)$:

logistic/linear reg. for linear classification
Regression for Classification

1. Run logistic/linear reg. on $D$ with $y_n \in \{-1, +1\}$ to get $w_{\text{REG}}$
2. Return $g(x) = \text{sign}(w_{\text{REG}}^T x)$

### PLA
- **Pros:** Efficient + strong guarantee if lin. separable
- **Cons:** Works only if lin. separable, otherwise needing pocket heuristic

### Linear Regression
- **Pros:** ‘easiest’ optimization
- **Cons:** Loose bound of $\frac{\text{err}_0}{1}$ for large $|ys|$

### Logistic Regression
- **Pros:** ‘easy’ optimization
- **Cons:** Loose bound of $\frac{\text{err}_0}{1}$ for very negative $ys$

- **Linear Regression** sometimes used to set $w_0$ for PLA/pocket/logistic regression
- **Logistic Regression** often preferred over pocket
Following the definition in the lecture, which of the following is not always $\geq \text{err}_{0/1}(y, s)$ when $y \in \{-1, +1\}$?

1. $\text{err}_{0/1}(y, s)$
2. $\text{err}_{SQR}(y, s)$
3. $\text{err}_{CE}(y, s)$
4. $\text{err}_{SCE}(y, s)$

Reference Answer: 3

Too simple, uh? :-) Anyway, note that $\text{err}_{0/1}$ is surely an upper bound of itself.
Two Iterative Optimization Schemes

For $t = 0, 1, \ldots$

$$w_{t+1} \leftarrow w_t + \eta v$$

when stop, return last $w$ as $g$

PLA
pick $(x_n, y_n)$ and decide $w_{t+1}$ by the one example
$O(1)$ time per iteration ::-

logistic regression (pocket)
check $D$ and decide $w_{t+1}$ (or new $\hat{w}$) by all examples
$O(N)$ time per iteration ::-(

logistic regression with $O(1)$ time per iteration?
Logistic Regression Revisited

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta \frac{1}{N} \sum_{n=1}^{N} \theta \left( -y_n \mathbf{w}_t^T \mathbf{x}_n \right) (y_n \mathbf{x}_n)$$

- want: update direction $\mathbf{v} \approx -\nabla E_{in}(\mathbf{w}_t)$
  while computing $\mathbf{v}$ by one single $(\mathbf{x}_n, y_n)$

- technique on removing $\frac{1}{N} \sum_{n=1}^{N}$:
  view as expectation $E$ over uniform choice of $n$!

stochastic gradient:
$$\nabla_{\mathbf{w}} \text{err}(\mathbf{w}, \mathbf{x}_n, y_n) \ \text{with random} \ n$$
true gradient:
$$\nabla_{\mathbf{w}} E_{in}(\mathbf{w}) = E_{\text{random} n} \nabla_{\mathbf{w}} \text{err}(\mathbf{w}, \mathbf{x}_n, y_n)$$
Stochastic Gradient Descent (SGD)

stochastic gradient = true gradient + zero-mean ‘noise’ directions

Stochastic Gradient Descent

- idea: replace true gradient by stochastic gradient
- after enough steps, average true gradient ≈ average stochastic gradient
- pros: simple & cheaper computation :-) —useful for big data or online learning
- cons: less stable in nature

SGD logistic regression, looks familiar? :-):

\[
\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta \theta \left( -y_n \mathbf{w}_t^T \mathbf{x}_n \right) \left( y_n \mathbf{x}_n \right) - \nabla \text{err}(\mathbf{w}_t, \mathbf{x}_n, y_n)
\]
PLA Revisited

SGD logistic regression:

\[ \mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta \cdot \theta \left(-y_n \mathbf{w}_t^T \mathbf{x}_n\right) (y_n \mathbf{x}_n) \]

PLA:

\[ \mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + 1 \cdot \left[y_n \neq \text{sign}(\mathbf{w}_t^T \mathbf{x}_n)\right] (y_n \mathbf{x}_n) \]

- SGD logistic regression \( \approx \) ‘soft’ PLA
- PLA \( \approx \) SGD logistic regression with \( \eta = 1 \) when \( \mathbf{w}_t^T \mathbf{x}_n \) large

Two practical rule-of-thumb:

- Stopping condition? \( t \) large enough
- \( \eta \)? 0.1 when \( \mathbf{x} \) in proper range
Consider applying SGD on linear regression for big data. What is the update direction when using the negative stochastic gradient?

1. $x_n$
2. $y_n x_n$
3. $2(w_t^T x_n - y_n)x_n$
4. $2(y_n - w_t^T x_n)x_n$

Reference Answer: 4

Go check lecture 9 if you have forgotten about the gradient of squared error. :-)

Anyway, the update rule has a nice physical interpretation: improve $w_t$ by ‘correcting’ proportional to the residual $(y_n - w_t^T x_n)$. 
Multiclass Classification

- $\mathcal{Y} = \{\square, \diamond, \triangle, \star\}$
  (4-class classification)
- many applications in practice, especially for ‘recognition’

next: use **tools for** $\{\times, \circ\}$ classification to $\{\square, \diamond, \triangle, \star\}$ classification
One Class at a Time

□ or not? \{□ = \circ, \diamond = \times, \triangle = \times, \star = \times\}
One Class at a Time

◊ or not? \{□ = ×, ◊ = ◦, △ = ×, ⋆ = ×\}
One Class at a Time

△ or not? \( \square = \times, \diamond = \times, \triangle = \circ, \star = \times \)
One Class at a Time

or not? \{\square = \times, \diamond = \times, \triangle = \times, \star = \circ\}
Multiclass Prediction: Combine Binary Classifiers

but ties? :-}
One Class at a Time Softly

\[ P(\square | \mathbf{x})? \{\square = \circ, \diamond = \times, \triangle = \times, \star = \times \} \]
One Class at a Time \textbf{Softly}

\[ P(\diamond | \mathbf{x})? \{\square = \times, \diamond = \circ, \triangle = \times, \star = \times \} \]
One Class at a Time \textbf{Softly}

$P(\triangle | \mathbf{x})$? \{□ = ×, ◊ = ×, △ = ○, ⋆ = ×\}
One Class at a Time \textbf{Softly}

\begin{align*}
P(\star | \mathbf{x})? \quad \{\square = \times, \diamond = \times, \triangle = \times, \star = \circ\}
\end{align*}
Multiclass Prediction: Combine **Soft** Classifiers

\[ g(x) = \arg\max_{k \in \mathcal{Y}} \theta (w_{[k]}^T x) \]
One-Versus-All (OVA) Decomposition

1. For $k \in \mathcal{Y}$, obtain $w[k]$ by running logistic regression on

   $$\mathcal{D}[k] = \{(x_n, y'_n = 2 [y_n = k] - 1)\}_{n=1}^{N}$$

2. Return $g(x) = \arg\max_{k \in \mathcal{Y}} (w^T[k] x)$

- **Pros**: efficient, can be coupled with any logistic regression-like approaches
- **Cons**: often unbalanced $\mathcal{D}[k]$ when $K$ large
- **Extension**: multinomial (‘coupled’) logistic regression

OVA: a simple multiclass meta-algorithm to keep in your toolbox
Which of the following best describes the training effort of OVA decomposition based on logistic regression on some $K$-class classification data of size $N$?

1. Learn $K$ logistic regression hypotheses, each from data of size $N/K$.
2. Learn $K$ logistic regression hypotheses, each from data of size $N \ln K$.
3. Learn $K$ logistic regression hypotheses, each from data of size $N$.
4. Learn $K$ logistic regression hypotheses, each from data of size $NK$.

Reference Answer: 3

Note that the learning part can be easily done in parallel, while the data is essentially of the same size as the original data.
Source of **Unbalance**: One versus All

idea: make binary classification problems more **balanced** by one versus one
One versus One at a Time

□ or ♦? \{□ = ◦, ♦ = ×, △ = nil, ⋆ = nil\}
One versus One at a Time

□ or △? \{□ = o, ◊ = nil, △ = x, ⋆ = nil\}
One versus One at a Time

□ or ★?

\{□ = ⭕, ◊ = nil, △ = nil, ★ = ✗\}
One versus One at a Time

◊ or △? \{□ = nil, ◊ = ○, △ = ×, ⋆ = nil\}
One versus One at a Time

◊ or ⋆? \{□ = nil, ◊ = ○, △ = nil, ⋆ = ✗\}
One versus One at a Time

△ or ★?
\[
\{ \square = \text{nil}, \diamond = \text{nil}, \triangle = \bigcirc, \ast = \times \}
\]
Multiclass Prediction: Combine **Pairwise** Classifiers

$$g(x) = \text{tournament champion} \left\{ w_{[k,\ell]}^T x \right\}$$

(voting of classifiers)
One-versus-one (OVO) Decomposition

1. for \((k, \ell) \in \mathcal{Y} \times \mathcal{Y}\)
   obtain \(w_{[k,\ell]}\) by running linear binary classification on
   
   \[ D_{[k,\ell]} = \{(x_n, y'_n = 2[y_n = k] - 1): y_n = k \text{ or } y_n = \ell\} \]

2. return \(g(x) = \text{tournament champion}\ \left\{ w_{[k,\ell]}^T x \right\} \)

• pros: efficient (‘smaller’ training problems), stable, can be coupled with any binary classification approaches

• cons: use \(O(K^2) w_{[k,\ell]}\)
   —more space, slower prediction, more training

OVO: another simple multiclass meta-algorithm to keep in your toolbox
Fun Time

Assume that some binary classification algorithm takes exactly $N^3$ CPU-seconds for data of size $N$. Also, for some 10-class multiclass classification problem, assume that there are $N/10$ examples for each class. Which of the following is total CPU-seconds needed for OVO decomposition based on the binary classification algorithm?

1. $\frac{9}{200}N^3$
2. $\frac{9}{25}N^3$
3. $\frac{4}{5}N^3$
4. $N^3$

Reference Answer: 2

There are 45 binary classifiers, each trained with data of size $(2N)/10$. Note that OVA decomposition with the same algorithm would take $10N^3$ time, much worse than OVO.
Summary

1. When Can Machines Learn?
2. Why Can Machines Learn?
3. How Can Machines Learn?

Lecture 10: Logistic Regression

Lecture 11: Linear Models for Classification

- Linear Models for Binary Classification
  three models useful in different ways
- Stochastic Gradient Descent
  follow negative stochastic gradient
- Multiclass via Logistic Regression
  predict with maximum estimated $P(k|x)$
- Multiclass via Binary Classification
  predict the tournament champion

- next: from linear to nonlinear

4. How Can Machines Learn Better?