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?

**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

4. How Can Machines Learn Better?
Linear Models Revisited

linear scoring function: $s = w^T x$
Linear Models Revisited

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

linear classification

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

plausible $\text{err} = 0/1$

**discrete $E_{in}(w)$:**
NP-hard to solve
Linear Models Revisited

-linear scoring function: \( s = w^T x \)

**linear classification**

\[ h(x) = \text{sign}(s) \]

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

**linear regression**

\[ h(x) = s \]

- friendly err = squared
- quadratic convex \( E_{in}(w) \): closed-form solution
Linear Models Revisited

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

### Linear Classification

- **Function:** \( h(\mathbf{x}) = \text{sign}(s) \)
- **Output:** \( s \)
- **Plausible error:** \( \text{err} = 0/1 \)
- **Discrete \( E_{\text{in}}(\mathbf{w}) \):** NP-hard to solve

### Linear Regression

- **Function:** \( h(\mathbf{x}) = s \)
- **Output:** \( s \)
- **Friendly error:** \( \text{err} = \text{squared} \)
- **Quadratic convex \( E_{\text{in}}(\mathbf{w}) \):** closed-form solution

### Logistic Regression

- **Function:** \( h(\mathbf{x}) = \theta(s) \)
- **Output:** \( s \)
- **Plausible error:** \( \text{err} = \text{cross-entropy} \)
- **Smooth convex \( E_{\text{in}}(\mathbf{w}) \):** gradient descent
Linear Models Revisited

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

linear classification

\[ h(\mathbf{x}) = \text{sign}(s) \]

plausible err = 0/1

discrete \( E_{in}(\mathbf{w}) \): NP-hard to solve

linear regression

\[ h(\mathbf{x}) = s \]

friendly err = squared

quadratic convex \( E_{in}(\mathbf{w}) \): closed-form solution

logistic regression

\[ h(\mathbf{x}) = \theta(s) \]

plausible err = cross-entropy

smooth convex \( E_{in}(\mathbf{w}) \): gradient descent

can linear regression or logistic regression help linear classification?
Linear Models for Classification

Error Functions Revisited

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

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

linear classification

\[
\begin{align*}
h(\mathbf{x}) &= \text{sign}(s) \\
\text{err}(h, \mathbf{x}, y) &= \mathbb{I}[h(\mathbf{x}) \neq y]
\end{align*}
\]

\[
\text{err}_{0/1}(s, y) = \begin{cases} 0 & \text{if } s = y \\ 1 & \text{otherwise} \end{cases}
\]
Linear Models for Classification

Error Functions Revisited

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

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

Linear classification

\[
\begin{align*}
  h(\mathbf{x}) &= \text{sign}(s) \\
  \text{err}(h, \mathbf{x}, y) &= \mathbb{I}[h(\mathbf{x}) \neq y] \\
  \text{err}_{0/1}(s, y) &= \mathbb{I}[\text{sign}(s) \neq y]
\end{align*}
\]
Error Functions Revisited

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

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

linear classification

\[
\begin{align*}
    h(\mathbf{x}) &= \text{sign}(s) \\
    \text{err}(h, \mathbf{x}, y) &= \mathbb{I}[h(\mathbf{x}) \neq y]
\end{align*}
\]

\[
\begin{align*}
    \text{err}_{0/1}(s, y) &= \mathbb{I}[\text{sign}(s) \neq y] \\
    &= \mathbb{I}[\text{sign}(ys) \neq 1]
\end{align*}
\]
Error Functions Revisited

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

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

**linear classification**

\[
\begin{align*}
  h(\mathbf{x}) &= \text{sign}(s) \\
  \text{err}(h, \mathbf{x}, y) &= \mathbb{I}[h(\mathbf{x}) \neq y] \\
  \text{err}_0/1(s, y) &= \mathbb{I}[\text{sign}(s) \neq y] \\
  &= \mathbb{I}[\text{sign}(ys) \neq 1]
\end{align*}
\]

**linear regression**

\[
\begin{align*}
  h(\mathbf{x}) &= s \\
  \text{err}(h, \mathbf{x}, y) &= (h(\mathbf{x}) - y)^2 \\
  \text{err}_{SQR}(s, y) &= \text{err}_0/1(s, y) \neq 1
\end{align*}
\]
### Error Functions Revisited

A **linear scoring function** is given by $s = \mathbf{w}^T \mathbf{x}$, where $\mathbf{w}$ is the weight vector and $\mathbf{x}$ is the feature vector.

For binary classification $y \in \{-1, +1\}$,

- **Linear Classification**
  
  $h(\mathbf{x}) = \text{sign}(s)$
  
  $\text{err}(h, \mathbf{x}, y) = \mathbb{I}[h(\mathbf{x}) \neq y]$

  $\text{err}_{0/1}(s, y) = \mathbb{I}[\text{sign}(s) \neq y]$
  
  $= \mathbb{I}[\text{sign}(ys) \neq 1]$

- **Linear Regression**
  
  $h(\mathbf{x}) = s$
  
  $\text{err}(h, \mathbf{x}, y) = (h(\mathbf{x}) - y)^2$

  $\text{err}_{\text{SQR}}(s, y) = (s - y)^2$
Error Functions Revisited

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

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

**linear classification**

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

**linear regression**

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

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

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

<table>
<thead>
<tr>
<th>Linear classification</th>
<th>Linear regression</th>
<th>Logistic regression</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h(\mathbf{x}) = \text{sign}(s)$</td>
<td>$h(\mathbf{x}) = s$</td>
<td>$h(\mathbf{x}) = \theta(s)$</td>
</tr>
<tr>
<td>$\text{err}(h, \mathbf{x}, y) = [h(\mathbf{x}) \neq y]$</td>
<td>$\text{err}(h, \mathbf{x}, y) = (h(\mathbf{x}) - y)^2$</td>
<td>$\text{err}(h, \mathbf{x}, y) = -\ln h(y \mathbf{x})$</td>
</tr>
<tr>
<td>$\text{err}_{0/1}(s, y) = [\text{sign}(s) \neq y]$</td>
<td>$\text{err}_{\text{SQR}}(s, y) = (s - y)^2$</td>
<td>$\text{err}_{\text{CE}}(s, y) = (ys - 1)^2$</td>
</tr>
</tbody>
</table>
Error Functions Revisited

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

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

**linear classification**

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

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

**linear regression**

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

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

**logistic regression**

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

\[
\begin{align*}
    \text{err}_{\text{CE}}(s, y) & = -\ln(1 + \exp(-ys))
\end{align*}
\]
Error Functions Revisited

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

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

**Linear Classification**

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

**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**

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

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

\[ \text{err}_{0/1}(s, y) = \lfloor \text{sign}(ys) \neq 1 \rfloor \]

- 0/1: 1 iff \( ys \leq 0 \)
Visualizing Error Functions

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

- **0/1**: 1 iff \( ys \leq 0 \)
- **sqr**: large if \( ys \ll 1 \)
  - **but** over-charge if \( ys \gg 1 \)
- \( \text{small} \ \text{err}_{\text{SQR}} \rightarrow \text{small} \ \text{err}_{0/1} \)
Visualizing Error Functions

\[ \text{0/1: } \text{err}^{0/1}_{0/1}(s, y) = [\text{sign}(ys) \neq 1] \]
\[ \text{sqr: } \text{err}^{\text{sqr}}(s, y) = (ys - 1)^2 \]
\[ \text{ce: } \text{err}^{\text{CE}}(s, y) = \ln(1 + \exp(-ys)) \]

- 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 \( \text{err}^{0/1} \)
Visualizing Error Functions

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

- **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 \( \hat{\text{err}} \)
Theoretical Implication of Upper Bound

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

$$\text{err}_{0/1}(s, y) \leq \text{err}_{\text{SCE}}(s, y) = \frac{1}{\ln 2} \text{err}_{\text{CE}}(s, y).$$
For any $ys$ where $s = w^T x$

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

$$\implies E_{\text{in}}^{0/1}(w) \leq$$
Theoretical Implication of Upper Bound

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

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

$$\implies E_{in}^{0/1}(w) \leq E_{in}^{SCE}(w) = \frac{1}{\ln 2} E_{in}^{CE}(w).$$
For any $ys$ where $s = \mathbf{w}^T \mathbf{x}$

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

$$\Rightarrow \quad E_{\text{in}}^{0/1}(\mathbf{w}) \leq E_{\text{in}}^{\text{SCE}}(\mathbf{w}) = \frac{1}{\ln 2} E_{\text{in}}^{\text{CE}}(\mathbf{w})$$

$$E_{\text{out}}^{0/1}(\mathbf{w}) \leq E_{\text{out}}^{\text{SCE}}(\mathbf{w}) = \frac{1}{\ln 2} E_{\text{out}}^{\text{CE}}(\mathbf{w})$$
Theoretical Implication of Upper Bound

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

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

$$\implies E_{0/1}^0(w) \leq E_{\text{SCE}}^0(w) = \frac{1}{\ln 2} E_{\text{CE}}^0(w)$$

$$E_{\text{out}}^0(w) \leq E_{\text{SCE}}^0(w) = \frac{1}{\ln 2} E_{\text{CE}}^0(w)$$

VC on 0/1:

$$E_{\text{out}}^{0/1}(w) \leq$$
Theoretical Implication of Upper Bound

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

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

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

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

VC on $0/1$:

$$E^{0/1}_{\text{out}}(w) \leq E^{0/1}_{\text{in}}(w) + \Omega^{0/1}$$
Theoretical Implication of Upper Bound

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

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

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

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

VC on 0/1:

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

$$\leq \frac{1}{\ln 2} E^{\text{CE}}_{\text{in}}(w) + \Omega^0/1$$
Theoretical Implication of Upper Bound

For any $y$s where $s = \mathbf{w}^T \mathbf{x}$

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

$$\Rightarrow \quad E_{\text{in}}^{0/1}(\mathbf{w}) \leq E_{\text{in}}^{\text{SCE}}(\mathbf{w}) = \frac{1}{\ln 2} E_{\text{in}}^{\text{CE}}(\mathbf{w})$$

$$E_{\text{out}}^{0/1}(\mathbf{w}) \leq E_{\text{out}}^{\text{SCE}}(\mathbf{w}) = \frac{1}{\ln 2} E_{\text{out}}^{\text{CE}}(\mathbf{w})$$

VC on 0/1:

$$E_{\text{out}}^{0/1}(\mathbf{w}) \leq E_{\text{in}}^{0/1}(\mathbf{w}) + \Omega^{0/1}$$

$$\leq \frac{1}{\ln 2} E_{\text{in}}^{\text{CE}}(\mathbf{w}) + \Omega^{0/1}$$
Theoretical Implication of Upper Bound

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

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

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

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

VC on 0/1:

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

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

VC-Reg on CE:

$$E_{\text{out}}^{0/1}(w) \leq \frac{1}{\ln 2} E_{\text{out}}^{\text{CE}}(w)$$
Theoretical Implication of Upper Bound

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

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

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

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

VC on 0/1:

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

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

VC-Reg on CE:

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

$$\leq \frac{1}{\ln 2} E_{\text{in}}^{\text{CE}}(w) + \frac{1}{\ln 2} \Omega^{\text{CE}}$$
Theoretical Implication of Upper Bound

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

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

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

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

VC on 0/1:

$$
E_{out}^{0/1}(w) \leq E_{in}^{0/1}(w) + \Omega_{0/1}^{0/1}
$$

$$
\leq \frac{1}{\ln 2} E_{in}^{CE}(w) + \Omega_{0/1}^{0/1}
$$

VC-Reg on CE:

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

$$
\leq \frac{1}{\ln 2} E_{in}^{CE}(w) + \frac{1}{\ln 2} \Omega_{CE}^{CE}
$$

small $E_{in}^{CE}(w) \implies$ small $E_{out}^{0/1}(w)$:
Theoretical Implication of Upper Bound

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

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

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

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

VC on 0/1:

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

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

VC-Reg on CE:

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

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

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

logistic/linear reg. for linear classification
1. run logistic/linear reg. on $\mathcal{D}$ with $y_n \in \{-1, +1\}$ to get $w_{\text{REG}}$
Regression for Classification

1. run logistic/linear reg. on $\mathcal{D}$ with $y_n \in \{-1, +1\}$ to get $w_{\text{REG}}$
2. return $g(x) = \text{sign}(w^T_{\text{REG}}x)$
Regression for Classification

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

linear regression

- **pros:**
  - 'easiest' optimization

- **cons:**
  - works only if linear separable, otherwise needing pocket heuristic
Regression for Classification

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

### linear regression

- **pros:**
  - ‘easiest’ optimization
- **cons:** loose bound of $\text{err}_{0/1}$ for large $|ys|$
Regression for Classification

1. Run logistic/linear reg. on $\mathcal{D}$ with $y_n \in \{-1, +1\}$ to get $w_{\text{REG}}$

2. Return $g(x) = \text{sign}(w_{\text{REG}}^T x)$

**Linear regression**
- **pros:** ‘easiest’ optimization
- **cons:** loose bound of $\text{err}_{0/1}$ for large $|y_s|$
Regression for Classification

1. Run **logistic/linear reg.** on $\mathcal{D}$ with $y_n \in \{-1, +1\}$ to get $\mathbf{w}_{\text{REG}}$

2. Return $g(\mathbf{x}) = \text{sign}(\mathbf{w}_{\text{REG}}^T \mathbf{x})$

### Linear Regression
- **Pros:** ‘easiest’ optimization
- **Cons:** loose bound of $\text{err}_{0/1}$ for large $|y_s|

### Logistic Regression
- **Pros:** ‘easy’ optimization
- **Cons:** loose bound of $\text{err}_{0/1}$ for very negative $y_s$
Regression for Classification

1. run **logistic/linear reg.** on $\mathcal{D}$ with $y_n \in \{-1, +1\}$ to get $\mathbf{w}_{\text{REG}}$
2. return $g(\mathbf{x}) = \text{sign}(\mathbf{w}_{\text{REG}}^T \mathbf{x})$

### PLA
- **pros:** efficient + strong guarantee if lin. separable

### Linear Regression
- **pros:** ‘easiest’ optimization
- **cons:** loose bound of $\text{err}_{0/1}$ for large $|ys|$  

### Logistic Regression
- **pros:** ‘easy’ optimization
- **cons:** loose bound of $\text{err}_{0/1}$ for very negative $ys$
Regression for Classification

1. run logistic/linear reg. on $\mathcal{D}$ 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 $err_{0/1}$ for large $|ys|$ for large $|ys|$ for very negative $ys$

**logistic regression**
- **pros:** ‘easy’ optimization
- **cons:** loose bound of $err_{0/1}$ for very negative $ys$
### Linear Models for Classification

**Linear Models for Binary Classification**

**Regression for Classification**

1. run **logistic/linear reg.** on $\mathcal{D}$ 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 $\text{err}_{0/1}$ for large $|ys|$ for

#### Logistic Regression
- **pros:** ‘easy’ optimization
- **cons:** loose bound of $\text{err}_{0/1}$ for very negative $ys$

**linear regression** sometimes used to set $w_0$ for PLA/pocket/logistic regression
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 $\frac{\text{err}_{0/1}}{1}$ for large $|ys|$ for large $|ys|$

**Logistic Regression**
- pros: ‘easy’ optimization
- cons: loose bound of $\frac{\text{err}_{0/1}}{1}$ 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.
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 \( t = 0, 1, \ldots \)

\[
\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta \mathbf{v}
\]

when stop, return last \( \mathbf{w} \) as \( g \)

**PLA**

pick \((x_n, y_n)\) and decide \( \mathbf{w}_{t+1} \) by the one example

\( O(1) \) time per iteration :-(

---

Hsuan-Tien Lin (NTU CSIE)
Two Iterative Optimization Schemes

For $t = 0, 1, \ldots$

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta \mathbf{v}$$

when stop, return last $\mathbf{w}$ as $g$

**PLA**

pick $(\mathbf{x}_n, y_n)$ and decide $\mathbf{w}_{t+1}$ by the one example

$O(1)$ time per iteration :-)

**logistic regression (pocket)**

check $\mathcal{D}$ and decide $\mathbf{w}_{t+1}$ (or new $\hat{\mathbf{w}}$) by all examples

$O(N)$ time per iteration :-(
Two Iterative Optimization Schemes

For $t = 0, 1, \ldots$

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta \mathbf{v}$$

when stop, return last $\mathbf{w}$ as $g$

**PLA**

pick $(\mathbf{x}_n, y_n)$ and decide $\mathbf{w}_{t+1}$ by the one example

$O(1)$ time per iteration :-)

**logistic regression (pocket)**

check $\mathcal{D}$ and decide $\mathbf{w}_{t+1}$ (or new $\hat{\mathbf{w}}$) by all examples

$O(N)$ time per iteration :-(

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

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

\[ -\nabla E_{\text{in}}(\mathbf{w}_t) \]
Logistic Regression Revisited

\[ w_{t+1} \leftarrow w_t + \eta \frac{1}{N} \sum_{n=1}^{N} \theta \left( -y_n w_t^T x_n \right) (y_n x_n) \]

- want: update direction \( \mathbf{v} \approx -\nabla E_{\text{in}}(w_t) \)
  while computing \( \mathbf{v} \) by one single \((x_n, y_n)\)
Logistic Regression Revisited

\[ w_{t+1} \leftarrow w_t + \eta \frac{1}{N} \sum_{n=1}^{N} \theta \left( -y_n w_t^T x_n \right) (y_n x_n) \]

- want: update direction \( v \approx -\nabla E_{in}(w_t) \) while computing \( v \) by one single \((x_n, y_n)\)

- technique on removing \( \frac{1}{N} \sum_{n=1}^{N} \)

view as expectation \( \mathbb{E} \) over uniform choice of \( n! \)
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 \( \mathbf{v} \approx -\nabla E_{\text{in}}(\mathbf{w}_t) \)

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 \( \mathcal{E} \) over uniform choice of \( n \)!

stochastic gradient:

\[ \nabla_{\mathbf{w}} \text{err}(\mathbf{w}, \mathbf{x}_n, y_n) \text{ with random } n \]
Logistic Regression Revisited

\[ w_{t+1} \leftarrow w_t + \eta \frac{1}{N} \sum_{n=1}^{N} \theta \left( -y_n w_T x_n \right) (y_n x_n) \]

\[-\nabla E_{in}(w_t)\]

- want: update direction \( v \approx -\nabla E_{in}(w_t) \)
  while computing \( v \) by one single \((x_n, y_n)\)

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

stochastic gradient:
\[ \nabla_w \text{err}(w, x_n, y_n) \text{ with random } n \]
true gradient:
\[ \nabla_w E_{in}(w) = \mathcal{E} \nabla_w \text{err}(w, x_n, y_n) \text{ random } n \]
Stochastic Gradient Descent (SGD)

\[ \text{stochastic gradient} = \text{true gradient} + \text{zero-mean ‘noise’ directions} \]
Stochastic Gradient Descent (SGD)

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

\[
\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta \theta (y_n \mathbf{x}_n^T \mathbf{x}_n) - \nabla \text{err} (\mathbf{w}_t, \mathbf{x}_n, y_n)
\]
Stochastic Gradient Descent (SGD)

**Stochastic gradient** = **true gradient** + **zero-mean ‘noise’ directions**

- **idea**: replace **true gradient** by **stochastic gradient**

**SGD logistic regression**: looks familiar? :-)

\[ w_{t+1} \leftarrow w_t + \eta \theta \left( -y_n w^T x_n \right) y_n x_n \]

\[ \approx \nabla \text{err}(w_t, x_n, y_n) \]
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**
Stochastic Gradient Descent (SGD)

\[
\text{stochastic gradient} = \text{true gradient} + \text{zero-mean ‘noise’ directions}
\]

- **idea:** replace \text{true gradient} by \text{stochastic gradient}
- **after enough steps,**
  \[
  \text{average true gradient} \approx \text{average stochastic gradient}
  \]
- **pros:** \text{simple & cheaper computation :-)}
  —useful for \text{big data} or \text{online learning}
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 \approx \text{average stochastic gradient}
- pros: simple & cheaper computation :-)
  —useful for big data or online learning
- cons: less stable in nature
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,
  \[ \text{average true gradient} \approx \text{average stochastic gradient} \]
- 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:

\[ w_{t+1} \leftarrow w_t + \eta \cdot \theta \left( -y_n w_T^n x_n \right) (y_n x_n) \]

PLA:

\[ w_{t+1} \leftarrow w_t + 1 \cdot \left[ y_n \neq \text{sign}(w_T^n x_n) \right] (y_n x_n) \]
PLA Revisited

SGD logistic regression:

\[ w_{t+1} \leftarrow w_t + \eta \cdot \theta \left( -y_n w_t^T x_n \right) (y_n x_n) \]

PLA:

\[ w_{t+1} \leftarrow w_t + 1 \cdot \left[ y_n \neq \text{sign}(w_t^T x_n) \right] (y_n x_n) \]

• SGD logistic regression \( \approx \) ‘soft’ PLA
PLA Revisited

SGD logistic regression:

\[ w_{t+1} \leftarrow w_t + \eta \cdot \theta \left( -y_n w_t^T x_n \right) (y_n x_n) \]

PLA:

\[ w_{t+1} \leftarrow w_t + 1 \cdot \left[ y_n \neq \text{sign}(w_t^T x_n) \right] (y_n x_n) \]

- SGD logistic regression \( \approx \) ‘soft’ PLA
- PLA \( \approx \) SGD logistic regression with \( \eta = 1 \) when \( w_t^T x_n \) large
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:
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?
PLA Revisited

SGD logistic regression:

\[ w_{t+1} \leftarrow w_t + \eta \cdot \theta \left( -y_n w_T^T x_n \right) (y_n x_n) \]

PLA:

\[ w_{t+1} \leftarrow w_t + 1 \cdot \left[ y_n \neq \text{sign}(w_T^T x_n) \right] (y_n x_n) \]

- SGD logistic regression \( \approx \) ‘soft’ PLA
- PLA \( \approx \) SGD logistic regression with \( \eta = 1 \) when \( w_T^T x_n \) large

Two practical rule-of-thumb:

- stopping condition? \( t \) large enough
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
- \( \eta \)?
PLA Revisited

SGD logistic regression:

\[ w_{t+1} \leftarrow w_t + \eta \cdot \theta \left( -y_n w^T x_n \right) (y_n x_n) \]

PLA:

\[ w_{t+1} \leftarrow w_t + 1 \cdot \left[ y_n \neq \text{sign}(w^T x_n) \right] (y_n x_n) \]

- SGD logistic regression \( \approx \) ‘soft’ PLA
- PLA \( \approx \) SGD logistic regression with \( \eta = 1 \) when \( w^T x_n \) large

Two practical rule-of-thumb:

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

1. $\mathbf{x}_n$
2. $y_n\mathbf{x}_n$
3. $2(\mathbf{w}_t^T \mathbf{x}_n - y_n)\mathbf{x}_n$
4. $2(y_n - \mathbf{w}_t^T \mathbf{x}_n)\mathbf{x}_n$
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

- \( \mathcal{Y} = \{\Box, \Diamond, \triangle, \star\} \)
  (4-class classification)
- many applications in practice, especially for ‘recognition’
Multiclass Classification

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

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

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

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

△ or not? \( \{\Box = \times, \lozenge = \times, \triangle = \circ, \star = \times\} \)
or not? \{□ = \times, \diamond = \times, \triangle = \times, \star = \circ\}
Multiclass Prediction: Combine Binary Classifiers

but ties? :-)

Hsuan-Tien Lin (NTU CSIE)
One Class at a Time \textbf{Softly}

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

\[ P(\diamond | \mathbf{x})? \{ \square = \times, \diamond = \circ, \triangle = \times, \star = \times \} \]
Linear Models for Classification

Multiclass via Logistic Regression

One Class at a Time **Softly**

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

\[ P(\star | \mathbf{x})? \{ \Box = \times, \diamond = \times, \triangle = \times, \star = \circ \} \]
Multiclass Prediction: Combine **Soft** Classifiers

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

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

   $$\mathcal{D}_k = \{(x_n, y_n' = 2[y_n = k] - 1)\}^N_{n=1}$$

   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
One-Versus-All (OVA) Decomposition

1. for $k \in \mathcal{Y}$
   obtain $w_k$ by running logistic regression on
   
   $$\mathcal{D}_k = \{(x_n, y'_n = 2 \left[ y_n = k \right] - 1)\}_{n=1}^N$$

2. return $g(x) = \arg\max_{k \in \mathcal{Y}} \left( w_k^T x \right)$
One-Versus-All (OVA) Decomposition

1. For $k \in \mathcal{Y}$
   - obtain $w[k]$ by running logistic regression on
     $$D[k] = \left\{ (x_n, y'_n = 2 \lfloor y_n = k \rfloor - 1) \right\}_{n=1}^{N}$$

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

- **Pros:**
  - Efficient,
  - Can be coupled with any logistic regression-like approaches

---

Hsuan-Tien Lin (NTU CSIE)  
Machine Learning Foundations  
18/25
One-Versus-All (OVA) Decomposition

1. for $k \in \mathcal{Y}$
   obtain $w_k$ by running logistic regression on
   \[ D_k = \{(x_n, y'_n = 2 \mathbb{1}[y_n = k] - 1)\}_{n=1}^N \]

2. return $g(x) = \arg\max_{k \in \mathcal{Y}} (w_k^T x)$

- **pros**: efficient, can be coupled with any logistic regression-like approaches
- **cons**: often *unbalanced* $D_k$ when $K$ large
One-Versus-All (OVA) Decomposition

1. for $k \in \mathcal{Y}$
   obtain $w_k$ by running logistic regression on
   \[ D_k = \{(x_n, y'_n = 2 \mathbb{1}[y_n = k] - 1)\}^N_{n=1} \]

2. return $g(x) = \arg\max_{k \in \mathcal{Y}} \left( w^T_k x \right)$

- pros: efficient, can be coupled with any logistic regression-like approaches
- cons: often unbalanced $D_k$ when $K$ large
- extension: multinomial (‘coupled’) logistic regression
One-Versus-All (OVA) Decomposition

1. for $k \in \mathcal{Y}$
   obtain $w[k]$ by running logistic regression on
   \[
   \mathcal{D}_k = \{(x_n, y'_n = 2 \mathbb{1}[y_n = k] - 1)\}_{n=1}^N
   \]

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

- **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 $NK$
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
Source of **Unbalance**: One versus All

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

□ or ◊? \{□ = o, ◊ = x, △ = 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 △?

\{\square = \text{nil}, \diamondsuit = \circ, \triangle = \times, \star = \text{nil}\}
One versus One at a Time

\[ \diamond \text{ or } \star \text{? } \{\square = \text{nil}, \diamond = \circ, \triangle = \text{nil}, \star = \times\} \]
One versus One at a Time

△ or ★? \{□ = nil, ◆ = nil, △ = ○, ★ = ×\}
Multiclass Prediction: Combine Pairwise Classifiers
Multiclass Prediction: Combine **Pairwise** Classifiers

\[ g(x) = \text{tournament champion} \left\{ w^T_{[k,\ell]} x \right\} \]

(voting of classifiers)
Multiclass Prediction: Combine **Pairwise** Classifiers

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

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

1. For \((k, \ell) \in \mathcal{Y} \times \mathcal{Y}\) obtain \(\mathbf{w}_{[k,\ell]}\) by running linear binary classification on

\[
\mathcal{D}_{[k,\ell]} = \{(\mathbf{x}_n, y'_n = 2 [y_n = k] - 1) : y_n = k \text{ or } y_n = \ell\}
\]
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

\[
\mathcal{D}_{k,\ell} = \{(x_n, y'_n = 2[y_n = k] - 1): y_n = k \text{ or } y_n = \ell\}
\]

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

- pros: efficient (‘smaller’ training problems), stable, can be coupled with any binary classification approaches
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 \(g(x) = \text{tournament champion } \{w_{[k,\ell]}^T x\}\)

- **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
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
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. $N^3$
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?

Hsuan-Tien Lin (NTU CSIE)