Lecture 5: Kernel Logistic Regression

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. Embedding Numerous Features: Kernel Models

Lecture 4: Soft-Margin Support Vector Machine
- allow some margin violations $\xi_n$ while penalizing them by $C$; equivalent to upper-bounding $\alpha_n$ by $C$

Lecture 5: Kernel Logistic Regression
- Soft-Margin SVM as Regularized Model
- SVM versus Logistic Regression
- SVM for Soft Binary Classification
- Kernel Logistic Regression

2. Combining Predictive Features: Aggregation Models

3. Distilling Implicit Features: Extraction Models
**Hard-Margin Primal**

\[
\min_{b,w} \quad \frac{1}{2} w^T w \\
\text{s.t.} \quad y_n (w^T z_n + b) \geq 1
\]

**Soft-Margin Primal**

\[
\min_{b,w,\xi} \quad \frac{1}{2} w^T w + C \sum_{n=1}^{N} \xi_n \\
\text{s.t.} \quad y_n (w^T z_n + b) \geq 1 - \xi_n, \xi_n \geq 0
\]

**Hard-Margin Dual**

\[
\min_{\alpha} \quad \frac{1}{2} \alpha^T Q \alpha - 1^T \alpha \\
\text{s.t.} \quad y^T \alpha = 0 \\
0 \leq \alpha_n
\]

**Soft-Margin Dual**

\[
\min_{\alpha} \quad \frac{1}{2} \alpha^T Q \alpha - 1^T \alpha \\
\text{s.t.} \quad y^T \alpha = 0 \\
0 \leq \alpha_n \leq C
\]

*soft-margin preferred in practice; linear: LIBLINEAR; non-linear: LIBSVM*
Slack Variables $\xi_n$

- record ‘margin violation’ by $\xi_n$
- penalize with margin violation

$$\min_{b, w, \xi} \frac{1}{2} w^T w + C \cdot \sum_{n=1}^{N} \xi_n$$

s.t.

$$y_n(w^T z_n + b) \geq 1 - \xi_n \text{ and } \xi_n \geq 0 \text{ for all } n$$

on any $(b, w)$, $\xi_n = \text{margin violation} = \max(1 - y_n(w^T z_n + b), 0)$

- $(x_n, y_n)$ violating margin: $\xi_n = 1 - y_n(w^T z_n + b)$
- $(x_n, y_n)$ not violating margin: $\xi_n = 0$

‘unconstrained’ form of soft-margin SVM:

$$\min_{b, w} \frac{1}{2} w^T w + C \sum_{n=1}^{N} \max(1 - y_n(w^T z_n + b), 0)$$
Kernel Logistic Regression

**Unconstrained Form**

\[
\min_{b,w} \frac{1}{2} w^T w + C \sum_{n=1}^{N} \max(1 - y_n(w^T z_n + b), 0)
\]

**familiar? :-)**

\[
\min \frac{1}{2} w^T w + C \sum \hat{e}_{\text{err}}
\]

**just L2 regularization**

\[
\min \frac{\lambda}{N} w^T w + \frac{1}{N} \sum e_{\text{err}}
\]

**with shorter w, another parameter, and special err**

**why not solve this? :-)**

- not QP, no (?) kernel trick
- \(\max(\cdot, 0)\) **not differentiable**, harder to solve
### SVM as Regularized Model

<table>
<thead>
<tr>
<th>Regularization by Constraint</th>
<th>Minimize</th>
<th>Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hard-margin SVM</td>
<td>$E_{in}$</td>
<td>$w^T w \leq C$</td>
</tr>
<tr>
<td>L2 Regularization</td>
<td>$\frac{\lambda}{N} w^T w + E_{in}$</td>
<td>$E_{in} = 0$ [and more]</td>
</tr>
<tr>
<td>Soft-margin SVM</td>
<td>$\frac{1}{2} w^T w + C N \hat{E}_{in}$</td>
<td></td>
</tr>
</tbody>
</table>

- Large margin $\iff$ fewer hyperplanes $\iff$ L2 regularization of short $w$
- Soft margin $\iff$ special $\hat{err}$
- Larger $C$ or $C$ $\iff$ smaller $\lambda$ $\iff$ less regularization

Viewing SVM as regularized model:

- Allows **extending/connecting** to other learning models
When viewing soft-margin SVM as regularized model, a larger $C$ corresponds to:

1. a larger $\lambda$, that is, stronger regularization
2. a smaller $\lambda$, that is, stronger regularization
3. a larger $\lambda$, that is, weaker regularization
4. a smaller $\lambda$, that is, weaker regularization
When viewing soft-margin SVM as regularized model, a larger $C$ corresponds to

1. a larger $\lambda$, that is, stronger regularization
2. a smaller $\lambda$, that is, stronger regularization
3. a larger $\lambda$, that is, weaker regularization
4. a smaller $\lambda$, that is, weaker regularization

Reference Answer: 4

Comparing the formulations on page 4 of the slides, we see that $C$ corresponds to $\frac{1}{2\lambda}$. So larger $C$ corresponds to smaller $\lambda$, which surely means weaker regularization.
Algorithmic Error Measure of SVM

\[
\min_{b,w} \frac{1}{2} w^T w + C \sum_{n=1}^{N} \max(1 - y_n(w^T z_n + b), 0)
\]

linear score \( s = w^T z_n + b \)

- \( \text{err}_{0/1}(s, y) = [ys \leq 0] \)
- \( \hat{\text{err}}_{SVM}(s, y) = \max(1 - ys, 0) \):
  - upper bound of \( \text{err}_{0/1} \)
  - often called \textbf{hinge error measure}

\( \hat{\text{err}}_{SVM} \): \textbf{algorithmic error measure} by \textbf{convex upper bound} of \( \text{err}_{0/1} \)
Algorithmic Error Measure of SVM

\[
\min_{b,w} \frac{1}{2} w^T w + C \sum_{n=1}^{N} \max(1 - y_n(w^T z_n + b), 0)
\]

linear score \( s = w^T z_n + b \)
- \( \text{err}_{0/1}(s, y) = \lceil ys \leq 0 \rceil \)
- \( \hat{\text{err}}_{\text{SVM}}(s, y) = \max(1 - ys, 0) \):
  upper bound of \( \text{err}_{0/1} \)
  —often called hinge error measure

\( \hat{\text{err}}_{\text{SVM}} \): algorithmic error measure
by convex upper bound of \( \text{err}_{0/1} \)
Connection between SVM and Logistic Regression

linear score \( s = w^T z_n + b \)

- \( \text{err}_{0/1}(s, y) = \lceil y s \leq 0 \rceil \)
- \( \hat{\text{err}}_{\text{SVM}}(s, y) = \max(1 - y s, 0) \): upper bound of \( \text{err}_{0/1} \)
- \( \text{err}_{\text{SCE}}(s, y) = \log_2(1 + \exp(-y s)) \): another upper bound of \( \text{err}_{0/1} \) used in logistic regression

### Graph

- \( y s \) vs. \( \text{err} \)
- SVM
- Scaled CE

### Approximations

- \( -\infty \approx -y s \)
- \( -y s = 0 \)
- \( -y s \approx 0 \)

**SVM \approx L2-regularized logistic regression**
### Linear Models for Binary Classification

<table>
<thead>
<tr>
<th>Method</th>
<th>Objective</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLA</td>
<td>Minimize $\text{err}_{0/1}$ specially</td>
<td>Pros: efficient if lin. separable</td>
<td>Cons: works only if lin. separable, otherwise needing pocket</td>
</tr>
<tr>
<td>soft-margin SVM</td>
<td>Minimize regularized $\hat{\text{err}}_{\text{SVM}}$ by QP</td>
<td>Pros: ‘easy’ optimization &amp; theoretical guarantee</td>
<td>Cons: loose bound of $\text{err}_{0/1}$ for very negative $y$s</td>
</tr>
<tr>
<td>regularized logistic regression for classification</td>
<td>Minimize regularized $\text{err}_{\text{SCE}}$ by GD/SGD/...</td>
<td>Pros: ‘easy’ optimization &amp; regularization guard</td>
<td>Cons: loose bound of $\text{err}_{0/1}$ for very negative $y$s</td>
</tr>
</tbody>
</table>

- **regularized LogReg $\Rightarrow$ approximate SVM**
- **SVM $\Rightarrow$ approximate LogReg (?)**
We know that $\hat{\text{err}}_{\text{SVM}}(s, y)$ is an upper bound of $\text{err}_{0/1}(s, y)$. When is the upper bound tight? That is, when is $\hat{\text{err}}_{\text{SVM}}(s, y) = \text{err}_{0/1}(s, y)$?

1. $ys \geq 0$
2. $ys \leq 0$
3. $ys \geq 1$
4. $ys \leq 1$
We know that $\hat{\text{err}}_{\text{SVM}}(s, y)$ is an upper bound of $\text{err}_{0/1}(s, y)$. When is the upper bound tight? That is, when is $\hat{\text{err}}_{\text{SVM}}(s, y) = \text{err}_{0/1}(s, y)$?

1. $ys \geq 0$
2. $ys \leq 0$
3. $ys \geq 1$
4. $ys \leq 1$

Reference Answer: 3

By plotting the figure, we can easily see that $\hat{\text{err}}_{\text{SVM}}(s, y) = \text{err}_{0/1}(s, y)$ if and only if $ys \geq 1$. In that case, both error functions evaluate to 0.
Kernel Logistic Regression

SVM for Soft Binary Classification

Naïve Idea 1
1. run SVM and get $(b_{SVM}, w_{SVM})$
2. return $g(x) = \theta(w_{SVM}^T x + b_{SVM})$

- ‘direct’ use of similarity — works reasonably well
- no LogReg flavor

Naïve Idea 2
1. run SVM and get $(b_{SVM}, w_{SVM})$
2. run LogReg with $(b_{SVM}, w_{SVM})$ as $w_0$
3. return LogReg solution as $g(x)$

- not really ‘easier’ than original LogReg
- SVM flavor (kernel?) lost

want: flavors from both sides
A Possible Model: Two-Level Learning

\[ g(x) = \theta(A \cdot (w_{SVM}^T \Phi(x) + b_{SVM}) + B) \]

- **SVM flavor**: fix hyperplane direction by \( w_{SVM} \)—kernel applies
- **LogReg flavor**: fine-tune hyperplane to match maximum likelihood by scaling \((A)\) and shifting \((B)\)
  - often \( A > 0 \) if \( w_{SVM} \) reasonably good
  - often \( B \approx 0 \) if \( b_{SVM} \) reasonably good

**new LogReg Problem:**

\[
\min_{A,B} \frac{1}{N} \sum_{n=1}^{N} \log \left( 1 + \exp \left( -y_n \left( A \cdot (w_{SVM}^T \Phi(x_n) + b_{SVM}) + B \right) \right) \right)
\]

**two-level learning:**

LogReg on SVM-transformed data
Platt’s Model of Probabilistic SVM for Soft Binary Classification

1. run **SVM** on $\mathcal{D}$ to get $(b_{\text{SVM}}, w_{\text{SVM}})$ [or the equivalent $\alpha$], and transform $\mathcal{D}$ to $z'_n = w_{\text{SVM}}^T \Phi(x_n) + b_{\text{SVM}}$
   — actual model performs this step in a more complicated manner

2. run **LogReg** on $\{(z'_n, y_n)\}_{n=1}^N$ to get $(A, B)$
   — actual model adds some special regularization here

3. return $g(x) = \theta(A \cdot (w_{\text{SVM}}^T \Phi(x) + b_{\text{SVM}}) + B)$

- **soft binary classifier** not having the same boundary as **SVM classifier**
  — because of $B$

- how to solve **LogReg**: GD/SGD/or better
  — because only **two variables**

kernel SVM $\Rightarrow$ approx. LogReg in $\mathcal{Z}$-space

exact LogReg in $\mathcal{Z}$-space?
Recall that the score $w_{\text{SVM}}^T \Phi(x) + b_{\text{SVM}} = \sum_{\text{SV}} \alpha_n y_n K(x_n, x) + b_{\text{SVM}}$ for the kernel SVM. When coupling the kernel SVM with $(A, B)$ to form a probabilistic SVM, which of the following is the resulting $g(x)$?

1. $\theta \left( \sum_{\text{SV}} B\alpha_n y_n K(x_n, x) + b_{\text{SVM}} \right)$
2. $\theta \left( \sum_{\text{SV}} B\alpha_n y_n K(x_n, x) + Bb_{\text{SVM}} + A \right)$
3. $\theta \left( \sum_{\text{SV}} A\alpha_n y_n K(x_n, x) + b_{\text{SVM}} \right)$
4. $\theta \left( \sum_{\text{SV}} A\alpha_n y_n K(x_n, x) + Ab_{\text{SVM}} + B \right)$
Recall that the score \( w_{\text{SVM}}^T \Phi(x) + b_{\text{SVM}} = \sum_{SV} \alpha_n y_n K(x_n, x) + b_{\text{SVM}} \) for the kernel SVM. When coupling the kernel SVM with \((A, B)\) to form a probabilistic SVM, which of the following is the resulting \( g(x) \)?

1. \[ \theta \left( \sum_{SV} B\alpha_n y_n K(x_n, x) + b_{\text{SVM}} \right) \]
2. \[ \theta \left( \sum_{SV} B\alpha_n y_n K(x_n, x) + Bb_{\text{SVM}} + A \right) \]
3. \[ \theta \left( \sum_{SV} A\alpha_n y_n K(x_n, x) + b_{\text{SVM}} \right) \]
4. \[ \theta \left( \sum_{SV} A\alpha_n y_n K(x_n, x) + Ab_{\text{SVM}} + B \right) \]

Reference Answer: ④

We can simply plug the kernel formula of the score into \( g(x) \).
Key behind Kernel Trick

one key behind kernel trick: optimal \( w_\star = \sum_{n=1}^{N} \beta_n z_n \)

because \( w_\star^T z = \sum_{n=1}^{N} \beta_n z_n^T z = \sum_{n=1}^{N} \beta_n K(x_n, x) \)

\[
\begin{align*}
\text{SVM} & : w_{\text{SVM}} = \sum_{n=1}^{N} (\alpha_n y_n) z_n \\
\alpha_n & \text{ from dual solutions}
\end{align*}
\]

\[
\begin{align*}
\text{PLA} & : w_{\text{PLA}} = \sum_{n=1}^{N} (\alpha_n y_n) z_n \\
\alpha_n & \text{ by # mistake corrections}
\end{align*}
\]

\[
\begin{align*}
\text{LogReg by SGD} & : w_{\text{LOGREG}} = \sum_{n=1}^{N} (\alpha_n y_n) z_n \\
\alpha_n & \text{ by total SGD moves}
\end{align*}
\]

when can optimal \( w_\star \) be represented by \( z_n \)?
**Kernel Logistic Regression**

**Representer Theorem**

Claim: for any L2-regularized linear model

\[
\min_w \frac{\lambda}{N} w^T w + \frac{1}{N} \sum_{n=1}^{N} \text{err}(y_n, w^T z_n)
\]

Optimal \(w_* = \sum_{n=1}^{N} \beta_n z_n\).

- Let optimal \(w_* = w_\parallel + w_\perp\), where \(w_\parallel \in \text{span}(z_n)\) & \(w_\perp \perp \text{span}(z_n)\)
  - Want \(w_\perp = 0\)
- What if not? Consider \(w_\parallel\)
  - Of same err as \(w_*: \text{err}(y_n, w_*^T z_n) = \text{err}(y_n, (w_\parallel + w_\perp)^T z_n)\)
  - Of smaller regularizer as \(w_*:\)
    \[
    w_*^T w_* = w_\parallel^T w_\parallel + 2w_\parallel^T w_\perp + w_\perp^T w_\perp > w_\parallel^T w_\parallel
    \]
    - \(w_\parallel\) ‘more optimal’ than \(w_*\) (contradiction!)

Any L2-regularized linear model can be **kernelized!**
Kernel Logistic Regression

solving L2-regularized logistic regression

\[
\min_w \ \frac{\lambda}{N} w^T w + \frac{1}{N} \sum_{n=1}^{N} \log \left(1 + \exp \left(-y_n w^T z_n\right)\right)
\]

yields optimal solution \( w_\ast = \sum_{n=1}^{N} \beta_n z_n \)

with out loss of generality, can solve for optimal \( \beta \) instead of \( w \)

\[
\min_\beta \ \frac{\lambda}{N} \sum_{n=1}^{N} \sum_{m=1}^{N} \beta_n \beta_m K(x_n, x_m) + \frac{1}{N} \sum_{n=1}^{N} \log \left(1 + \exp \left(-y_n \sum_{m=1}^{N} \beta_m K(x_m, x_n)\right)\right)
\]

—how? GD/SGD/... for unconstrained optimization

kernel logistic regression:
use representer theorem for kernel trick
on L2-regularized logistic regression
Kernel Logistic Regression (KLR) : Another View

\[
\min_{\beta} \frac{\lambda}{N} \sum_{n=1}^{N} \sum_{m=1}^{N} \beta_n \beta_m K(x_n, x_m) + \frac{1}{N} \sum_{n=1}^{N} \log \left( 1 + \exp \left( -y_n \sum_{m=1}^{N} \beta_m K(x_m, x_n) \right) \right)
\]

- \[\sum_{m=1}^{N} \beta_m K(x_m, x_n)\]: inner product between variables \(\beta\) and transformed data \((K(x_1, x_n), K(x_2, x_n), \ldots, K(x_N, x_n))\)
- \[\sum_{n=1}^{N} \sum_{m=1}^{N} \beta_n \beta_m K(x_n, x_m)\]: a special regularizer \(\beta^T K \beta\)
- KLR = linear model of \(\beta\)
  with kernel as transform & kernel regularizer;
  = linear model of \(w\)
  with embedded-in-kernel transform & L2 regularizer

- similar for SVM

**warning**: unlike coefficients \(\alpha_n\) in SVM, coefficients \(\beta_n\) in KLR often non-zero!
When viewing KLR as linear model of $\beta$ with embedded-in-kernel transform & kernel regularizer, what is the dimension of the $\mathcal{Z}$ space that the linear model operates on?

1. $d$, the dimension of the original $\mathcal{X}$ space
2. $N$, the number of training examples
3. $\tilde{d}$, the dimension of some feature transform $\Phi(x)$ that is embedded within the kernel
4. $\lambda$, the regularization parameter
When viewing KLR as linear model of $\beta$ with embedded-in-kernel transform & kernel regularizer, what is the dimension of the $Z$ space that the linear model operates on?

1. $d$, the dimension of the original $X$ space
2. $N$, the number of training examples
3. $\tilde{d}$, the dimension of some feature transform $\Phi(x)$ that is embedded within the kernel
4. $\lambda$, the regularization parameter

Reference Answer: 2

For any $x$, the transformed data is $(K(x_1, x), K(x_2, x), \ldots, K(x_N, x))$, which is $N$-dimensional.
Summary

1 Embedding Numerous Features: Kernel Models

Lecture 5: Kernel Logistic Regression

- Soft-Margin SVM as Regularized Model
  L2-regularization with hinge error measure
- SVM versus Logistic Regression
  \( \approx \) L2-regularized logistic regression
- SVM for Soft Binary Classification
  common approach: two-level learning
- Kernel Logistic Regression
  representer theorem on L2-regularized LogReg

- next: kernel models for regression

2 Combining Predictive Features: Aggregation Models

3 Distilling Implicit Features: Extraction Models