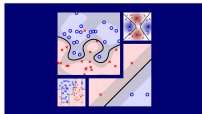


Machine Learning Techniques (機器學習技法)



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** ξ_n while penalizing them by C ; equivalent to **upper-bounding** α_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

Wrap-Up

Hard-Margin Primal

$$\begin{aligned} \min_{b, \mathbf{w}} \quad & \frac{1}{2} \mathbf{w}^T \mathbf{w} \\ \text{s.t.} \quad & y_n(\mathbf{w}^T \mathbf{z}_n + b) \geq 1 \end{aligned}$$

Soft-Margin Primal

$$\begin{aligned} \min_{b, \mathbf{w}, \xi} \quad & \frac{1}{2} \mathbf{w}^T \mathbf{w} + \textcolor{red}{C} \sum_{n=1}^N \xi_n \\ \text{s.t.} \quad & y_n(\mathbf{w}^T \mathbf{z}_n + b) \geq 1 - \xi_n, \xi_n \geq 0 \end{aligned}$$

Hard-Margin Dual

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \alpha^T Q \alpha - \mathbf{1}^T \alpha \\ \text{s.t.} \quad & \mathbf{y}^T \alpha = 0 \\ & 0 \leq \alpha_n \end{aligned}$$

Soft-Margin Dual

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \alpha^T Q \alpha - \mathbf{1}^T \alpha \\ \text{s.t.} \quad & \mathbf{y}^T \alpha = 0 \\ & 0 \leq \alpha_n \leq \textcolor{red}{C} \end{aligned}$$

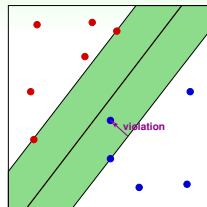
soft-margin preferred in practice;
linear: LIBLINEAR; non-linear: LIBSVM

Slack Variables ξ_n

- record '**margin violation**' by ξ_n
- penalize with **margin violation**

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

$$\text{s.t.} \quad y_n(\mathbf{w}^T \mathbf{z}_n + b) \geq 1 - \xi_n \text{ and } \xi_n \geq 0 \text{ for all } n$$



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

- (\mathbf{x}_n, y_n) violating margin: $\xi_n = 1 - y_n(\mathbf{w}^T \mathbf{z}_n + b)$
- (\mathbf{x}_n, y_n) not violating margin: $\xi_n = 0$

'unconstrained' form of soft-margin SVM:

$$\min_{b, \mathbf{w}} \quad \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{n=1}^N \max(1 - y_n(\mathbf{w}^T \mathbf{z}_n + b), 0)$$

Unconstrained Form

$$\min_{b, \mathbf{w}} \quad \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{n=1}^N \max(1 - y_n(\mathbf{w}^T \mathbf{z}_n + b), 0)$$

familiar? :-)

$$\min \quad \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum \widehat{\text{err}}$$

just L2 regularization

$$\min \quad \frac{\lambda}{N} \mathbf{w}^T \mathbf{w} + \frac{1}{N} \sum \text{err}$$

with shorter \mathbf{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

	minimize	constraint
regularization by constraint	E_{in}	$\mathbf{w}^T \mathbf{w} \leq C$
hard-margin SVM	$\mathbf{w}^T \mathbf{w}$	$E_{\text{in}} = 0$ [and more]
L2 regularization	$\frac{\lambda}{N} \mathbf{w}^T \mathbf{w} + E_{\text{in}}$	
soft-margin SVM	$\frac{1}{2} \mathbf{w}^T \mathbf{w} + C N \widehat{E}_{\text{in}}$	

large margin \iff fewer hyperplanes \iff L2 regularization of short \mathbf{w}

soft margin \iff special $\widehat{\text{err}}$

larger C or C \iff smaller λ \iff less regularization

viewing SVM as regularized model:

allows **extending/connecting** to other learning models

Fun Time

When viewing soft-margin SVM as regularized model, a larger C corresponds to

- 1 a larger λ , that is, stronger regularization
- 2 a smaller λ , that is, stronger regularization
- 3 a larger λ , that is, weaker regularization
- 4 a smaller λ , that is, weaker regularization

Fun Time

When viewing soft-margin SVM as regularized model, a larger C corresponds to

- ① a larger λ , that is, stronger regularization
- ② a smaller λ , that is, stronger regularization
- ③ a larger λ , that is, weaker regularization
- ④ a smaller λ , that is, weaker regularization

Reference Answer: ④

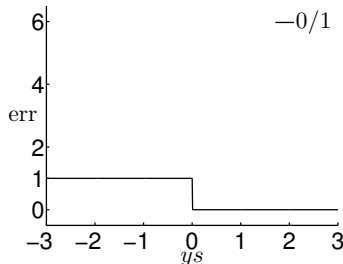
Comparing the formulations on page 4 of the slides, we see that C corresponds to $\frac{1}{2\lambda}$. So larger C corresponds to smaller λ , which surely means weaker regularization.

Algorithmic Error Measure of SVM

$$\min_{b, \mathbf{w}} \quad \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{n=1}^N \max(1 - y_n(\mathbf{w}^T \mathbf{z}_n + b), 0)$$

linear score $s = \mathbf{w}^T \mathbf{z}_n + b$

- $\text{err}_{0/1}(s, y) = \mathbb{I}[ys \leq 0]$
- $\widehat{\text{err}}_{\text{SVM}}(s, y) = \max(1 - ys, 0)$:
upper bound of $\text{err}_{0/1}$
—often called **hinge error measure**



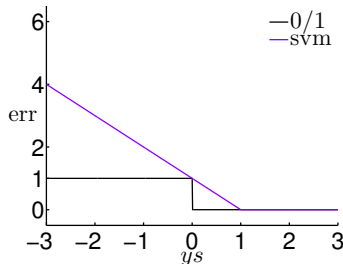
$\widehat{\text{err}}_{\text{SVM}}$: **algorithmic error measure**
by **convex upper bound** of $\text{err}_{0/1}$

Algorithmic Error Measure of SVM

$$\min_{b, \mathbf{w}} \quad \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{n=1}^N \max(1 - y_n(\mathbf{w}^T \mathbf{z}_n + b), 0)$$

linear score $s = \mathbf{w}^T \mathbf{z}_n + b$

- $\text{err}_{0/1}(s, y) = \mathbb{I}[ys \leq 0]$
- $\widehat{\text{err}}_{\text{SVM}}(s, y) = \max(1 - ys, 0)$:
upper bound of $\text{err}_{0/1}$
—often called **hinge error measure**

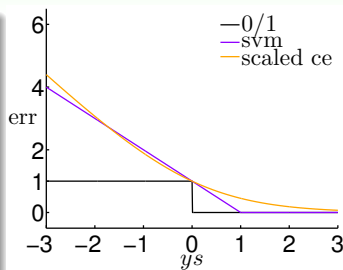


$\widehat{\text{err}}_{\text{SVM}}$: algorithmic error measure
by **convex upper bound** of $\text{err}_{0/1}$

Connection between SVM and Logistic Regression

linear score $s = \mathbf{w}^T \mathbf{z}_n + b$

- $\text{err}_{0/1}(s, y) = \mathbb{I}[ys \leq 0]$
- $\widehat{\text{err}}_{\text{SVM}}(s, y) = \max(1 - ys, 0)$:
upper bound of $\text{err}_{0/1}$
- $\text{err}_{\text{SCE}}(s, y) = \log_2(1 + \exp(-ys))$:
another upper bound of $\text{err}_{0/1}$ used in
logistic regression



$-\infty$	\longleftarrow	ys	\longrightarrow	$+\infty$
$\approx -ys$		$\widehat{\text{err}}_{\text{SVM}}(s, y)$		$= 0$
$\approx -ys$		$(\ln 2) \cdot \text{err}_{\text{SCE}}(s, y)$		≈ 0

SVM \approx L2-regularized **logistic regression**

Linear Models for Binary Classification

PLA

minimize $\text{err}_{0/1}$ specially

- pros: **efficient if lin. separable**
- cons: works only if lin. separable, otherwise needing **pocket**

soft-margin SVM

minimize regularized $\widehat{\text{err}}_{\text{SVM}}$ by QP

- pros: **'easy' optimization** & theoretical guarantee
- cons: loose bound of $\text{err}_{0/1}$ for very negative **ys**

regularized logistic regression for classification

minimize regularized err_{SCE} by GD/SGD/...

- pros: **'easy' optimization** & regularization guard
- cons: loose bound of $\text{err}_{0/1}$ for very negative **ys**

regularized LogReg \implies approximate SVM
SVM \implies approximate LogReg (?)

Fun Time

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

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

Fun Time

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

- 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 $\widehat{\text{err}}_{\text{SVM}}(s, y) = \text{err}_{0/1}(s, y)$ if and only if $ys \geq 1$. In that case, both error functions evaluate to 0.

SVM for Soft Binary Classification

Naïve Idea 1

- 1 run SVM and get $(b_{\text{SVM}}, \mathbf{w}_{\text{SVM}})$
- 2 return
$$g(\mathbf{x}) = \theta(\mathbf{w}_{\text{SVM}}^T \mathbf{x} + b_{\text{SVM}})$$

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

Naïve Idea 2

- 1 run SVM and get $(b_{\text{SVM}}, \mathbf{w}_{\text{SVM}})$
- 2 run LogReg with $(b_{\text{SVM}}, \mathbf{w}_{\text{SVM}})$ as \mathbf{w}_0
- 3 return LogReg solution as $g(\mathbf{x})$

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

want: flavors from both sides

A Possible Model: Two-Level Learning

$$g(\mathbf{x}) = \theta(A \cdot (\mathbf{w}_{\text{SVM}}^T \Phi(\mathbf{x}) + b_{\text{SVM}}) + B)$$

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

new LogReg Problem:

$$\min_{A, B} \frac{1}{N} \sum_{n=1}^N \log \left(1 + \exp \left(-y_n \left(A \cdot \underbrace{(\mathbf{w}_{\text{SVM}}^T \Phi(\mathbf{x}_n) + b_{\text{SVM}})}_{\Phi_{\text{SVM}}(\mathbf{x}_n)} + B \right) \right) \right)$$

two-level learning:

LogReg on **SVM-transformed** data

Probabilistic SVM

Platt's Model of Probabilistic SVM for Soft Binary Classification

- 1 run **SVM** on \mathcal{D} to get $(b_{\text{SVM}}, \mathbf{w}_{\text{SVM}})$ [or the equivalent α], and transform \mathcal{D} to $\mathbf{z}'_n = \mathbf{w}_{\text{SVM}}^T \Phi(\mathbf{x}_n) + b_{\text{SVM}}$
—actual model performs this step in a more complicated manner
 - 2 run **LogReg** on $\{(\mathbf{z}'_n, y_n)\}_{n=1}^N$ to get (A, B)
—actual model adds some special regularization here
 - 3 return $g(\mathbf{x}) = \theta(A \cdot (\mathbf{w}_{\text{SVM}}^T \Phi(\mathbf{x}) + b_{\text{SVM}}) + B)$
- **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 \implies approx. LogReg in \mathcal{Z} -space
exact LogReg in \mathcal{Z} -space?

Fun Time

Recall that the score $\mathbf{w}_{\text{SVM}}^T \Phi(\mathbf{x}) + b_{\text{SVM}} = \sum_{\text{SV}} \alpha_n y_n K(\mathbf{x}_n, \mathbf{x}) + b_{\text{SVM}}$ for the kernel SVM. When coupling the kernel SVM with (A, B) to form a probabilistic SVM, which of the following is the resulting $g(\mathbf{x})$?

- 1 $\theta \left(\sum_{\text{SV}} B \alpha_n y_n K(\mathbf{x}_n, \mathbf{x}) + b_{\text{SVM}} \right)$
- 2 $\theta \left(\sum_{\text{SV}} B \alpha_n y_n K(\mathbf{x}_n, \mathbf{x}) + B b_{\text{SVM}} + A \right)$
- 3 $\theta \left(\sum_{\text{SV}} A \alpha_n y_n K(\mathbf{x}_n, \mathbf{x}) + b_{\text{SVM}} \right)$
- 4 $\theta \left(\sum_{\text{SV}} A \alpha_n y_n K(\mathbf{x}_n, \mathbf{x}) + A b_{\text{SVM}} + B \right)$

Fun Time

Recall that the score $\mathbf{w}_{\text{SVM}}^T \Phi(\mathbf{x}) + b_{\text{SVM}} = \sum_{\text{SV}} \alpha_n y_n K(\mathbf{x}_n, \mathbf{x}) + b_{\text{SVM}}$ for the kernel SVM. When coupling the kernel SVM with (A, B) to form a probabilistic SVM, which of the following is the resulting $g(\mathbf{x})$?

- ① $\theta \left(\sum_{\text{SV}} B \alpha_n y_n K(\mathbf{x}_n, \mathbf{x}) + b_{\text{SVM}} \right)$
- ② $\theta \left(\sum_{\text{SV}} B \alpha_n y_n K(\mathbf{x}_n, \mathbf{x}) + B b_{\text{SVM}} + A \right)$
- ③ $\theta \left(\sum_{\text{SV}} A \alpha_n y_n K(\mathbf{x}_n, \mathbf{x}) + b_{\text{SVM}} \right)$
- ④ $\theta \left(\sum_{\text{SV}} A \alpha_n y_n K(\mathbf{x}_n, \mathbf{x}) + A b_{\text{SVM}} + B \right)$

Reference Answer: ④

We can simply plug the kernel formula of the score into $g(\mathbf{x})$.

Key behind Kernel Trick

one key behind kernel trick: optimal $\mathbf{w}_* = \sum_{n=1}^N \beta_n \mathbf{z}_n$

$$\text{because } \mathbf{w}_*^T \mathbf{z} = \sum_{n=1}^N \beta_n \mathbf{z}_n^T \mathbf{z} = \sum_{n=1}^N \beta_n K(\mathbf{x}_n, \mathbf{x})$$

SVM

$$\mathbf{w}_{\text{SVM}} = \sum_{n=1}^N (\alpha_n y_n) \mathbf{z}_n$$

α_n from **dual solutions**

PLA

$$\mathbf{w}_{\text{PLA}} = \sum_{n=1}^N (\alpha_n y_n) \mathbf{z}_n$$

α_n by **# mistake corrections**

LogReg by SGD

$$\mathbf{w}_{\text{LOGREG}} = \sum_{n=1}^N (\alpha_n y_n) \mathbf{z}_n$$

α_n by **total SGD moves**

when can **optimal \mathbf{w}_*** be **represented** by \mathbf{z}_n ?

Representer Theorem

claim: for any L2-regularized linear model

$$\min_{\mathbf{w}} \quad \frac{\lambda}{N} \mathbf{w}^T \mathbf{w} + \frac{1}{N} \sum_{n=1}^N \text{err}(y_n, \mathbf{w}^T \mathbf{z}_n)$$

optimal $\mathbf{w}_* = \sum_{n=1}^N \beta_n \mathbf{z}_n$.

- let optimal $\mathbf{w}_* = \mathbf{w}_{\parallel} + \mathbf{w}_{\perp}$, where $\mathbf{w}_{\parallel} \in \text{span}(\mathbf{z}_n)$ & $\mathbf{w}_{\perp} \perp \text{span}(\mathbf{z}_n)$
—want $\mathbf{w}_{\perp} = \mathbf{0}$
 - what if **not**? Consider \mathbf{w}_{\parallel}
 - of same err as \mathbf{w}_* : $\text{err}(y_n, \mathbf{w}_*^T \mathbf{z}_n) = \text{err}(y_n, (\mathbf{w}_{\parallel} + \mathbf{w}_{\perp})^T \mathbf{z}_n)$
 - of smaller regularizer as \mathbf{w}_* :

$$\mathbf{w}_*^T \mathbf{w}_* = \mathbf{w}_{\parallel}^T \mathbf{w}_{\parallel} + 2\mathbf{w}_{\parallel}^T \mathbf{w}_{\perp} + \mathbf{w}_{\perp}^T \mathbf{w}_{\perp} > \mathbf{w}_{\parallel}^T \mathbf{w}_{\parallel}$$
- \mathbf{w}_{\parallel} ‘**more optimal**’ than \mathbf{w}_* (**contradiction!**)

any L2-regularized linear model
can be **kernelized!**

Kernel Logistic Regression

solving L2-regularized logistic regression

$$\min_{\mathbf{w}} \quad \frac{\lambda}{N} \mathbf{w}^T \mathbf{w} + \frac{1}{N} \sum_{n=1}^N \log \left(1 + \exp \left(-y_n \mathbf{w}^T \mathbf{z}_n \right) \right)$$

yields optimal solution $\mathbf{w}_* = \sum_{n=1}^N \beta_n \mathbf{z}_n$

with out loss of generality, can solve for optimal β instead of \mathbf{w}

$$\min_{\beta} \quad \frac{\lambda}{N} \sum_{n=1}^N \sum_{m=1}^N \beta_n \beta_m K(\mathbf{x}_n, \mathbf{x}_m) + \frac{1}{N} \sum_{n=1}^N \log \left(1 + \exp \left(-y_n \sum_{m=1}^N \beta_m K(\mathbf{x}_m, \mathbf{x}_n) \right) \right)$$

—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

$$\min_{\beta} \frac{\lambda}{N} \sum_{n=1}^N \sum_{m=1}^N \beta_n \beta_m K(\mathbf{x}_n, \mathbf{x}_m) + \frac{1}{N} \sum_{n=1}^N \log \left(1 + \exp \left(-y_n \sum_{m=1}^N \beta_m K(\mathbf{x}_m, \mathbf{x}_n) \right) \right)$$

- $\sum_{m=1}^N \beta_m K(\mathbf{x}_m, \mathbf{x}_n)$: inner product between variables β and transformed data $(K(\mathbf{x}_1, \mathbf{x}_n), K(\mathbf{x}_2, \mathbf{x}_n), \dots, K(\mathbf{x}_N, \mathbf{x}_n))$
- $\sum_{n=1}^N \sum_{m=1}^N \beta_n \beta_m K(\mathbf{x}_n, \mathbf{x}_m)$: a special regularizer $\beta^T K \beta$
- KLR = linear model of β
 with kernel as transform & kernel regularizer;
 = linear model of \mathbf{w}
 with embedded-in-kernel transform & L2 regularizer
- similar for SVM

warning: unlike coefficients α_n in SVM,
 coefficients β_n in KLR often non-zero!

Fun Time

When viewing KLR as **linear model of β** 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 $\Phi(\mathbf{x})$ that is embedded within the kernel
- 4 λ , the regularization parameter

Fun Time

When viewing KLR as **linear model of β** 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 $\Phi(\mathbf{x})$ that is embedded within the kernel
- 4 λ , the regularization parameter

Reference Answer: (2)

For any \mathbf{x} , the transformed data is $(K(\mathbf{x}_1, \mathbf{x}), K(\mathbf{x}_2, \mathbf{x}), \dots, K(\mathbf{x}_N, \mathbf{x}))$, 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