

Kernel Logistic Regression

这篇笔记开始介绍SVM的一些实际应用，本篇是对于logistic regression的改进

一、SVM as Regularization Model

Recap:

Wrap-Up			
Hard-Margin Primal		Soft-Margin Primal	
$\min_{b, \mathbf{w}} \quad \frac{1}{2} \mathbf{w}^T \mathbf{w}$		$\min_{b, \mathbf{w}, \xi} \quad \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{n=1}^N \xi_n$	
s.t. $y_n(\mathbf{w}^T \mathbf{z}_n + b) \geq 1$		s.t. $y_n(\mathbf{w}^T \mathbf{z}_n + b) \geq 1 - \xi_n, \xi_n \geq 0$	
Hard-Margin Dual		Soft-Margin Dual	
$\min_{\alpha} \quad \frac{1}{2} \alpha^T Q \alpha - \mathbf{1}^T \alpha$		$\min_{\alpha} \quad \frac{1}{2} \alpha^T Q \alpha - \mathbf{1}^T \alpha$	
s.t. $\mathbf{y}^T \alpha = 0$ $0 \leq \alpha_n$		s.t. $\mathbf{y}^T \alpha = 0$ $0 \leq \alpha_n \leq C$	

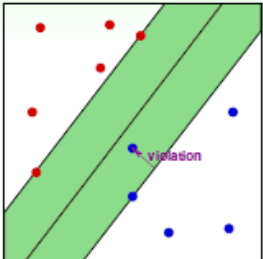
对于松弛变量 ξ_n 的解释

Slack Variables ξ_n

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

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

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

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

我们就得到了 ξ_n 的公式：

$$\xi_n = \max(1 - y_n(w^T)z_n + b, 0) \quad (1)$$

由此观之，SVM项可以看作是regularization项，这就得到了SVM Loss

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

我们总结一下各种regularization:

其中L2 regularization是对regularization constraint的一种改进:

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	$\frac{1}{2} \mathbf{w}^T \mathbf{w} + C \widehat{E}_{\text{in}}$	

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 $\lambda \iff$ less regularization

viewing SVM as regularized model:

allows extending/connecting to other learning models

C 越大，正则化的程度就越小!

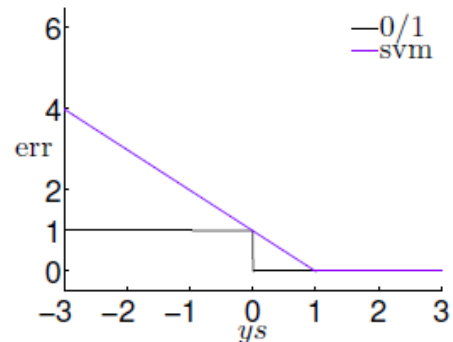
二、SVM and Logistic Regression

对比0/1 loss 和 SVM loss:

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

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

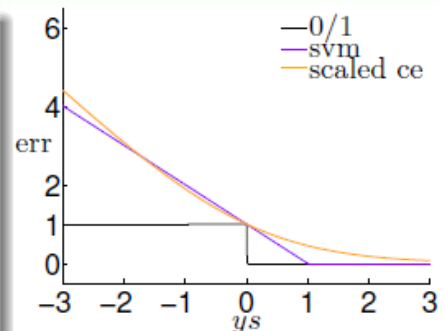
注意一个结论：如果loss function 是0/1-loss的 upper bound，那么就可以优化该loss-function来进行操作

利用SVM-Loss的regression叫做：hinge regression

引入logistic regression里面的loss function

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



$-\infty$	\leftarrow	ys	\rightarrow	$+\infty$
$\approx -ys$		$\widehat{\text{err}}_{\text{svm}}(s, y)$		$= 0$
$\approx -ys$		$(\ln 2) \cdot \text{err}_{\text{sce}}(s, y)$		≈ 0

对比三者，发现SVM-Loss 和 L2-regularized logistic regression类似！

Linear Models for Binary Classification

PLA	soft-margin SVM	regularized logistic regression for classification
minimize $\text{err}_{0/1}$ specially <ul style="list-style-type: none"> pros: efficient if lin. separable cons: works only if lin. separable, otherwise needing pocket 	minimize regularized $\widehat{\text{err}}_{\text{SVM}}$ by QP <ul style="list-style-type: none"> pros: 'easy' optimization & theoretical guarantee cons: loose bound of $\text{err}_{0/1}$ for very negative <i>ys</i> 	minimize regularized err_{SCE} by GD/SGD/... <ul style="list-style-type: none"> pros: 'easy' optimization & regularization guard cons: loose bound of $\text{err}_{0/1}$ for very negative <i>ys</i>

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

对比一下, $y_s \geq 1$ 的时候 hinge 和 0/1 一样

三、SVM 用作 binary classification

1. naive idea: 直接拿 SVM-loss 来代替 logistic regression 或者 把 SVM 的结果作为 gradient descent 的初始解

Naïve Idea 1	Naïve Idea 2
<ol style="list-style-type: none"> run SVM and get $(b_{\text{SVM}}, \mathbf{w}_{\text{SVM}})$ return $g(\mathbf{x}) = \theta(\mathbf{w}_{\text{SVM}}^T \mathbf{x} + b_{\text{SVM}})$ <ul style="list-style-type: none"> 'direct' use of similarity — works reasonably well no LogReg flavor 	<ol style="list-style-type: none"> run SVM and get $(b_{\text{SVM}}, \mathbf{w}_{\text{SVM}})$ run LogReg with $(b_{\text{SVM}}, \mathbf{w}_{\text{SVM}})$ as \mathbf{w}_0 return LogReg solution as $g(\mathbf{x})$ <ul style="list-style-type: none"> not really 'easier' than original LogReg SVM flavor (kernel?) lost

want: flavors from both sides

2. 融合两个模型进行一步操作:

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

我们就有一个新的SVM模型了

3. 我们总结一下流程: Platt's Model

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 \Rightarrow approx. LogReg in \mathcal{Z} -space
exact LogReg in \mathcal{Z} -space?

做的比较好的时候, 一般 $A > 0, B \approx 0$

4. 存在的问题:

没有办法直接在 \mathcal{Z} -space 里面找到自己想要的解!

四、Kernel Logistic Regression

1. Recap:

我们可以使用kernel trick的一个原因是因为W可以表示成关于Z的一个线性组合，进而我们算出来的score就是关于Z的一个内积。

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 ?

Key: W是Z的线性组合！

模型的系数是模型的本质差异

2. Representer Theorem: 表示定理

claim: for any L2-regularized linear model

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

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

- 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!**

L2正则化的模型，我们最优的结果W一定都是关于Z的一个线性组合

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}

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

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

kernel logistic regression:

use **representer theorem** for kernel trick
on L2-regularized logistic regression

我们利用结论，把关于 w 的最佳化问题转化成了关于 β 的最佳化问题。我们有如下的结论：

Kernel Logistic Regression

Kernel 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 $\beta^T \mathbf{K} \beta$
- 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!