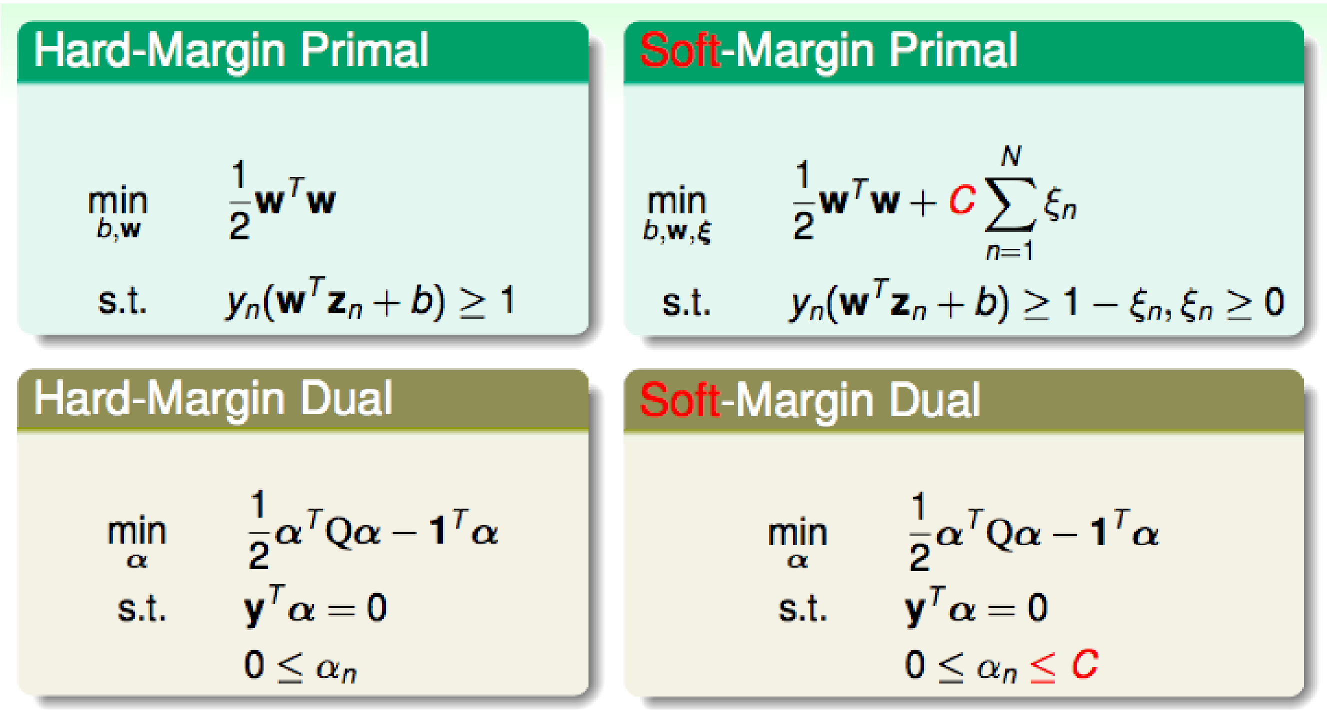
Lecture 5：Kernel Logistic Regression

> 课件链接：[Hsuan-Tien Lin - kernel logistic regression]
>
> **Kernel Logistic Regression(核Logistic回归)**
>
> - Soft-Margin SVM as Regularized Model: 作为正则化模型的软间隔SVM
> - SVM versus Logistic Regression: SVM与Logistic回归
> - SVM for Soft Binary Classification: 软二元分类的SVM
> - Kernel Logistic Regression: 核Logistic回归

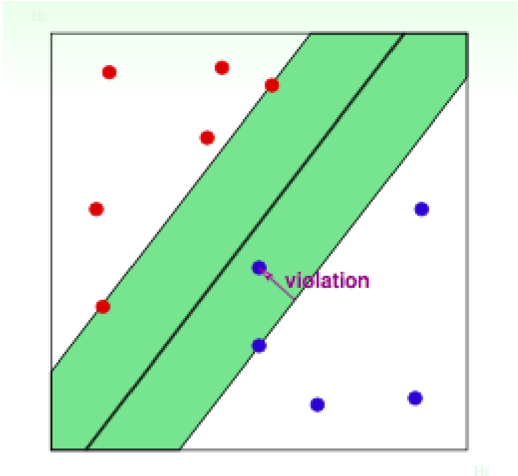
1. Soft-Margin SVM as Regularized Model: 作为正则化模型的软间隔SVM

在上一章中我们提到，设计Soft-Margin SVM的一个原因是**希望避免过拟合**——因为Hard-Margin会坚持将样本全部正确分类而不犯任何错误，这可能会使模型容易受到杂讯的影响而出现过拟合。因此，**Soft-Margin SVM具有更优秀的正则化效果**。本节将通过数学推导，**把Soft-Margin Primal问题转化为L2-Regularized的无约束形式。**

首先，用一张图回顾Hard-Margin SVM Primal & Dual与Soft-Margin SVM Primal & Dual：



对于Soft-Margin Primal，引入松弛变量\xi_n记录每个样本点的"**margin violation**"，即每个样本点对于margin的违反程度(如下图所示)：



对于某样本点(\mathbf{z}_n, y_n)：

- 如果没有违反margin，则\xi_n = 0；
- 如果违反了margin，则\xi_n = 1-y_n(\mathbf{w}^T\mathbf{z}_n+b)。

综上，可以将其\xi_n合写为：

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

因此，我们可以将Soft-Margin SVM Primal写成无约束条件的形式：

从上式可看出，**Soft-Margin SVM Primal实质是L2-正则化**，因为L2-正则化的一般形式为：

这里SVM的err是：

那么，为何一开始不直接介绍SVM的这种无约束形式？因为：

- 该最优化问题不是QP问题，很难直接使用kernel trick；
- 损失函数不可微分，不好解。

最后，我们将SVM与正则化模型进行比较：

	minimize	constraint
regularization by constraint	E_{in}	$\mathbf{w}^T\mathbf{w} \le C$
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	$\frac{1}{2}\mathbf{w}^T\mathbf{w} + CN\widehat{E}_{\text{in}}$	

Soft-Margin SVM中的C越小，正则化力度越大，相当于L2正则化模型中更大的\lambda。

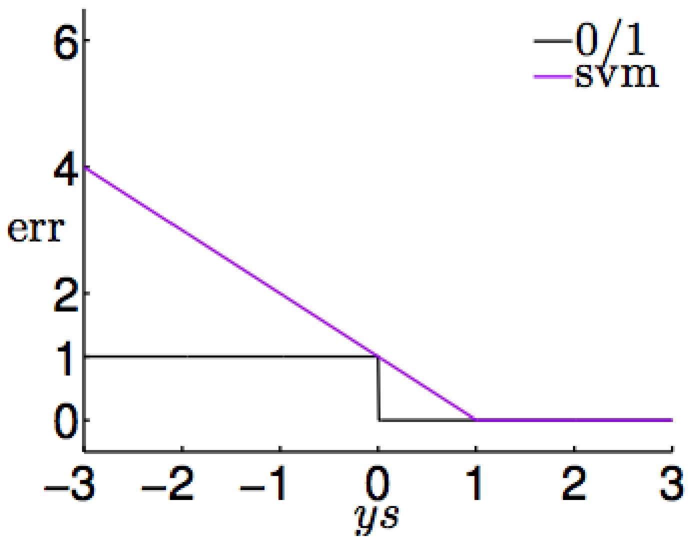
2. SVM versus Logistic Regression: SVM与Logistic回归

线性模型(linear model)的**共性**是：都要算一个"**分数(linear score)**"，然后根据该分数进行进一步的简单运算与判断：

基于此分数，我们有0-1误差：

现在，对于Soft-Margin SVM Primal，我们又有了新的误差函数：

该误差函数是0-1误差函数的**凸上界(convex upper bound)**，被称为**合页误差函数(hinge loss)**：

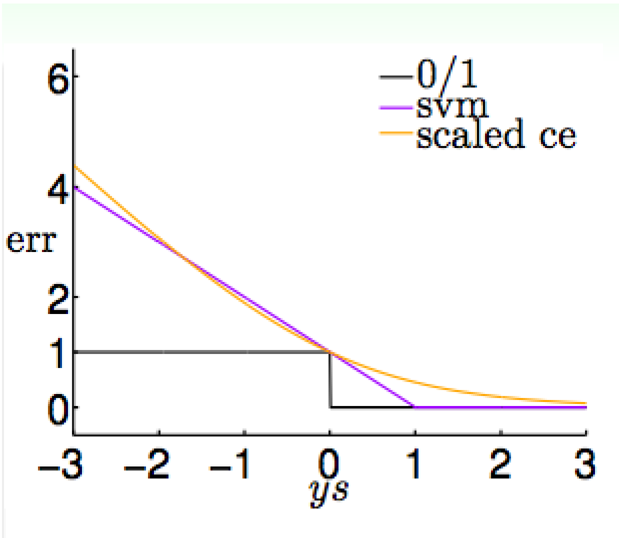


因此，把hinge loss"做好"(限制的很小)，等于间接将0-1误差"做好"。

回忆，Logistic Regression的误差函数是**交叉熵(cross entropy)**：

将其进行**1/ln2**的放缩，得到**Scaled cross entropy**：

该误差函数也是0-1误差函数的凸上界：

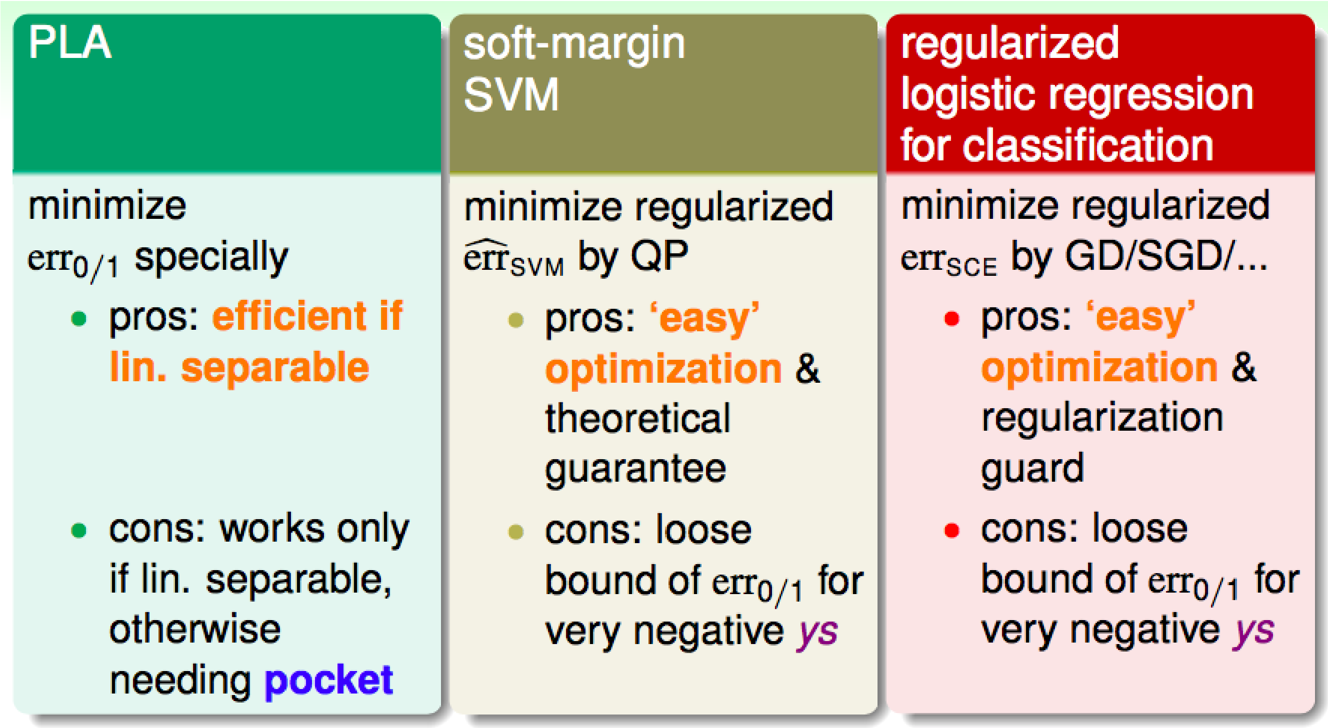


交叉熵误差函数与Hinge误差函数比较相近，因为：

- 当ys趋近于负无穷时，前者约等于-ys，后者约等于-ys；
- 当ys趋近于正无穷时，前者约等于0，后者等于0；

因此，**Soft-Margin SVM"近似"等于L2正则化的LR。**

小结：二元分类问题的线性模型

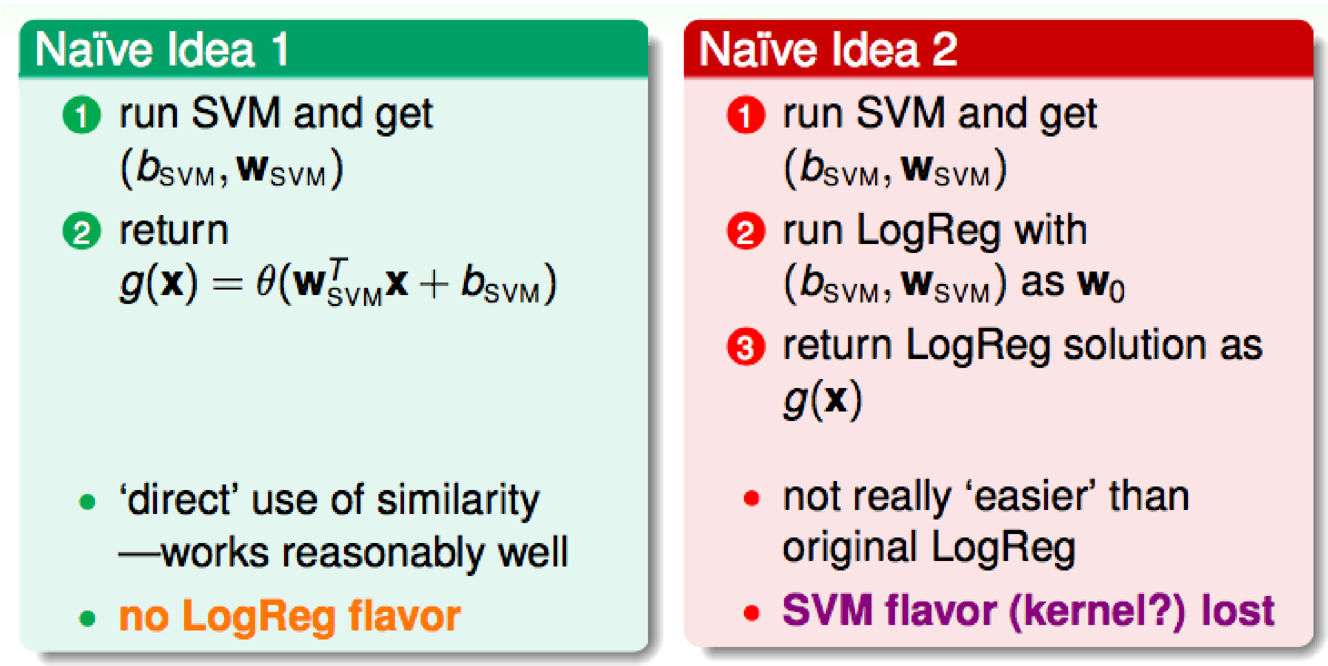


解释：关于Soft-Margin SVM与L2-LogReg的缺点(cons)——loose bound for very negative ys：因为在ys"很负"的时候，0-1误差函数的值为1，而交叉熵误差与Hinge Loss的值均为一个非常大的正数。因此，使用上述两种误差函数对0-1误差函数进行替代后，E_{in}的最小值，实际上是E_{in}^{0/1}的一个很松的上界：

- 如果E_{in}最够小，E_{in}^{0/1}肯定也足够小；
- 如果E_{in}比较大，无法说明E_{in}^{0/1}的信息。

3. SVM for Soft Binary Classification: 软二元分类的SVM

如果我们希望SVM的最终输出不仅是样本的预测分类结果，而且是样本的预测分类结果的**概率**，就如同LogReg的输出一样(回忆，我们将其称为**Soft Classification**)，我们大致可以有两种做法：



Idea 1：直接利用Soft-Margin SVM与L2-LogReg的相似性

1. 解SVM问题，得到(\mathbf{w}_{SVM}, b_{SVM})；
2. 计算分数并送给sigmoid函数，得到概率值(与LogReg的处理方式一样)：g(\mathbf{x})=\theta(\mathbf{w}_{SVM}^T \Phi(\mathbf{x}) + b_{SVM})。

实务上，这种做法的表现还不错，但是丧失了LogReg的特点，例如**最大似然估计。**

Idea 2：将Soft-Margin SVM的结果作为LogReg的初始化值

1. 解SVM问题，得到(\mathbf{w}_{SVM}, b_{SVM})；
2. 将上述解作为LogReg的初始化值\mathbf{w}_0；
3. 回传LogReg的最终解作为g(\mathbf{x})。

缺点是，丧失了SVM的特点，例如Kernel Trick无法在LogReg步骤里使用。

因此，我们考虑这样一个**Two-Level Learning**：

上述模型既保留了SVM的特点，也保留的LogReg的特点：

- SVM flavor：分离超平面的法向量(\mathbf{w})被SVM确定——\mathbf{w}_{SVM}(A只不过是放缩动作，影响长度但不影响方向)，这样我们就可以使用kernel trick了；
- LogReg flavor：在第二层学习中，通过A的放缩动作与B的平移动作微调SVM得到的分离超平面，使之符合最大似然估计的结果：
  - 往往A > 0
  - 往往B \approx 0

因此，新的LogReg问题为：

new LogReg Problem:

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

该模型被称为**Platt's Model**，或者**Probabilistic SVM**，算法如下：

1. 在训练集D上跑SVM，得到(b_{SVM}, \mathbf{w}_{SVM})，或者等价的\alpha_n；然后将训练集D上的数据进行转换：\mathbf{z}_n' = \mathbf{w}^T_{SVM} \Phi(\mathbf{x}_n) + b_{SVM}——这里转换得到的数据是1维的；
2. 在数据集\{(\mathbf{z}_n', y_n)\}_{n=1}^N上跑LogReg，得到(A,B)；
3. 回传g(\mathbf{x}) = \theta \Bigg(A· (\mathbf{w}^T_{SVM} \Phi(\mathbf{x})+b_{SVM})+ B\Bigg)。

然而，Probabilistic SVM并不是Kernel LogReg。因为Probabilistic SVM并没有真正在Z空间中解LogReg问题，而是利用Soft-Margin SVM与LogReg的相似性，在Z空间中解Soft-Margin SVM问题，然后利用LogReg进行微调——如果我们就是要解Z空间中的LogReg问题呢？下一讲，真正的Kernel LogReg。

4. Kernel Logistic Regression: 核Logistic回归

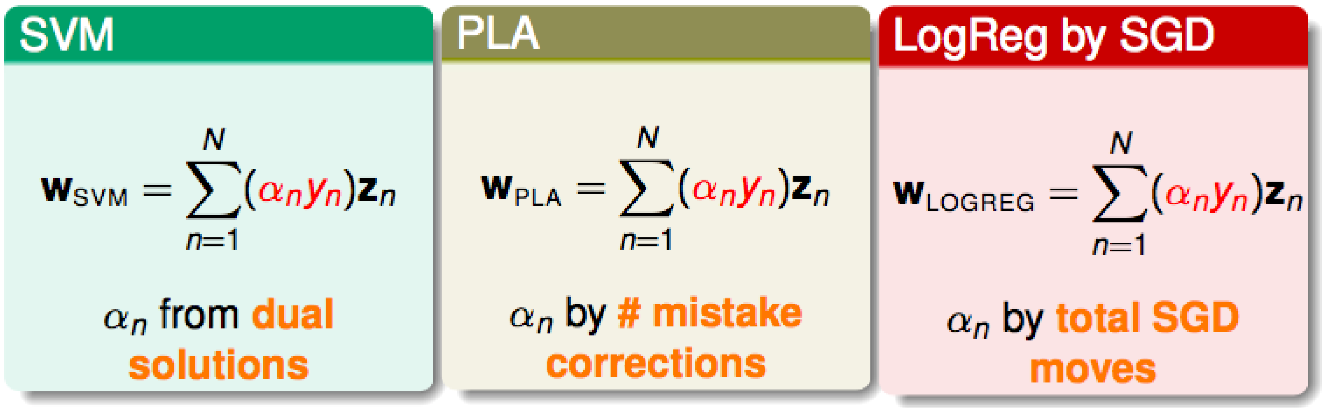
Kernel Trick的实质：将Z空间的内积转换成在X空间内可以轻易计算的函数。**Kernel Trick之所以会起作用**，是因为：

1. linear model，需要算optimal \mathbf{w}_*和\mathbf{z}的内积；
2. optimal \mathbf{w}_*可以用\mathbf{z}_n线性表示：

这样：

因此，能够使用kernel trick的关键是：**最佳的w是z的线性组合。**

SVM \ PLA \ LogReg by SGD，他们都有这样的性质：



Representer Theorem：对于任何的**L2-正则化线性模型**：

最佳的\mathbf{w}_* = \sum_{n=1}^N \beta_n \mathbf{z}_n。

证明：

- 我们将最佳的w分拆成两项之和——一项是在Span\ \{\mathbf{z}_n\}中向量，记做\mathbf{w}_{||}；另一项是正交于Span\ \{\mathbf{z}_n\}的向量，记做\mathbf{w}_{\perp}。故有：
- 假设\mathbf{w}_{\perp}不为零向量，考虑向量\mathbf{w}_{||}
  - err(y_n, \mathbf{w}_*^T\mathbf{z}_n) = err(y_n, (\mathbf{w}_{||}+\mathbf{w}_{\perp})^T\mathbf{z}_n) = err(y_n, \mathbf{w}_{||}^T \mathbf{z}_n)
  - 但是\mathbf{w}_*^T\mathbf{w}_* > \mathbf{w}_{||}^T\mathbf{w}_{||}
  - 因此\mathbf{w}_{||}比\mathbf{w}_*更优，产生矛盾，证毕。

综上，任何L2-正则化的线性模型，都可以使用kernel trick，

Kernel Logistic Regression

由Representer Theorem知，Z空间中LogReg的解可以表示为：

将其代入LogReg的损失函数中即可：

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

这是一个无约束的最优化问题，可以使用GD/SGD等方法很容易地求解。因此，KLR可以看做是：**use representer theorem for kernel trick on L2-regularized logistic regression**。

另一种视角：Another View

对于：

我们可以将其看做是向量\mathbf{\beta}和转换后的数据：

的内积。

对于：

可以看做是一个特殊的正则项：

因此，KLR可以看做是\mathbf{\beta}的线性模型，with：

- kernel as transform；
- kernel regularizer。

注意：\beta_n往往是non-zero，不像SVM中的\alpha_n是sparse的。

5. Summary

- 通过对\xi_n含义的重新梳理，我们得到了Soft-Margin SVM Primal的无约束条件形式——L2正则化，误差函数是Hinge Loss；C越小，正则化力度越大；
- Hinge Loss与Cross Entropy十分相近，因此Soft-Margin SVM"约等于"L2-LogReg；
- Idea 1与Idea 2使得SVM能够输出概率，即能够进行Soft Classification；但更好的方法是使用两层学习，即Platt's Model——使用Soft-Margin SVM先跑，再用LogReg微调分离超平面；
- Representer Theorem，表示理论，任何L2正则化线性模型的最佳w都可以被样本点z线性表示；
- KLR的两种观点：linear model of \mathbf{w} & linear model of \mathbf{\beta}。