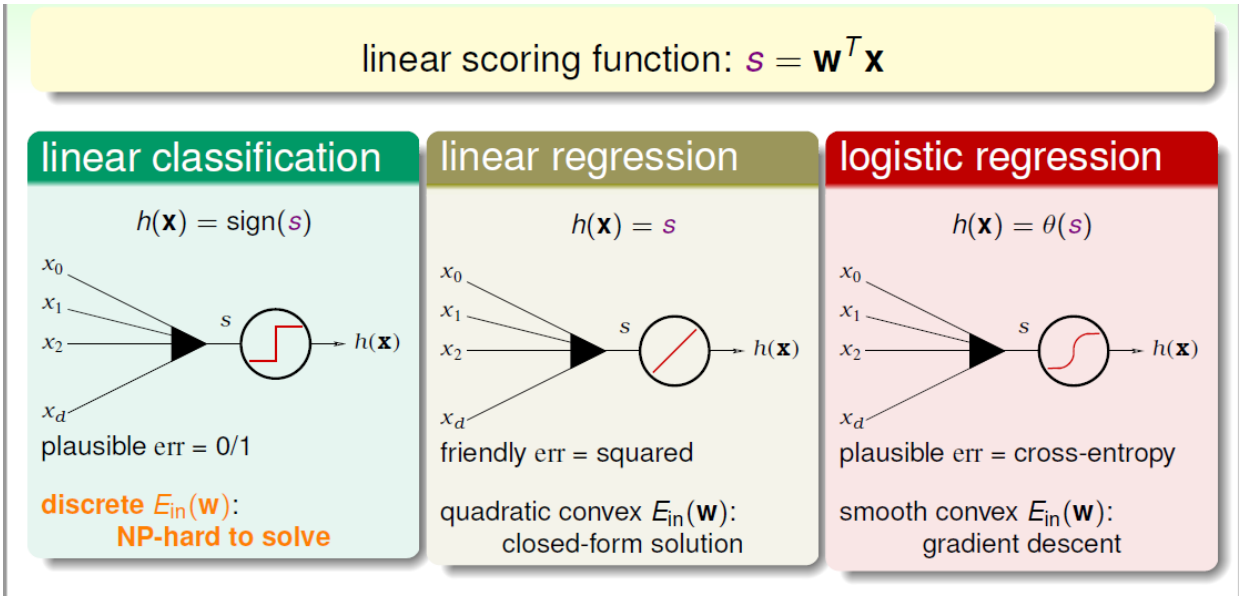
Lecture 11: Linear Models for Classification

1 .Linear Models for Binary Classification



三种线性模型:

- linear classification。线性分类模型的hypothesis为$h()x = sign(x)$,取值范围为{1,+1}两个值，它的err是0/1的，所以对应的是离散的，并不好解，这是个NP_hard问题。
- linear regression。线性回归模型的hypothesis为$h(x) = s$，取值范围为整个实数空间，它的err是squared的，所以对应的是开口向上的二次曲线，其解是closed-form的，直接用线性最小二乘法求解即可。
- logistic regression。逻辑回归模型的hypothesis为$h(x) = \theta(s)$，取值范围(-1,1)之间，它的err是cross-entropy的，所有对应的是平滑的凸函数，可以使用梯度下降算法求最小值。

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

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

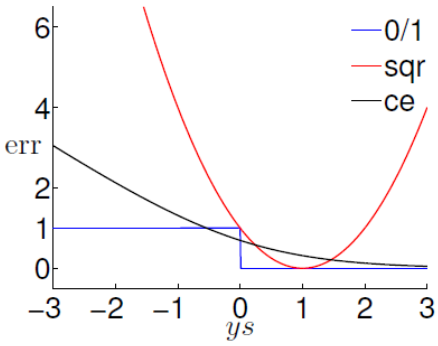
linear classification	linear regression	logistic regression
$h(\mathbf{x}) = \text{sign}(s)$	$h(\mathbf{x}) = s$	$h(\mathbf{x}) = \theta(s)$
$\text{err}(h, \mathbf{x}, y) = [\![h(\mathbf{x}) \neq y]\!]$	$\text{err}(h, \mathbf{x}, y) = (h(\mathbf{x}) - y)^2$	$\text{err}(h, \mathbf{x}, y) = -\ln h(y\mathbf{x})$
$\text{err}_{0/1}(s, y) = [\![\text{sign}(s) \neq y]\!] = [\![\text{sign}(ys) \neq 1]\!]$	$\text{err}_{\text{SQR}}(s, y) = (s - y)^2 = (ys - 1)^2$	$\text{err}_{\text{CE}}(s, y) = \ln(1 + \exp(-ys))$

(ys): classification correctness score

三种模型都引入了ys变量，ys就是指分类的正确率得分，其值越大越好，得分越高。

Visualizing Error Functions

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



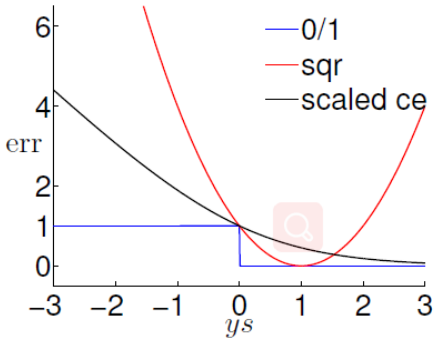
- 0/1: 1 iff $ys \leq 0$
- sqr: large if $ys \ll 1$
  but over-charge $ys \gg 1$
  small $\text{err}_{\text{SQR}} \rightarrow$ small $\text{err}_{0/1}$
- ce: monotonic of ys
  small $\text{err}_{\text{CE}} \leftrightarrow$ small $\text{err}_{0/1}$
- scaled ce: a proper upper bound of 0/1
  small $\text{err}_{\text{SCE}} \leftrightarrow$ small $\text{err}_{0/1}$

upper bound:
useful for designing algorithmic error $\widehat{\text{err}}$

如上图所示，ys是横坐标轴,$err_{0/1}$是呈阶梯状的，在ys>0时， 恒取最小值0。err_{SQR}呈抛物线形式，在$ys = 1$时，取得最小值，且在ys=1左右很小区域内，$err_{0/1}$和err_{SQR}近似。err_{CE}是呈指数下降的单调函数，ys越大，其值越小。同样在ys=1左右很小区域内，err_{CE}和$err_{0/1}$近似。但是我们发现并不是始终在$err_{0/1}$之上，所以为了计算讨论方便，我们把做幅值上的调整，引入$err_{SCE} = log_2(1 + exp(-ys)) = \frac{1}{ln2}\, err_{CE}$，这样能保证$err_{SCE}$始终在$err_{0/1}$上面，如下图所示：

Visualizing Error Functions

$$
\begin{aligned}
0/1 \quad \mathrm{err}_{0/1}(s,y) &= [\![\mathrm{sign}(ys) \neq 1]\!] \\
\mathrm{sqr} \quad \mathrm{err}_{\mathrm{SQR}}(s,y) &= (ys-1)^2 \\
\mathrm{ce} \quad \mathrm{err}_{\mathrm{CE}}(s,y) &= \ln(1+\exp(-ys)) \\
\text{scaled ce} \quad \mathrm{err}_{\mathrm{SCE}}(s,y) &= \log_2(1+\exp(-ys))
\end{aligned}
$$



- 0/1: 1 iff $ys \leq 0$
- sqr: large if $ys \ll 1$
  but over-charge $ys \gg 1$
  small $\mathrm{err}_{\mathrm{SQR}} \rightarrow$ small $\mathrm{err}_{0/1}$
- ce: monotonic of ys
  small $\mathrm{err}_{\mathrm{CE}} \leftrightarrow$ small $\mathrm{err}_{0/1}$
- scaled ce: a proper upper bound of 0/1
  small $\mathrm{err}_{\mathrm{SCE}} \leftrightarrow$ small $\mathrm{err}_{0/1}$

upper bound:
useful for designing algorithmic error $\widehat{\mathrm{err}}$

Theoretical Implication of Upper Bound

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

$$
\begin{aligned}
\mathrm{err}_{0/1}(s,y) &\leq \mathrm{err}_{\mathrm{SCE}}(s,y) = \frac{1}{\ln 2}\mathrm{err}_{\mathrm{CE}}(s,y). \\
\Longrightarrow \quad E_{\mathrm{in}}^{0/1}(\mathbf{w}) &\leq E_{\mathrm{in}}^{\mathrm{SCE}}(\mathbf{w}) = \frac{1}{\ln 2}E_{\mathrm{in}}^{\mathrm{CE}}(\mathbf{w}) \\
E_{\mathrm{out}}^{0/1}(\mathbf{w}) &\leq E_{\mathrm{out}}^{\mathrm{SCE}}(\mathbf{w}) = \frac{1}{\ln 2}E_{\mathrm{out}}^{\mathrm{CE}}(\mathbf{w})
\end{aligned}
$$

VC on 0/1:

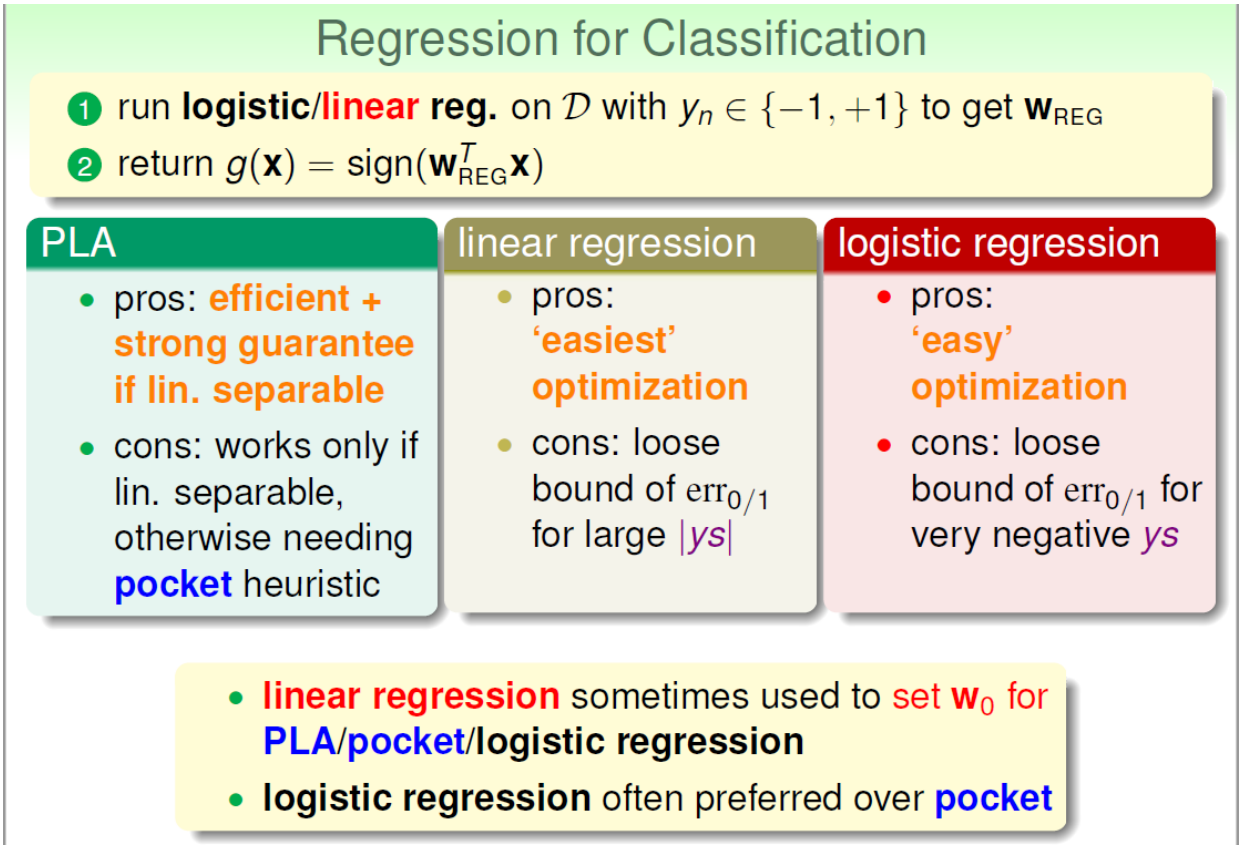
$$
\begin{aligned}
E_{\mathrm{out}}^{0/1}(\mathbf{w}) &\leq E_{\mathrm{in}}^{0/1}(\mathbf{w}) + \Omega^{0/1} \\
&\leq \frac{1}{\ln 2}E_{\mathrm{in}}^{\mathrm{CE}}(\mathbf{w}) + \Omega^{0/1}
\end{aligned}
$$

VC-Reg on CE :

$$
\begin{aligned}
E_{\mathrm{out}}^{0/1}(\mathbf{w}) &\leq \frac{1}{\ln 2}E_{\mathrm{out}}^{\mathrm{CE}}(\mathbf{w}) \\
&\leq \frac{1}{\ln 2}E_{\mathrm{in}}^{\mathrm{CE}}(\mathbf{w}) + \frac{1}{\ln 2}\Omega^{\mathrm{CE}}
\end{aligned}
$$

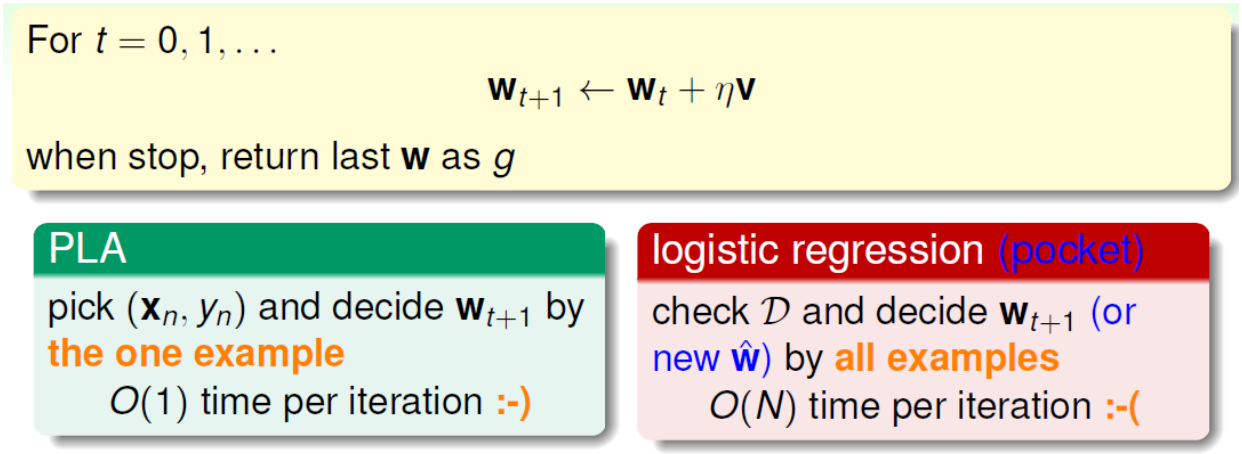
small $E_{\mathrm{in}}^{\mathrm{CE}}(\mathbf{w}) \Longrightarrow$ small $E_{\mathrm{out}}^{0/1}(\mathbf{w})$:
logistic/linear reg. for linear classification

通过上面的分析，我们看到$err_{0/1}$是被限定在一个上界中。这个上界是由logistic regression模型的error function决定的。而linear regression其实也是linear classification的一个upper bound，只是随着ys偏离1的位置越来越远，linear regression的error function偏差越来越大。综上所述，linear regression和logistic regression都可以用来解决linear classification的问题。



通常，我们使用linear regression来获得初始化的W_0，再用logistic regression模型进行最优化解。

2. Stochastic Gradient Descent



PLA算法和logistic regression算法，都是用到了迭代操作。PLA每次迭代只会更新一个点，它每次迭代的时间复杂度是O(1)；而logistic regression每次迭代要对所有N个点都进行计算，它的每时间复杂度是O(N)。

为了提高logistic regression中gradient descent算法的速度，可以使用另一种算法：随机梯度下降算法(StochasticGradient Descent)。随机梯度下降算法每次迭代只找到一个点，计算该点的梯度，作为我们下一步更新w的依据。这样就保证了每次迭代的计算量大大减小，我们可以把整体的梯度看成这个随机过程的一个期望值。

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta \underbrace{\frac{1}{N}\sum_{n=1}^{N} \theta\left(-y_n \mathbf{w}_t^T \mathbf{x}_n\right)\left(y_n \mathbf{x}_n\right)}_{-\nabla E_{\text{in}}(\mathbf{w}_t)}$$

- 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:
$\nabla_{\mathbf{w}} \ \text{err}(\mathbf{w}, \mathbf{x}_n, y_n)$ with random n
true gradient:
$\nabla_{\mathbf{w}} \ E_{\text{in}}(\mathbf{w}) = \underset{\text{random } n}{\mathcal{E}} \ \nabla_{\mathbf{w}} \ \text{err}(\mathbf{w}, \mathbf{x}_n, y_n)$

随机梯度下降可以看成是真实的梯度加上均值为零的随机噪声方向。单次迭代看，好像会对每一步找到正确梯度方向有影响，但是整体期望值上看，与真实梯度的方向没有差太多，同样能找到最小值位置。随机梯度下降的优点是减少计算量，提高运算速度，而且便于online学习；缺点是不够稳定，每次迭代并不能保证按照正确的方向前进，而且达到最小值需要迭代的次数比梯度下降算法一般要多。

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 $\approx$ average stochastic gradient
- pros: **simple & cheaper computation :-)**
  —useful for **big data** or **online learning**
- cons: less stable in nature

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

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta \underbrace{\theta\left(-y_n \mathbf{w}_t^T \mathbf{x}_n\right)\left(y_n \mathbf{x}_n\right)}_{-\nabla \text{err}(\mathbf{w}_t, \mathbf{x}_n, y_n)}$$

SGD logistic regression:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta \cdot \theta\left(-y_n \mathbf{w}_t^T \mathbf{x}_n\right)\left(y_n \mathbf{x}_n\right)$$

PLA:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + 1 \cdot \left[\!\left[y_n \neq \text{sign}(\mathbf{w}_t^T \mathbf{x}_n)\right]\!\right]\left(y_n \mathbf{x}_n\right)$$

- SGD logistic regression ≈ 'soft' PLA
- PLA ≈ SGD logistic regression with $\eta = 1$ when $\mathbf{w}_t^T \mathbf{x}_n$ large

two practical rule-of-thumb:

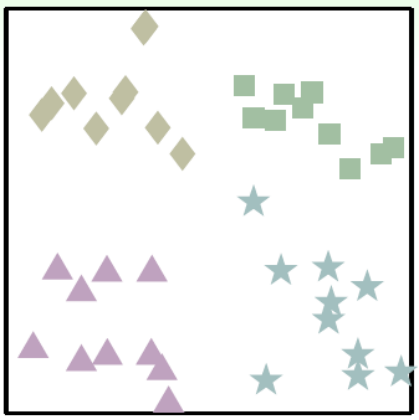
- stopping condition? t large enough
- η? 0.1 when $\mathbf{x}$ in proper range

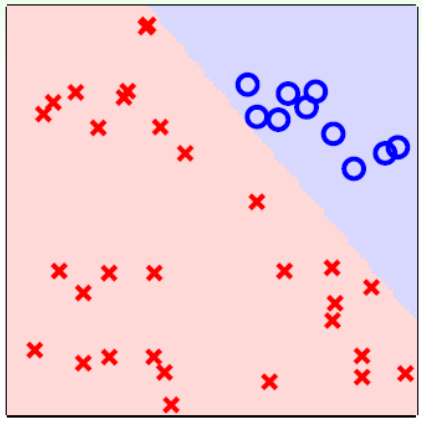
我们把SGD logistic regression称之为'soft' PLA，因为PLA只对分类错误的点进行修正，而SGD logistic regression每次迭代都会进行或多或少的修正。另外，当$\eta = 1$且$w_t^T x_n$足够大的时候，PLA近似等于SGD。
另外，需要注意的两点：

- SGD的终止迭代条件:没有统一的终止条件，一般让迭代次数足够多即可；
- 学习速率η：η的取值是根据实际情况来定的，一般取值0.1（0.1126）就可以了，

3. Multiclass via Logistic Regression

One Class at a Time

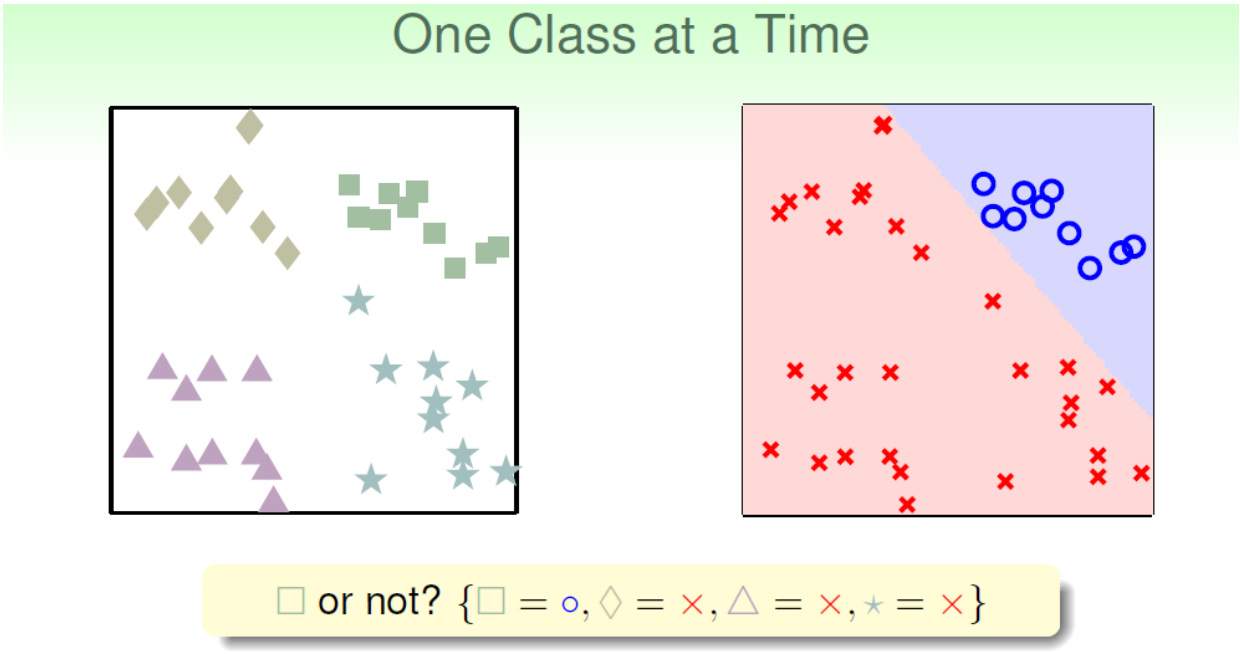


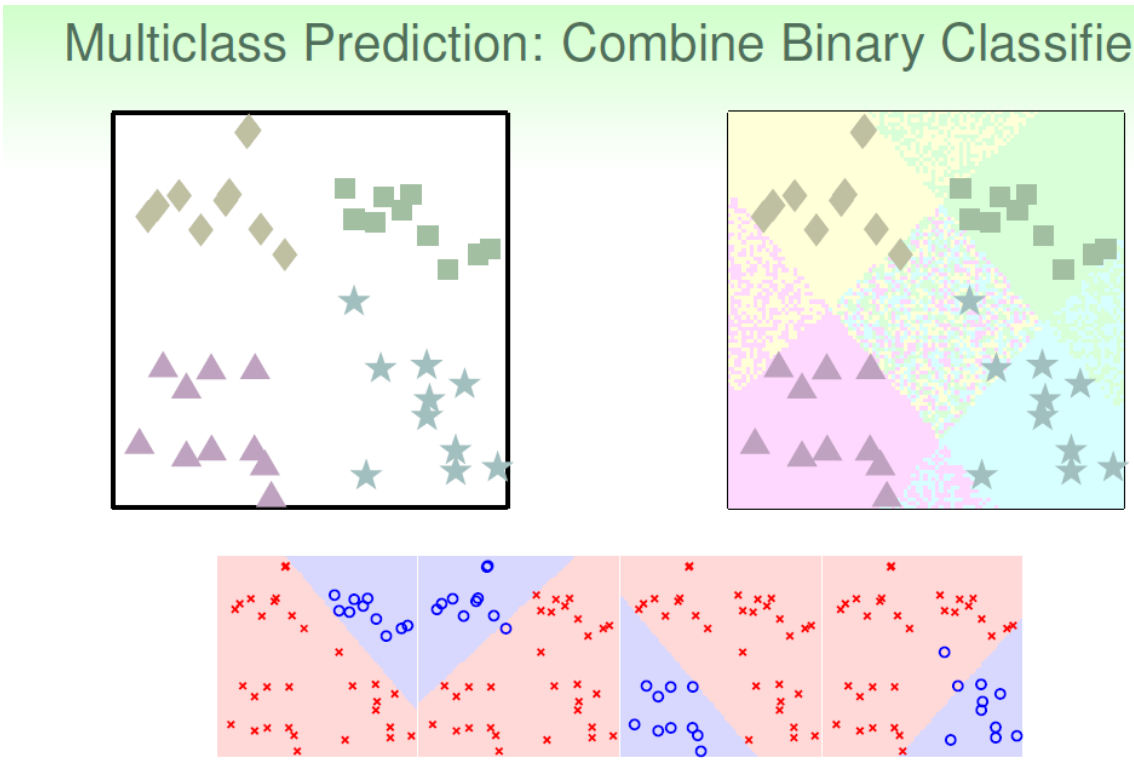


□ or not? $\{\square = \circ, \Diamond = \times, \triangle = \times, \star = \times\}$

假设平面上有四个类，分别是正方形、菱形、三角形和星形，现在要通过linear classification模型解决多分类问题。首先就是先把正方形作为正类，其他三种形状都是负类，即把它当成一个二分类问题，通过linear classification模型进行训练，得出平面上某个图形是不是正方形，且只有

{1,+1}两种情况。然后再分别以菱形、三角形、星形为正类，进行二元分类。这样进行四次二分类之后，就完成了这个多分类问题。

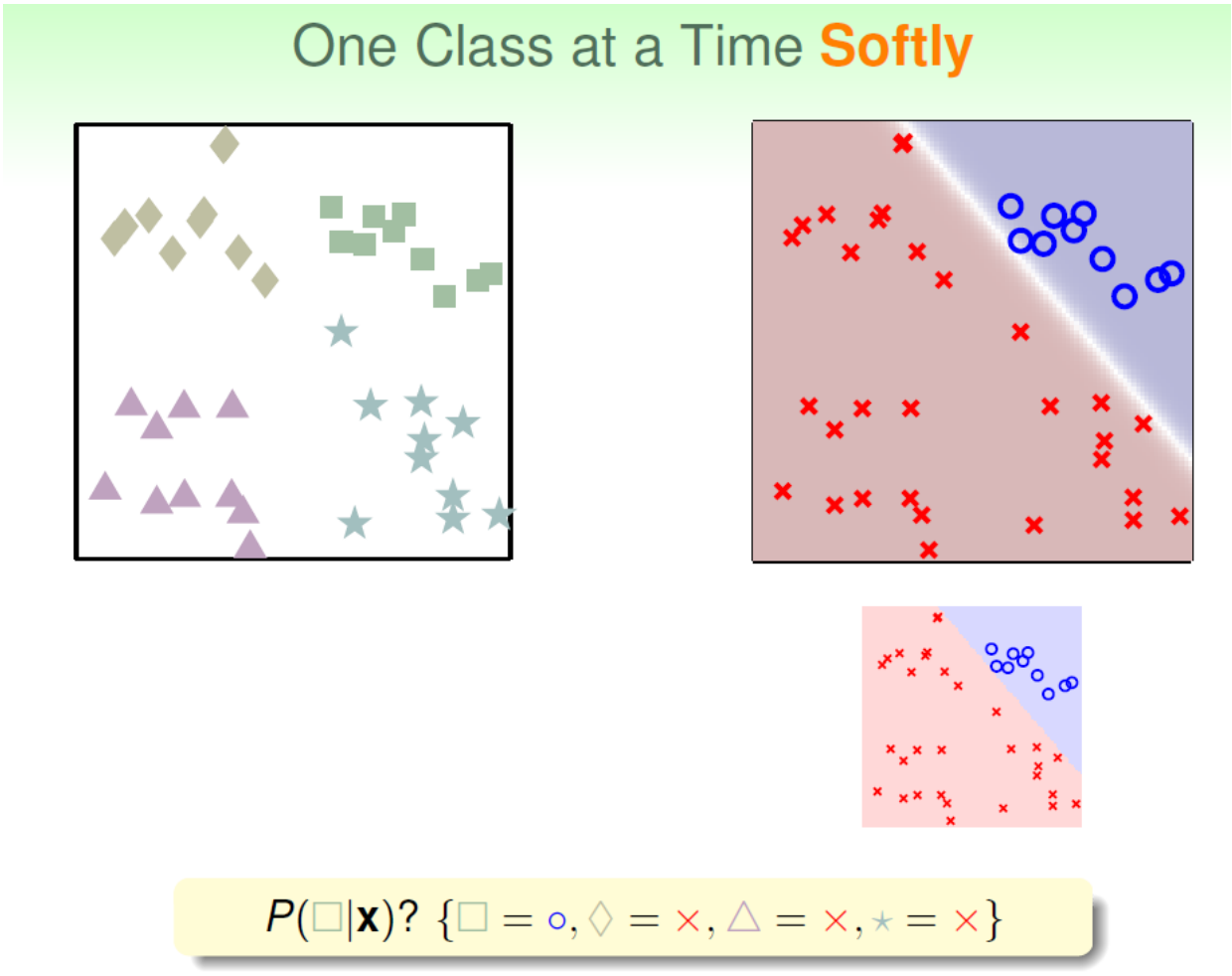


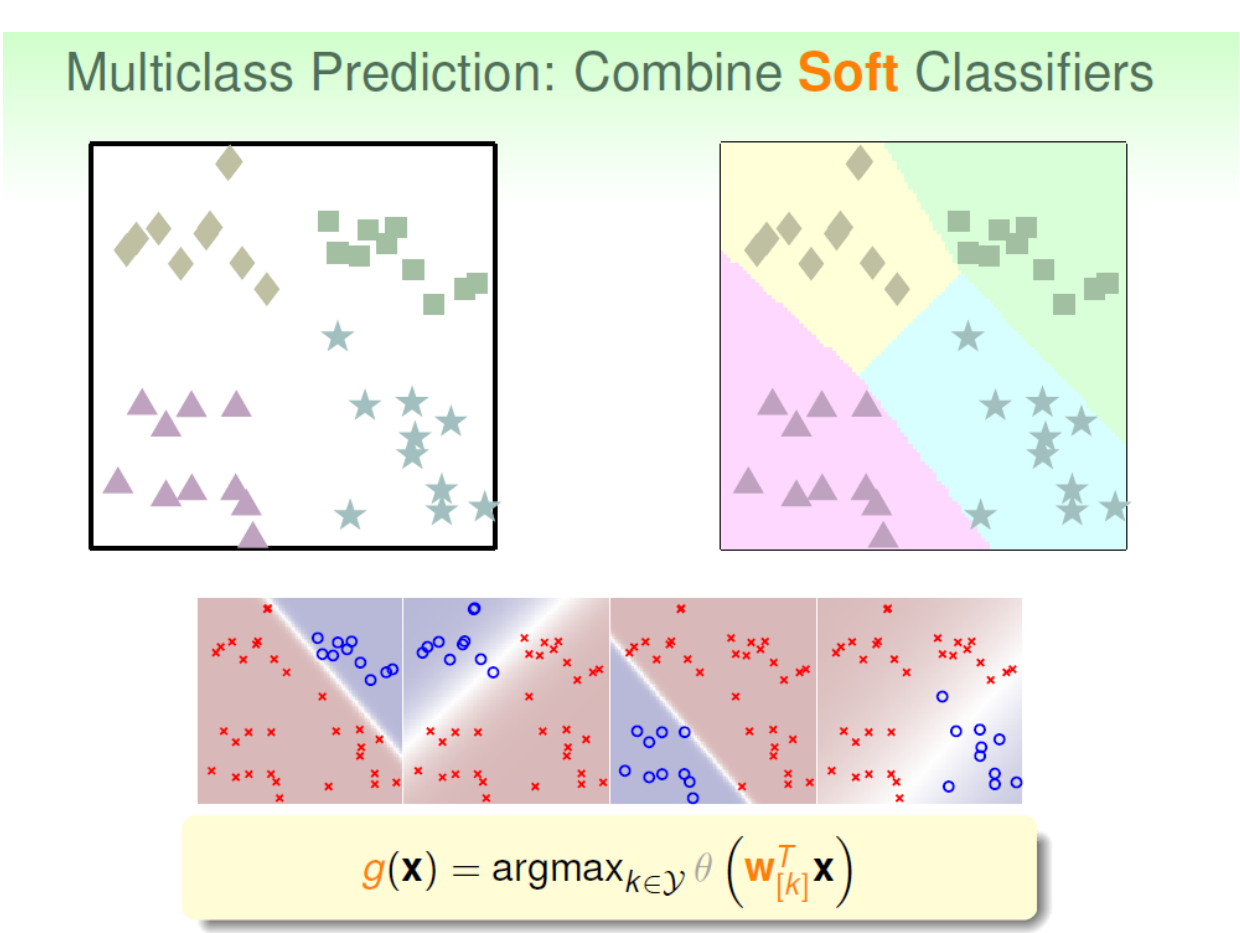


但是，通过上述的二分类方法会带来一些问题，因为我们只用{1，+1}两个值来标记，那么平面上某些可能某些区域都被上述四次二分类模型判断为负类，即不属于四类中的任何一类,如上图中的中心区域；也可能会出现某些区域同时被两个类甚至多个类同时判断为正类，比如某个区域又判定为正方形又判定为菱形。那么对于这种情况，我们就无法进行多类别的准确判断，所以对于多类别，简单的binary classification不能解决问题。

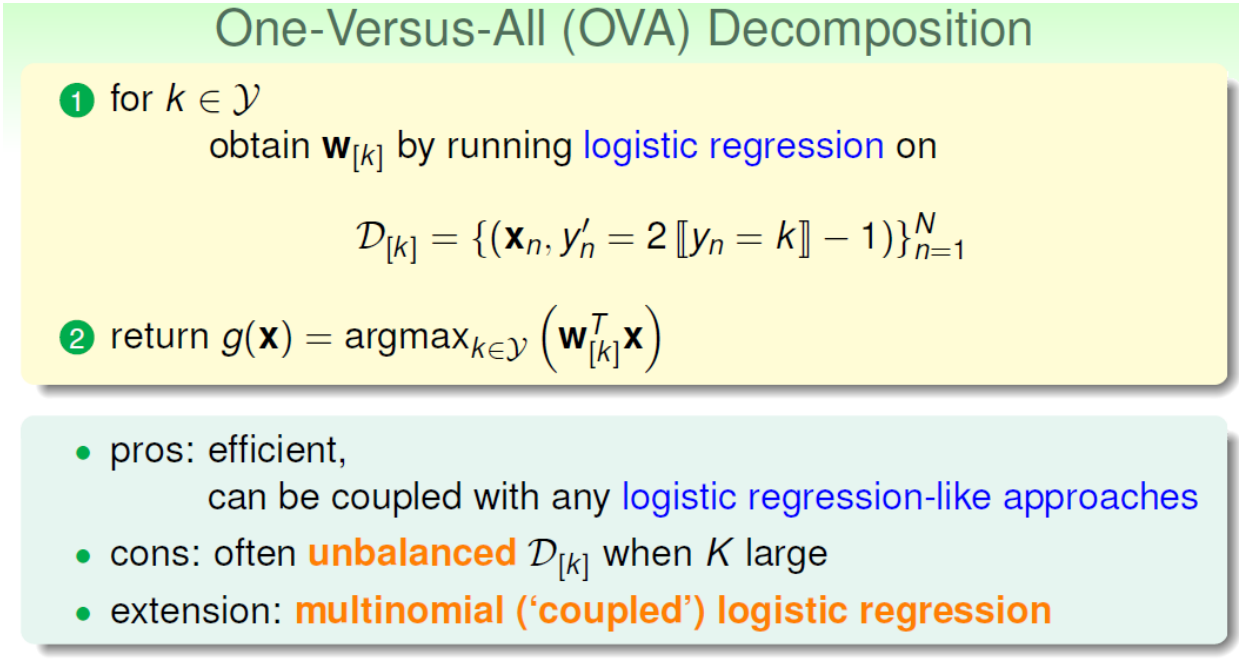
针对这种问题，我们可以使用另外一种方法来解决：soft软性分类，即不用{1，+1}这种binary classification，而是使用logistic regression，计算某点属于某类的概率、可能性，取概率最大的值为那一类就好。soft classification的处理过程和之前类似，同样是分别令某类为正，其他三类为

负，不同的是得到的是概率值，而不是{1，+1}。最后得到某点分别属于四类的概率，取最大概率对应的哪一个类别就好。效果如下图所示：





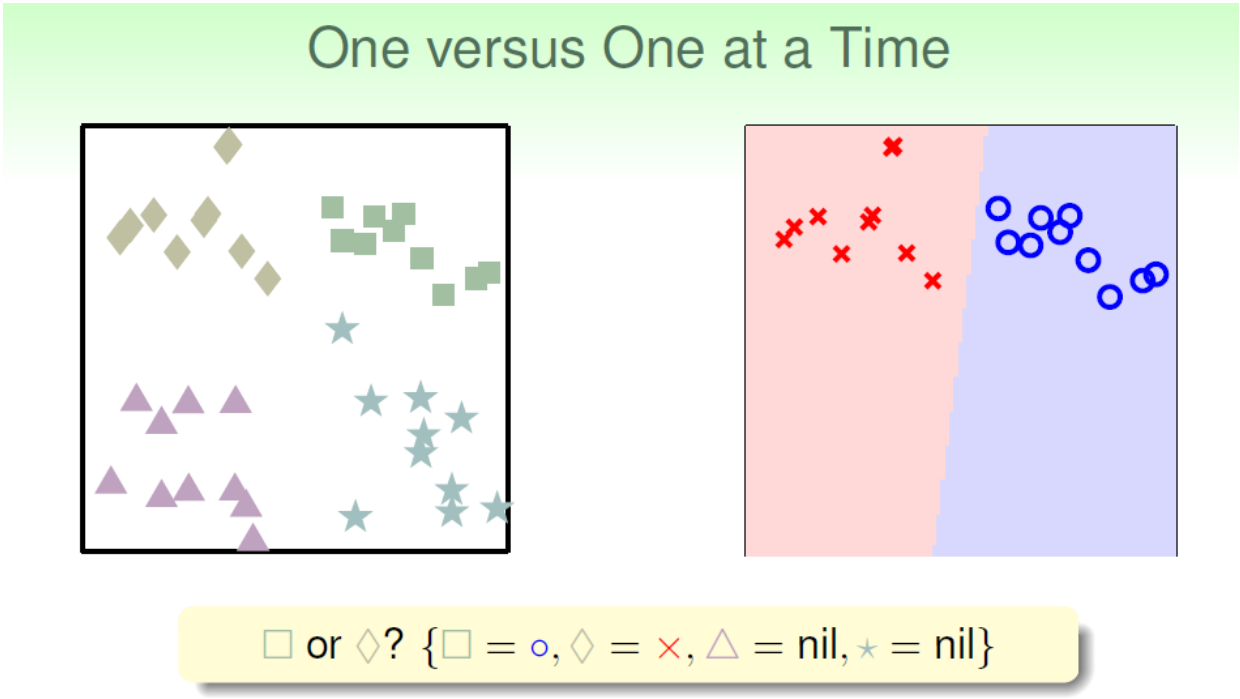
这种多分类的处理方式，我们称之为One-Versus-All(OVA) Decomposition(称OVR：One vs Rest更标准些)。这种方法的优点是简单高效，可以使用logistic regression模型来解决；缺点是如果数据类别很多时，那么每次二分类问题中，正类和负类的数量差别就很大，数据不平衡，这样会影响分类效果。

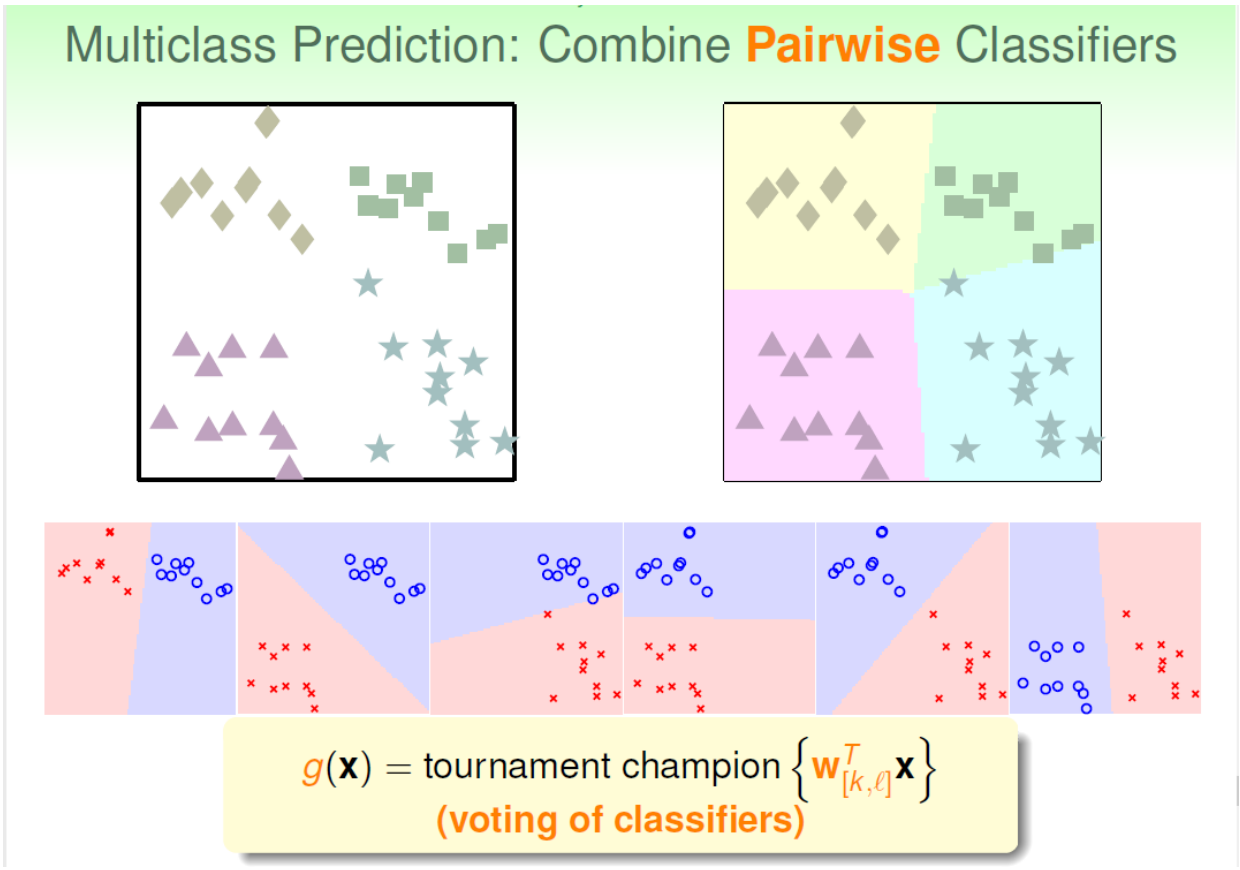


4.Multiclass via Binary Classification

多分类算法OVA存在一个问题，就是当类别k很多的时候，造成正负类数据不平衡，会影响分类效果，表现不好。现在，使用另一种方法OVO来解决这个不平衡问题。

这种方法呢，每次只取两类进行binary classification，取值为{1，+1}。假k=4，那么总共需要进行$C_4^2 = 6$次binary classification。那么，六次分类之后，如果平面上有个点，有三个分类器判断它是正方形，一个分类器判断是菱形，另外两个判断是三角形，那么取最多的那个判断，即判断它属于正方形，我们的分类就完成了。这种形式就如同k个足球对进行单循环的比赛，每场比赛都有一个队赢，一个队输，赢了得1分，输了得0分。那么总共进行了C_k^2次的比赛，最终取得分最高的那个队就可以了.





这种多分类方法叫做One-Versus-One(OVO)。这种方法的优点是更加高效，因为虽然需要进行的分类次数增加了，但是每次只需要进行两个类别的比较，也就是说单次分类的数量减少了。而且一般不会出现数据unbalanced的情况。缺点是需要分类的次数多，时间复杂度和空间复杂度可能都比较高。

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
   $$\mathcal{D}_{[k,\ell]} = \{(\mathbf{x}_n, y'_n = 2[\![y_n = k]\!] - 1) : y_n = k \text{ or } y_n = \ell\}$$
2. return $g(\mathbf{x}) =$ tournament champion $\left\{\mathbf{w}_{[k,\ell]}^T \mathbf{x}\right\}$

- pros: efficient ('smaller' training problems), stable,
  can be coupled with any binary classification approaches
- cons: use $O(K^2)$ $\mathbf{w}_{[k,\ell]}$
  —**more space, slower prediction, more training**

5. 总结

本节课主要介绍了分类问题的三种线性模型：linear classification、linear regression和logistic regression都可以来做binary classification。然后介绍了比梯度下降算法更加高效的SGD算法来进行logistic regression分析。最后讲解了两种多分类方法，一种是OVA，用一个类别跟其他 类别比，然后来看看哪一个的可能性最高。另一种是OVO。两个类别之间两两比较，然后从所有的预测结果中选择出现次数最多的预测结果。这两种方法各有优缺点，当类别数量k不多的时候，建议选择OVA，以减少分类次数。

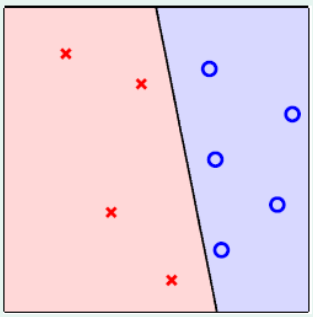
Lecture 12: Nonlinear Transformation

1. Quadratic Hypothesis

线性模型的优点就是，它的VC Dimension比较小，保证了$E_{in} \approx E_{out}$。但是缺点也很明显，对某些非线性问题，可能会造成E_{in}很大，虽然$E_{in} \approx E_{out}$，但是也造成E_{out}很大，分类效果不佳。

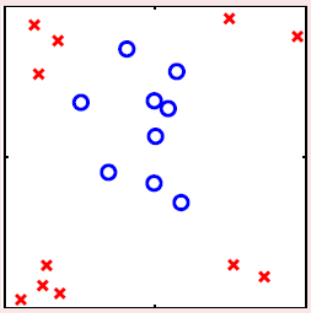
Linear Hypotheses

up to now: linear hypotheses



- visually: **'line'-like** boundary
- mathematically: linear scores $s = \mathbf{w}^T\mathbf{x}$

but limited . . .



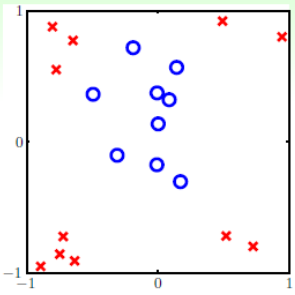
- theoretically: d_{VC} **under control :-)**
- practically: on some $\mathcal{D}$, **large** E_{in} for every line **:-(**

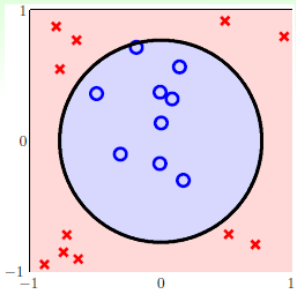
为了解决线性模型的缺点，我们可以使用非线性模型来进行分类。例如数据集D不是线性可分的，而是圆形可分的，圆形内部是正类，外面是负类。假设它的hypotheses可以写成：

$$h_{SEP}(x) = sign(-x_1^2 - x_2^2 + 0.6)$$

这样，计算每个点到圆心的距离的平方，如果这个平方大于0.6就判断为叉叉，小于就判断为圈圈。

Circular Separable



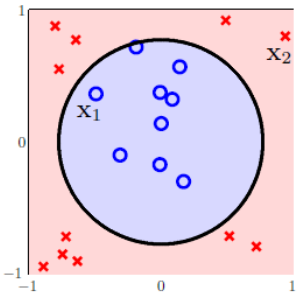


- $\mathcal{D}$ not linear separable
- but **circular separable** by a circle of radius $\sqrt{0.6}$ centered at origin:

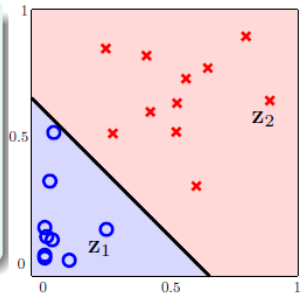
$$h_{\text{SEP}}(\mathbf{x}) = \text{sign}\left(-x_1^2 - x_2^2 + 0.6\right)$$

Circular Separable and Linear Separable

$$h(\mathbf{x}) = \text{sign}\left(\underbrace{0.6}_{\tilde{w}_0}\cdot\underbrace{1}_{z_0} + \underbrace{(-1)}_{\tilde{w}_1}\cdot\underbrace{x_1^2}_{z_1} + \underbrace{(-1)}_{\tilde{w}_2}\cdot\underbrace{x_2^2}_{z_2}\right) = \text{sign}\left(\tilde{\mathbf{w}}^T\mathbf{z}\right)$$



- $\{(\mathbf{x}_n, y_n)\}$ circular separable $\Longrightarrow$ $\{(\mathbf{z}_n, y_n)\}$ linear separable
- $\mathbf{x} \in \mathcal{X} \xrightarrow{\Phi} \mathbf{z} \in \mathcal{Z}$: **(nonlinear) feature transform Φ**



circular separable in $\mathcal{X}$ $\Longrightarrow$ linear separable in $\mathcal{Z}$

vice versa?

这种的$x_n \longrightarrow z_n$转换可以看成是x空间的点映射到z空间中去，而在z域中，可以用一条直线进行分类，也就是从x空间的圆形可分映射到z空间的线性可分。z域中的直线对应于x域中的圆形。因此，我们把$x_n \longrightarrow z_n$这个过程称之为特征转换（Feature Transform）。通过　这种特征转换，可以将非线性模型转换为另一个域中的线性模型。

已知x域中圆形可分在z域中是线性可分的，那么反过来，如果在z域中线性可分，是否在x域中一定是圆形可分的呢？答案是否定的。由于权重向量w取值不同，x域中的hypothesis可能是圆形、椭圆、双曲线等等多种情况.

General Quadratic Hypothesis Set

a 'bigger' $\mathcal{Z}$-space with $\Phi_2(\mathbf{x}) = (1, x_1, x_2, x_1^2, x_1x_2, x_2^2)$

perceptrons in $\mathcal{Z}$-space $\Longleftrightarrow$ quadratic hypotheses in $\mathcal{X}$-space

$$\mathcal{H}_{\Phi_2} = \left\{h(\mathbf{x}) : h(\mathbf{x}) = \tilde{h}(\Phi_2(\mathbf{x})) \text{ for some linear } \tilde{h} \text{ on } \mathcal{Z}\right\}$$

- can implement **all possible quadratic curve boundaries**: circle, ellipse, **rotated** ellipse, hyperbola, parabola, . . .

  ellipse $2(x_1 + x_2 - 3)^2 + (x_1 - x_2 - 4)^2 = 1$

  $\Longleftarrow \tilde{\mathbf{w}}^T = [33, -20, -4, 3, 2, 3]$
- include **lines and constants as degenerate cases**

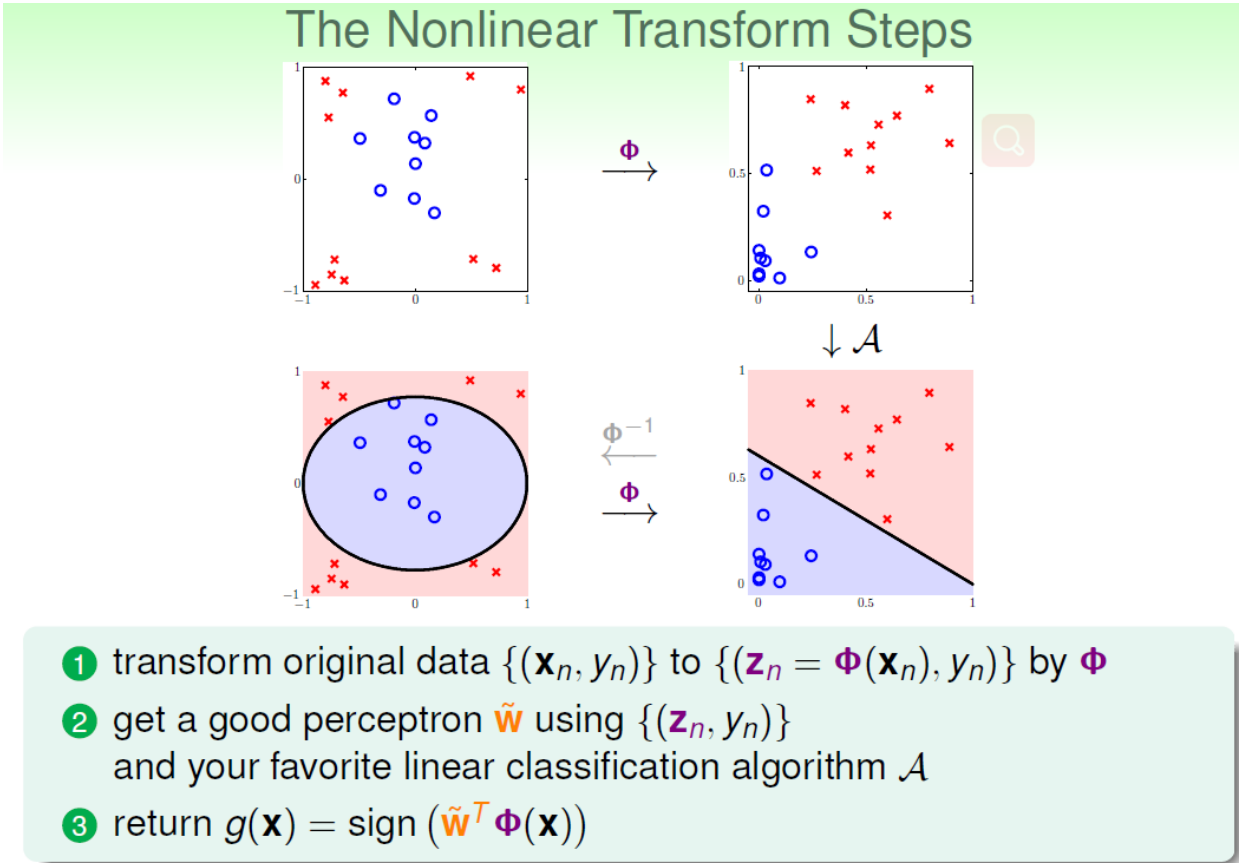
目前讨论的x域中的圆形都是圆心过原点的，对于圆心不过原点的一般情况， 映射公式包含的所有项为：

$$\Phi_2(x) = (1, x_1, x_2, x_1^2, x_2^2, x_2^2)$$

对于二次hypothesis，它包含二次项、一次项和常数项1，z域中每一条线对应x域中的某二次曲线的分类方式，也许是圆，也许是椭圆，也许是双曲线等等。

2.Nonlinear Transform

如何设计一个好的二次hypothesis来达到良好的分类效果。那么目标就是在z域中设计一个最佳的分类线。利用映射变换的思想，通过映射关系，把x域中的最高阶二次的多项式转换为z域中的一次向量，也就是从quardratic hypothesis转换成了perceptrons问题。用z值代替x多项式，其中向量z的个数与x域中x多项式的个数一致（包含常数项）。这样就可以在z域中利用线性分类模型进行分类训练。训练好的线性模型之后，再将z替换为x的多项式就可以了。具体过程如下：



整个过程就是通过映射关系，换个空间去做线性分类，重点包括两个：

- 特征转换
- 训练线性模型

3.Price of Nonlinear Transform

Computation/Storage Price

Q-th order polynomial transform: $\mathbf{\Phi}_Q(\mathbf{x}) = \Big($ $1,$
$x_1, x_2, \ldots, x_d,$
$x_1^2, x_1x_2, \ldots, x_d^2,$
$\ldots,$
$x_1^Q, x_1^{Q-1}x_2, \ldots, x_d^Q\Big)$

$\underbrace{1}_{\tilde{w}_0} + \underbrace{\tilde{d}}_{\text{others}}$ dimensions

= # ways of $\leq Q$-combination from d kinds with repetitions

$= \binom{Q+d}{Q} = \binom{Q+d}{d} = O(Q^d)$

= efforts needed for computing/storing $\mathbf{z} = \mathbf{\Phi}_Q(\mathbf{x})$ and $\tilde{\mathbf{w}}$

Q large $\Longrightarrow$ **difficult to compute/store**

若x特征维度是d维的，也就是包含d个特征.如果x特征维度是2维的，那么它的二次多项式为$(1, x_1, x_2, x_1^2, x_1x_2, x_2^2)$，有6个。现在，如果阶数更高，假设阶数为Q，那么对于x特征维度是d维的，它的z域特征维度为：

$$\frac{(Q+d)!}{Q!d!} = C_{Q+d}^{Q} = C_{Q+d}^{Q} = O(Q^d)$$

由上式可以看出，计算z域特征维度个数的时间复杂度是$O(Q^d)$，随着Q和d的增大，计算量会变得很大。同时，空间复杂度也大。也就是说，这种特征变换的一个代价是计算的时间、空间复杂度都比较大。

Model Complexity Price

Q-th order polynomial transform: $\Phi_Q(\mathbf{x}) = \Big($ $1,$ $x_1, x_2, \ldots, x_d,$ $x_1^2, x_1x_2, \ldots, x_d^2,$ $\ldots,$ $x_1^Q, x_1^{Q-1}x_2, \ldots, x_d^Q\Big)$

$\underbrace{1}_{\tilde{w}_0} + \underbrace{\tilde{d}}_{\text{others}}$ dimensions $= O(Q^d)$

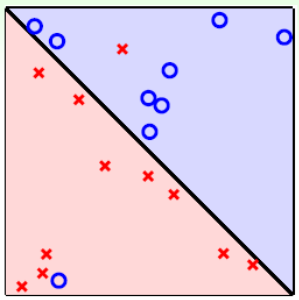
- number of free parameters $\tilde{w}_i = \tilde{d} + 1 \approx d_{\text{VC}}(\mathcal{H}_{\Phi_Q})$
- $d_{\text{VC}}(\mathcal{H}_{\Phi_Q}) \le \tilde{d} + 1$, why?

any $\tilde{d} + 2$ inputs not shattered in $\mathcal{Z}$
$\Longrightarrow$ any $\tilde{d} + 2$ inputs not shattered in $\mathcal{X}$

Q large $\Longrightarrow$ **large** d_{VC}

另一方面，z域中特征个数随着Q和d增加变得很大，同时权重w也会增大，即自由度增加，VC Dimension增大。令z域中的特征维是$\tilde{d}+1$，则在域中，任何$\tilde{d}+2$的输入都不能被shattered；同样，在x域中，任何$\tilde{d}+2$的输入也不能被shattered。$\tilde{d}+1$是VC Dimension的上界，如果$\tilde{d}+1$很大的时候，相应的VC Dimension就会很大。根据之前章节课程的讨论，VC Dimension过大，模型的泛化能力会比较差。

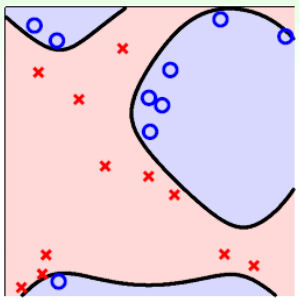
Generalization Issue



Φ_1 (original $\mathbf{x}$)

which one do you prefer? :-)

- Φ_1 'visually' preferred
- Φ_4: $E_{\text{in}}(g) = 0$ but overkill



Φ_4

1. can we make sure that $E_{\text{out}}(g)$ is close enough to $E_{\text{in}}(g)$?
2. can we make $E_{\text{in}}(g)$ small enough?

trade-off:	$\tilde{d}$ (Q)	1	2
	higher	:-(	**:-D**
	lower	**:-D**	:-(

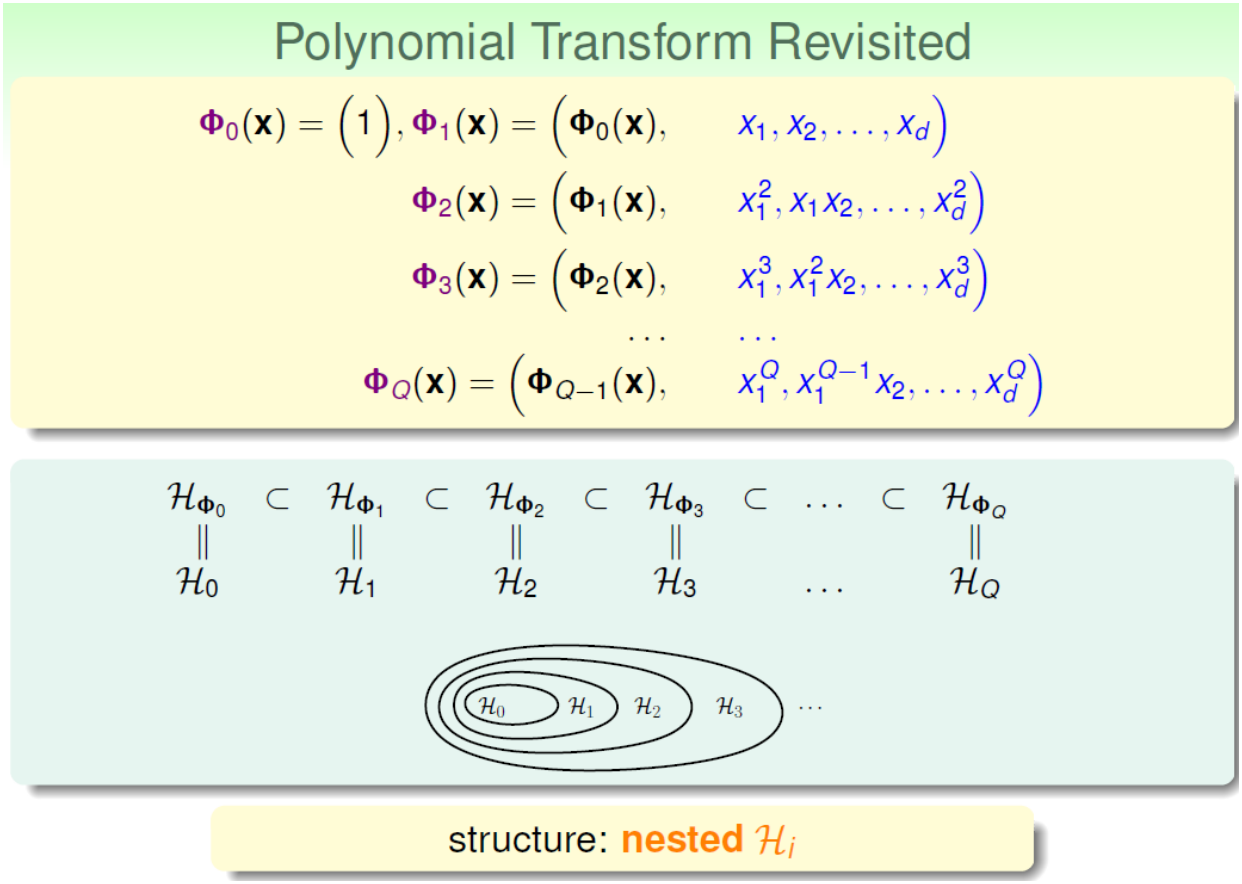
上图中，左边是用直线进行线性分类，有部分点分类错误；右边是用四次曲线进行非线性分类，所有点都分类正确，那么哪一个分类效果好呢？单从平面上这些训练数据来看，四次曲线的分类效果更好，但是四次曲线模型很容易带来过拟合的问题，虽然它的E_{in}比较小，从泛化能力上来说，还是左边的分类器更好一些。也就是说VC Dimension过大会带来过拟合问题，$\tilde{d}+1$不能太大

了。

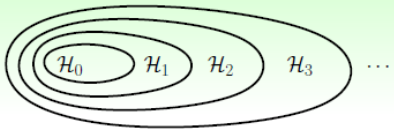
那么如何选择合适的Q，来保证不会出现过拟合问题，使模型的泛化能力强呢？一般情况下，为了尽量减少特征自由度，我们会根据训练样本的分布情况，人为地减少、省略一些项。但是，这种人为地删减特征会带来一些“自我分析”代价，虽然对训练样本分类效果好，但是对训练样本外的样本，不一定效果好。所以，一般情况下，还是要保存所有的多项式特征，避免对训练样本的人为选择。

4. Structured Hypothesis Sets

从x域到z域的多项式变幻如下图所示：

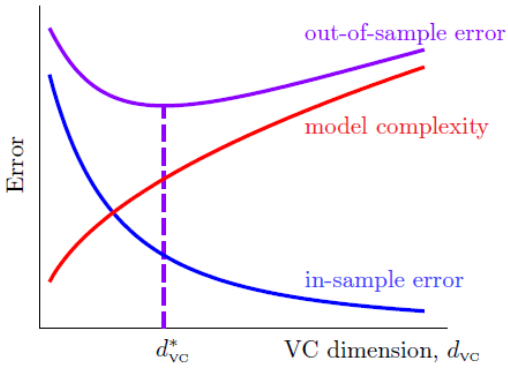


Structured Hypothesis Sets



Let $g_i = \text{argmin}_{h \in \mathcal{H}_i} E_{\text{in}}(h)$:

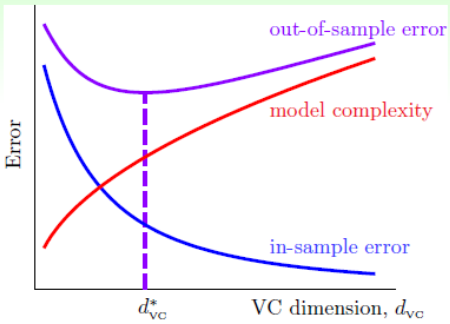
$$\begin{array}{ccccccccc} \mathcal{H}_0 & \subset & \mathcal{H}_1 & \subset & \mathcal{H}_2 & \subset & \mathcal{H}_3 & \subset & \cdots \\ d_{\text{VC}}(\mathcal{H}_0) & \leq & d_{\text{VC}}(\mathcal{H}_1) & \leq & d_{\text{VC}}(\mathcal{H}_2) & \leq & d_{\text{VC}}(\mathcal{H}_3) & \leq & \cdots \\ E_{\text{in}}(g_0) & \geq & E_{\text{in}}(g_1) & \geq & E_{\text{in}}(g_2) & \geq & E_{\text{in}}(g_3) & \geq & \cdots \end{array}$$



use $\mathcal{H}_{1126}$ won't be good! :-(

从上图中也可以看到，随着变换多项式的阶数增大，虽然E_{in}逐渐减小，但是model complexity会逐渐增大，造成E_{out}很大，所以阶数不能太高。

Linear Model First



- tempting sin: use $\mathcal{H}_{1126}$, low $E_{\text{in}}(g_{1126})$ to fool your boss
  —**really? :-(a dangerous path of no return**
- safe route: $\mathcal{H}_1$ first
  - if $E_{\text{in}}(g_1)$ good enough, **live happily thereafter :-)**
  - otherwise, move right of the curve
    with nothing lost except 'wasted' computation

linear model first:
simple, efficient, **safe**, **and workable!**

那么，如果选择的阶数很大，确实能使E_{in}接近于0，但是泛化能力通常很差，我们把这种情况叫做tempting sin。所以，一般最合适的做法是先从低阶开始，如先选择一阶hypothesis，看看E_{in}是否很小，如果足够小的话就选择一阶，如果E_{in}大的话，再逐渐增加阶数，直到满足要求为止。也就是说，尽量选择低阶的hypothes，这样才能得到较强的泛化能力。

5.总结

这节课主要介绍了非线性分类模型，通过非线性变换，将非线性模型映射到另一个空间，转换为线性模型，再来进行线性分类。本节课完整介绍了非线性变换的整体流程，以及非线性变换可能会带来的一些问题：时间复杂度和空间复杂度的增加。最后介绍了在要付出代价的情况下，使用非线性变换的最安全的做法，尽可能使用简单的模型，而不是模型越复杂越好。