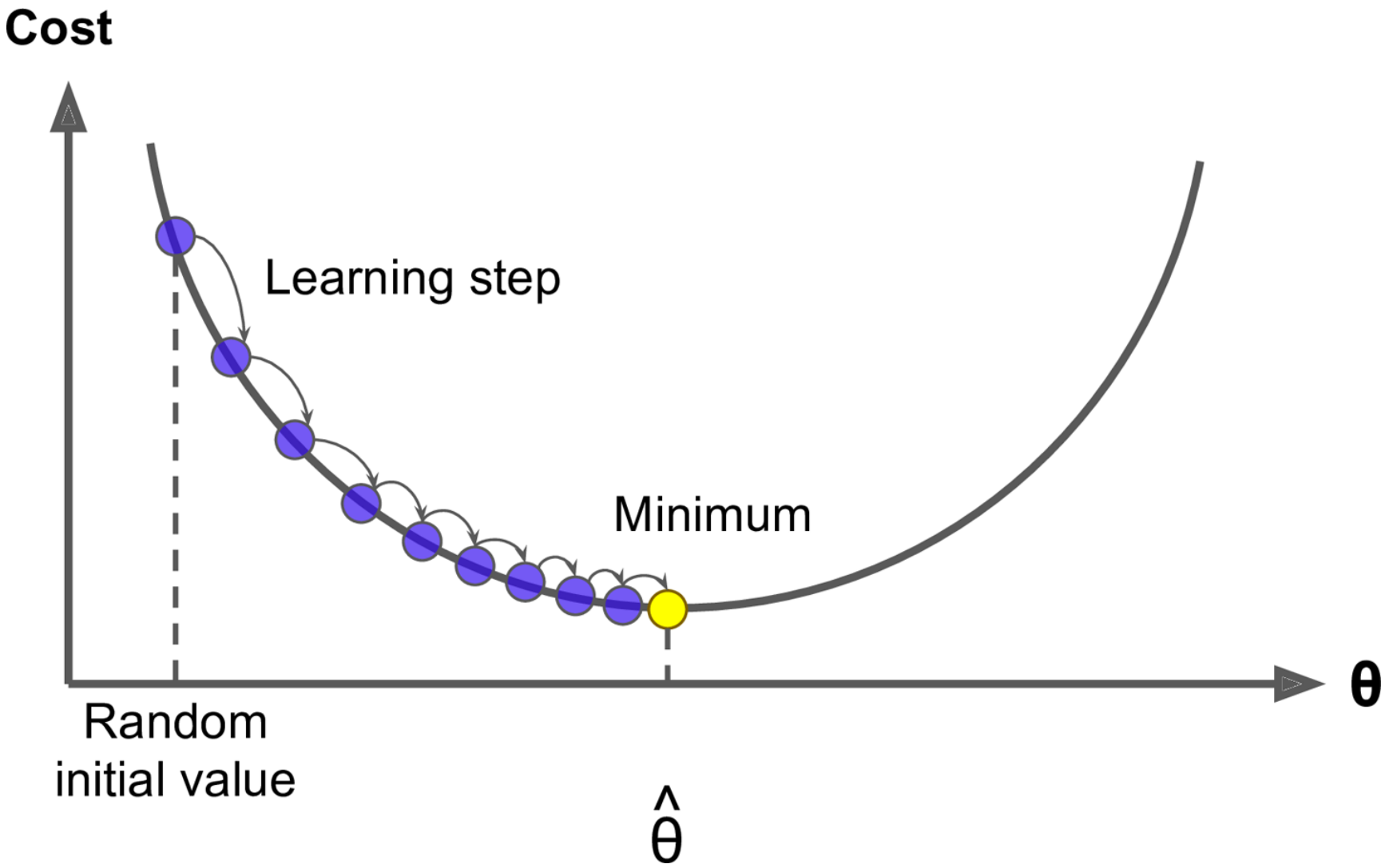
线性回归（续）

梯度下降法

- 上面利用公式求解里面对称阵是N维乘以N维的，复杂度是是O N的三次方，换句话说，就是如果你的特征数量翻倍，你的计算时间大致上要2的三次方，8倍的慢

梯度下降法



求导

$$\theta = \theta - \alpha \cdot \frac{\partial J(\theta)}{\partial \theta}$$

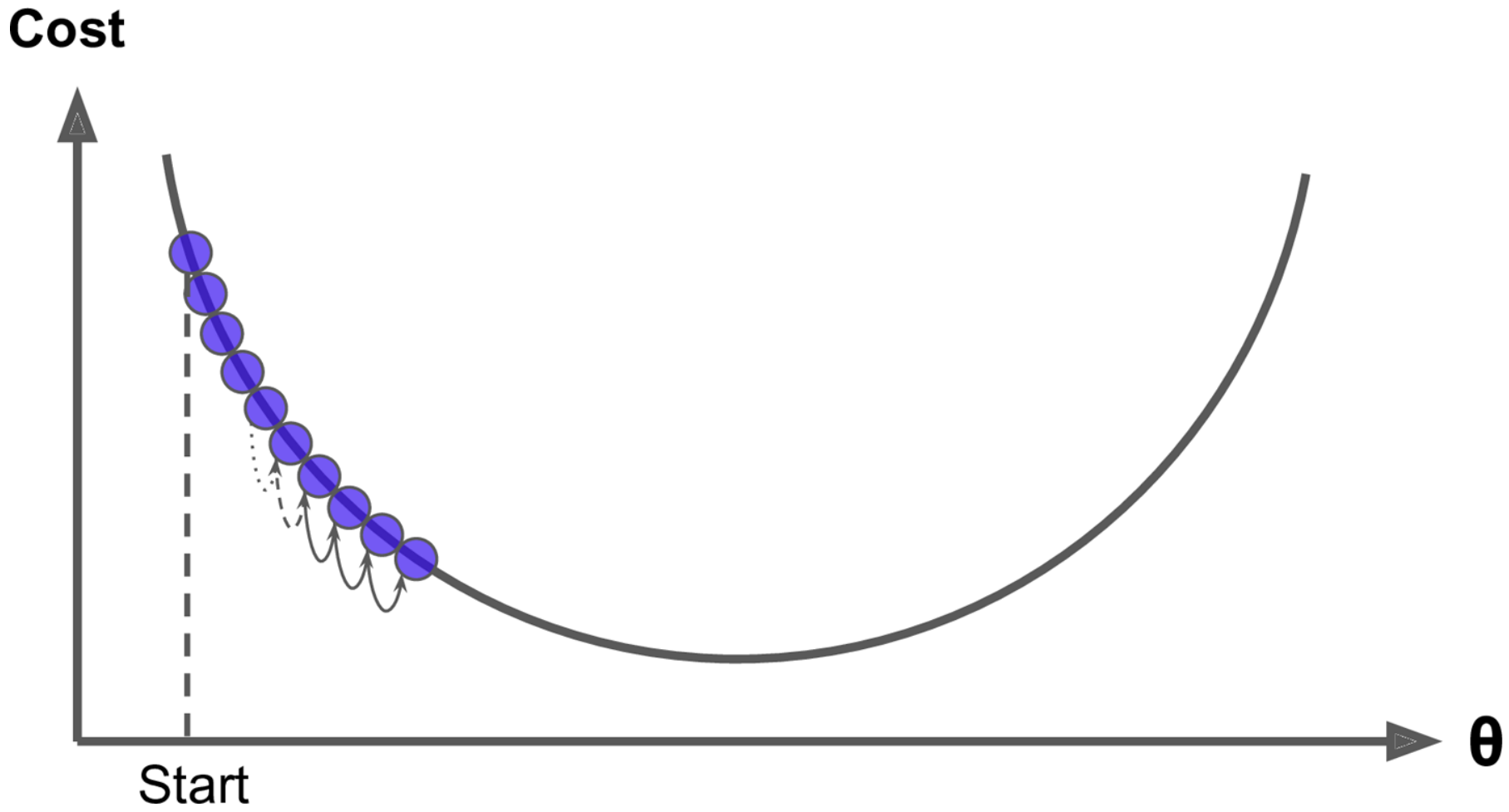
$$\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} \left(h_\theta(x) - y\right)^2 \\
&= 2 \cdot \frac{1}{2} \left(h_\theta(x) - y\right) \cdot \frac{\partial}{\partial \theta_j} \left(h_\theta(x) - y\right) \\
&= \left(h_\theta(x) - y\right) \cdot \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^{n} \theta_i x_i - y \right) \\
&= \left(h_\theta(x) - y\right) x_j
\end{aligned}$$

批量梯度下降

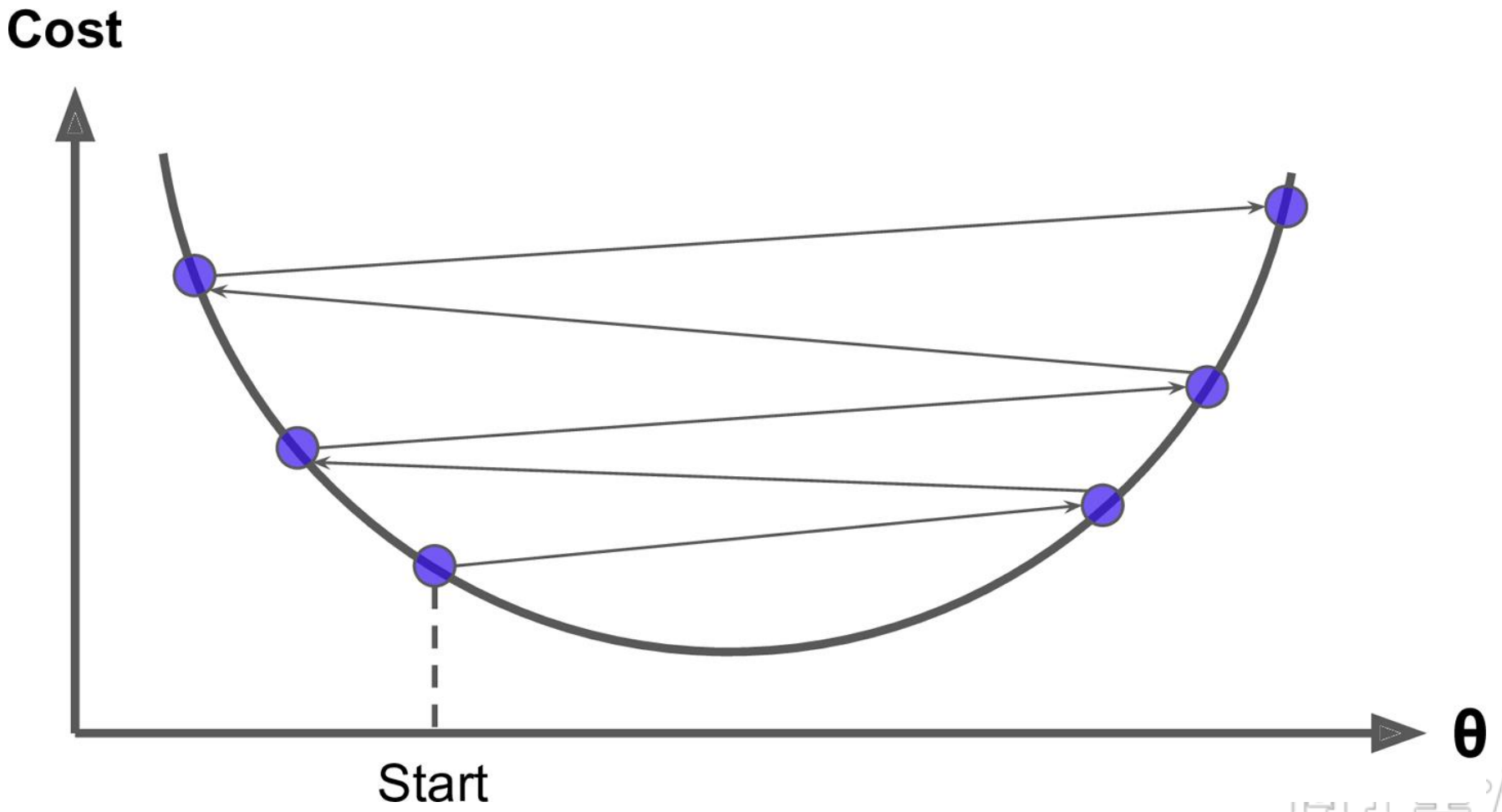
$$\theta_j := \theta_j + \alpha \sum_{i=1}^{m} \left(y^{(i)} - h_\theta(x^{(i)}) \right) x_j^{(i)}$$

- 初始化W，随机w，给初值
- 沿着负梯度方向迭代，更新后的w使得J(w)更小
- 如果w维度是几百维度，直接算SVD是可以的，几百维度以上一般是梯度下降算法，这个更像是机器学习，学习嘛，埃尔法是学习率、步长
- 上面这个公式其实是把所有样本梯度加起来的结果，因为有个加和符合从1到m，然后更新每个w的
- 卷积神经网络，最常见的算法我们就是用梯度下降

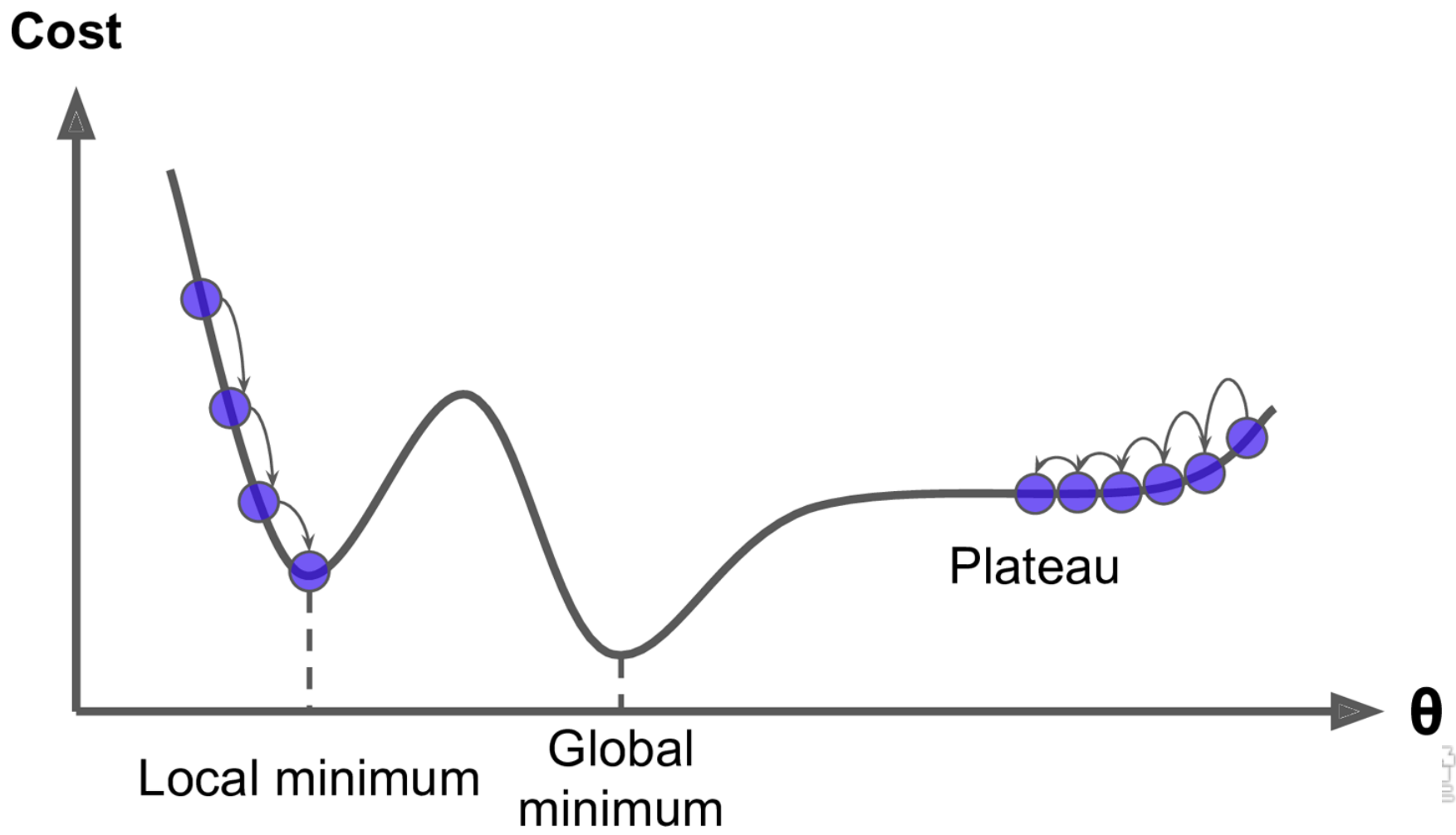
当Step小的时候



当Step大的时候



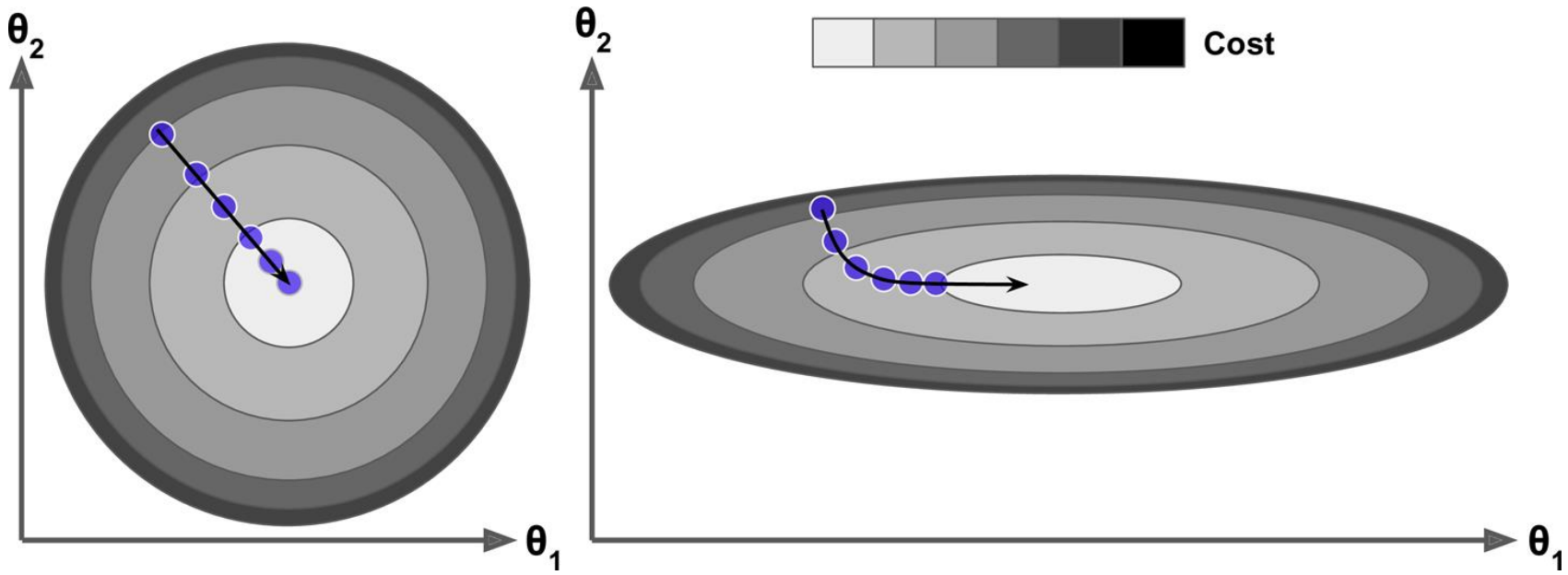
全局最优解



全局最优解

- 其实很多时候大家最后人工智能做多了，就发现很多时候不要去纠结全局最优，就像找终身伴侣怎么找的？你并不是在全球30亿女性中找一个最好的，你一般是在你朋友圈里面找一个，一个模型是堪用的，work的
- 我们只不过在线性回归模型中，目标函数求二阶导，是半正定的，是凸函数的
-
- 批量梯度下降是稳健的，贪心算法，不一定找到最优，但是一定可以找到更好的

归一化和没有归一化的特征

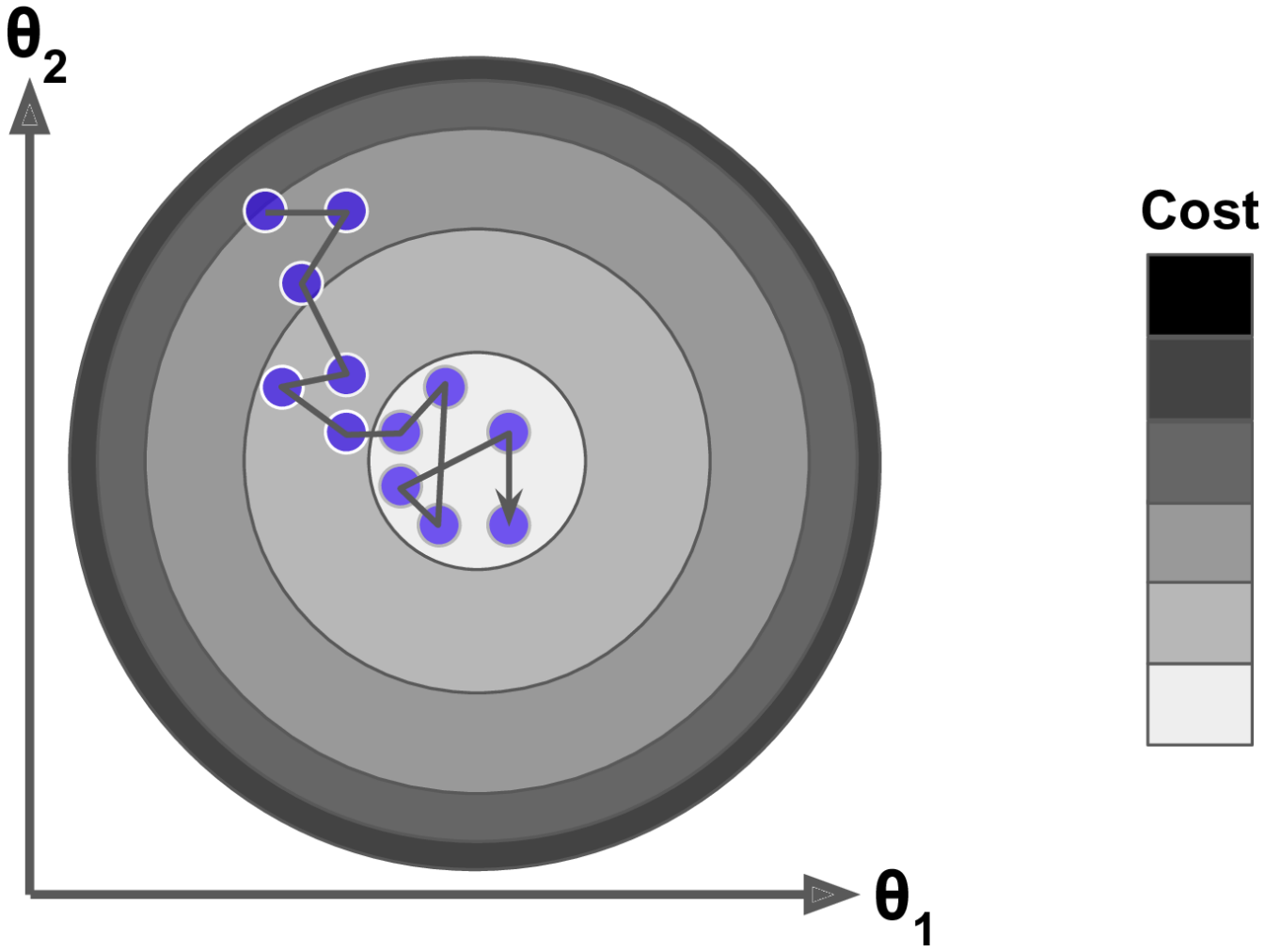


随机梯度下降

- 优先选择随机梯度下降
- 有些时候随机梯度下降可以跳出局部最小值

$$
\begin{aligned}
&\text{for i=1 to m, } \{ \\
&\quad \theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)}) \right) x_j^{(i)} \\
&\}
\end{aligned}
$$

随机梯度下降



L1正则 L2正则

$$\mathrm{MSE}(\mathbf{X}, h_\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(\theta^T \cdot \mathbf{x}^{(i)} - y^{(i)}\right)^2$$

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n \qquad \hat{y} = h_\theta(\mathbf{x}) = \theta^T \cdot \mathbf{x}$$

- Lasso Regression

$$J(\theta) = \mathrm{MSE}(\theta) + \alpha \sum_{i=1}^{n} \left|\theta_i\right|$$

- Ridge Regression

$$J(\theta) = \mathrm{MSE}(\theta) + \alpha \frac{1}{2} \sum_{i=1}^{n} \theta_i^2$$

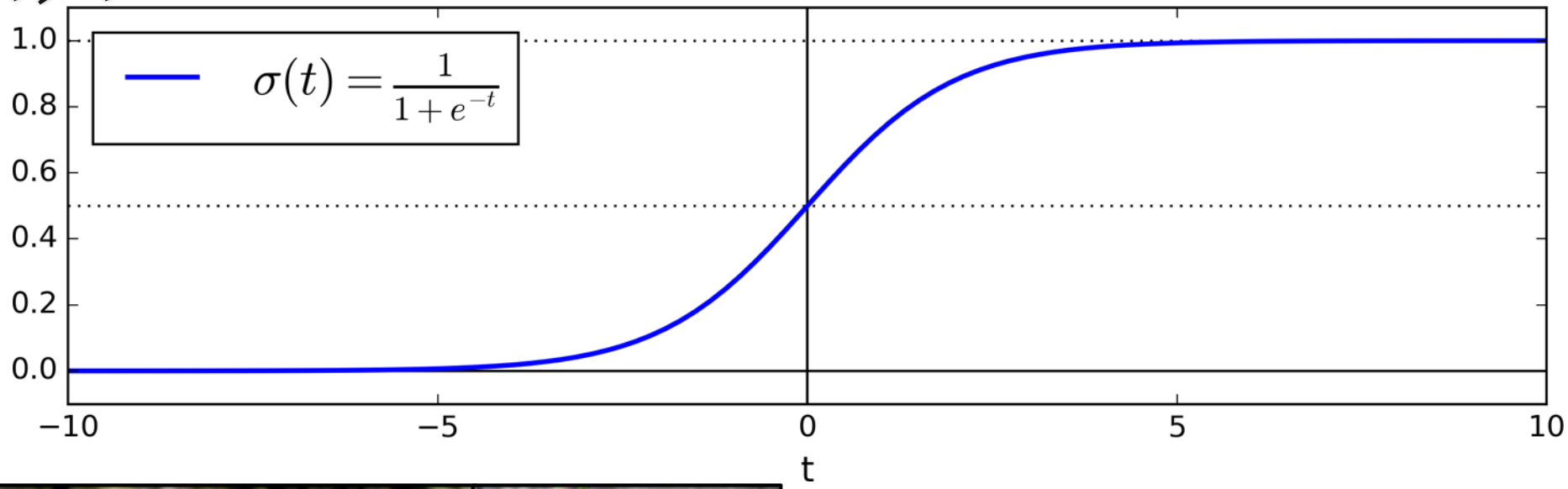
ElasticNet

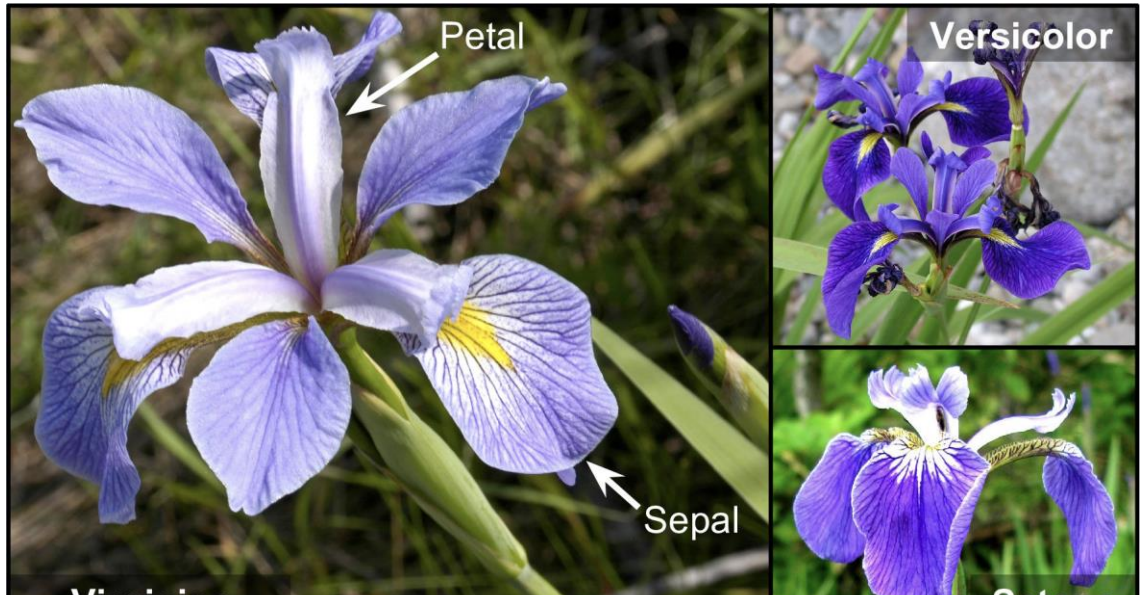
- 默认情况下Ridge是不错的
- 如果你怀疑只有少数特征有用
- 那么用Lasso或ElasticNet比较好
- 如果特征数大于了样本数量
- Lasso会不稳定，那么最好ElasticNet

$$J(\theta) = \mathrm{MSE}(\theta) + r\alpha \sum_{i=1}^{n} |\theta_i| + \frac{1-r}{2}\alpha \sum_{i=1}^{n} \theta_i^2$$

逻辑回归

$$\hat{p} = h_\theta(\mathbf{x}) = \sigma\left(\theta^T \cdot \mathbf{x}\right)$$





$$\sigma(t) = \frac{1}{1 + \exp(-t)}$$

逻辑回归

- 阈值为0.5
- 误差函数
- Log loss
- 求偏导

$$\hat{y} = \begin{cases} 0 \text{ if } \hat{p} < 0.5, \\ 1 \text{ if } \hat{p} \geq 0.5. \end{cases}$$

$$c(\theta) = \begin{cases} -\log(\hat{p}) & \text{if } y = 1, \\ -\log(1-\hat{p}) & \text{if } y = 0. \end{cases}$$

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}log\left(\hat{p}^{(i)}\right) + \left(1-y^{(i)}\right)log\left(1-\hat{p}^{(i)}\right)\right]$$

$$\frac{\partial}{\partial\theta_j}J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(\sigma\left(\theta^T \cdot \mathbf{x}^{(i)}\right) - y^{(i)}\right)x_j^{(i)}$$

Soft-max

$$s_k(\mathbf{x}) = {\theta_k}^T \cdot \mathbf{x}$$

$$\hat{p}_k = \sigma(\mathbf{s}(\mathbf{x}))_k = \frac{\exp\left(s_k(\mathbf{x})\right)}{\sum_{j=1}^{K} \exp\left(s_j(\mathbf{x})\right)}$$

- 类别K的分数
- Softmax函数
- 预测分类的方法

$$\hat{y} = \underset{k}{\operatorname{argmax}}\ \sigma(\mathbf{s}(\mathbf{x}))_k = \underset{k}{\operatorname{argmax}}\ s_k(\mathbf{x}) = \underset{k}{\operatorname{argmax}}\ \left({\theta_k}^T \cdot \mathbf{x}\right)$$

Soft-max

$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left(\hat{p}_k^{(i)}\right)$$

- 交叉熵损失函数
- 如果K是2，其实就是逻辑回归
- 求导得梯度

$$\nabla_{\theta_k} J(\Theta) = \frac{1}{m}\sum_{i=1}^{m}\left(\hat{p}_k^{(i)} - y_k^{(i)}\right)\mathbf{x}^{(i)}$$

交叉熵

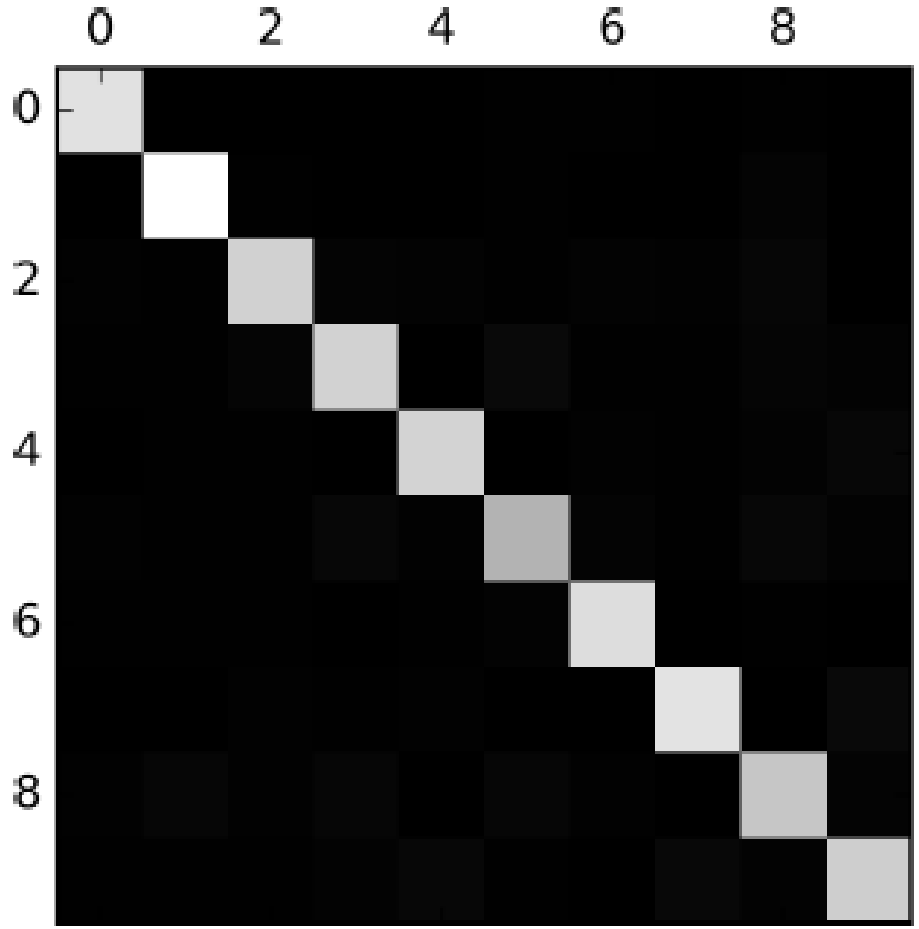
- 来自于信息论
- 交叉熵可在神经网络(机器学习)中作为损失函数，p表示真实标记的分布，q则为训练后的模型的预测标记分布，交叉熵损失函数可以衡量p与q的相似性。

评估指标

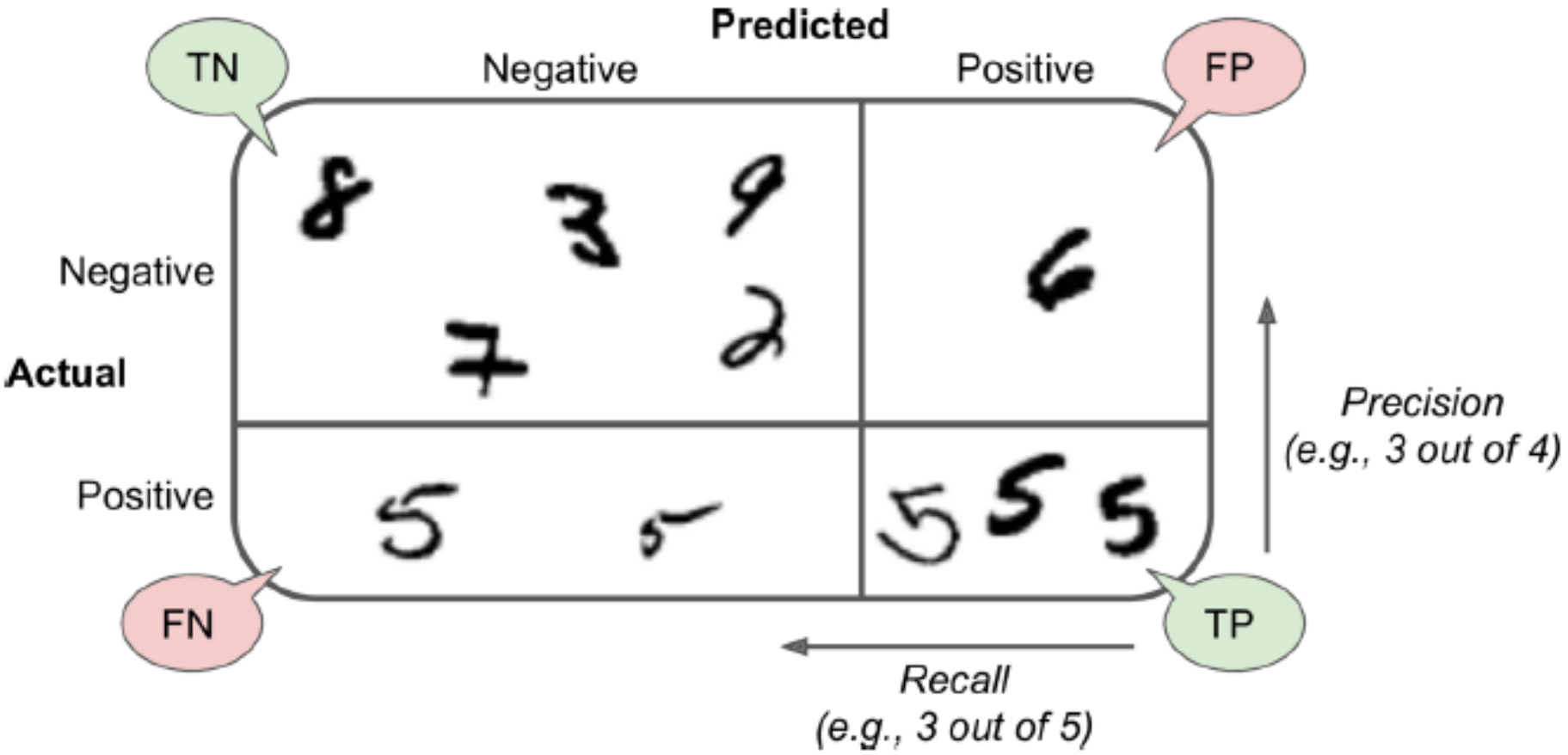
- K折交叉验证
- 交叉验证(Cross-validation)主要用于
- 建模应用中，例如PCR 、PLS 回归建模
- 在给定的建模样本中，拿出大部分
- 样本进行建模型，留小部分样本用刚
- 建立的模型进行预报，并求这小部分
- 样本的预报误差，记录们的平方加和

混淆矩阵

```
array([[5725,    3,   24,    9,   10,   49,   50,   10,   39,    4],
       [   2, 6493,   43,   25,    7,   40,    5,   10,  109,    8],
       [  51,   41, 5321,  104,   89,   26,   87,   60,  166,   13],
       [  47,   46,  141, 5342,    1,  231,   40,   50,  141,   92],
       [  19,   29,   41,   10, 5366,    9,   56,   37,   86,  189],
       [  73,   45,   36,  193,   64, 4582,  111,   30,  193,   94],
       [  29,   34,   44,    2,   42,   85, 5627,   10,   45,    0],
       [  25,   24,   74,   32,   54,   12,    6, 5787,   15,  236],
       [  52,  161,   73,  156,   10,  163,   61,   25, 5027,  123],
       [  43,   35,   26,   92,  178,   28,    2,  223,   82, 5240]])
```



准确率和召回率



$$\text{precision} = \frac{TP}{TP + FP}$$

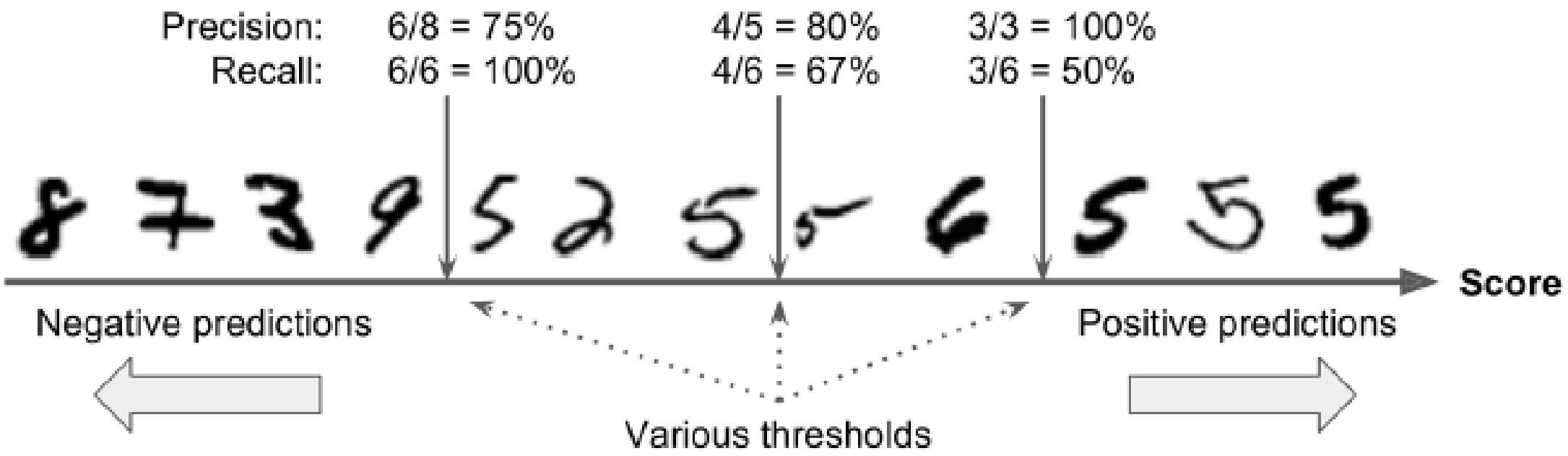
$$\text{recall} = \frac{TP}{TP + FN}$$

F1-Score

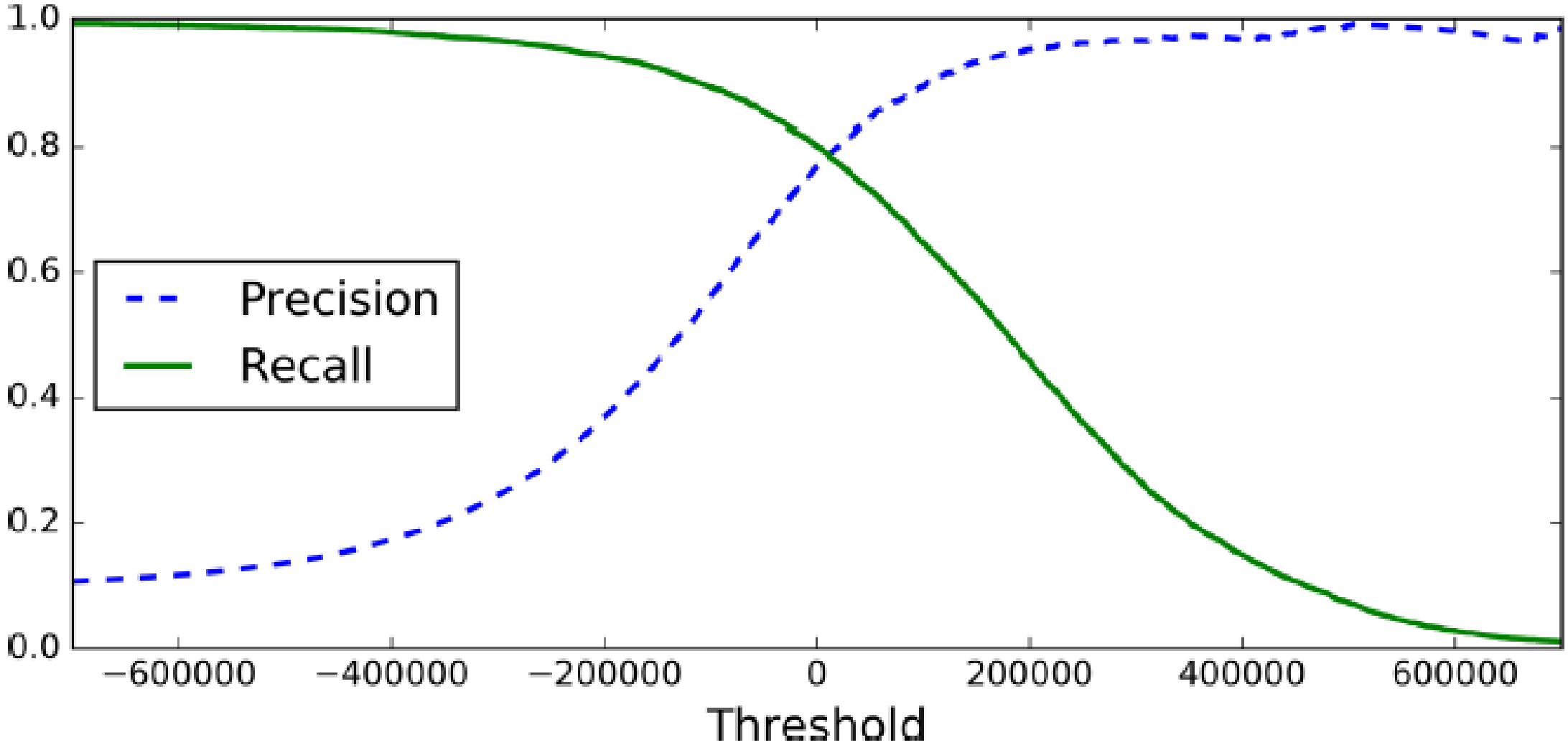
$$F_1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = \frac{TP}{TP + \frac{FN + FP}{2}}$$

- 如果训练分类器去检测视频是否对应小孩是安全的，宁愿拒绝很多好的视频，低召回率去保证好的视频，也就是要高准确率
- 如果监视录像中去检测商店小偷，那么我们可以要高召回率，低准确率，所以这里有TradeOff

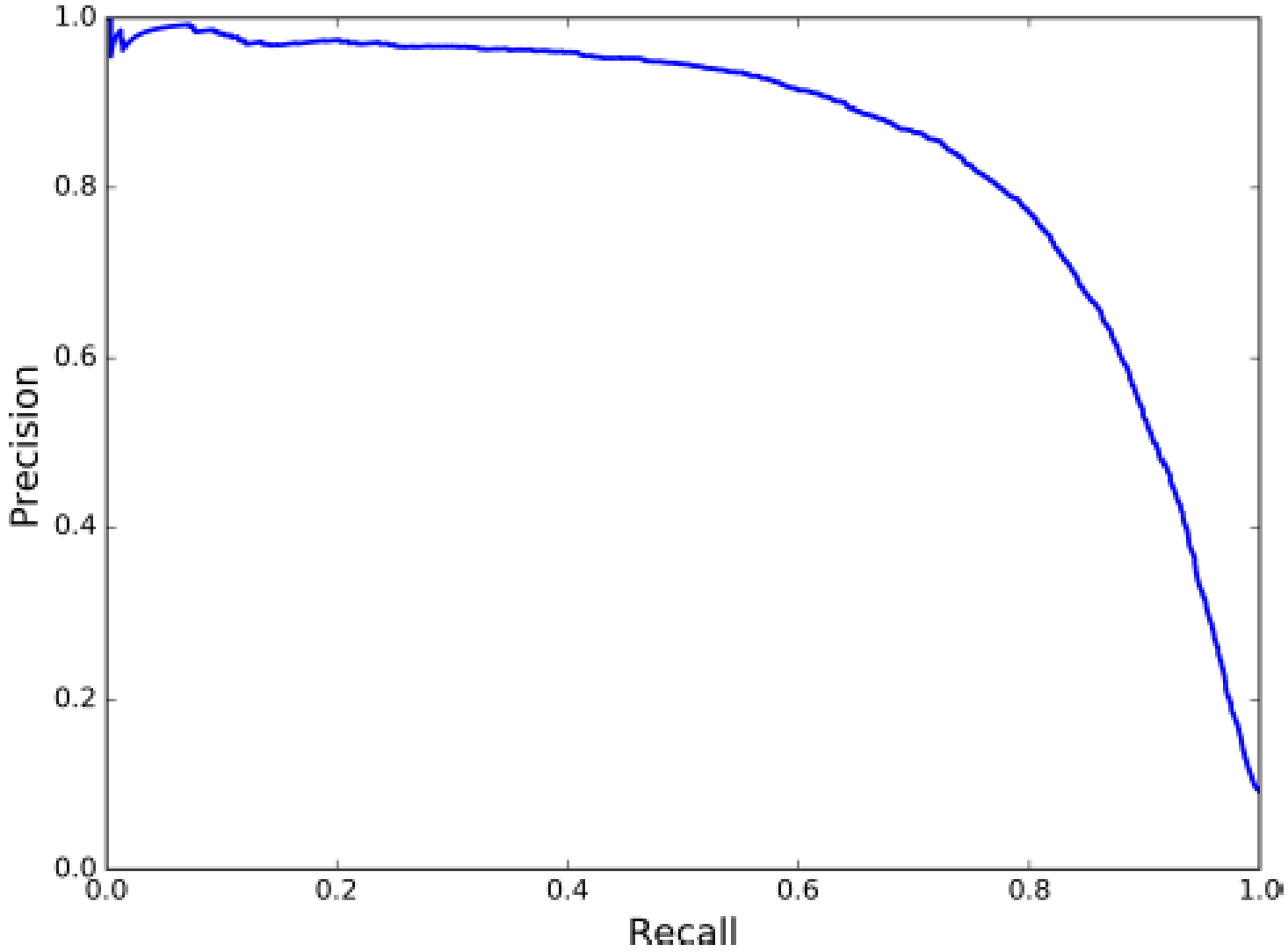
TradeOff



TradeOff

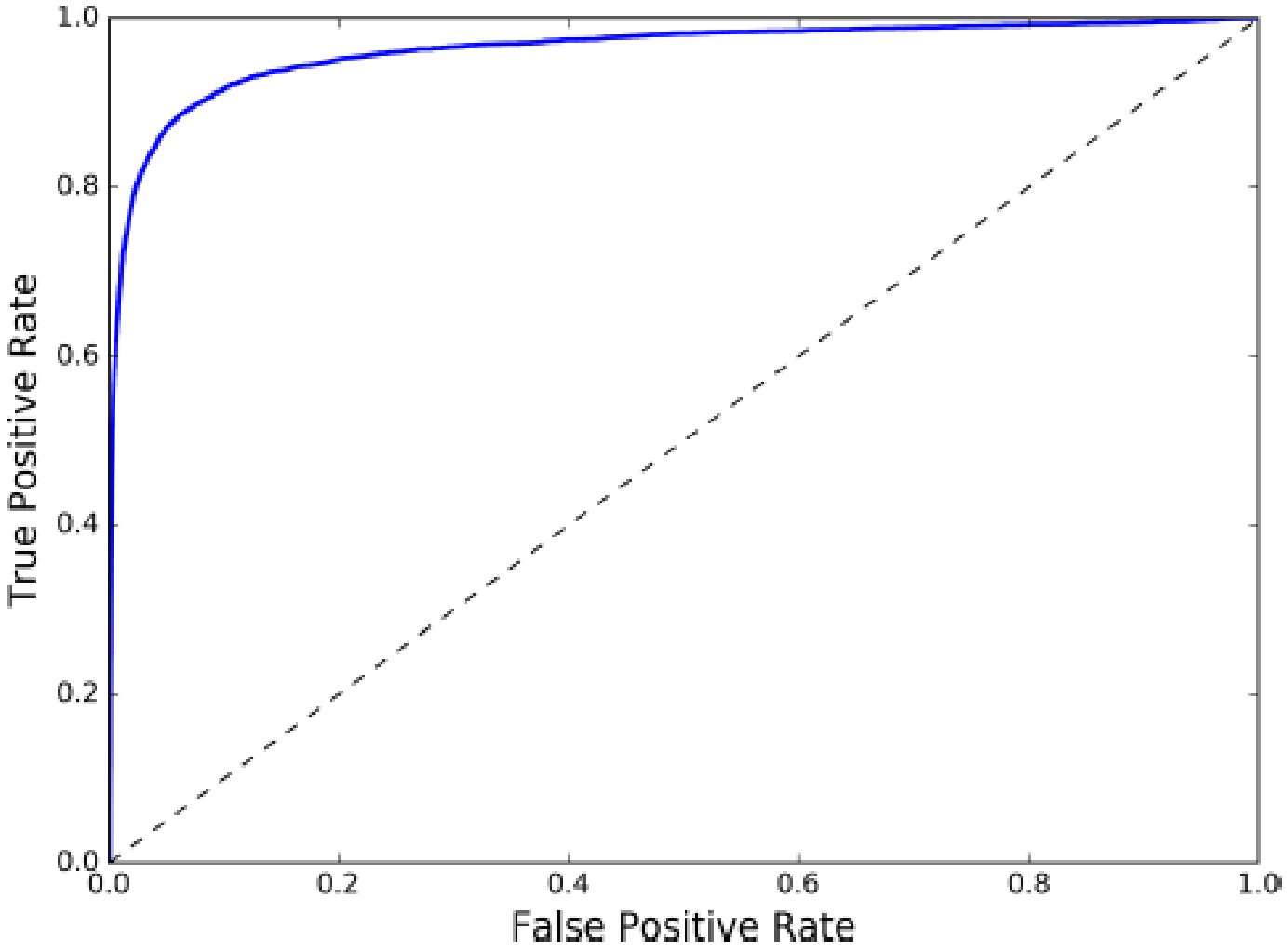

1.0
0.8
0.6
0.4
0.2
0.0
Precision
Recall
−600000
−400000
−200000
0
200000
400000
600000
Threshold

TradeOff



ROC曲线

- 0,0点
- 1,1点
- 0,1点



AUC面积

- 曲线下的面积
- ROC、AUC更看重正例
- 在正例少于负例的时候要结合PR曲线