Coursera 機器學習基石

November 17, 2015

1 Week 1

1.1 Perceptron Learning Algorithm (PLA)

简单的线性分类模型。输入 $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}$，每个 $\mathbf{x_i} = (x_1, x_2, \ldots, x_d)$ 维度为 d，对应 d 种特征。输出即类别 $\mathcal{Y} = \{-1, +1\}$。

考虑对每个特征赋一个权值 w_i，所有特征的权值构成权值向量 $\mathbf{w}$，对每个输入可以计算出总的分数，并根据分数是否超过某阈值 $threshold$ 来决定判定输出：

$$h(\mathbf{x}) = sign(\sum_{i=1}^{d} w_i x_i - threshold)$$

令 $w_0 = -threshold$，$x_0 = 1$，上式可简化为：

$$h(\mathbf{x}) = sign(\sum_{i=0}^{d} w_i x_i) = sign(\mathbf{w}^T \cdot \mathbf{x})$$

算法开始时初始化 $\mathbf{w} = \mathbf{w_0}$，然后基于训练数据不断对 $\mathbf{w}$ 做出修正，直到 $\mathbf{w}$ 将训练数据全部正确分类。

若在第 t 轮迭代中，PLA 发现对训练数据 $\mathbf{x}_{n(t)}$ 分类错误，即：

$$sign(\mathbf{w}_t^T \cdot \mathbf{x}_{n(t)}) \neq y_{n(t)}$$

则可以做出修正：

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_{n(t)}\mathbf{x}_{n(t)}$$

因为内积运算结果的符号由向量夹角的余弦表示，所以若发现分类结果与“标准答案”异号，则可以通过以上修正让 $\mathbf{w}_{t+1}$ 更加接近 $\mathbf{x}_{n(t)}$。

如此不断更新 $\mathbf{w}_t$ 直到其对所有训练数据都能正确分类。训练数据可以随机顺序访问，或按照某个预先设定的顺序循环访问。

1.2 PLA 的收敛性

为什么 PLA 不会一直运行下去?

设 $\mathbf{w}_f$ 为我们要学习的目标权重向量（未知）。如果数据线性可分，则必有：

$$\min_n y_n \mathbf{w}_f^T \mathbf{x} > 0$$

即，从训练数据中计算出来的最小总分和“参考答案”也必须是同号的。

同时最优的那个 $\mathbf{w}_f$ 要满足：

$$y_{n(t)}\mathbf{w}_f^T\mathbf{x}_{n(t)} \geq \min_n y_n\mathbf{w}_f^T\mathbf{x} > 0$$

联想一下点到直线的距离公式，最优的 $\mathbf{w}_f$ 到各个数据点应该具有一个最小的 margin。

首先证明 PLA 更新之后内积会逐渐变小，方法是比较更新后的权重向量和最优向量：

$$\begin{aligned}\mathbf{w}_f^T\mathbf{w}_{t+1} &= \mathbf{w}_f^T(\mathbf{w}_t + y_{n(t)}\mathbf{x}_{n(t)}) \\ &= \mathbf{w}_f^T\mathbf{w}_t + y_{n(t)}\mathbf{x}_f^T\mathbf{x}_{n(t)} \\ &\geq \mathbf{w}_f^T\mathbf{w}_t + \min_n y_n\mathbf{w}_f^T\mathbf{x}_n \\ &> \mathbf{w}_f^T\mathbf{w}\end{aligned} \tag{1}$$

接下来证明 PLA 每次更新的幅度不大，方法是估计更新后 $\mathbf{w}_{t+1}$ 的范数的上界。

注意到仅有出错时才更新。出错时 $y_{n(t)}\mathbf{w}_t^T\mathbf{x}_{n(t)} \leq 0$，即异号，所以有：

$$\begin{aligned}\|\mathbf{w}_{t+1}\|^2 &= \|\mathbf{w}_t + y_{n(t)}\mathbf{x}_{n(t)}\|^2 \\ &= \|\mathbf{w}_t\|^2 + 2y_{n(t)}\mathbf{w}_t^T\mathbf{x}_{n(t)} + \|y_{n(t)}\mathbf{x}_{n(t)}\|^2 \\ &\leq \|\mathbf{w}_t\|^2 + \|y_{n(t)}\mathbf{x}_{n(t)}\|^2 \\ &\leq \|\mathbf{w}_t\|^2 + \max_n \|y_n\mathbf{x}_n\|^2\end{aligned} \tag{2}$$

所以更新之后 $\|\mathbf{w}_{t+1}\|^2$ 最多比 $\|\mathbf{w}_t\|^2$ 增长 $\max_n \|y_n\mathbf{x}_n\|^2$ （最远的点）。

设 PLA 最多进行 T 次更新，由 **??** 得：

$$\frac{\mathbf{w}_f^T\mathbf{w}_T}{\|\mathbf{w}_f\|} \geq T \cdot \min_n \frac{y_n\mathbf{w}_f^T\mathbf{x}_n}{\|\mathbf{w}_f\|} = T\rho$$

由 **??** 得：

$$\|\mathbf{w}_T\|^2 \leq TR^2$$

综上，

$$1 \geq \frac{\mathbf{w}_f^T\mathbf{w}_T}{\|\mathbf{w}_f\| \cdot \|\mathbf{w}_T\|} \geq \frac{T\rho}{\sqrt{T}R} = \sqrt{T} \cdot \frac{\rho}{R}$$

可得更新次数的上界为

$$T \leq \frac{R^2}{\rho^2}$$

1.3 非线性可分问题

Pocket 在 PLA 基础上，维护几个量：

1. 当前最佳 $\mathbf{w}_t$
2. 当前连续正确分类的次数

3. 当前最佳连续分类次数

每次正确分类后，若连续正确分类次数超过历史最优，则计算当前 $\mathbf{w}_t$ 对所有训练数据的分类错误率，若犯错比历史值更少，则将当前 $\mathbf{w}_t$ 保存为 $\hat{\mathbf{w}}$。超过预设更新次数后，输出 $\hat{\mathbf{w}}$。

2 Week 2

2.1 Learning 是否可行?

实际的 hypothesis f 是永远无法获知的，无论如何设计学习算法，都有一些"个人考虑"在里面，不能达到最终的 f，例子：3×3 黑白方格分类问题。所以从 possibility 的角度讲，learning 是 impossible 的。但 impossible 并不代表 not probable。所以需要建立关于"learning is probable"的理论基础。

所谓算法"学到东西"，是指学习算法输出的 g 在测试数据上也要有较好的表现。那么，算法在有限的训练数据上的表现，能否提供训练数据之外的信息呢?

2.2 Bin model 和 Hoeffding 不等式

考虑装有红绿两色球的罐子，要估计罐子中红球的比例 μ，独立地随机抽 N 个球，通过样本中红球的比例 ν 来估计 μ。此模型满足 Hoeffding 不等式

$$\mathbb{P}[|\mu - \nu| > \epsilon] \leq 2e^{-2N\epsilon^2}$$

其实就是大数定理的一个形式，要满足较小的误差，就需要抽足够大的样本。Hoeffding 不等式即给出了"误差太大"这一坏事发生的概率上界。

将 bin model 联系到 learning 当中，对于任意一个确定的 h，对训练数据 $x_1, x_2, \ldots, x_N$ 依次计算 $h(x_n)$，$n = 1, 2, \ldots, x_N$，若 $h(x_n) \neq y_n$ 则将球涂成红色（出错），否则涂绿色，即可得到一个样本。如果训练数据和测试数据都由同一个分布 $P(\mathcal{X})$ 生成，即可根据 Hoeffding 不等式，用算法在训练数据上的错误率 ν 来估计其在测试数据上的错误率 μ。此时 ν 和 μ 分别称为 in-sample error 和 out-of-sample error，分别记为 E_{in} 和 E_{out}，即

$$\mathbb{P}[|E_{in} - E_{out}| > \epsilon] \leq 2e^{-2N\epsilon^2} \tag{3}$$

注意到这里应用 Hoeffding 不等式的前提是固定 h，所以不等式实际上是为 verification 提供了理论保证：对某个 h，E_{in} 和 E_{out} 差别不会太大。

但学习算法是在 $\mathcal{H}$ 中选定一个 g，所以我们希望把 Hoeffding 不等式提供的保障推广到 $|\mathcal{H}| = M$ 个 hypothesis 的情况。取 union bound，只要 $\mathcal{H}$ 中任意一个 h 可能发生"坏事"，则最后选出来的 g 也可能发生，即

$$\begin{aligned}
\mathbb{P}[|E_{in}(g) - E_{out}(g)| > \epsilon] &= \mathbb{P}[\bigvee_{1 \leq m \leq M} (E_{in}(h_m) - E_{out}(h_m) > \epsilon)] \\
&\leq \sum_{1 \leq m \leq M} \mathbb{P}[|E_{in}(h_m) - E_{out}(h_m)|] \\
&= \sum_{1 \leq m \leq M} 2e^{-2N\epsilon^2} \\
&= 2Me^{-2N\epsilon^2}
\end{aligned}$$

这个上界的特点：

1. 非常松
2. 依赖于模型的复杂程度 M：若要 $E_{in}(g)$ 接近 $E_{out}(g)$，需要 M 小，但太简单的模型往往不理想，即 $E_{in}(g)$ 会比较大
3. 对于 $M = +\infty$ 没有意义，但很多问题都有 $M = +\infty$，PLA 就是

3 Week 3

3.1 Error bar

现在我们知道

$$\mathbb{P}[|E_{in}(g) - E_{out}(g)| > \epsilon] \leq 2Me^{-2N\epsilon^2}$$

假设允许坏事发生的概率最大为 δ，即至少可以 $1-\delta$ 的概率保证 $|E_{in}(g) - E_{out}(g)| \leq \epsilon$，变形得：

$$E_{out}(g) \leq E_{in}(g) + \sqrt{\frac{1}{2N}\ln\frac{2M}{\delta}}$$

右边根号项称为 error bar，即将 g 从训练数据推广到测试数据产生的误差的界。因为带有 M，所以不能处理 $M = +\infty$ 的情况。

3.2 寻找更紧的上界

主要想法是缩小 $\mathcal{H}$ 的大小。以 PLA 为例，两条分类线如果斜率很接近，分类结果也会很接近，E_{in} 也会很接近。尝试找出 $\mathcal{H}$ 的“有效大小”，实际在 $M = +\infty$ 时也是有限值，从而形成一个上界。

Dichotomy 定义 $\mathcal{H}$ 在 $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N \in \mathcal{X}$ 上形成的 dichotomy 为：

$$\mathcal{H}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N) = \{\, (h(\mathbf{x}_1), h(\mathbf{x}_2), \ldots, h(\mathbf{x}_N)) \,|\, h \in \mathcal{H} \,\}$$

一个 dichotomy 即所有 h 在一些数据上可能形成的结果，表明 $\mathcal{H}$ 的“多样性”。

Shattered set 设 class C 和集合 A，如果对每个 $T \subset A$ 都存在 $U \subset C$ 使得 $U \cap A = T$，则称 C *shatter* A。即，当幂集 $P(A) = \{U \cap A \,|\, U \in C\}$ 时，C shatters A。

例如，二维 PLA 无法 shatter 平面上任意 4 个点，但可以 shatter 任意 3 个点。

Growth function 为了衡量一个 $\mathcal{H}$ shatter 数据集能力的能力，定义 $\mathcal{H}$ 的 growth function 为：

$$m_{\mathcal{H}}(N) = \max_{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N \in \mathcal{X}} |\mathcal{H}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)|$$

即这个 $\mathcal{H}$ 在任意 N 个点上能产生的最大 dichotomy 大小。对于二元分类问题，显然 $m_{\mathcal{H}}(N) \leq 2^N$。

$m_{\mathcal{H}}(N)$ 满足性质：

1. 若 $m_{\mathcal{H}}(N) = 2^N$，则表明存在大小为 N 的集合，能够被 $\mathcal{H}$ shatter，因为从中可以拿出集合去求交，得到幂集中所有可能的集合大小
2. 若对某个 $n > 1$ 有 $m_{\mathcal{H}}(m) < 2^m$，那么对所有 $n > m$ 也有 $m_{\mathcal{H}}(n) < 2^n$，因为如果 $\mathcal{H}$ 不能 shatter 某个大小 m，更大的集合中仍然有不能 shatter 的情况存在

Break point 若 $\mathcal{H}$ 不能 shatter 大小为 k 的数据集，则称 k 是 $\mathcal{H}$ 的一个 break point。由此可得，若 $\mathcal{H}$ 有 break point k，则 $m_{\mathcal{H}}(k) < 2^k$。

Growth function 的界 在 break point 存在的情况下，hypothesis 的有效数量会被极大地限制，所以要找到 growth function 的上界。而如果我们能用 growth function 取代 error bar 中的 M，那么，只要 growth function 有多项式的上界，error bar 右边取对数之后，不管多项式次数多少，都是按 N 对数增长，当 N 足够大的时候，E_{in} 和 E_{out} 就有可能非常接近。

设 $B(N,k)$ 为 N 个点中任意 k 个都不被 shatter 的最大 dichotomy 数量。如果 $m_{\mathcal{H}}(N)$ 有 break point k，则

$$m_{\mathcal{H}}(N) \leq B(N,k)$$

因为 growth function 衡量的是 $\mathcal{H}$ 的"划分"能力，而 $B(N,k)$ 是任意 N 个点的最大 dichotomy 数。

$B(N,k)$ 显然满足

$$\begin{aligned} B(N,1) &= 1 \\ B(1,k) &= 2 \qquad for \qquad k > 1 \end{aligned} \tag{4}$$

现在设 $N \geq 2$、$k \geq 2$，尝试找到 $B(N,k)$ 的递推关系。假设有 $B(N,k)$ 个 dichotomy，我们把这些 dichotomy 分为两类：

1. $\mathcal{H}_1 = \{\, (h(\mathbf{x}_1), h(\mathbf{x}_2), \ldots, h(\mathbf{x}_{N-1})) \,\}$，即 $\mathcal{H}$ 在前 $N-1$ 个点上产生的 dichotomy
2. $\mathcal{H}_2 = \{\, (y_1, y_2, \ldots, y_{N-1}) \in \mathcal{H}_1 \,|\, \exists h, h' \in \mathcal{H}\, s.t.\, h(\mathbf{x}_i) = h'(\mathbf{x}_i) = y_i, \forall i \in [1, N-1] \wedge h(\mathbf{x}_N) \neq h'(\mathbf{x}_N) \,\}$，即仅对 $\mathbf{x}_N$ 结果不同的所有 h 在前 $N-1$ 个点上产生的 dichotomy

则有

$$B(N,k) = |\mathcal{H}_1| + |\mathcal{H}_2|$$

由以上定义，设仅在 $\mathbf{x}_N$ 上不同的 hypothesis 产生的 dichotomy 数量为 2β，剩下 dichotomy 数量为 α，则

$$B(N,k) = \alpha + 2\beta$$

这 β 个 dichotomy 显然满足

$$\beta \leq B(N-1, k-1)$$

否则若加上 $\mathbf{x}_N$ 上的结果，就会 shatter $k+1$ 个点，不满足 $B(N,k)$ 的假设。

另外，因为 $B(N,k)$ 保证了没有任意 k 个点被 shatter，所以不考虑 $\mathbf{x}_N$，就应该有

$$\alpha + \beta \leq B(N-1, k)$$

以上两不等式相加，得：

$$B(N,k) = \alpha + \beta \leq B(N-1, k-1) + B(N-1, k)$$

在此基础上，有 Sauer 引理：

$$B(N,k) \leq \sum_{i=0}^{k-1} \binom{N}{i}$$

根据 $B(N,k)$ 的递推关系用数学归纳法即可证明，或参见这里。
由此可以得出 growth function 的上界

$$m_{\mathcal{H}}(N) \leq B(N,k) \leq \sum_{i=0}^{k-1} \binom{N}{i}$$

4 Week 4

4.1 VC Dimension

上面那个 $k-1$ 即 $\mathcal{H}$ 的 VC 维，记为 $d_{VC}(\mathcal{H})$，也就是 $\mathcal{H}$ 能够 shatter 的最大点数。若对所有 N 都有 $m_{\mathcal{H}}(N) = 2^N$，则 $d_{VC}(\mathcal{H}) = \infty$。

VC Generalization Bound 对任意 $\delta > 0$，下式以概率 $\geq 1-\delta$ 成立：

$$E_{out}(g) \leq E_{in}(g) + \sqrt{\frac{8}{N} \ln \frac{4m_{\mathcal{H}}(2N)}{\delta}}$$

即 Vapnik-Chervonenkis 不等式

$$\mathbb{P}\left[\sup_{h \in \mathcal{H}} |E_{in}(h) - E_{out}(h)| > \epsilon\right] \leq 4m_{\mathcal{H}}(2N) e^{-\frac{1}{8}N\epsilon^2}$$

注意几点：

1. $m_{\mathcal{H}}(N)$ 替代了原来的 M。因为对于某个 $\mathcal{D}$，很多 hypothesis 会共享一些 dichotomy，所以在计算 error 时，这些 hypothesis 要错一起错。Growth function 能做的就是计算这种重复。书里举了个例子，因为训练数据 $\mathcal{D}$ 是随机性的唯一来源，想象所有 $\mathcal{D}$ 的空间为一张白纸，某个 $\mathcal{D}$ 如果最后 $E_{in}(g)$ 和 $E_{out}(g)$ 相差很大，就把这个 $\mathcal{D}$ 对应的的区域涂色。而因为 dichotomy 会重叠，多个 $\mathcal{D}$ 的区域也会重叠，而之前 union bound 实际上是把这些重叠部分全部都当做了不相交，所以当 $\mathcal{H}$ 变大时，涂色区域会很快占满整张纸。而引入 VC 维之后的上界则能处理这种重叠的情况。

2. N 变为了 $2N$。因为 $|E_{in}(h) - E_{out}(g)|$ 不仅依赖于 $\mathcal{D}$，也依赖于 $\mathcal{X}$，所以引入了一个同样大小的样本 $\mathcal{D}'$，变为 $|E_{in}(h) - E'_{in}(h)|$。

3. VC 维很松，因为它不依赖于 $\mathcal{H}$、$\mathcal{P}$ 等，但好处是可以扩展到各种问题和 hypothesis set。

4. $\sqrt{\frac{8}{N}\ln\frac{4m_{\mathcal{H}}(2N)}{\delta}}$ 称为模型复杂度。模型越复杂，d_{VC} 越大，E_{in} 越小，但上界也越松，即 penalty of model complexity。

VC 不等式的证明思路:

1. 引入 ghost data。要 bound 的是 $|E_{in}-E_{out}|$，而 E_{out} 未知，所以再抽样一组和 $\mathcal{D}$ 独立同分布且大小相同的 $\mathcal{D}'$ 来近似 E_{out}，即用 $|E_{in}-E'_{in}|$ 来估计之
2. 对大小为 $2N$ 的数据 $\mathcal{D}$ 和 $\mathcal{D}'$ 使用 growth function，限制 hypothesis 数量。
3. 用 Hoeffding 的另一个定理估计上界。

这东西证明起来还挺复杂，看了几个版本，用 Chebyshev 不等式的版本严重伤害了我们这些低智商码农的感情。要好理解而不失严谨请看 *Learning From Data* 书附录。

4.2 交集和并集的 VC 维

交集

$$0 \le d_{VC}(\bigcap_{k=1}^{K}\mathcal{H}_k) \le min\{d_{VC}(\mathcal{H}_k)\}_{k=1}^{K}$$

并集

$$max\{d_{VC}(\mathcal{H}_k)\}_{k=1}^{K} \le d_{VC}(\bigcup_{k=1}^{K}\mathcal{H}_k) \le K-1+\sum_{k=1}^{K}d_{VC}(\mathcal{H}_k)$$

这个要证明一下。假设只有两个 hypothesis set $\mathcal{H}_1$ 和 $\mathcal{H}_2$，并集 $\mathcal{H}=\mathcal{H}_1\cap\mathcal{H}_2$，VC 维分别是 d_1、d_2，则显然两者的并集最多能 shatter 两者各自能 shatter 的那么多个点，设为 m。即

$$m_{\mathcal{H}}(m) \le m_{\mathcal{H}_1}(m)+m_{\mathcal{H}_2}(m)$$

根据 Sauer's Lemma，

$$m_{\mathcal{H}}(m) \le \sum_{i=0}^{d_1}\binom{m}{i}+\sum_{i=0}^{d_2}\binom{m}{i}$$

替换一下变量，

$$m_{\mathcal{H}}(m) \le \sum_{i=0}^{d_1}\binom{m}{i}+\sum_{i=0}^{d_2}\binom{m}{m-i}=\sum_{i=0}^{d_1}\binom{m}{i}+\sum_{i=m-d_2}^{m}\binom{m}{i}$$

当 $m-d_2>d_1+1$ 即 $m\ge d_1+d_2+2$ 时

$$m_{\mathcal{H}}(m) \le \sum_{i=0}^{d_1}\binom{m}{i}-\binom{m}{d_1+1}=2^m-\binom{m}{d_1+1}<2^m$$

也就是说并集的 VC 维应该小于 m，而 m 最小为 d_1+d_2+2，所以并集 VC 维最大只能取 d_1+d_2+1。

参考论文 http://www.davideisenstat.com/cv/EisenstatA07.pdf

4.3 估算样本数据规模

根据 VC 不等式，如果要以至少 $1-\delta$ 的 confidence 保证误差不超过 ϵ，则需要

$$\delta \leq 4m_{\mathcal{H}(2N)}e^{-\frac{1}{8}N\epsilon^2}$$

变形即得

$$N \geq \frac{8}{\epsilon^2}\ln\frac{4m_{\mathcal{H}}(2N)}{\delta}$$

如果用 growth function 的多项式上界 N_{VC}^d+1 来估计它，则可得

$$N \geq \frac{8}{\epsilon^2}\ln\frac{4((2N)^{d_{VC}}+1)}{\delta}$$

据此便可估计 N 的大小。

4.4 Noise & Error

$\mathcal{D}$ 有噪声时，相当于每个 label y 也是由一个概率分布产生的，即 $\mathbb{P}[y|\mathbf{x}]$。所以学习算法除了要找 target function f，还要决定每个 label 到底取什么值，也就是所谓的 mini-target。此时 y 不一定告诉你正确的 label，我们看得到的只有被噪声影响的 $f(\mathbf{x})$。y 不是被 $\mathbf{x}$ 决定（$f(\mathbf{x})$），而是受 $\mathbf{x}$ 影响（$\mathbb{P}[y|\mathbf{x}]$）。

Error measure 用来衡量我们找到的 hypothesis 所犯的错误多少。常用的 measure 有分类器常用的 0-1 measure $\mathbf{1}\{\tilde{y}\neq y\}$，和回归问题常用的平方误差 $(\tilde{y}-y)^2$。

Error measure 的选择会影响学习算法。例子：三类别分类器，用以上两种 measure 会得到不同的 label 值。

Error measure 也是与算法应用的具体场景相关的。例子：指纹识别，超市根据识别结果判断是否打折，CIA 判断门禁是否通过。两者需要的 measure 对不同的错判结果有不同的惩罚。设计算法时，可能无法知道这个最佳的 error measure，所以一般会用一个近似的 $\widehat{err}$。

衡量与 f 的差距仍然要用理想的 f。对 0-1 error 而言：

1. 衡量 h 和 f 的差距要用 $\mathbf{1}\{h(\mathbf{x})\neq f(\mathbf{x})\}$
2. $E_{out}(g)=\mathbb{E}_{\mathbf{x}}\left[\mathbf{1}\{f(\mathbf{x})\neq g(\mathbf{x})\}\right]$ 一定要跟 $f(\mathbf{x})$ 来比较

所谓 weighted classification 就是将模型犯错情况的惩罚考虑进去。例如 CIA 门禁的例子，错放了一个人问题很严重，所以需要加大模型犯这种错误的惩罚。在此情况下，PLA 无需改变，只要数据仍然线性可分即可。对 pocket 算法而言，weighted classification 可以转化为普通的 0-1 error 问题，方法是所谓的 virtual copying，即，把需要惩罚的错判数据复制很多份，再对这个扩大后的数据集执行修改版 pocket 算法：

1. 不用实际复制数据，而是在 pocket 评估 $\mathbf{w}$ 表现时，算入复制后的数据
2. 随机访问某个点 $\mathbf{x}$ 时，要将复制后的点的访问概率提高

5 Week 5

5.1 线性回归

线性回归要求输出某个实数值，而不是 label，可以用线性分类同样的“加权”方法来计算“分数”，从而得到 hypothesis：

$$h(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$$

使用平方误差 $(\hat{y} - y)^2$ 来测算 hypothesis 与实际数据的误差，得到 in- 和 out-of-sample error 的计算方法：

$$\begin{aligned} E_{in}(\mathbf{w}) &= \frac{1}{N}\sum_{n=1}^{N}(\mathbf{w}^T\mathbf{x}_n - y_n)^2 \\ E_{out}(\mathbf{w}) &= \mathbb{E}_{(\mathbf{x},y)}[(\mathbf{w}^T\mathbf{x} - y)^2] \end{aligned} \tag{5}$$

将 E_{in} 改写为矩阵形式即得

$$E_{in}(\mathbf{w}) = \frac{1}{N}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2$$

由于 E_{in} 凸、连续、可微（补充一下证明方法……），所以直接求令其梯度为 0 的 $\mathbf{w}$ 即可：

$$\begin{aligned} E_{in}(\mathbf{w}) &= \frac{1}{N}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 \\ &= \frac{1}{N}\left(\mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w} - 2\mathbf{w}^T\mathbf{X}^T\mathbf{y} + \mathbf{y}^T\mathbf{y}\right) \\ \nabla E_{in}(\mathbf{w}) &= \frac{2}{N}(\mathbf{X}^T\mathbf{X}\mathbf{w} - \mathbf{X}^T\mathbf{y}) \end{aligned} \tag{6}$$

令 $\nabla E_{in}(\mathbf{w}) = 0$ 可得

$$\mathbf{w}_{lin} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \mathbf{X}^\dagger\mathbf{y}$$

这里大部分情况下 $\mathbf{X}^T\mathbf{X}$ 是可逆的，如果不可逆，可以通过其他方式定义 $\mathbf{X}^\dagger$，最好用现有的线性代数库来实现。

5.2 线性回归的可靠性

线性回归有 closed form，考虑 E_{in} 和 E_{out} 的期望。此处假设每个样本点的噪声都服从 $\epsilon \sim \mathcal{N}(0, \sigma^2)$

首先计算 $\mathbb{E}[E_{in}(\mathbf{w})]$。具体可见 *Learning From Data* 练习 3.3 和 3.4，得到 $E_{in}(\mathbf{w}) = \frac{1}{N}\epsilon^T(\mathbf{I} - \mathbf{H})\epsilon$。其中 $\mathbf{H}$ 为 hat matrix $\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$，对称、幂等，用于将 $\mathbf{y}$ 变为 $\hat{\mathbf{y}}$。

故

$$
\begin{aligned}
\mathbb{E}[E_{in}(\mathbf{w})] &= \frac{1}{N}\mathbb{E}[\epsilon^T\epsilon] - \frac{1}{N}\mathbb{E}[\epsilon^T\mathbf{H}\epsilon] \\
&= \sigma^2 - \frac{1}{N}\mathbb{E}[\sum_{i=1}^{N}\epsilon_i^2 h_{ii} + \sum_{i\neq j}\epsilon_i\epsilon_j h_{ij}] \\
&= \sigma^2 - \frac{1}{N}(\sum_{i=1}^{N}(\mathbb{E}[\epsilon_i^2]\mathbb{E}[h_{ii}]) + \sum_{i\neq j}\mathbb{E}[\epsilon_i]\mathbb{E}[\epsilon_j]\mathbb{E}[h_{ij}]) \\
&= \sigma^2 - \frac{1}{N}(\sigma^2\mathbb{E}[\sum_{i=1}^{N}h_{ii}] + 0) \\
&= \sigma^2 - \frac{1}{N}(\sigma^2 trace(H)) \\
&= \sigma^2 - \frac{1}{N}(\sigma^2(d+1)) \\
&= \sigma^2(1 - \frac{d+1}{N})
\end{aligned}
\tag{7}
$$

同时

$$
\mathbb{E}[E_{out}(\mathbf{w})] = \sigma^2(1 + \frac{d+1}{N})
$$

可见当 N 变大时，E_{in} 和 E_{out} 逐渐趋近 σ^2 即 noise level，故 generalization error 的期望值为 $\frac{2(d+1)}{N}$。

5.3 与线性分类的比较

注意到，平方错误 $(\mathbf{w}^T\mathbf{x} - y)^2$ 是 0-1 错误 $\mathbf{1}\{sign(\mathbf{w}^T\mathbf{x}) \neq y\}$ 的上界，所以在 generalization 时，线性回归算法得到的 $\mathbf{w}$ 会有更松的上界。如果我们直接对二元分类的问题执行线性回归，然后返回 $sign(\mathbf{w}_{lin})$ 也是可行的，但相当于用 generalization bound 去换了执行效率。

$\mathbf{w}_{lin}$ 可以作为线性分类的初始 $\mathbf{w}$，从一个较好的 $\mathbf{w}$ 开始迭代，缩短分类算法的运行时间。

5.4 Logistic 回归 & 梯度下降法

分类问题的 mini-target 是 $f(\mathbf{x}) = \mathbb{P}[+1|\mathbf{x}] \in [0,1]$，但有些问题需要在给定 0-1 label 的基础上直接输出 $\mathbb{P}[+1|\mathbf{x}]$。比如，要预测人患病的概率，我们手里的数据只有某个人是否得病。

所以需要把线性模型中的“加权分值”换算成一个 $[0,1]$ 上的值，例如 logistic 函数

$$
\theta(s) = \frac{1}{1+e^{-s}}
$$

此函数光滑、单调，满足 $\lim_{s\to-\infty}\theta(s) = 0$，$\lim_{s\to+\infty}\theta(s) = 1$，$\theta(0) = \frac{1}{2}$。将其作用到加权分值上即得：

$$
h(\mathbf{x}) = \theta(\mathbf{w}^T\mathbf{x})
$$

我们要学习的目标为 $f(\mathbf{x}) = \mathbb{P}[+1|\mathbf{x}]$，即在给定数据 $\mathbf{x}$ 的基础上，$+1$（或 -1）出现的概率。由于样本中并没有概率，而只有结果，所以相当于样本产生于一个有噪声的目标 $\mathcal{P}(y|\mathbf{x})$：

$$\mathcal{P}(y|\mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{for } y = +1 \\ 1 - f(\mathbf{x}) & \text{for } y = -1 \end{cases} \tag{8}$$

在此基础上，相当于用 logistic 函数输出的概率 $h(\mathbf{x})$ 来“匹配”训练数据：

$$\mathcal{P}(y|\mathbf{x}) = \begin{cases} h(\mathbf{x}) & \text{for } y = +1 \\ 1 - h(\mathbf{x}) & \text{for } y = -1 \end{cases} \tag{9}$$

注意到 logistic 函数满足 $h(-x) = 1 - h(x)$，所以，综合以上两种情况，即可得 $\mathcal{P}(y|\mathbf{x}) = h(y\mathbf{x}) = \theta(y\mathbf{w}^T\mathbf{x})$。于是可以用似然函数来衡量 hypothesis 与样本的相似程度：

$$\mathcal{L}(\mathbf{w}) = \prod_{n=1}^{N} \mathcal{P}(y_n|\mathbf{x}_n) = \prod_{n=1}^{N} \theta(y_n\mathbf{w}^T\mathbf{x}_n)$$

最大化似然函数等价于最小化：

$$-\frac{1}{N}\ln\left(\prod_{n=1}^{N}\mathcal{P}(y_n|\mathbf{x}_n)\right) = \frac{1}{N}\sum_{n=1}^{N}\ln\frac{1}{\mathcal{P}(y_n|\mathbf{x}_n)}$$

最小化这个式子相当于我们把它看做 error measure，即：

$$E_{in}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\ln(1 + e^{-y_n\mathbf{w}^T\mathbf{x}_n})$$

这里求和的 error measure $err(\mathbf{w}, \mathbf{x}, y) = \ln(1 + e^{-y\mathbf{w}^T\mathbf{x}})$ 称为 cross entropy error，是一个 pointwise 的 error measure。对它求梯度，可得

$$\nabla E_{in}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}(-y_n\mathbf{x}_n)\theta(-y_n\mathbf{w}^T\mathbf{x}_n)$$

注意到，这里无法直接令梯度等于 0 来获得一个 analytical solution，但 $E_{in}(\mathbf{w})$ 函数连续、一二阶可微、凸，可用梯度下降法（gradient descent）迭代求解。

梯度下降法类似于 PLA 的迭代过程，每一步对 $\mathbf{w}$ 做适当更新，使 $\mathbf{w}$ 沿着 $E_{in}(\mathbf{w})$ 减小最快（负梯度）方向逐步下降，直至找到局部最小。

假设每次更新方式为

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta\hat{\mathbf{v}}$$

其中 $\eta > 0$ 为 learning rate，$\hat{\mathbf{v}}$ 为更新方向的单位向量。

更新的目标是让每次更新后 $E_{in}(\mathbf{w})$ 最小。如果 η 足够小，在更新的局部，可以用 Taylor 展开来近似原函数 $E_{in}(\mathbf{w})$，即，在 $\mathbf{w}_t$ 处展开 $E_{in}(\mathbf{w})$：

$$\begin{aligned} E_{in}(\mathbf{w}) &= E_{in}(\mathbf{w}_t) + \nabla E_{in}(\mathbf{w}_t)^T(\mathbf{w} - \mathbf{w}_t) + O((\mathbf{w} - \mathbf{w}_t)^2) \\ E_{in}(\mathbf{w}_t + \eta\hat{\mathbf{v}}) &= E_{in}(\mathbf{w}_t) + \eta\hat{\mathbf{v}}^T\nabla E_{in}(\mathbf{w}_t) + O(\eta^2) \\ &\geq E_{in}(\mathbf{w}_t) - \eta\|\nabla E_{in}(\mathbf{w}_t)\| \end{aligned} \tag{10}$$

最后一步等号成立仅当 $\hat{\mathbf{v}}$ 和 $\nabla E_{in}(\mathbf{w}_t)$ 反向。所以更新方向为

$$\hat{\mathbf{v}} = -\frac{\nabla E_{in}(\mathbf{w}_t)}{\|\nabla E_{in}(\mathbf{w}_t)\|}$$

对于 η，如果固定取值，则可能因梯度大小不同而引起麻烦，比如函数减小很剧烈时，若 η 太小，则可能速度很慢，反之则可能让 $E_{in}(\mathbf{w})$ 不稳定。所以希望根据梯度来动态地决定 η 取值。直观上看，让 $\eta \propto \|\nabla E_{in}(\mathbf{w}_t)\|$ 比较好。此时

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta \frac{\nabla E_{in}(\mathbf{w}_t)}{\|\nabla E_{in}(\mathbf{w}_t)\|}$$

即相当于

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta \nabla E_{in}(\mathbf{w}_t)$$

这个新的 η 就是原来的 $\dfrac{\eta}{\|\nabla E_{in}(\mathbf{w}_t)\|}$，称为 fixed learning rate gradient descent。

综合以上步骤即可得到 Logistic 回归的具体操作：

1. 初始化 $\mathbf{w}_0$
2. 计算 $\nabla E_{in}(\mathbf{w}_t) = \dfrac{1}{N}\sum_{n=1}^{N}(-y_n\mathbf{x}_n)\theta(-y_n\mathbf{w}_t^T\mathbf{x}_n)$
3. 更新 $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta\nabla E_{in}(\mathbf{w}_t)$
4. 重复迭代直到 $\nabla E_{in}(\mathbf{w}_{t+1}) = 0$ 或足够小
5. 返回最终的 $\mathbf{w}_{t+1}$ 作为 g

从中提取出梯度下降法的一般情况，对于 $F(\mathbf{x})$，用 Taylor 公式在点 $\mathbf{a}$ 处展开得

$$F(\mathbf{x}) = F(\mathbf{a}) + \nabla F(\mathbf{a})(\mathbf{x}-\mathbf{a}) + O((\mathbf{x}-\mathbf{a})^2)$$

对 $\mathbf{a}$ 附近的 $\mathbf{b} = \mathbf{a} + \eta\mathbf{v}$，其中 $\eta > 1$ 且 $\|\mathbf{v}\| = 1$，有

$$\begin{aligned} F(\mathbf{b}) &= F(\mathbf{a}) + \eta\mathbf{v}^T\nabla F(\mathbf{a}) + O(\eta^2) \\ &\leq F(\mathbf{a}) - \eta\|\nabla F(\mathbf{a})\| \end{aligned} \tag{11}$$

即，仅当 $\mathbf{v}$ 取负梯度方向时等号成立。所以

$$F(\mathbf{a}) - F(\mathbf{b}) = \eta\|\nabla F(\mathbf{a})\| \geq 0$$

也就是说沿着负梯度方向走，有 $F(\mathbf{a}) \geq F(\mathbf{b})$。

6 Week 6

6.1 线性模型 vs 线性分类

线性回归和 Logistic 回归都可以用来分类，需要考察一下这样做的好坏。

设 $s = \mathbf{w}^T\mathbf{x}$，则有：

1. 线性分类：0-1 错误 $err_{0-1}(s, y) = \mathbf{1}sign(ys) \neq 1$
2. 线性回归：平方误差 $err_{SQR}(s, y) = (ys - 1)^2$
3. Logistic 回归：cross entropy $err_{CE}(s, y) = \ln(1 + e^{-ys})$

将 Logistic 回归的 error measure 换底改写为 scaled cross entropy $err_{SCE}(s, y) = \log_2(1 + e^{-ys})$，然后把三个 ys 的函数画图，可以看出：

1. 平方误差和 scaled cross entropy 都是 0-1 错误的上界
2. 平方误差相当于在线性回归中把输出目标定为 1，所以抛物线顶点为 1，越往两侧 penalty 越大，与 0-1 错误相差也越大
3. Scaled cross entropy 在 $ys \ll 0$ 时 penalty 很大，与 0-1 错误相差也越大

于是可以得到

$$\begin{aligned} E_{in}^{0-1}(\mathbf{w}) &\leq E_{in}^{SCE}(\mathbf{w}) = \frac{1}{\ln 2} E_{in}^{CE}(\mathbf{w}) \\ E_{out}^{0-1}(\mathbf{w}) &\leq E_{out}^{SCE}(\mathbf{w}) = \frac{1}{\ln 2} E_{out}^{CE}(\mathbf{w}) \end{aligned} \tag{12}$$

以上两个 $\leq$ 成立是因为线性分类和 Logistic 回归的 E_{in} 都是基于 point-wise error 对所有样本求平均。

有了上面这些关系，套用到 VC 不等式即可知，两个回归模型也可以用于分类，但相应的 generalization bound 更宽松。

相比于线性分类，两个回归模型算法运行更简单，可以用来初始化 $\mathbf{w}_0$。

6.2 随机梯度下降

前面说了 Logistic 回归中使用梯度下降法，在每轮迭代中需要计算 $\nabla E_{in}(\mathbf{w})$，以更新 $\mathbf{w}$，而这个计算需要 $O(N)$ 时间遍历所有训练数据，我们希望像 PLA 那样，让每次更新只依赖于一个点，即只用 $O(1)$ 时间。

随机梯度下降法（stochastic gradient descent，SGD）随机选取一个点，只用这个点对原来梯度的贡献来估计原来的梯度，即更新步骤改为

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

这样做相当于是把“随机梯度方向”看做“真实梯度方向”加上 zero-mean 的“噪声方向”（补充下理论证明），好处当然是简化了计算，尤其是对训练数据量很大以及 online learning 的情况适用。但停止条件不好找，实践上一般是让算法执行足够长的时间。（估计这个跟 SGD 的随机性有关?）

上面这个更新跟 PLA 很像，把 Logistic 函数换成 $\mathbf{1}\{y_n \neq sign(\mathbf{w}_t^T\mathbf{x}_n)\}$ 即得到 PLA 的更新操作。

6.3 多类别分类

One-Versus-All（OVA） 或称 one-versus-rest。有多个类别时，直接用 hard classifier 来输出 +1/-1 label 可能产生歧义，所以通常是用 soft classifier 输出一个概率，即认为某个点是某个类别的 confidence，然后再汇总。

OVA 对于训练数据中的 K 个类别，用 Logistic 回归训练 K 个分类器，每个分类器使用的训练数据为

$$\mathcal{D}_k = \{(\mathbf{x}_n, y'_n = 2 \times \mathbf{1}\{y_n = k\} - 1)\}_{n=1}^N$$

对于每个数据点，选取这 K 个分类器中 confidence 最大的那个 label:

$$g(\mathbf{x}) = \operatorname*{argmax}_{k \in \mathcal{Y}} \theta(\mathbf{w}_k^T \mathbf{x})$$

上式中，因为 Logistic 函数是单调的，所以可以直接比大小，不用计算一次函数值。

OVA 的好处是简单，而且可以把 Logistic 回归换成任何类似的算法。坏处是当 K 很大时，一般 -1 会比 +1 多很多，所以有可能最后输出的 -1 而非 +1。

One-Versus-One（OVO） 顾名思义，每次在所有 label 中选出一对来训练二元分类器，故一共要训练 $\binom{N}{2}$ 个分类器。每个分类器使用的训练数据为

$$\mathcal{D}_{k,l} = \{(\mathbf{x}_n, y'_n = y_n) \quad | \quad y = k \vee y = l\}_{n=1}^N$$

然后用每个分类器的结果对数据点进行投票，票数最高的 label 作为该数据点的 label。

OVO 对于类别数 K 较小的问题比较有效，而且可以搭配任何二元分类算法。缺点是需要花 $O(K^2)$ 空间来保存这些分类器的结果，而且需要的训练量更大。

6.4 非线性变换

非线性变换通过变换特征到 $\mathcal{Z}$-space，使非线性模型转换为线性模型。例子：以原点为圆心的圆形 PLA。

一种典型变换:

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

包含了所有原始特征的 0、1、2 次多项式组合。

用非线性变换加线性模型即可实现非线性模型，方法是先变换特征，然后在 $\mathcal{Z}$-space 中用线性模型解决问题，最后把结果对应到原来的数据。

特征变换是常用的手段，尤其是当手上只有 raw feature 时，需要转换而成为具有明确意义的 concrete feature。

非线性变换可能代价高昂。考虑 $\Phi_Q(\mathbf{x})$，将 d 维数据变换为 Q 次多项式:

$$\Phi_Q(\mathbf{x}) = (1, x_1, \ldots, x_d, x_1^2, \ldots, x_d^Q)$$

变换后数据的维度为 $\binom{Q+d}{Q}$，即 $O(Q^d)$。设变换后维度为 $1 + \tilde{d}$，则相当于训练一个 $\tilde{d}$ 维度的线性模型，其 VC 维也会很大。模型复杂化之后带来的问题是 E_{in} 小了，但 generalize 后 E_{in} 和 E_{out} 差距会较大。

选择模型时要避免“窥探”数据，基于数据特征来选择模型，得到的最终 hypothesis 可能 generalization 表现不好，不要对数据做人肉的推断。

将非线性变换推广到所有的多项式变换，递归地定义如下：

$$\begin{aligned}\Phi_0(\mathbf{x}) &= (1)\\ \Phi_1(\mathbf{x}) &= (\Phi_0(\mathbf{x}), x_1, \ldots, x_d)\\ &\cdots\\ \Phi_Q(\mathbf{x}) &= (\Phi_{Q-1}(\mathbf{x}), x_1^Q, x_1^{Q-1}x_2, \ldots, x_d^Q)\end{aligned} \tag{13}$$

它们对应的 hypothesis set 显然满足

$$\mathcal{H}_{\Phi_0} \subset \mathcal{H}_{\Phi_1} \subset \cdots \subset \mathcal{H}_{\Phi_Q}$$

所以 VC 维满足

$$d_{VC}(\mathcal{H}_{\Phi_0}) \leq d_{VC}(\mathcal{H}_{\Phi_1}) \leq \cdots \leq d_{VC}(\mathcal{H}_{\Phi_Q})$$

In-sample error 满足

$$E_{in}(g_0) \geq E_{in}(g_1) \geq \cdots \geq E_{in}(g_Q)$$

因为 generalization 的问题存在，所以并不是直接用高次的模型就好。一般是从简单有效的模型入手，如果 E_{in} 不理想再逐步提升模型复杂度。

7 Week 7

7.1 Overfitting

Overfitting 即在降低 E_{in} 的过程中增加了 E_{out}。Underfitting 是 E_{in} 本身就比较大，E_{out} 也较高。回忆两个原则：降低 E_{in}，保持 E_{in} 和 E_{out} 接近。

可能造成 overfitting 的因素：

1. 训练数据少，未展示足够的 pattern
2. Stochastic noise 太大，干扰了数据本身的 pattern，模型会花力气去模拟噪声
3. Deterministic noise 太大，即目标太复杂，当前的 $\mathcal{H}$ 竭尽全力也无法模拟 f，影响效果相当于噪声
4. Hypothesis 太强，overkill，比如用 10 次多项式拟合 2 次曲线上的点

针对训练数据和噪声的一些应对措施：

1. Data cleaning 修改一些错标的数据
2. Data pruning 删掉错误数据
3. Data hinting 基于已有数据生成一些新的数据作为补充，但需要注意新增数据要满足一定的随机性，即产生于 $P(\mathbf{x}|y)$ 否则会影响 generalization。

7.2 Regularization

Regularization 即对模型施加一些限制，来减小 overfitting 的可能性。

例子：线性回归 $\mathcal{H}_2$ 可以看做 $\mathcal{H}_{10}$ 的特例，即规定高次权重为 0，所以低次多项式的 hypothesis set 是高次的子集，VC 维也更小。这种次数上的限制是一种 hard constraint。通常使用的是 soft constraint，例如限制次数的平方和在一定范围内 $\sum_{q=0}^{10} w_q^2 \leq C$，这样形成的 hypothesis set 在取不同的 C 时可能会有交集，但总的来说，C 越大时限制越宽松。此时的 hypothesis set 称为 regularized hypothesis set $\mathcal{H}(C)$

对回归问题而言，现在的优化问题变为，在有限制的条件下最小化 E_{in}。写成矩阵形式即

$$\min_{\mathbf{w}\in\mathbb{R}^{d+1}} E_{in}(\mathbf{w}) = \frac{1}{N}(\mathbf{Z}\mathbf{w}-\mathbf{y})^T(\mathbf{Z}\mathbf{w}-\mathbf{y}) \quad \text{s.t.} \quad \mathbf{w}^T\mathbf{w} \leq C$$

有约束时，若最优解在约束范围内，则与原问题一致；若最优解在约束范围外，显然最优解应取在约束区域边缘，否则继续沿负梯度方向往边缘走会得到更优的解。所以只考虑 $\mathbf{w}^T\mathbf{w} \leq C$ 即可。根据 Lagrange multiplier，只要沿着约束曲线前进的速度方向在负梯度方向有分量，就能得到更优的解，所以终止条件是速度方向与负梯度方向垂直，即当前 $\mathbf{w}$ 的法线方向与梯度平行，即 $-\nabla E_{in}(\mathbf{w}_{REG}) \propto \mathbf{w}_{REG}$，因为此时约束为一个球，球面任意一点的法线即 $\mathbf{w}$。也就是说，只需要找到 Lagrange multiplier $\lambda > 0$ 和 $\mathbf{w}_{REG}$ 使得

$$\nabla E_{in}(\mathbf{w}_{REG}) + \frac{2\lambda}{N}\mathbf{w}_{REG} = 0$$

两边积分，也就相当于求函数

$$E_{in}(\mathbf{w}_{REG}) + \frac{\lambda}{N}\mathbf{w}^T\mathbf{w}$$

的最小值。这个函数称为 augmented error E_{aug}。有约束 C 时的优化问题相当于这个无约束的优化问题。这里 $\lambda > 0$ 是因为我们要限制 $\mathbf{w}$ 且最小化 E_{in}，若取 $\lambda < 0$ 则无法得到较小的 $\mathbf{w}$。

如果是线性回归问题，则

$$\nabla E_{in}(\mathbf{w}_{REG}) + \frac{2\lambda}{N}\mathbf{w}_{REG} = 0$$

即

$$\frac{2}{N}(\mathbf{Z}^T\mathbf{Z}\mathbf{w}_{REG} - \mathbf{Z}^T\mathbf{y}) + \frac{2\lambda}{N}\mathbf{w}_{REG} = 0$$

最优解为

$$\mathbf{w}_{REG} = (\mathbf{Z}^T\mathbf{Z} + \lambda\mathbf{I})^{-1}\mathbf{Z}^T\mathbf{y}$$

也称为 ridge regression。Regularizer $\frac{\lambda}{N}\mathbf{w}^T\mathbf{w}$ 称为 weight-decay regularizer，因为它会限制 $\mathbf{w}$ 不要增长太大。

7.3 与 VC 不等式的联系

VC 不等式

$$E_{out}(\mathbf{w}) \leq E_{in}(\mathbf{w}) + \Omega(\mathcal{H}(C))$$

保证了在有约束 C 时 E_{out} 的上界，这里的 Ω 项是对整个 hypothesis set 而言，惩罚较复杂的 hypothesis set。

而使用 augmented error 时，

$$E_{aug}(\mathbf{w}) = E_{in}(\mathbf{w}) + \frac{\lambda}{N}\mathbf{w}^T\mathbf{w}$$

直观地讲，如果对某个具体 h 而言，惩罚项 $\frac{\lambda}{N}\Omega(\mathbf{w})$ 是一个对其复杂度加以限制的方式，那么它很有可能对整个 hypothesis set 也能起到限制作用。也就是说，模型的 effective VC dimension 减小了。所以最小化 augmented error 有可能限制 E_{out} 从而达到更好的 generalization。

这里提出 N，因为训练数据量越大时，overfitting 的可能性就越小，所以也就越不需要 regularization。

λ 和惩罚项的选择取决于 target，但 target 通常是未知的。与 error measure 的选择一样，可以根据具体问题和优化的需要，选择适当的 λ 和惩罚项。

8 Week 8

8.1 模型选择

在有很多模型可选的情况下，希望从一堆模型里面选出具有最小 E_{out} 的，但显然 E_{out} 无法得知。而且也不能用 E_{in}，因为用 E_{in} 的话复杂的模型必然会完胜简单模型，容易导致 overfitting。所以我们需要的是一组与 $\mathcal{D}$ 同分布产生的测试数据。但一般不容易再获取到这样的数据。

所以可以取一种折中的办法，将训练数据分成两部分，一部分做训练，另一部分，称为 validation set 作为测试，来评估各种模型的好坏，称为 validation。

具体做法是从 $\mathcal{D}$ 中随机选择 K 组数据作为 $\mathcal{D}_{val}$。用剩下的数据训练出 g_m^-，VC 不等式保证了 $E_{out}(g_m^-)$ 会受 $E_{val}(g_m^-)$ 的约束。但此时训练数据量变小了，所以 $E_{out}(g_m^-)$ 应该比实际最好的 $E_{out}(g_m^*)$ 大。实践上通过 validation 选择出最佳模型后，会再将这个模型作用于整个 $\mathcal{D}$ 再训练一遍，最终输出这最后一个训练结果。

Validation set 的选择要慎重，选多了训练数据量太小，会使得 $E_{out}(g_m^-)$ 不能很好地近似 E_{out}。选少了不能很好地评估模型的好坏。

经验值 $K = \frac{N}{5}$。

8.2 Leave-One-Out Validation

做法是取 $K = 1$，每次从 N 组数据中选一组作为 validation set，用剩下的 $N-1$ 组数据训练模型，然后做 validation 得到一个 e_n。如此重复，对每一组数据都做一遍 validation，再用所有的 e_n 取平均，作为对 $E_{out}(g)$ 的估计，此过程称为 cross validation，因为数据会交替地作为训练和验证：

$$E_{loocv}(\mathcal{H}, \mathcal{A}) = \frac{1}{N}\sum_{n=1}^{N} e_n$$

最后取 E_{loocv} 最小的模型即可。
为什么这样做有效？可以取 $E_{loocv}(\mathcal{H},\mathcal{A})$ 的期望：

$$\begin{aligned}
\mathbb{E}_{\mathcal{D}}[E_{loocv((H,A))}] &= \mathbb{E}_{\mathcal{D}}[\frac{1}{N}\sum_{n=1}^{N} e_n] \\
&= \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}_{\mathcal{D}}[e_n] \\
&= \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}_{\mathcal{D}_n}[\mathbb{E}_{(\mathbf{x}_n,y_n)}[err(g_n^-(\mathbf{x}_n),y_n)]] \\
&= \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}_{\mathcal{D}_n}[E_{out}(g_n^-)] \\
&= \frac{1}{N}\sum_{n=1}^{N}\bar{E}_{out}(N-1) \\
&= \bar{E}_{out}(N-1)
\end{aligned} \tag{14}$$

其中第三步是将每次隔出来的 validation set 与训练数据分开。第四步是相当于在全部 $(\mathbf{x}_n,y_n)$ 上面取误差的期望，即 E_{out}。第五步是在全部 $N-1$ 组数据组成的数据集上取期望，即 $\bar{E}_{out}(N-1)$。于是可得，用 leave-one-out cross validation 来评估模型，可以在期望上接近模型在 $N-1$ 个数据上的 E_{out}。

此方法缺点是计算量太大，每个模型要额外训练很多次。除非是线性回归这种有 analytic solution 的模型，否则不太实用。

8.3 V-Fold Cross Validation

前一种方法的改进，目的是减小计算量。方法是将 $\mathcal{D}$ 分成 V 等分，然后再在这 V 等分上做 cross validation，得到

$$E_{CV}(\mathcal{H},\mathcal{A}) = \frac{1}{V}\sum_{v=1}^{V} E_{val}^{(v)}(g_v^-)$$

然后根据这个结果来选择合适的模型。
经验值 $V=10$。

8.4 总结

1. V-fold 一般比 single validation 效果好
2. 通常不需要做 leave-one-out validation
3. Validation 仍然比实际测试要乐观，也就是说即使 validation 告诉你某个模型很好，实际测试也有可能出乱子。Testing 仍然要以实际测试数据为准。

8.5 学习准则

Occam's Razor 奥卡姆剃刀：如无必要，勿增实体（Entia non sunt multiplicanda praeter necessitatem）

简单即使美。简单的模型有更好的 significance。也就是说，如果这个模型在某个数据集上表现很好，你可以说服自己它确实反映了数据的某些规律。而复杂的模型很可能即使是喂给它完全随机的数据也能表现很好，但实际上数据中并没有什么规律可言。

Sampling Bias 杜鲁门 vs 杜威的例子，电话民调选中的大都是有钱人，预测结果当然不准确。

实际上就是得到训练数据和测试数据的分布不一致，得不到 VC 理论的保障。

实践准则是，尽可能地模拟实际的测试场景。例如，如果知道测试场景会倾向于某些数据，则在训练时调整参数，使模型更偏向它们，验证时也用类似于测试场景的数据。

Data Snooping 为了 VC 理论的安全请不要以任何形式偷看数据，避免 data-driven learning。