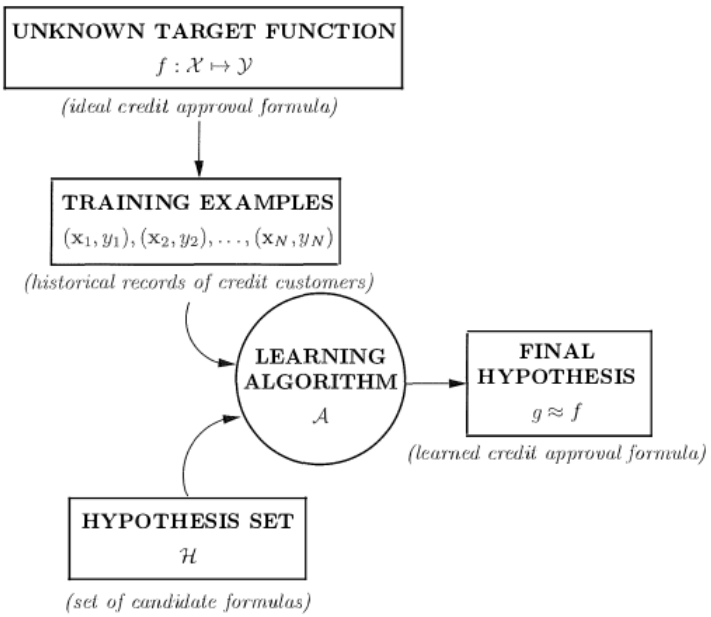
Resumo A1

Matheus Popst e Vitória Guardieiro Inferência e Aprendizagem

16 de abril de 2019

1. The learning problem

O framework mais simples de aprendizado é:



O exemplo mais importante de conjunto de hipóteses $\mathcal{H}$ é o **perceptron**.

- **Unknown target function:** $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = -1, 1$, como espaço de saída, sim ou não.
- **Hypothesis set** Cada $h(\mathbf{x}) \in \mathcal{H}$ é tal que escolhemos diferentes valores para multiplicar cada dimensão de $\mathbf{x}$. Também escolhemos um limiar. Para no futuro não precisar de limiar, costuma-se definir $\mathcal{X} = \{1\} \times \mathbb{R}^d$, de tal forma que apenas escolhemos um peso da dimensão 0.
- A fórmula compacta fica

$$h(\mathbf{x}) = \text{sinal}\left(\sum_{i=0}^{n} w_i x_i\right)$$

- Quão mais importante uma dimensão x_i (se a pessoa tem o nome no Serasa, na análise de crédito), maior será w_i.
- Com a notação simplificada temos que

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

O PLA (Perceptron Aprendendo Algoritmo) é um algoritmo iterativo em $t \in \mathbb{N}$ que somente funciona com dados linearmente separáveis. Se $\mathbf{w}(t)$ for tal que todos os dados estão corretamente classificados, o algoritmo para. Se não for o caso, o algoritimo escolhe um exemplo de $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$ que está atualmente desclassificado, o chama de $(\mathbf{x}(t), y(t))$ e toma

$$\mathbf{w}(t+1) = \mathbf{w}(t) + y(t)\mathbf{x}(t).$$

Primeiro a intuição. Se $\mathbf{x} \in P$, o que deveria ser a $\mathbf{w}(t)^T\mathbf{x}$? Positivo, é claro. Mas se está mal classificado, deu negativo. Ou seja

$$\mathbf{w}(t)^T\mathbf{x}(t) < 0$$

Agora o que acontece se iterarmos?

$$\mathbf{w}(t+1)^T\mathbf{x}(t) = (\mathbf{w}(t) + y(t)\mathbf{x}(t))^T\mathbf{x}(t) = \mathbf{w}(t)^T\mathbf{x}(t) + \mathbf{x}(t)^T y(t)\mathbf{x}(t) > \mathbf{w}(t)^T\mathbf{x}(t)$$

Então, intuitivamente é por isso que converge.

1.1. Is learning feasible?

Para que o aprendizado seja possível, o que acontece dentro a amostra tem que se refletir fora da amostra.

Para **uma** hipótese $h_1 \in \mathcal{H}$, temos a desigualdade de Hoeffding, com h_1 fixo antes de se gerar o dataset $\mathcal{D}$!

$$\mathbb{P}\left[|E_{\text{in}}(h_1) - E_{\text{out}}(h_1)\right] \leq 2e^{-2\epsilon^2 N}$$

. A hipótese final, que é outputada do nosso algoritmo $\mathcal{A}$ é feita depois de se gerar o dataset $\mathcal{D}$, usando todas as hipóteses!

Não queremos que

$$\mathbb{P}\left[E_{\text{in}}\left(h_m\right) - E_{\text{out}}\left(h_m\right) | > \epsilon\right] \text{é pequeno}$$

para qualquer particular m, mas na realidade que

$$\mathbb{P}\left[E_{\text{in}}\left(g\right) - E_{\text{out}}\left(g\right) | > \epsilon\right] \text{é pequeno}$$

para a nossa hipótese final g. Supondo que $\#\mathcal{H} = M \in \mathbb{N}$, existe uma maneira idiota e bruta de se resolver. Se fizermos g de maneira que não dependa de qual h_m o algoritmo escolha, $E_{\text{in}}\left(g\right) - E_{\text{out}}\left(g\right) | > \epsilon$ significa que ao menos um $E_{\text{in}}\left(g\right) - E_{\text{out}}\left(g\right) | > \epsilon$. Ou seja:

$$\mathbb{P}\left[E_{\text{in}}\left(g\right) - E_{\text{out}}\left(g\right) | > \epsilon\right] \leq \sum_{m=1}^{M} \mathbb{P}\left[E_{\text{in}}\left(h_m\right) - E_{\text{out}}\left(h_m\right) | > \epsilon\right] \Rightarrow \mathbb{P}\left[E_{\text{in}}\left(g\right) - E_{\text{out}}\left(g\right) | > \epsilon\right] \leq 2Me^{-2\epsilon^2 N}$$

O que é horroroso, mas é alguma coisa. Antes de continuar vale a pena insistir que esse tipo de ideia só funciona se adotarmos um framework probabilístico para f, a função misteriosa. Isto é, para cada $\mathbf{x}$, $f(\mathbf{x})$ pode retornar valores distintos, de acordo com as probabilidades. Podemos pensar em $\mathbf{x}$ como os parâmetros de uma f.d.p. $f(y|\mathbf{x})$.

2. Training versus Testing

2.1. Teoria da generalização

Pela inequalidade de Hoeffing:

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

Onde e é o erro da generalização, M o tamanho do espaço de hipóteses e N é quantidade de amostras.

Ao pegarmos um nível de tolerância δ:

$$E_{out}(g) \leq E_{in}(g) + \sqrt{\frac{1}{2N} \ln \frac{2M}{8}}$$

com probabilidade de no mínimo $1 - \delta$.

- **Definição:** Sejam $x_1, ..., x_N \in X$. A *dicotomia* gerada por H nesses pontos é definida por
$$H(x_1, ..., x_N) = (h(x_1), ..., h(x_N)|h \in H$$
- **Definição:** A *função de crescimento* é definida para um conjunto de hipóteses H como:
$$m_H(N) = max_{x_1,...,x_N \in X}|H(x_1, ..., x_N)|$$
onde $|.|$ é a cardinalidade.
Temos que $m_H(N) \leq \{-1, +1\}^N = 2^N$.
Se $H(x_1, ..., x_N) = 2^N$ dizemos que H *estilhaça* (ou *shatter*) $x_1, ..., x_N$.
- **Definição:** Se nenhum conjunto de dados de tamanho k pode ser estilhaçado por H, então k é dito ser um *ponto de quebra* (ou *break point*) para H.
Assim, $m_H(N) < 2^k$.
- **Teorema:** Se $m_H(N) < 2^k$ para algum valor de k, então
$$m_H(N) \leq \sum_{i=0}^{k-1} \binom{N}{i}$$
para todo N.
Obs1: Isso foi mostrado no capítulo, mas acho que não vale a pena adicionar no resumo.
Obs2: O somatório é um polinômio em N de grau $k - 1$.
- **Definição:** A *dimensão VC* de um conjunto de hipóteses H, denotada por $d_{VC}(H)$, é o maior valor de N tal que $m_H(N) = 2^N$.
Com isso, $k = d_{VC} + 1$.
Usando o teorema anterior, temos que
$$m_H(N) \leq N^{d_{VC}} + 1$$

- **Teorema:** Para qualquer tolerância $\delta > 0$

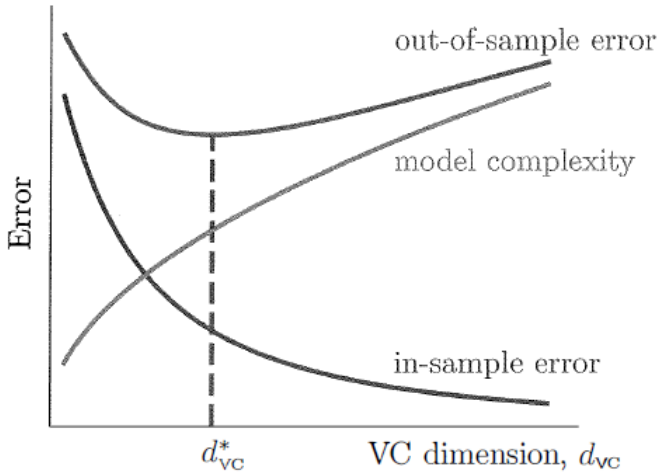
$$E_{out}(g) \leq E_{in}(g) + \sqrt{\frac{8}{N} \ln \frac{5m_H(2N)}{\delta}}$$

com probabilidade maior que $1 - \delta$.

Obs3: Não foi demonstrado no livro, mas aparentemente é um exercício.

2.2. Interpretando o limite de generalização

O gráfico é suficiente:



2.2.1. Outros tipos de alvo

A análise VC é baseada em funções binárias, mas podemos querer trabalhar com funções reais. Podemos usar a função de erro quadrático para definir:

$$E_{out}(h) = E[(h(x) - f(x))^2]$$

$$E_{in}(h) = \frac{1}{N} \sum_{n=1}^{N} (h(x_n) - f(x_n))^2$$

2.3. Tradeoff de generalização-aproximação

O erro fora da amostra é

$$E_{out}(g^{(D)}) = E_x[(g^{(D)}(x) - g^{(D)}(x))^2]$$

onde $g^{(D)}$ reforça que depende dos dados e E_x que o valor esperado é calculado em função de x.

O valor esperado do erro fora da amostra é

$$E_D[E_{out}(g^{(D)})] = E_D[E_x[(g^{(D)}(x) - g^{(D)}(x))^2]]$$

Chamando $E_D[g^{(D)}(x)]$ de $\bar{g}(x)$, temos:

$$E_D[E_{out}(g^{(D)})] = E_x[E_D[(g^{(D)}(x) - \bar{g}(x))^2] + (\bar{g} - f(x))^2]$$

Com isso, separamos em duas estimativas de erro:

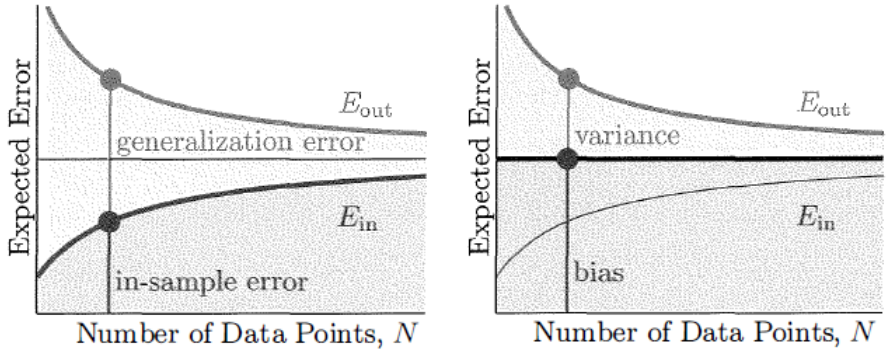
$$bias(x) = (\bar{g}(x) - f(x))^2$$
$$var(x) = E_D[(g^{(D)}(x) - \bar{g}(x))^2]$$

Assim

$$E_{out}(g^{(D)}) = E_x[bias(x) + var(x)] = bias + var$$

O tradeoff é explicado na imagem:

Fazendo o paralelo entre análise VC e análise de bias-variância:



3. The Linear Model

3.1. Non-Separable Data

Em alguns casos, não é possível separar os dados com uma reta, sendo necessário aceitarmos $E_{in} \neq 0$. Isso pode se dar tanto pelo problema em si quanto por ruídos ou outliers. Para encontrar a hipótese com E_{in} mínimo, precisamos resolver o problema de otimização:

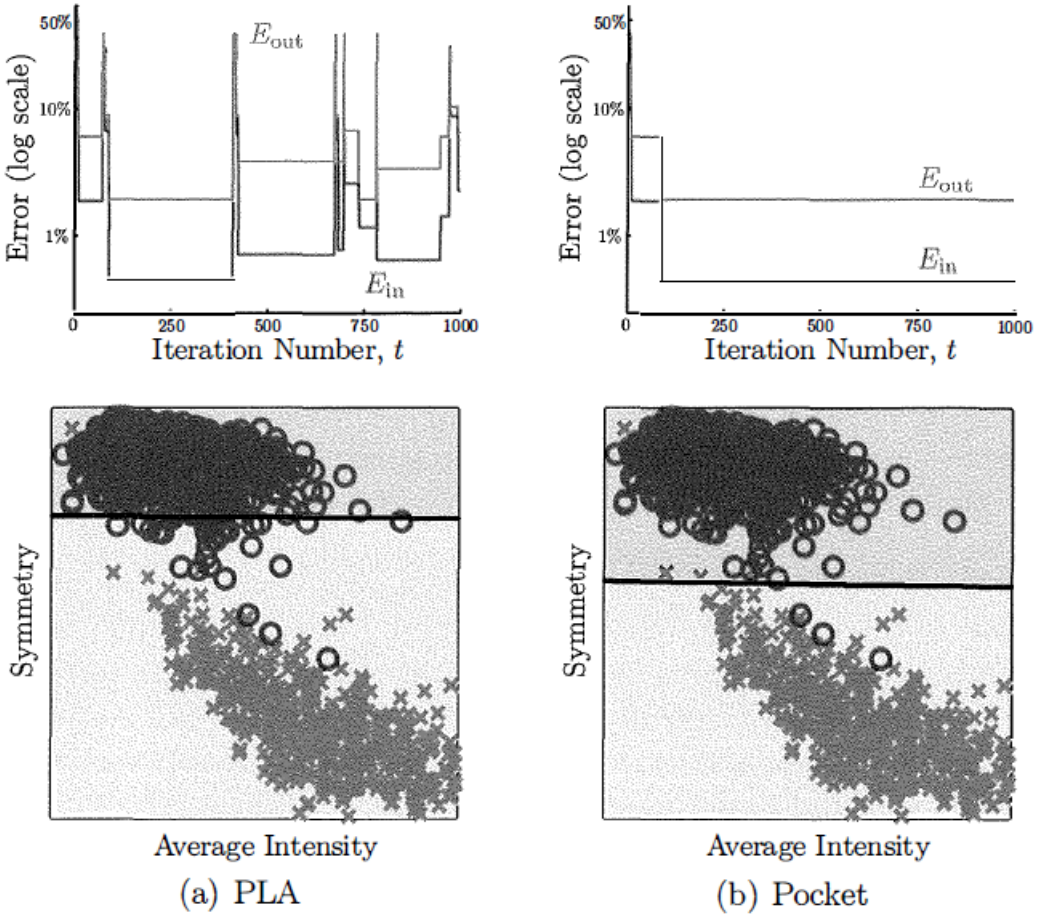
$$min_{w \in \mathbb{R}^{d+1}} \frac{1}{N} \sum_{n=1}^{N} [sign(w^T x_n) \neq y_n]$$

Mas esse é um problema NP-hard. Assim, há o algoritmo *pocket* para lidar com esses casos:

The pocket algorithm:

1: Set the pocket weight vector $\hat{\mathbf{w}}$ to $\mathbf{w}(0)$ of PLA.
2: **for** $t = 0, \ldots, T \quad 1$ **do**
3: Run PLA for one update to obtain $\mathbf{w}(t+1)$.
4: Evaluate $E_{\text{in}}(\mathbf{w}(t+1))$.
5: If $\mathbf{w}(t+1)$ is better than $\hat{\mathbf{w}}$ in terms of E_{in}, set $\hat{\mathbf{w}}$ to $\mathbf{w}(t+1)$.
6: Return $\hat{\mathbf{w}}$.

Ele é bem mais lento que o PLA, mas resolve para casos de dados com ruídos, como mostra a imagem:



(a) PLA (b) Pocket

3.2. Linear Regression

Utilizado para funções reais. É baseado na minimização do erro quadrado entre $h(x)$ e y.

$$E_{out}(h) = E[(h(x) - y)^2]$$

Como a distribuição $P(x, y)$ é desconhecida, E_{out} não pode ser computada. Assim como fizemos com o problema da classificação, iremos considerar a versão amostral então:

$$E_{in}(h) = \frac{1}{N}\sum_{n=1}^{N}(h(x_n) - y_n)^2$$

h toma a forma de uma combinação linear dos componentes de x, ou seja:

$$h(x) = \sum_{i=0}^{N} w_i x_i = w^T x$$

onde $x_0 = 1$ e $x \in \{1\} \times \mathbb{R}^d$ e $w \in \mathbb{R}^{d+1}$.

Podemos expressar E_{in} como uma função de w e os dados X, y:

$$E_{in}(h) = \frac{1}{N}\sum_{n=1}^{N}(h(x_n) - y_n)^2 = \frac{1}{N}||Xw - y||^2 = \frac{1}{N}(w^T X^T X w - 2w^T X^T y + y^T y)$$

Para minimizar E_{in}, encontramos sua derivada:

$$\nabla E_{in}(w) = \frac{2}{N}(X^T X w - X^T y)$$

e igualamos a zero:

$$X^T X w = X^T y$$

Se $X^T X$ é invertível, então $w = (X^T X)^{-1} X^T y$, onde $(X^T X)^{-1} X^T$ é chamada pseudo-inversa. Se não for, então uma pseudo-inversa ainda pode ser definida, mas a solução não será única.

Com isso

$$\hat{y} = XXw_{lin} = X(X^T X)^{-1} X^T y = Hy$$

O erro de generalizção do modelo é:

$$E_D[E_{in}(w_{lin})] = \sigma^2\Big(1 - \frac{d+1}{N}\Big)$$

Isso é demonstrado no exercício 3,4, que o Targino pediu pra fazermos. Se quiser, posso adicionar minha resolução (ainda incompleta) aqui.

3.3. Logistic Regression

Caso queiramos um modelo para a probabilidade de um evento, numa função que é mais suave que o classificador linear, podemos utilizar o modelo de regressão logística:

$$h(x) = \theta(w^T x)$$

onde θ é a função logística $\theta(s) = \frac{e^s}{1+e^s}$ cuja saída é entre 0 e 1.

Para encontrar a medida de erro, utilizamos a noção de likelihood para chegar em:

$$E_{in}(w) = \frac{1}{N}\sum_{n=1}^{N}\ln\left(1 + e^{-y_n w^T x_n}\right)$$

Nota-se que a medida de erro é pequena quando $y_n w^T x_n$ é grande e positivo, o que implicaria que $sign(w^T x_n) = y_n$. Assim, como se espera, a medida de erro encoraja w a çlassificarçada x_n corretamente.

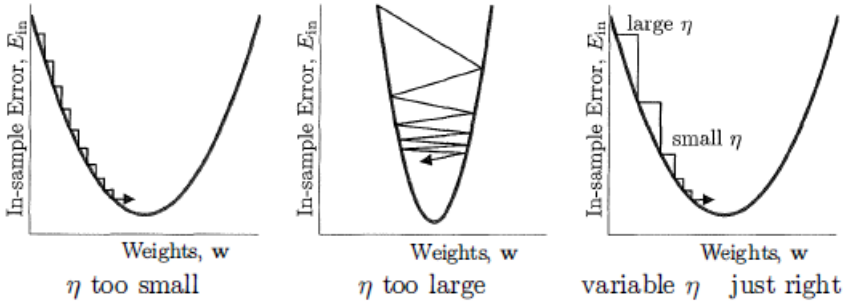
No exercício 3,7 mostramos que:

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

Como $-y_n w^T x_n > 0$ para um exemplo mal classificado e $-y_n w^T x_n < 0$ para um exemplo bem classificado, temos que pontos mal classificados impactam mais para o gradiente, já que $\theta(s)$ é uma função crescente, para o bem classificado $\theta(s)$ ficará mais próximo de 0 e para o mal classificado $\theta(s)$ ficará mais próximo de 1.

3.3.1. Gradiente Descendent

Gradiente descendente é uma técnica geral para minimizar uma função duas vezes diferenciável, tal como $E_{in}(w)$ na regressão logística. A ideia é basicamente andar na direção contrária do gradiente, ou seja, a cada ponto ir para um ponto menor. Para isso é definido um tamanho do passo (em alguns casos chamado de *learning rate*) que é indicado por η. Esse valor não pode ser muito grande, pois podemos acabar não chegando no mínimo dessa forma. Também é desejável que não seja muito pequeno, por precisarmos de uma quantidade muito alta de passos para chegar ao mínimo. Geralmente é utilizado 0,1.



η too small η too large variable η just right

Uma técnica interessante é utilizar um η variável, que começa grande e termina pequeno para estimarmos de forma mais rápida e certeira o mínimo.

Ao inicializar o algoritmo, é bom começar com $w(0)$ aleatório, geralmente com cada w_i independente de uma distribuição normal com média zero e variância pequena. Para determinar o ponto de parada, geralmente utiliza-se um número máximo de iterações, junto com um máximo para a melhora marginal do erro e um valor baixo para o erro em si.

Também podemos utilizar o **gradiente descendente estocástico** onde em cada iteração escolhemos um ponto aleatório, calculamos o gradiente e o utilizamos par atualizar os pesos de w. Isso reduz a complexidade do gradiente descendente e gera um resultado bom no final

3.4. Nonlinear Transformation

Em alguns casos, não conseguimos encontrar um classificador linear para o modelo. Ao transformar os inputs x_1, x_2 de uma forma não linear, podemos conseguir separar dados com regiões mais complicadas ainda utilizando um PLA.

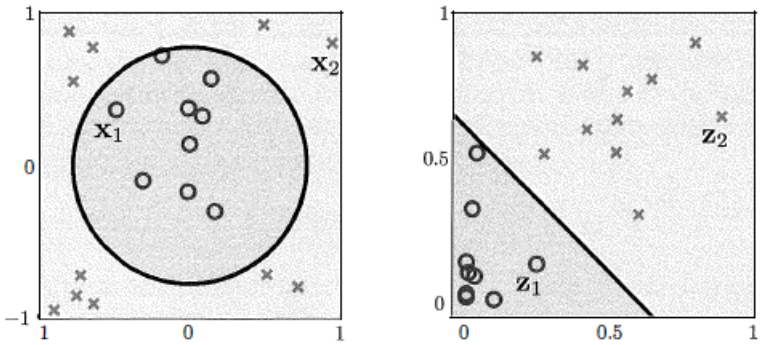
Podemos encontrar um vetor z da forma

$$z = \Phi(x)$$

Assim, podemos representar a hipótese não linear h do espaço X em uma hipótese linear $\tilde{h}$ no espaço Z:

$$h(x) = \tilde{h}(\Phi(x))$$

Com isso, utilizamos algum dos modelos lineares que aprendemos para encontrar $\tilde{h}(z)$.



(a) Data in $\mathcal{X}$-space (b) Transformed data in $\mathcal{Z}$ space

$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix} \qquad \mathbf{z} = \Phi(\mathbf{x}) = \begin{bmatrix} 1 \\ x_1^2 \\ x_2^2 \end{bmatrix}$$

Podemos querer utilizar mais de uma transformação. A transformação Φ_Q é chamada transformação polinomial de Q-ésima ordem. Embora utilizar um Q maior nós dê uma flexibilidade maior ao lidar com X, há consequências nisso. Para começar, uma transformação Φ_Q mapeia um vetor bi-dimensional x em $\tilde{d} = \frac{Q(Q+3)}{2}$ dimensões. Com isso temos que $d_{VC}(H_\Phi) <= \frac{Q(Q+3)}{2} + 1$, fazendo com que o segundo lado da restrição VC cresça bastante.

4. Overfitting

Overfitting ocorre quando o ruído dos dados faz com que cheguemos a uma função com baixo E_{in}, mas alto E_{out} por basicamente "decorarmos.ºs dados.

(Achei muito chata a parte inicial, resume aí pra mim)

4.1. Regularization

Uma forma de olhar a regularização é através do limite VC, que limita E_{out} usando um modelo de complexidade $\Omega(H)$:

$$E_{out}(h) \leq E_{in}(h) + \Omega(H)$$

para todo $h \in H$.

Ao invés de minimizar apenas $E_{in}(h)$, na regularização iremos minimizar a combinação de $E_{in}(h)$ e $\Omega(h)$. Isso evita o overfitting ao forçar o algoritmo de aprendizagem a aproximar bem os dados utilizando uma hipótese simples.

Uma técnica popular de regularização é *decaimento de pesos* (ou *weight decay*), que mede a complexidade de uma hipóteses h pelo tamanho dos coeficientes usados para representar h. Essa eurística prefere linhas linhas suaves com um pequeno deslocamento e inclinação a linhas com maior deslocamento e inclinação.

A decomposição bias-variância nos ajuda a entender como a

5. Overfitting

5.1. When does overfitting occur?

Um dos principais conceitos é o conceito de erro determinístico. Isso acontece quando pegamos o hypotesys set $\mathcal{H}$ tal que é garantido que $f \notin \mathcal{H}$.

A viés-variância decomposição, discutida no capítulo 2 é importante para entender

$$\mathbb{E}_{\mathcal{D}} = \sigma^2 + \text{bias} + \text{var}.$$

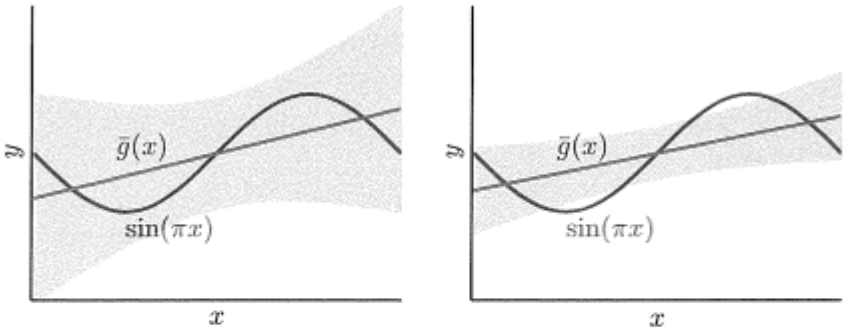
O σ^2, naturalmente, vem do ruído estocástico. O viés é diretamente relacionado com o ruído determinístico, no qual captura a inabilidade do modelo de aproximar f. A variância é o caso do quão certo você está do viés.

5.2. Regularização

Uma maneira de se enxergar a regularização é através da dimensão VC

$$E_{\text{out}}(h) \leq E_{\text{in}}(h) + \Omega(h) \text{ for all } h \in \mathcal{H}$$

Onde Ω é uma função de complexidade. A beleza dessa ideia reside em você impor a regularização em h e não em $\mathcal{H}$. Ou seja, você pode até ter um $\mathcal{H}$ complicado, desde que seu h seja simples e bom.



without regularization
bias = 0.21;
var = 1.69.

with regularization
bias = 0.23;
var = 0.33.

Average hypothesis $\bar{g}$ (red) with $\text{var}(x)$ indicated by the gray shaded region that is $\bar{g}(x) \pm \sqrt{\text{var}(x)}$.