Relevance Vector Machines

Seminário de “Bayesian statistical learning” - Insper

Mark Andrew Gannon

Departamento de Estatística
Instituto de Matemática e Estatística - Universidade de São Paulo (IME-USP)

18 março, 2016

Estrutura da apresentação

Introdução

- Aprendizado supervisionado
- Exemplos de vetores "input" (de preditores) $\{\mathbf{x}_n\}_{n=1}^N$ e respostas observados $\{t_n\}_{n=1}^N$, onde t pode ser um número real ou um rótulo de categoria
- Queremos fazer predições de t_n para dados $\mathbf{x}_n,\ n > N$

Introdução

- Função $y(\mathbf{x})$ definida em todo o espaço de possíveis entradas $\mathbf{x}$
- Candidatas: $y(\mathbf{x}; \mathbf{w}) = \sum_{i=1}^{M} w_i \phi_i(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x})$
- Linear nos "pesos" $\{w_i\}$
- Queremos evitar "overfitting"

Introdução: SVM

- Consideramos funções de um tipo correspondente às utilizadas pelo SVM
- $y(\mathbf{x};\, \mathbf{w}) = \sum_{i=1}^{N} w_i K(\mathbf{x}, \mathbf{x}_i) + w_0$
- Uma função de base por cada registro nos dados de aprendizado
- Tentamos minimizar erro nos dados de treinamento/aprendizado e simultaneamente maximizar a "margem" entre as categorias
- Evita "overfitting" construindo modelo "esparso" com poucas funções K : aquelas associadas a "support vectors" na fronteira ou no lado "errado" dela

Overfitting

Overfitting: resultado muito bom nos dados de aprendizado, potencialmente muito ruim para futuros dados

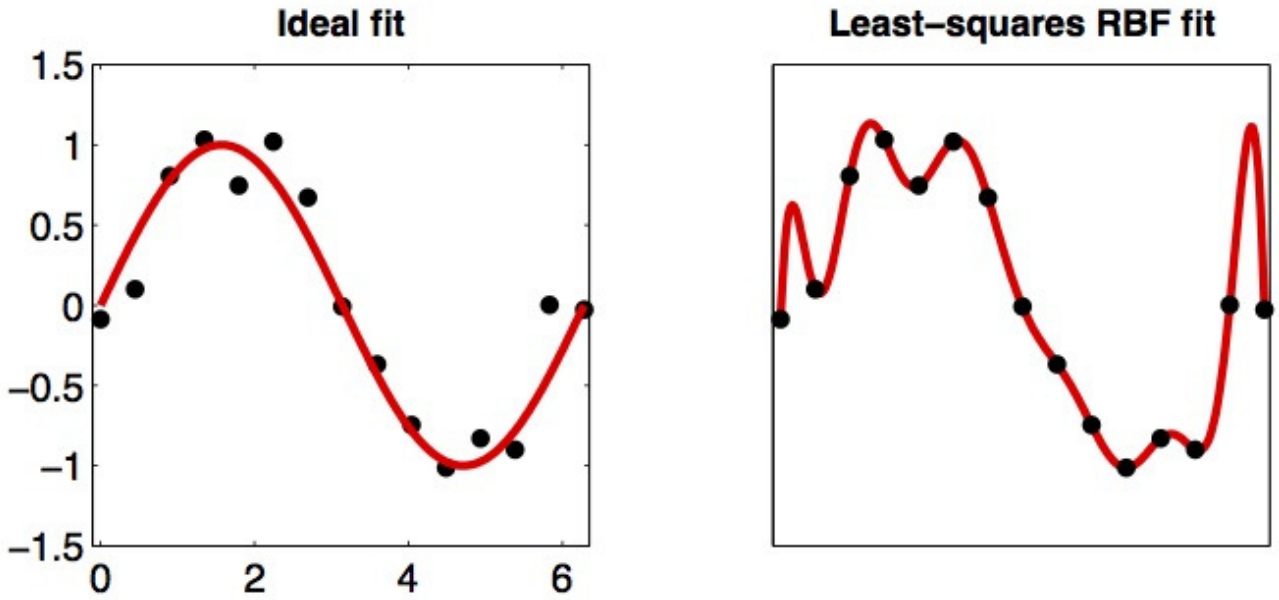

Figure 1: Overfitting? The 'ideal fit' is shown on the left, while the least-squares fit using 15 basis functions is shown on the right and perfectly interpolates all the data points.

Introdução:Desvantagens do SVM

O SVM já foi aplicado e produziu (e produz) resultados, mas a tecnologia tem algumas desvantagens:

- SVMs produzem modelos relativamente esparsos, mas o número de “support vectors” cresce linearmente com a quantidade de dados de aprendizado
- Pós-processamento freqüentemente necessário para reduzir a complexidade computacional

Introdução:Desvantagens do SVM

- Predições não são probabilísticas - estimativa pontual em regressão e decisão binária "hard" em classificação
  - Regressão : "error bars"
  - Classificação: precisamos de probabilidades *a posteriori* para adaptação a diferentes prioris de categoria e custos asimétricos de erros de classificação

  Existem estimativas de posterioris para SVMs, mas não são confiáveis
- Precisamos de estimativas para C, o parâmetro que controla o peso relativo de erros e margem e, para regressão, o parâmetro de "sensibilidade" ϵ. - Temos que usar validação cruzada - custo computacional e de dados
- $K(\mathbf{x}, \mathbf{x}_i)$ restritas

Introdução: Desvantagens do SVM

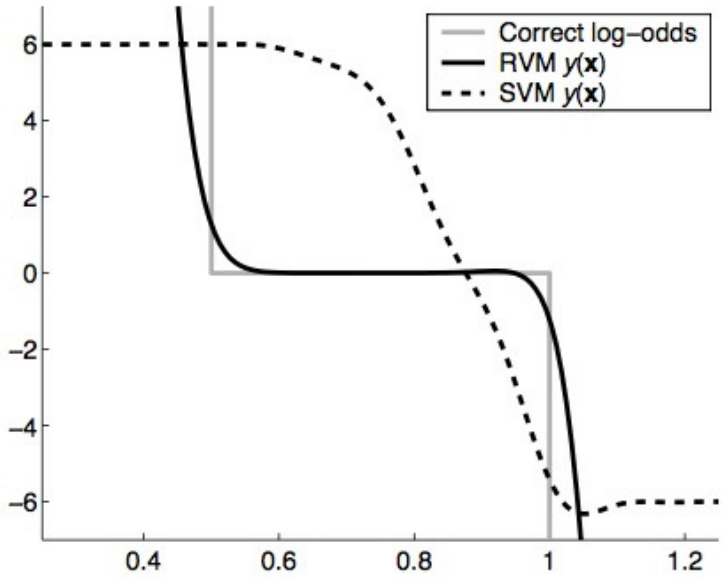

Figure 10: Output of an RVM and SVM trained on overlapping, uniformly distributed data, along with the true log-odds of membership of $\mathcal{C}_{+1}$ over $\mathcal{C}_{-1}$. The SVM output has been rescaled without prejudice for ease of comparison with the RVM.

Introdução: Desvantagens do SVM

O SVM já foi aplicado e produziu (e produz) resultados, mas a tecnologia tem algumas desvantagens:

- SVMs produzem modelos relativamente esparsos, mas o número de "support vectors" cresce linearmente com a quantidade de dados de aprendizado
- Pós-processamento freqüentemente necessário para reduzir a complexidade computacional
- Predições não são probabilísticas - estimativa pontual em regressão e decisão binária "hard" em classificação
  - Regressão : "error bars"
  - Classificação: abordagem probabilística necessária - probabilidades *a posteriori* para adaptação a diferentes prioris de categoria e custos asimétricos de erros de classificação

Especificação do Modelo

Modelo de regressão bayesiano e procedimento de inferência

- Tratamos dados $\{\mathbf{x}_n, t_n\}_{n=1}^N$ como amostras do modelo com ruído: $t_n = y(\mathbf{x}_n; \mathbf{w}) + \epsilon_n$, onde $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$
- $p(t_n|\mathbf{x}) = \mathcal{N}(t_n|y(\mathbf{x}_n, \sigma^2)$
- Funções de base $\phi_i(\mathbf{x}) = K(\mathbf{x}, \mathbf{x}_i)$

Especificação do Modelo

- Se os t_n são independentes, a verossimilhança do conjunto de dados é

$$p(\mathbf{t}|\mathbf{w},\sigma^2) = (2\pi\sigma^2)^{N/2} \exp\left\{-\frac{1}{2\sigma^2}||\mathbf{t}-\Phi\mathbf{w}||^2\right\}, \quad (1)$$

onde $\mathbf{t} = (t_1,...,t_N)^T$, $\mathbf{w} = (w_0,...,w_N)^T$ e

$\boldsymbol{\Phi}$ é a "design matrix" de tamanho $N \times (N+1)$
$\boldsymbol{\Phi} = [\phi(\mathbf{x}_1),...,\phi(\mathbf{x}_N)]^T$,
com $\phi(\mathbf{x}_n) = [K(\mathbf{x}_n,\mathbf{x}_1), K(\mathbf{x}_n,\mathbf{x}_2),...,K(\mathbf{x}_n,\mathbf{x}_N)]^T$

Especificação do Modelo

Como temos um parâmetro para cada registro no conjunto de dados, estimativas de máxima verossimilhança levariam a "overfitting". Agora poderíamos colocar restrições nos parâmetros, mas adotamos uma abordagem bayesiana e...

Introduzimos MAIS (!!) $N + 1$

Especificação do Modelo

Mas esses $N + 1$ parâmetros "a mais" são os hiperparâmetros das prioris para $\mathbf{w}$. A preferência por funções suaves (menos complexas) é expressada por prioris normais com média zero (e, portanto, mais probabilidade concentrado perto de zero):

$$p(\mathbf{w}|\boldsymbol{\alpha}) = \prod_{i=0}^{N} \mathcal{N}(w_i|0, \alpha_i^{-1})$$

Hiperprioris sobre $\boldsymbol{\alpha}$ e $\beta = \sigma^{-2}$:

$$p(\boldsymbol{\alpha}) = \prod_{i=0}^{N} \text{Gamma}(\alpha_i|a, b)$$

$$p(\beta) = \text{Gamma}(\beta|c, d)$$

Distribuição Gamma

$$\text{Gamma}(\alpha|a,b) = \Gamma(a)^{-1} b^a \alpha^{a-1} e^{-b\alpha}$$

$\Gamma(a) = \int_0^\infty t^{a-1} e^{-t} dt$ é a função "gamma"

Para n inteiro e positivo, $\Gamma(n) = (n-1)!$

Como o modelo fica esparso

Num apêndice do artigo, Tipping considera hiperprioris "Gamma" em geral, mas na maior parte do artigo, usa $a = b = c = d = 0$ para hiperprioris uniformes.

Esse esquema acaba sendo uma espécie de "automatic relevance determination".

Como o modelo fica esparso

Usar hiperprioris largas sobre os hiperparâmetros permite que a probabilidade *a posteriori* fique concentrada em valores grandes de alguns dos α_i.

O resultado disso é que a probabilidade *a posteriori* do parâmetro (peso) associado, w_i, fica concentrada em zero. Isso significa que podemos ignorar o preditor correspondente x_i em nosso modelo.

Inferência

O Teorema de Bayes nos dá a posteriori

$$p(\mathbf{w}, \boldsymbol{\alpha}, \sigma^2|\mathbf{t}) = \frac{p(\mathbf{t}|\mathbf{w}, \boldsymbol{\alpha}, \sigma^2)p(\mathbf{w}, \boldsymbol{\alpha}, \sigma^2)}{p(\mathbf{t})}$$

Dado um novo ponto de teste $\mathbf{x}_*$, predições são feitas sobre t_* em termos da preditiva

$$p(t_*|\mathbf{t}) = \int p(t_*|\mathbf{w}, \boldsymbol{\alpha}, \sigma^2)p(\mathbf{w}, \boldsymbol{\alpha}, \sigma^2|\mathbf{t})d\mathbf{w}d\boldsymbol{\alpha}d\sigma^2$$

Na prática, não conseguimos fazer esses cálculos. Precisamos de aproximações.

Inferência

Não podemos calcular a posteriori $p(\mathbf{w}, \boldsymbol{\alpha}, \sigma^2|\mathbf{t})$ porque não sabemos calcular o denominador,
$p(\mathbf{t}) = \int p(\mathbf{t}|\mathbf{w}, \boldsymbol{\alpha}, \sigma^2)p(\mathbf{w}, \boldsymbol{\alpha}, \sigma^2)d\mathbf{w}d\boldsymbol{\alpha}d\sigma^2$

Para contornar esse problema, expressamos a posteriori assim:

$$p(\mathbf{w}, \boldsymbol{\alpha}, \sigma^2|\mathbf{t}) = p(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha}, \sigma^2)p(\alpha, \sigma^2|\mathbf{t})$$

e podemos calcular a integral de normalização $p(\mathbf{t}|\boldsymbol{\alpha}, \sigma^2) = \int p(\mathbf{t}|\mathbf{w}, \sigma^2)p(\mathbf{w}, \boldsymbol{\alpha})d\mathbf{w}$, uma convolução de normais.

Inferência

A posteriori sobre os pesos **w** é

$$p(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha}, \sigma^2) = \frac{p(\mathbf{t}|\mathbf{w}, \sigma^2)p(\mathbf{w}|\boldsymbol{\alpha})}{p(\mathbf{t}|\boldsymbol{\alpha}, \sigma^2)}$$

$$p(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha}, \sigma^2) = (2\pi)^{-(N+1)/2}|\Sigma|^{-1/2} exp\left\{-\frac{1}{2}(\mathbf{w}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{w}-\boldsymbol{\mu})\right\},$$

onde a covariância e média *a posteriori* são

$$\boldsymbol{\Sigma} = (\sigma^{-2}\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \mathbf{A})^{-1}$$

$$\boldsymbol{\mu} = \sigma^{-2}\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\mathbf{t}$$

e $\mathbf{A} = \text{diag}(\alpha_0, \alpha_1, ..., \alpha_N)$

Inferência - aproximações

Trocamos a posteriori dos hiperparâmetros $p(\boldsymbol{\alpha}, \sigma^2|\mathbf{t})$ por uma "função delta" nos valores mais prováveis (moda) $\boldsymbol{\alpha}_{MP}$ e σ^2_{MP}

$$\int p(t_*|\boldsymbol{\alpha}, \sigma^2)\delta(\boldsymbol{\alpha}_{MP}, \sigma^2_{MP})d\boldsymbol{\alpha}d\sigma^2 \approx \int p(t_*|\boldsymbol{\alpha}, \sigma^2)p(\boldsymbol{\alpha}, \sigma^2|\mathbf{t})d\boldsymbol{\alpha}d\sigma^2$$

Tipping afirma que toda a evidência apresentada no artigo sugere que essa aproximação é boa em geral

Inferência

Aprendizado com "relevance vectors" acaba sendo a procura pela moda da posteriori dos hiperparâmetros, i.e., a maximização de $p(\boldsymbol{\alpha}, \sigma^2|\mathbf{t}) \propto p(\mathbf{t}|\boldsymbol{\alpha}, \sigma^2)p(\boldsymbol{\alpha})p(\sigma^2)$ com respeito a $\boldsymbol{\alpha}$ e β.

Com hiperprioris uniformes, só precisamos maximizar $p(\mathbf{t}|\boldsymbol{\alpha}, \sigma^2)$, que podemos calcular:

$$p(\mathbf{t}|\boldsymbol{\alpha}, \sigma^2) = \int p(\mathbf{t}|\mathbf{w}, \sigma^2)p(\mathbf{w}|\boldsymbol{\alpha})d\mathbf{w}$$

$$= (2\pi)^{-N/2}|\sigma^2\mathbf{I} + \boldsymbol{\Phi}\mathbf{A}^{-1}\boldsymbol{\Phi}^T|^{-1/2} exp\left\{-\frac{1}{2}\mathbf{t}^T(\sigma^2\mathbf{I} + \boldsymbol{\Phi}\mathbf{A}^{-1}\boldsymbol{\Phi}^T)^{-1}\mathbf{t}\right\}$$

Inferência

$p(\mathbf{t}|\boldsymbol{\alpha}, \sigma^2)$: "marginal likelihood" (verossimilhança marginal) ou "evidênca dos hiperparâmetros"

Maximização da verossimilhança marginal: "evidence procedure"

Otimização dos Hiperparâmetros

Não podemos obter "de forma fechada" (uma equação) os valores de $\boldsymbol{\alpha}$ e σ^2 que maximizam a verossimilhança marginal. Usamos reestimativa iterativa.

Quando a derivada da verossimilhança marginal é zero, temos

$$\alpha_i^{\text{novo}} = \frac{\gamma_i}{\mu_i^2},$$

onde μ_i é o i-ésimo "peso" médio *a posteriori* e $\gamma_i = 1 = \alpha_i \Sigma_{ii}$, onde Σ_{ii} é o i-ésimo elemento diagonal da covariância *a posteriori* dos "pesos" $\mathbf{w}$, calculados utilizando valores atuais de $\boldsymbol{\alpha}$ e σ^2.

Otimização dos Hiperparâmetros

Maximização da verossimilhança marginal com respeito a σ^2 resulta em

$$(\sigma^2)^{\text{novo}} = \frac{||\mathbf{t} - \boldsymbol{\Phi}\boldsymbol{\mu}||^2}{N - \sum_i \gamma_i}$$

Algoritmo de Aprendizado

Aplicamos as fórmulas para α_i^{novo} e $(\sigma^2)^{\text{novo}}$ e atualizamos as estatísticas *a posteriori* $\boldsymbol{\mu}$ e $\boldsymbol{\Sigma}$. Repetimos até satisfazer algum critério de convergência.

Na prática, Tipping e seus colaboradores descobrirarm que durante o processo de reestimativa, muitos dos α_i ficam grandes ("tendem ao ∞").

$p(w_i|\mathbf{t}, \boldsymbol{\alpha}, \sigma^2)$ fica muito concentrada perto de $w_i = 0$. Podemos "podar" as funções de base correspondentes.

Algoritmo de Aprendizado

Faul e Tipping estudaram a maximização da verossimilhança marginal com respeito aos hiperparâmetros.

Provaram que a verossimilhança marginal, considerada como função de um único hiperparâmetro, tem um único máximo que pode ser calculado analiticamente. Sob um critério de “sparseness”, esse máximo é equivalente a “podar” esse hiperparâmetro (e a função-base e parâmetro correspondentes) do modelo.

Predições

Quando o procedimento de otimização dos hiperparâmetros converge, fazemos predições baseadas na posteriori sobre os "pesos" (coeficientes no modelo **w**), condicionada nos valores α_{MP} e σ^2_{MP}
Podemos calcular a distribuição preditiva. Para um novo x_*,

$$p(t_*|\mathbf{t}, \alpha_{MP}, \sigma^2_{MP}) = \int p(t_*|\mathbf{w}, \sigma^2_{MP}) p(\mathbf{w}|\mathbf{t}, \alpha_{MP}, \sigma^2_{MP}) d\mathbf{w}$$

As duas partes do integrando são normais. Portanto, podemos fazer esse cálculo

Predições

$$p(t_*|\mathbf{t}, \boldsymbol{\alpha}_{MP}, \sigma^2_{MP}) = \mathcal{N}(t_*|y_*, \sigma^2_*),$$

onde

$$y_* = \boldsymbol{\mu}^T \boldsymbol{\phi}(\mathbf{x}_*)$$
$$\sigma^2_* = \sigma^2_{MP} + \boldsymbol{\phi}(\mathbf{x}_*)\boldsymbol{\Sigma}\boldsymbol{\phi}(\mathbf{x}_*)$$

A média preditiva é $y(\mathbf{x}_*; \boldsymbol{\mu})$, as funções de base ponderadas pelos "pesos" médios μ_i - muitos são zero

A variância preditiva ("error bars") tem duas partes: a estimativa do ruído nos dados e a incerteza na predição dos "pesos"

Classificação

Duas categorias - queremos a probabilidade *a posteriori* de um novo ponto pertencer a cada uma, dado $\mathbf{x}$

Aplicamos função sigmóide $\sigma(y) = 1/(1 + e^{-y})$ a $y(\mathbf{x})$ e, adotando uma Bernoulli para $P(t|\mathbf{x})$, a verossimilhança é

$$P(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} \sigma(y(\mathbf{x}_n; \mathbf{w}))^{t_n} [1 - \sigma(y(\mathbf{x}_n; \mathbf{w}))]^{1-t_n}$$

Aqui não podemos integrar os pesos analiticamente. Não temos expressões pra posteriori dos pesos $p(\mathbf{w}|\mathbf{t}, \alpha)$, nem pra verossimilhança marginal $P(\mathbf{t}|\alpha)$. Usamos outra aproximação.

Classificação: algoritmo

- Pros atuais (fixos) valores de $\boldsymbol{\alpha}$, encontramos os pesos "mais prováveis" $\mathbf{w}_{MP}$ (a moda).
  Como $p(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha}) \propto P(\mathbf{t}|\mathbf{w})p(\mathbf{w}|\boldsymbol{\alpha})$, é equivalente é procurar o máximo, sobre $\mathbf{w}$, de

$$\log[P(\mathbf{t}|\mathbf{w})p(\mathbf{w}|\boldsymbol{\alpha})] = \sum_{n=1}^{N}[t_n \log y_n + (1-t_n)\log(1-y_n)] - \frac{1}{2}\mathbf{w}^T\mathbf{A}\mathbf{w},$$

  com $y_n = \sigma(y(\mathbf{x}_n; \mathbf{w}))$.

Classificação: algoritmo

- O método de Laplace: aproximação quadrática da log-posteriori perto da sua moda. Tomando duas derivadas da log-posteriori, obtemos

$$\nabla_w \nabla_w \log p(\mathbf{w}|\mathbf{t}, \alpha)|_{\mathbf{w}_{MP}} = -(\boldsymbol{\Phi}^T \mathbf{B} \boldsymbol{\Phi} + \mathbf{A})^{-1}$$

$$\mathbf{B} = \text{diag}(\beta_1, \beta_2, ..., \beta_N) \text{ com } \beta_n = \sigma(y(\mathbf{x}_n))[1 - \sigma(y(\mathbf{x}_n))]$$

Isso é invertido para obter a covariância Σ de uma aproximação normal à posteriori sobre presos com moda em $\mathbf{w}_{MP}$

Classificação: algoritmo

- Usando as estatísticas Σ e $\mathbf{w}_{MP}$ da aproximação normal, os hiperparâmetros α são atualizados usando as mesmas expressões usadas em regressão.

  Na moda de $p(\mathbf{w}|\mathbf{t}, \alpha)$, usando a expressão do slide anterior e o fato $\nabla_w \log p(\mathbf{w})|\mathbf{t}, \alpha)|_{\mathbf{w}_{MP}} = 0$, podemos escrever

$$\Sigma = (\Phi^T \mathbf{B} \Phi + \mathbf{A})^{-1}$$
$$\mathbf{w}_{MP} = \Sigma \Phi^T \mathbf{B} \hat{\mathbf{t}}$$
$$\hat{\mathbf{t}} = \mathbf{F}\mathbf{w} + \mathbf{B}^{-1}(\mathbf{t} - \mathbf{y})$$

Regressão: função "sinc"

Função $\text{sinc}(x) = \text{sen}(x)/x$.

- Imaginamos um "tubo" $\pm\epsilon$ em volta da função
- Dentro desse tubo, erros não são penalizados
- "Support vectors" ficam fora dessa região ou na sua fronteira.
- Kernel "linear spline"

$$K(x_m, x_n) = 1 + x_m x_n + x_m x_n \min(x_m, x_n) - \frac{x_m + x_n}{2} \min(x_m, x_n)^2 + \frac{\min(x_m, x_n)^3}{3} \quad (2)$$

- $\epsilon = 0,01$

Regressão: função "sinc"

- Com RVM, modelamos os dados com o mesmo kernel, para definir funções de base $\phi_n(x) = K(x, x_n)$
- Fixa variância do ruído em $0,01^2$ e fazer estimativa só de $\boldsymbol{\alpha}$.
- RVM utiliza apenas 9 "relevance vectors"
- Erro menor do que SVM

Regressão: função "sinc"

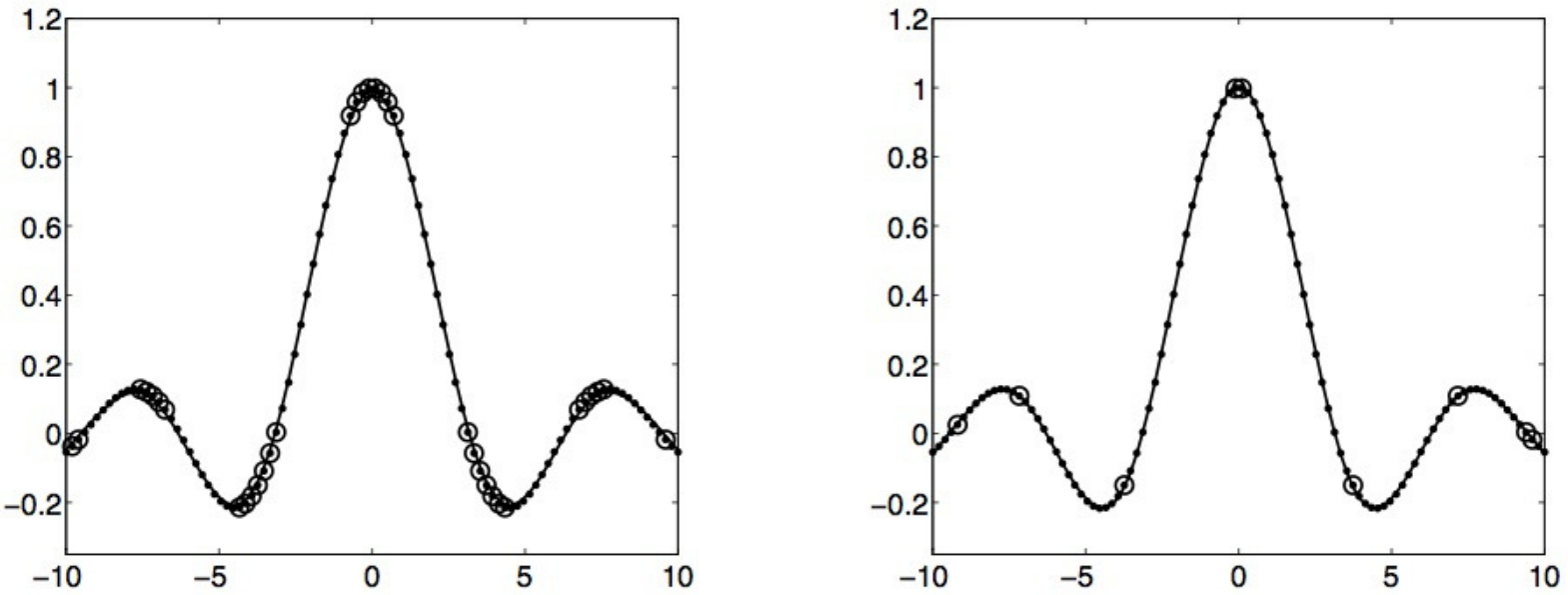

Figure 1: Support (left) and relevance (right) vector approximations to sinc(x) from 100 noise-free examples using 'linear spline' basis functions. The estimated functions are drawn as solid lines with support/relevance vectors shown circled.

Regressão: função "sinc"

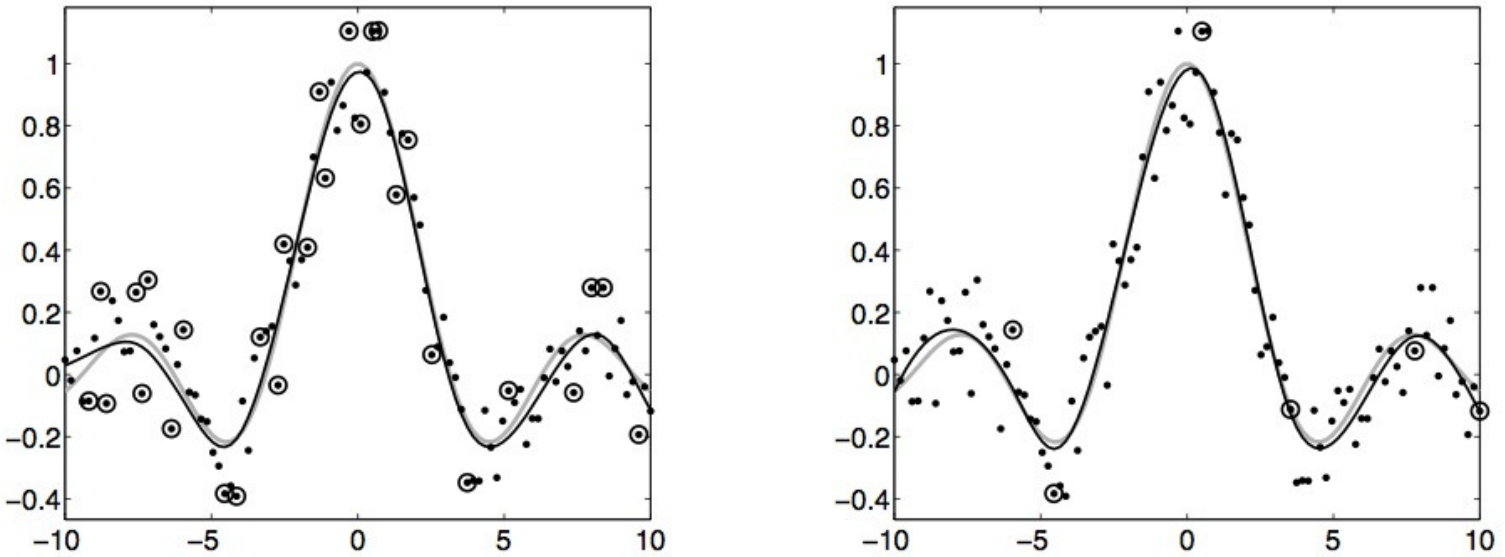

Figure 2: Support (left) and relevance (right) vector approximations to sinc(x), based on 100 noisy samples. The estimated functions are drawn as solid lines, the true function in grey, and support/relevance vectors are again shown circled.

Extensões

- Função em duas dimensões:

$$y(x_1, x_2) = sinc(x_1) + 0,1x_2$$

- Linear em x_2 - difícil de modelar com superposição de funções não-lineares
- A parte não-linear, sinc(x_1), só depende de x_1. x_2 acrescentará apenas "ruído"

Extensões

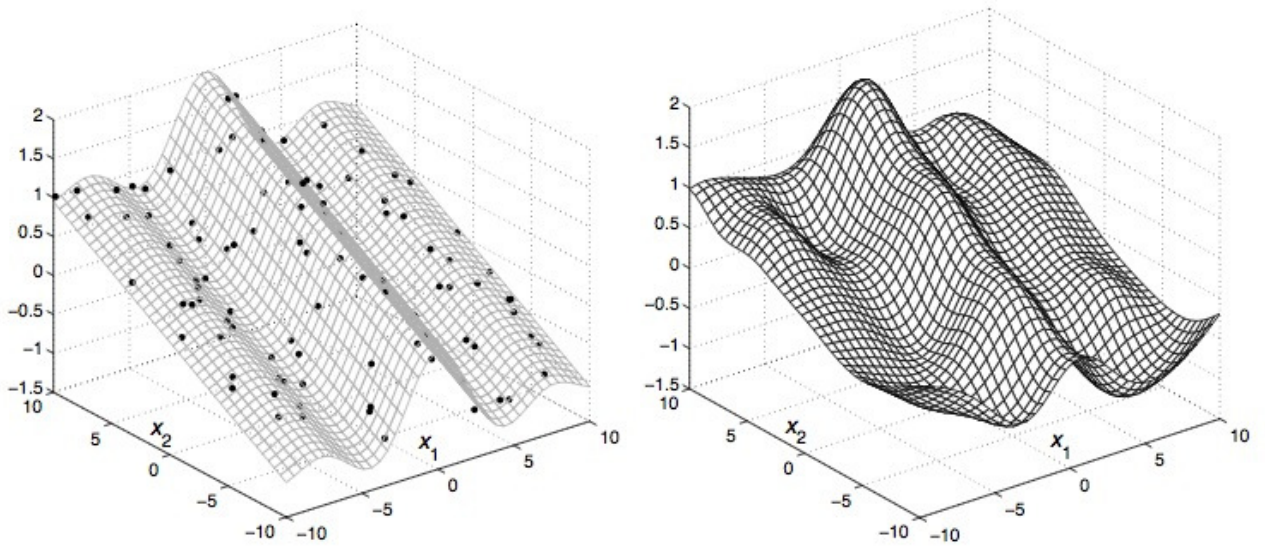

Figure 4: LEFT: the function $\text{sinc}(x_1) + 0.1x_2$ along with the training data, and RIGHT: its approximation by a support vector model with a Gaussian kernel ($r = 3$).

Extensões

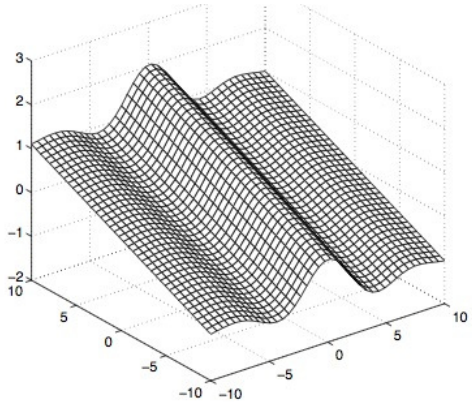

Figure 5: Approximation by a relevance vector model, with additional linear inputs and optimisation of the input scales. The noise estimate was $\sigma = 0.101$.

Model	error	# functions
SVM	0.0194	75
RVM	0.0053	8

Table 1: Root-mean-square error and number of basis functions required in approximation of the function $\sin(x_1)/x_1 + 0.1x_2$, using an SVM, and an RVM with additional linear input functions and optimised input scale parameters.

Comparação - Regressão

			errors		*vectors*	
Data set	N	d	**SVM**	**RVM**	**SVM**	**RVM**
Sinc (Gaussian noise)	100	1	0.378	0.326	45.2	6.7
Sinc (Uniform noise)	100	1	0.215	0.187	44.3	7.0
Friedman #2	240	4	4140	3505	110.3	6.9
Friedman #3	240	4	0.0202	0.0164	106.5	11.5
Boston Housing	481	13	8.04	7.46	142.8	39.0
Normalised Mean			1.00	**0.86**	1.00	**0.15**

Correção: erros do SVM e RVM para a função sinc são
0.0378, 0.0326;
0.0215, 0.0187

Comparação - Classificação

			errors		*vectors*	
Data set	N	d	**SVM**	**RVM**	**SVM**	**RVM**
Pima Diabetes	200	8	20.1%	19.6%	109	4
U.S.P.S.	7291	256	4.4%	5.1%	2540	316
Banana	400	2	10.9%	10.8%	135.2	11.4
Breast Cancer	200	9	26.9%	29.9%	116.7	6.3
Titanic	150	3	22.1%	23.0%	93.7	65.3
Waveform	400	21	10.3%	10.9%	146.4	14.6
German	700	20	22.6%	22.2%	411.2	12.5
Image	1300	18	3.0%	3.9 %	166.6	34.6
Normalised Mean			1.00	**1.08**	1.00	**0.17**

Priori Sobre os Pesos

Pra priori "Gamma" sobre os hiperparâmetros, podemos integrar sobre α independentemente para cada peso w_i, para obter uma priori sobre os pesos:

$$p(w_i) = \int p(w_i|\alpha_i)p(\alpha_i)d\alpha_i$$

$$p(w_i) = \frac{b^a \Gamma(a+\frac{1}{2})}{(2\pi)^{1/2}\Gamma(a)}(b + w_i^2/2)^{-(a+\frac{1}{2})}$$

Essa é uma densidade "student's t" sobre os w_i.

Priori Sobre os Pesos

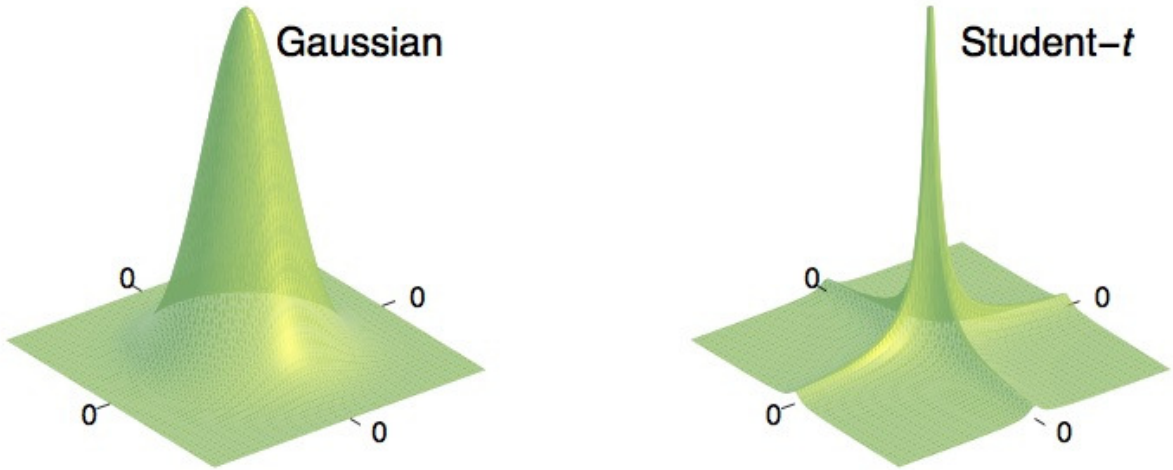

Figure 6: LEFT: an example Gaussian prior $p(\mathbf{w}|\boldsymbol{\alpha})$ in two dimensions. RIGHT: the prior $p(\mathbf{w})$, where the hyperparameters have been integrated out to give a product of Student-t distributions. Note that the probability mass is concentrated both at the origin and along 'spines' where one of the two weights is zero.

Implementações

Com as descrições detalhadas do algoritmo original (no artigo de Tipping de 2001) e do algoritmo mais rápido (no artigo de Tipping e Faul de 2003), é possível escrever um programa para implementar o RVM. Mas não é necessário, pois algumas implementações estão disponíveis

- Através do site dele, Tipping disponibiliza uma implementação do RVM. Essa implementação utiliza o algoritmo mais rápido descrito no artigo de Tipping e Faul (2003).

(continua no próximo slide)

Implementações

- Ainda há páginas sobre uma biblioteca para C++ chamada `Kernel-Machine Library`, mas o código não está disponível. Parece que a biblioteca incluía um RVM para regressão.
- A biblioteca `dLib` para C++ contém uma implementação do RVM para classificação e para regressão. Utiliza o novo algoritmo do artigo de Tipping e Faul (2003).
  Exemplo de classificação:
  `http://dlib.net/rvm_ex.cpp.html`
  Exemplo de regressão:
  `http://dlib.net/rvm_regression_ex.cpp.html`

(continua no próximo slide)

Implementações

- O pacote `kernlab` pro `R` tem uma implementação do RVM para regressão. Veja `http://www.inside-r.org/packages/cran/kernlab/docs/rvm`
- Um programador fez uma versão em python da implementação (para Matlab) de Tipping, utlizando a API `scikit-learn`. Veja `https://github.com/JamesRitchie/scikit-rvm`

Patente

Tipping e Microsoft patentearam um algoritmo de RVM em 1999. A patente, que pertence à Microsoft, vence em setembro de 2019.

`http://www.google.com/patents/US6633857`

PORÉM... Existem implementações “software livre” em C++, python e Matlab, essa última sendo GPLv2 e disponível no site de Tipping...

(continua no próximo slide)

Patente

Há um comentário numa discussão sobre `scikit-learn` em que um programador alega ter perguntado para Tipping sobre a patente. Segundo o programador, a patente cobre o algoritmo original, mas **não** o novo algoritmo descrito no artigo de Tipping e Faul (2003). A existência das implementações “software livre” faria mais sentido nesse contexto. Parece que todas as implementações mencionadas aqui utilizam o novo algoritmo.

`https://github.com/scikit-learn/scikit-learn/issues/1513`

Veja o comentário do usuário **jhallock7** em 7 de agosto de 2015 (“Aug 7, 2015”).

Patente

Enviei um e-mail para Tipping, expressando minha preocupação com a patente. Perguntei se o RVM pode ser utliizado num ambiente acadêmico e sob quais condições, e se pode ser utilizado num ambiente de negócios e sob quais condições. Se Tipping responder, atualizarei esta apresentação com a resposta.

Conclusões

- Resultados comparáveis com outros métodos de “statistical learning”
- Modelos “aprendidos” são bem esparsos
- Classificação: obtemos a probabilidade *a posteriori* de pertencer a uma classe
- Não há limites no número e tipo de funções de base que podemos usar (mas temos que considerar recursos computacionais)

Referências

O artigo principal que li é

- Tipping, M. E. (2001). Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research* **1**, 211–244.

No site de Tipping, há uma página sobre RVMs:
`http://miketipping.com/sparsebayes.htm`
Vários artigos e apresentações sobre RVMs e uma implementação do RVM para Matlab estão disponíveis lá.

Referências

- Faul, A. C. e M. E. Tipping (2002). Analysis of sparse Bayesian learning. Em T. G. Dietterich, S. Becker, and Z. Ghahramani (Eds.), *Advances in Neural Information Processing Systems 14*, pp. 383–389. MIT Press.

Faul e Tipping consideram alguns detalhes da verossimilhança marginal e sua maximização.

Referências

- Tipping, M. E. and A. C. Faul (2003). Fast marginal likelihood maximisation for sparse Bayesian models. In C. M. Bishop and B. J. Frey (Eds.), Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics, Key West, FL, Jan 3-6.

Tipping e Faul apresentam um novo algoritmo para a maximização da verossimilhança marginal. Segundo o site de Tipping, o novo algoritmo é mais rápido do que o original do artigo de Tipping de 2001, utiliza menos memória no computador e "poda" as funções-base analiticamente e não numericamente.