Máquinas de Vetores de Suporte

Prof. Marcelo Keese Albertini
Faculdade de Computação
Universidade Federal de Uberlândia

19 de Junho de 2017

Máquinas de Vetores de Suporte

- Support Vector Machines (SVM)
- O que é?
- Perceptron revisitado
- Kernels (núcleos)
- Otimização de pesos
- Trabalhando com ruídos

Tópicos envolvidos

- Teoria de aprendizado
- Teoria de kernels (núcleos)
- Otimização quadrática com restrições

O que é SVM?

- Três partes:
  - Um subconjunto dos exemplos de treino chamados de **vetores de suporte**
    - cada **vetor** é um exemplo
  - Um vetor de pesos para os exemplos α
  - Uma função de similaridade $K(x, x')$ chamada de **kernel**

Predição de classe para um novo exemplo x_n é um tipo de votação com pesos:

$$f(x_n) = sinal\left(\sum_i \alpha_i y_i K(x_n, x_i)\right)$$

onde as classes podem ser

$$y_i \in \{-1, 1\}$$

e $sinal(x) = 1$ se $x \geq 0$ e $sinal(x) = -1$ se $x < 0$ e

O que é SVM?

- Então SVM é uma forma de aprendizado baseado em instâncias.
- Às vezes são apresentados como uma generalização do perceptron.
- Qual é a relação entre perceptrons e aprendizado baseado em instâncias?

SVM $\sim$ perceptron + kernel + margem + variáveis de folga

Perceptron revisitado

O perceptron é um caso especial de kNN ponderado que obtém-se quando a função de similaridade é o produto interno:

$$f(x_q) = sinal\left[\sum_j w_j x_{qj}\right]$$

Porém

$$w_j = \sum_i \alpha_i y_i x_{ij}$$

Então

$$f(x_q) = sinal\left[\sum_j \left(\sum_i \alpha_i y_i x_{ij}\right) x_{qj}\right] = sinal\left[\sum_i \alpha_i y_i (x_q \cdot x_i)\right]$$

Outra perspectiva da SVM

- Pegar o perceptron
- Substituir o produto interno com uma função de similaridade qualquer
- Agora tem um algoritmo de aprendizado mais capaz
- Matriz kernel: $K(x, x')$ para x, $x' \in \mathit{Dados}$
- Se uma matriz simétrica K é semi-definida positiva (ou seja, não tem auto-valores não-positivos), então $K(x, x')$ é ainda um produto interno, mas em um espaço transformado:

$$K(x, x') = \phi(x) \cdot \phi(x')$$

- Também garante resultar em um problema de otimização convexo
- "Truque" aplicável a muitas situações

Exemplos de Kernels

- Linear

$$K(x, x') = x \cdot x'$$

- Polinomial

$$K(x, x') = (x \cdot x')^d$$

- Normal (Gaussiana):

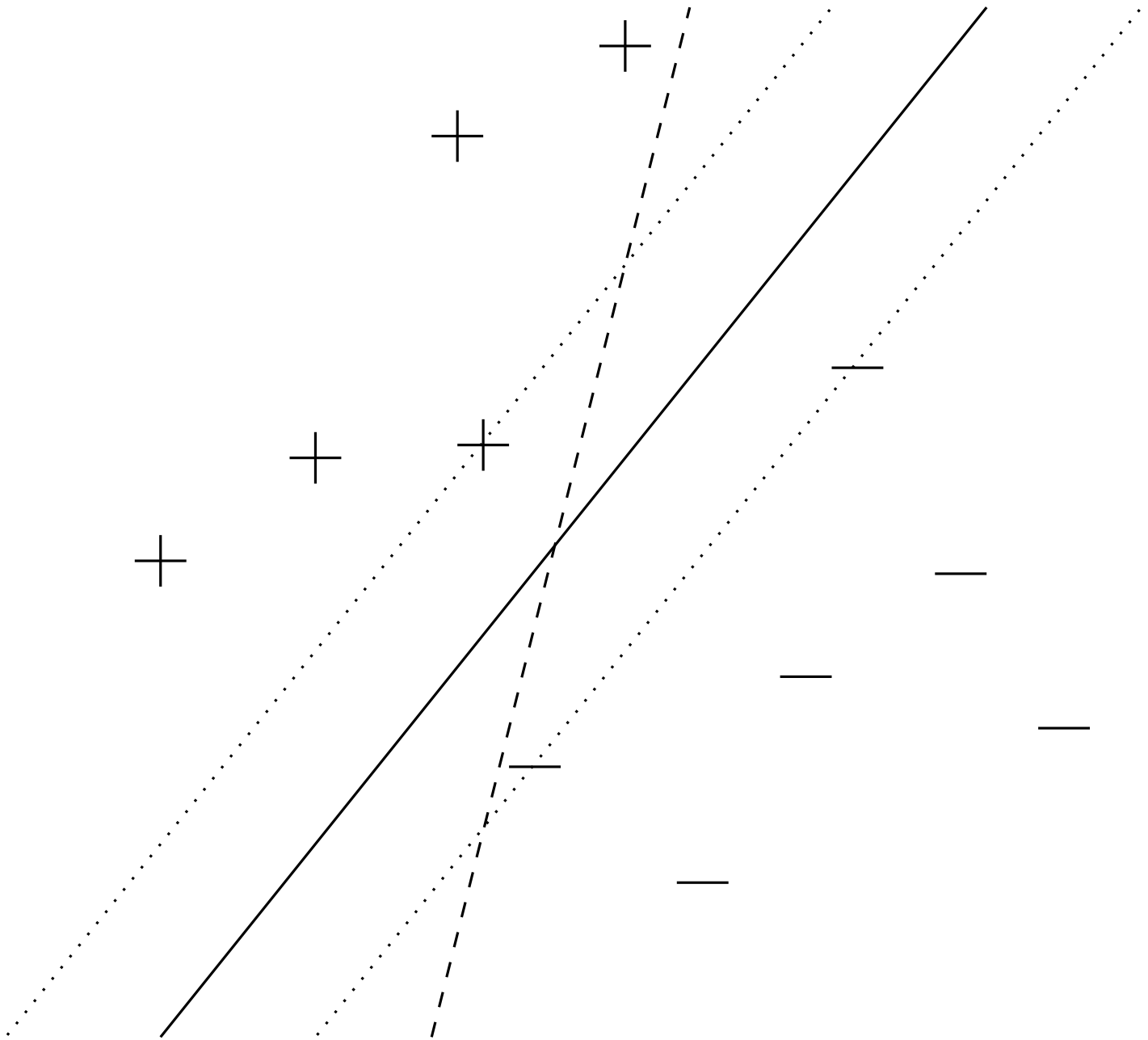
$$K(x, x') = \exp\left(-\frac{1}{2}\|x - x'\|/\sigma\right)$$

Aprendizado de SVMs

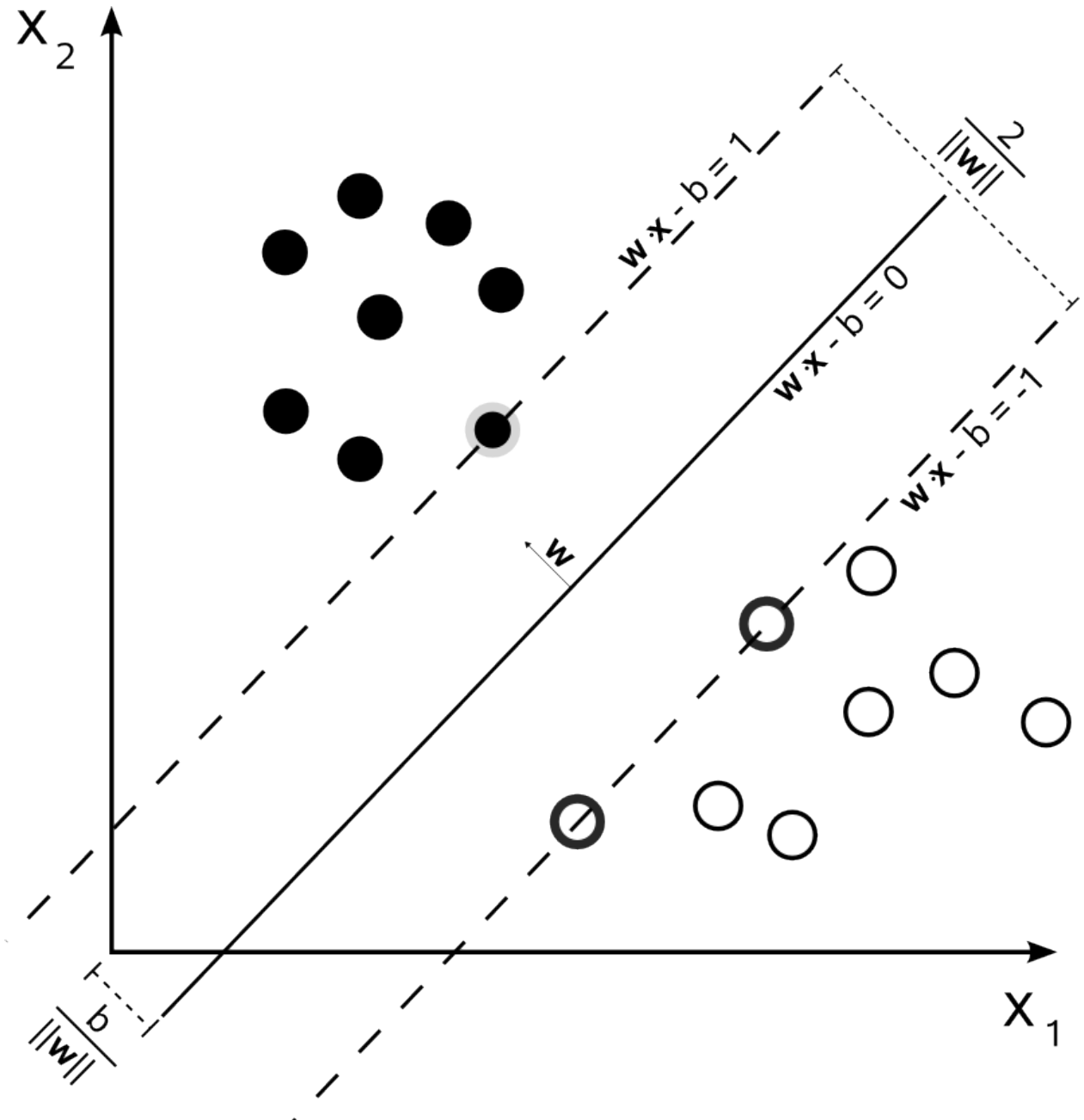
Como

- Escolher o kernel? Depende da aplicação
- Escolher os exemplos? Efeito colateral da escolha de pesos
- Escolher os pesos? Maximizar a margem

Maximizando a margem



Maximizando a margem (imagem: wikipedia)



Problema da otimização dos pesos

- Distância à margem é $\frac{2}{||w||}$.
- Queremos aumentar a margem e portanto reduzir $||w||$.
- Em vez de usar $||w||$, usar o mais conveniente $||w||^2 = w \cdot w$

Formulação do problema

Minimizar $w \cdot w$

Sujeito às condições $y_i(w \cdot x_i + b) \leq 1$ para todo i

- Podemos transformar as condições em parte da função de minimização

Otimização com restrições simplificada

- Quando temos um problema da forma:

  Minimizar $f(w)$

  Sujeito a $h_i(w) = 0$ para $i = 1, 2, \ldots$
- Na solução w^*, $\nabla f(w^*)$ deve estar no subespaço gerado por $\{\nabla h_i(w^*) : i = 1, 2, \ldots\}$
- Podemos usar no lugar da função objetivo original, a **função lagrangeana**

$$L(w, \alpha) = f(w) + \sum_i \alpha_i h_i(w)$$

- Os α_i são os multiplicadores de Lagrange
- Resolver $\nabla L(w^*, \alpha^*) = 0$

Formulação primal

- Queremos transformar as condições em parte da função de minimização para ficar mais fácil encontrar a solução.

Formulação primal

Minimizar $w \cdot w$

Sujeito às condições $y_i(w \cdot x_i + b) \leq 1$ para todo exemplo de treino i

Formulação dual com a função lagrangiana

Minimizar em w **e** b **e maximizar em** α

$$L(w, b, \alpha) = w \cdot w - \sum_i \alpha_i (y_i(w \cdot x_i + b) - 1)$$

Sujeito às condições $\alpha_i \geq 0$ para todo exemplo de treino i

Otimização

Formulação dual com a função lagrangiana

Minimizar em w e b e maximizar em α

$$L(w, b, \alpha) = w \cdot w - \sum_i \alpha_i (y_i (w \cdot x_i + b) - 1)$$

Sujeito às condições $\alpha_i \geq 0$ para todo exemplo de treino i

No ponto ótimo de $L(w, b, \alpha)$ temos

- $\frac{\partial L}{\partial b} = 0$ e $\frac{\partial L}{\partial w} = 0$
- A partir do que podemos obter:

  $\sum_i \alpha_i y_i = 0$ e $2w = \sum_{i=1}^{M} \alpha_i y_i x_i$

Problemas primais e duais

- Problema sobre w é o **primal**
- Resolver equações para w e substituir
- Problema resultante sobre α é o **dual**
- Se for mais fácil, resolver o dual em vez do primal
- Em SVMs:
  - Problema primal é sobre pesos dos atributos
  - Problema dual é sobre os pesos das instâncias

Formulação dual

- Maximizar em $\alpha_i \geq 0$ para cada exemplo i:

$$L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j) - \lambda \sum_{i} \alpha_i y_i$$

- Ao fazer a maximização de $L(\alpha)$ obtemos para cada exemplo i de treino um α_i
- Com os α_i, obtemos w que define a margem:

$$\vec{w} = \sum_{i=1}^{n} \alpha_i y_i \vec{x_i}$$

- Então a classificação de um exemplo novo x_n é realizada de acordo com o sinal de

$$b + \sum_{i} \alpha_i y_i K(x_i, x_n)$$

- onde $b = \vec{w} \cdot \vec{x_i} - y_i$

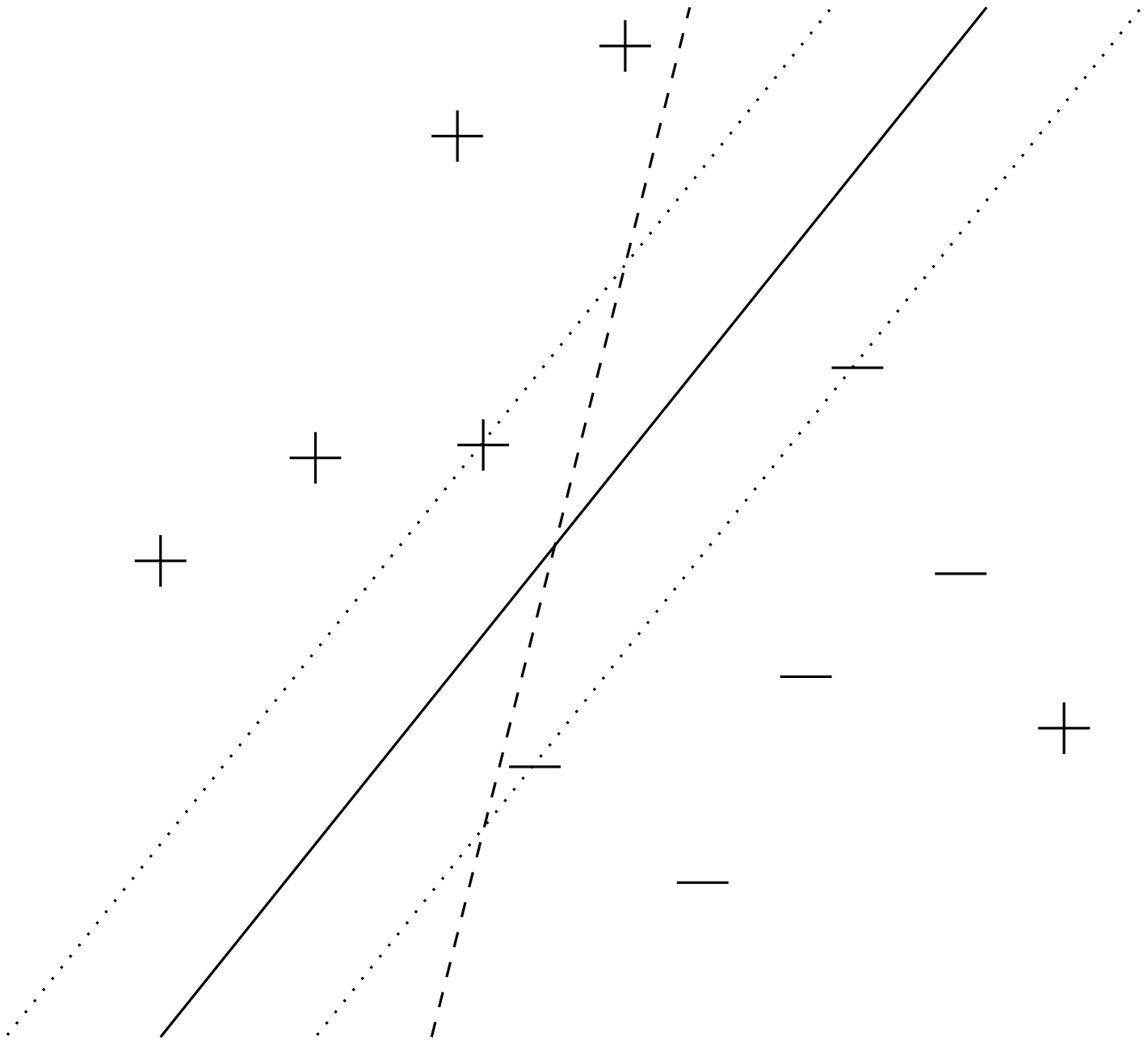
Técnicas de resolução

- Opções:
  - Usar algoritmo genérico para resolver problemas quadráticos
  - Usar algoritmo de otimização especializado

Exemplo SMO (Otimização Minimal Sequencial)

- Método mais simples: atualizar um α_i por vez
- O que viola restrições
- Iterar até convergência:
  - Encontrar exemplo x_i que viola restrições de KKT
  - Escolher segundo exemplo x_j heuristicamente
  - Otimizar juntos α_i e α_j

Considerando ruídos



Considerando ruídos

- Usar **variáveis de folga**: ξ_i

  Minimizar

$$w \cdot w + C \sum_i \xi_i$$

  Sujeito a

$$y_i(w \cdot x_i) \geq 1 - \xi_i \text{ para todo exemplo } i$$

Provavelmente Aproximadamente Correto

Erro de generalização descresce com a margem

$$\epsilon(m, \gamma, \delta) = \frac{1}{m}\left(k \log(\frac{8em}{k}) \log(32m) + \log(\frac{8m}{\delta})\right)$$

- ou seja, teremos $(1 - \delta)$ de confiança do erro ser no máximo $\epsilon(m, \gamma, \delta)$
- para m exemplos e $\gamma = 1/||w||$

  onde $k = \lfloor 577R^2/\gamma^2 \rfloor$,

  R é o raio da distribuição de suporte dos exemplos,

Referência

Simple Learning Algorithms for Training Support Vector Machines, C. Campbell e N. Cristianini

Máquinas de Vetores de Suporte: resumo

- O que é uma máquina de vetores de suporte?
- O perceptron revisitado
- Funções kernel
- Otimização de pesos
- Considerando ruídos
- Provavelmente Aproximadamente Correto

SVM: pacote e1071

```
require(e1071)
data(iris)
model <- svm(Species ~ ., data = iris)
print(model)

##
## Call:
## svm(formula = Species ~ ., data = iris)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.25
##
## Number of Support Vectors:  51
```

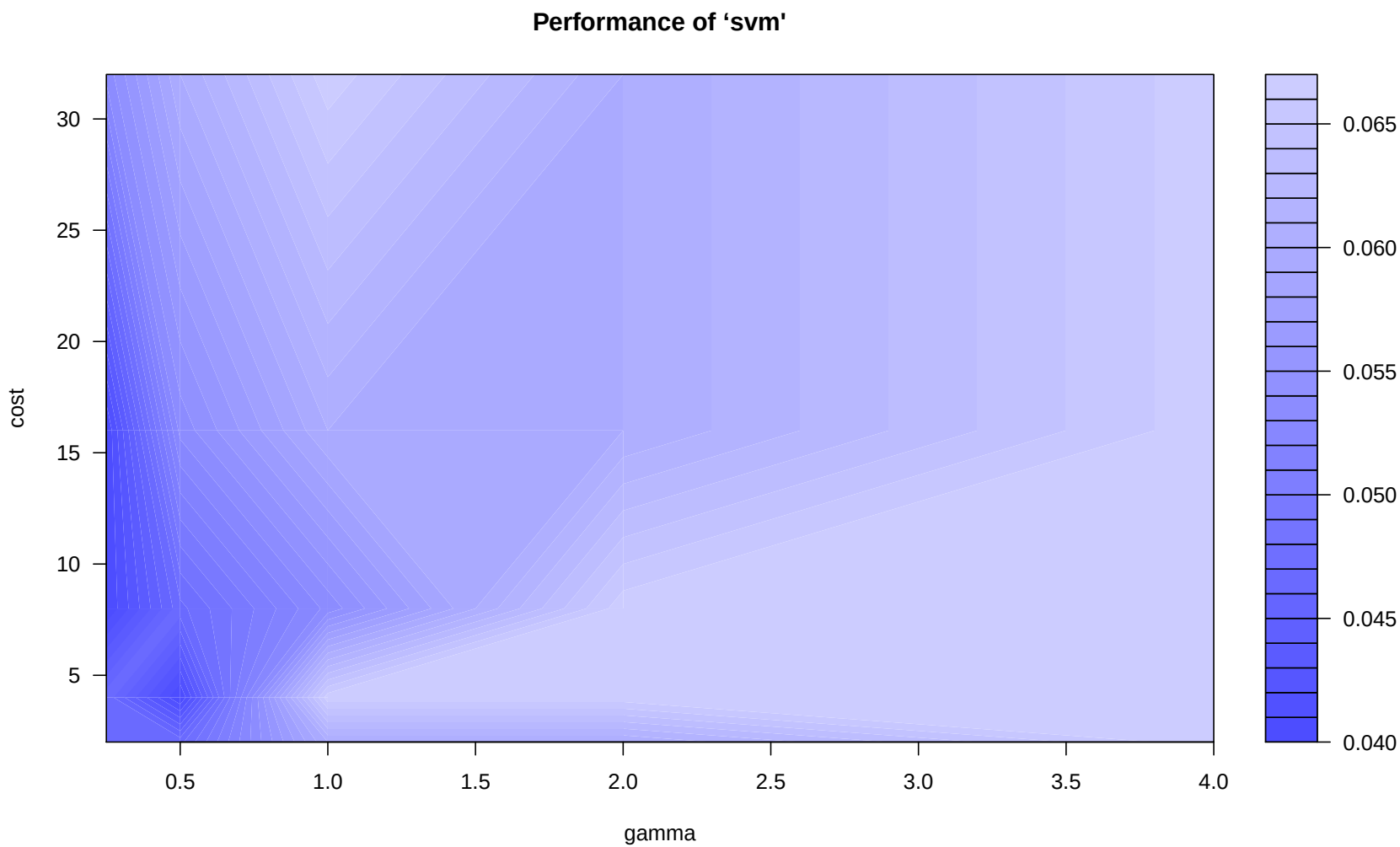
```
summary(model)

##
## Call:
## svm(formula = Species ~ ., data = iris)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.25
##
## Number of Support Vectors:  51
##
##  ( 8 22 21 )
##
##
## Number of Classes:  3
```

```r
table(predict(model, iris),iris$Species)
```

```
##
##                setosa versicolor virginica
##   setosa           50          0         0
##   versicolor        0         48         2
##   virginica         0          2        48
```

```
obj <- tune(svm, Species~., data = iris,
            ranges = list(gamma = 2^(-2:2), cost = 2^(1:5)))
plot(obj)
```



```r
tunsvm <-svm(Species ~., iris,gamma=0.5,cost=4,cross=10)
1-tunsvm$tot.accuracy/100 # errorEstimated

## [1] 0.04666667

table(predict(tunsvm, iris),iris$Species)

##
##                setosa versicolor virginica
##   setosa           50          0         0
##   versicolor        0         48         1
##   virginica         0          2        49
```