ENTRENAMIENTO DE MAQUINAS DE SOPORTE VECTORIAL

Jesús Alejandro Flores Hernández, Ricardo Armando Barrera Cámara, José Alonso Pérez Cruz, Carlos de la Cruz Dorantes

Universidad Autónoma del Carmen CA Tecnologías de la Información, CA Ciencias de la Computación, Calle 31 s/n Ciudad del Carmen, Campeche, México
{jflores, rbarrera, japcruz, cdelacruz}@pampano.unacar.mx

Abstract. Este trabajo describe el entrenamiento de maquinas de soporte vectorial que permiten discriminar puntos en un conjunto, usando el algoritmo de chunking suave, este inicia con un subconjunto aleatorio de vectores para hallar los vectores de soporte que nos den una separación para dos subconjuntos del conjunto dado. Dada la muestra aleatoria se procede (según el algoritmo) a construir hiperplanos de separación entre los subconjunto a partir de la muestra inicial cambiando los vectores de la muestra de ser necesario, hasta tener los que mejor sirvan para trazar el hiperplano; si en la muestra se tuviese vectores mas cercanos a los de soporte, el proceso de entrenamiento seria mas rápido, en este trabajo se analizan los resultados para algunos conjuntos de datos de un preprocesamiento que permita seleccionar un conjunto no tan aleatorio de vectores iniciales para reducir el tiempo de entrenamiento, se revisan los algoritmos de GetBorder, TransRed y algoritmos genéticos.

Keywords: Inteligencia artificial, Máquinas de soporte vectorial, Algoritmos genéticos, Optimización y Clasificación.

1 Introducción

El reconocimiento de patrones se realiza usando uno de los tres enfoques siguientes: Estadístico. Se extrae un conjunto de características (rasgos) de los datos de entrada y se utilizan para clasificar los datos. El modelo obtenido consiste en probabilidades o funciones de densidad de probabilidad; Sintáctico. Consiste en relacionar la estructura de los patrones con la sintaxis de un lenguaje definido formalmente; Neuronal.
Consiste tomar la manera en que los sistemas neuronales biológicos clasifican los datos.

Necesitamos entonces con alguno de estos enfoques construir una que pueda clasificar los patrones. Esto nos lleva a la necesidad de entrenar la máquina a clasificar los datos.

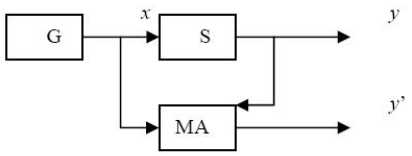

Fig. 1. Modelo de aprendizaje, G es un generador de vectores aleatorios, S es un supervisor y MA es una máquina de aprendizaje.

Durante la etapa de aprendizaje, una máquina de aprendizaje (MA) observa las parejas de entrenamiento (x,y) y después del entrenamiento es capaz de, para un valor é cerca de la respuesta del supervisor. Para escoger la mejor aproximación posible a la respuesta del supervisor, se mide la discrepancia entre la respuesta del supervisor y la de la máquina, el objetivo es encontrar una función que minimice el funcional del riesgo (que es el valor esperado de la discrepancia). G es un generador de vectores aleatorios obtenidos a partir de una función de probabilidad desconocida, S es el supervisor que regresa un valor de clasificación para cada vector de entrada y MA es una máquina de aprendizaje capaz de implementar un conjunto de funciones f(x,w) con w un conjunto de parámetros.

2 SVM

Los SVM son una herramienta usada en la clasificación de objetos puntuales de dos clases que ha resultado ser muy poderosa, se basa en encontrar una superficie de clasificación determinada por ciertos puntos de un conjunto de entrenamiento, este conjunto de vectores debe estar en la frontera entre los dos subconjuntos en que se clasificarán los puntos, estos vectores son llamados vectores de soporte y describen

Las SVM son un paradigma aparte de la Redes Neuronales, pero a pesar de tener similitudes están mejor fundamentadas en la teoría y tienen mucho mejor capacidad de generalización.

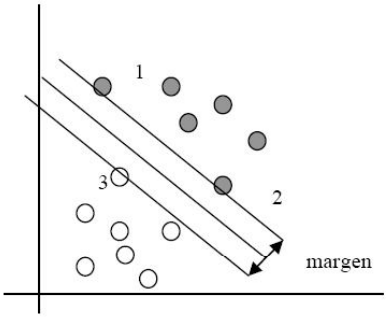

Fig. 2. Conjunto de vectores separados por un hiperplano (en este caso una línea = hiperplano en dos dimensiones).

En la grafica se muestran los 3 vectores de soporte para esta muestra en el plano. Para obtener la superficie de separación se deberá resolver un problema de programación cuadrática convexa, con restricciones lineales, cuyo número de variables es igual al número de datos de entrenamiento. construir una SVM que sepa mapear los valores de los vectores xi a los valores de las etiquetas yi correspondientes, por medio de una función de decisión y = f : Rn {1,- [1]. los puntos a clasificar están en Rn y debemos clasificarlos en uno u otro subconjunto {1, 1}. Formulando el problema de clasificación con SVM, dentro del marco de la teoría de optimización, podemos aprovechar métodos bien establecidos para resolverlo, se sabe además que según el teorema de Lagrange, un problema de optimización no lineal puede (con alguna restricción) resolverse por medio de la función de lagrange dual (Lagrangiano Dual):

$$\max(\alpha) L_D = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \quad (1)$$

Sujeto a

$$\sum_{i=1}^{l} \alpha_i y_i = 0$$

$$\alpha_i \geq 0 \text{ para } i = 1,2,\ldots,l$$

La solución a este problema será planteada en estos términos.

3 Entrenamiento

Para entrenar las SVM como se dijo anteriormente, se formula el problema por medio de programación cuadrática convexa, resolver un problema de este tipo con unos pocos miles de datos demanda muchos recursos de cómputo, para problemas prácticos, el número de datos de entrenamiento puede exceder 50,000, lo cual nos plantea un problema grave en el uso de recursos de máquina para entrenar una SVM. En estos casos se usan los métodos de Conjunto de trabajo, se puede utilizar entre otros el método de chunking, que consiste en tomar un conjunto arbitrario inicial de vectores a partir de los cuales se entrena el SVM, a continuación se toma un nuevo pedazo, incluyendo los vectores fuera del primer pedazo que mas violen las condiciones del problema, se itera hasta tener un conjunto óptimo.

1

reconocimiento de CIMAT, julio de 2002.

4 Preprocesamiento.

El conjunto de trabajo inicial, se elige aleatoriamente lo que no representa mucho problema si el número de puntos es muy parecido en cada clase, sin embargo para casos como el que se plantea una clase con un número proporcionalmente menor de vectores que la otra, muchos vectores si no todos, pueden pertenecer a la misma clase, con lo que se obtienen resultados muy malos en las primeras iteraciones que pueden hacer muy lento el algoritmo. Para esto existen técnicas de selección que intentan encontrar un mejor conjunto de trabajo inicial. En el paper Acelerater Training Support Vector escrito por Rycheltsky, Ortmann, Ullman y Glesner. De la universidad tecnológica Darmstadt, mencionan dos métodos de reducción del tiempo de entrenamiento de las SVM, usando el método de pedaceo, pues mencionan que si los posibles vectores de soporte se conocen de antemano se puede acelerar el entrenamiento usando pedaceo, los métodos mencionados son: TransRed y GetBorder.

TransRed.

Este algoritmo encuentra vectores que son posibles vectores de soporte, por medio de la estimación de la posición del hiperplano, el cual es construido en un alto espacio dimensional durante la optimización SVM. El plano estimado es usado para calcular la distancia de los vectores de entrenamiento a este plano. Los vectores son seleccionados por su distancia al plano para el primer pedazo. La manera más simple consiste en dividir en dos clases la muestra, en un espacio de dimensión alta, después aproximar las áreas de clase por hiperesferas, entonces la posición y penetración de las dos esferas puede ser calculada. La primera aproximación del radio y centro de las esferas es calculado.

Los puntos se dividen en Xmas y Xmenos

```
%calcular el radio de la esfera para Xmas rmas=0;
aux=0;
for i=1:length(Xmas)
   aux1=Cmas(1)*Cmas(1)+Cmas(2)*Cmas(2);
   aux2=-2*(Cmas(1)*Xmas(i,1)+Cmas(2)*Xmas(i,2));
   aux3=Xmas(i,1)*Xmas(i,1)+Xmas(i,2)*Xmas(i,2);
   aux=aux1+aux2+aux3;
   if aux>rmas
      rmas=aux;
    end
end
rmas=sqrt(rmas);
%centro de la esfera Xmenos
Cmenos=[0 0];
for i=1:length(Xmenos)
   Cmenos(1)=Cmenos(1)+Xmenos(i,1);
   Cmenos(2)=Cmenos(2)+Xmenos(i,2);
end
Cmenos(1)=Cmenos(1) / length(Xmas);
Cmenos(2)=Cmenos(2) / length(Xmas);
```

Se repite esto para Xmenos. El radio de la hiperesfera es dado por la distancia del punto medio de la clase al más distante vector en la misma clase. Esto puede ser determinado mediante el cálculo de la distancia máxima de todos los vectores en elconjunto Tr+ y Tr- a su media. Las distancias pueden ser calculadas por:

```
% Calcular el radio de las hiperesferas rhXmas=0;
for i=1:length(Xmas)
   aux=kernel(Cmas,Cmas)-
   2*kernel(Cmas,Xmas(i,:))+kernel(Xmas(i,:),Xmas(i,:));
   if aux>rhXmas
          rhXmas=aux;
   end
end
rhXmas=sqrt(rhXmas);
% para la hiperesfera Xmenos rhXmenos=0;
for i=1:length(Xmenos)
   aux=kernel(Cmenos,Cmenos)-
   2*kernel(Cmenos,Xmenos(i,:))+kernel(Xmenos(i,:),Xm(i,:))
   ;
   if aux>rhXmenos
          rhXmenos=aux;
   end
end
rhXmenos=sqrt(rhXmenos);
```

Enseguida se construye un plano M que aproxime el hiperplano de separación, y éste deberá ser encontrado por la optimización de los vectores de soporte.

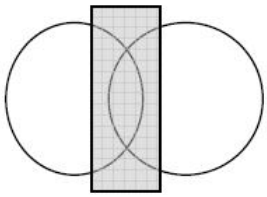

Fig. 3. Dos conjuntos de puntos en las hiper esfereas (circulos en este caso), el área sombrada corresponde al hiperplano de separación

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%calcular m+, m-, la distancia entre el plano y el
%centro de las calses
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Mmas=(dmasmenos + (rhXmas - rhXmenos)) / 2;
%Mmas=(dmasmenos)/2; Mmenos=dmasmenos - Mmas;
%calculando las distancias de los vectores Xmas al
hiperplano
distXmas=zeros(length(Xmas),1);
```

```
dmxi=zeros(length(Xmas),1);
%b+0= (K(xc+, xc+)-K(xc+,xc-))/d+-
bias=(Kdexcixcj(Xmas,Xmas)-
Kdexcixcj(Xmas,Xmenos))/dmasmenos;
% la distancia a este plano se puede escribir como
for i=1:length(Xmas)
 %d de M+<->x de a
 distXmas(i)=(1/dmasmenos*(kernel(Cmas,Xmas(i,:))-
 kernel(Cmenos,Xmas(i,:))))+bias;
 aux1=distXmas(i)-Mmas;
 %calculando la norma
 dmxi(i)=sqrt(aux1*aux1);
end
%ordenar los resultados
Maux1=Xmas;
[Y1, I]=sort(dmxi);
for i=1:length(I)
 dmxi(I(i));
 Maux1(i,:)=Xmas(I(i),:);
end
%calculando la distancia de los vectores Xmenos al
hiperplano
distXmenos=zeros(length(Xmenos),1);
dmxi2=zeros(length(Xmenos),1);
%bias es el misnmo
for i=1:length(Xmas)
  distXmenos(i)=(1/dmasmenos*(kernel(Cmas,Xmenos(i,:))-
  kernel(Cmenos,Xmenos(i,:))))+bias;
  aux1=distXmenos(i)-Mmenos;
  %calculando la norma
  dmxi2(i)=sqrt(aux1*aux1);
end
%ordenar los resultados
Maux2=Xmenos;
[Y1, I]=sort(dmxi2);
for i=1:length(I)
 dmxi2(I(i));
 Maux2(i,:)=Xmenos(I(i),:);
end
```

Los primeros vectores de la lista que son los más próximos al hiperplano de separación se tomarán como conjunto inicial de la muestra.

Entrenamiento Chungking suave				
	Tiempo de entrenamiento			
	Con preprocesamineto			Entrenamiento sin preprocesamiento
Conjunto de datos	Preprocesamiento	Entrenamiento	Total	
A	1.94	0.24	2.18	1.51
B	1.92	0.28	2.2	1.52
C	1.60	0.49	2.09	3.19

D	1.74	0.72	2.46	9.02

Se aprecia una mejora de entre 7 y 18 por ciento en promedio aunque en algunos casos el tiempo total (preprocesamiento + entrenamiento es mayor) que al hacerlo sin preprocesamiento.

GetBorder

Este método desarrollado para determinar un posible primer pedazo para el algoritmo de pedaceo, al igual que TransRed trabaja con un espacio de dimensión alta pero sin usar un hiperplano, TransRed de manera optimista asume que los puntos pueden ser descritos en un espacio de dimensión alta, por medio de hiperesferas, lo cual no siempre es cierto. Otra aproximación es buscar los vectores cerca del borde de las dos clases, incluso con una arbitraria distribución de datos en un alto espacio dimensional. En muchos casos sólo los vectores pertenecientes a una clase, los cuales están cerca de los vectores de la otra clase son probables vectores de soporte. Para todos los vectores de entrenamiento en la clase, iniciamos buscando un vector de la clase contraria, que tenga la menor distancia, en la proyección del espacio de alta dimensión. En el segundo paso se toma una lista con los vectores cercanos y se repite con el conjunto contrario, estos son los candidatos al primer pedazo.

Se calcula el vector de distancias

```
%vector de distancias
vdXmas=zeros(length(X
mas),1); Maux1=Xmas;
%Vector de soporte para Xmas segun getBorder
dmenorXmas=0;
dmenorXmas=kernel(Xmas(1,:), Xmenos(1,:));
for i=1:length(Xmas)
   for j=1:length(Xmenos)
      aux=kernel(Xmas(i,:),Xmenos(j,:));
      vdXmas(i)=vdXmas(i)+aux;
      if aux<dmenorXmas
         dmenorXmas=aux; XmasSoporte=Xmas(i,:);
      end
   end
   vdXmas(i)=vdXmas(i)/length(Xmas);
   %distancia promedio del vec i a los j
end
%ordenar las distancias medias de menor a mayor para
obtener los vectores de soporte
[Y1, I]=sort(vdXmas);
%creamos la matriz para los vectores de soporte
vAux=[1:cuantos; 1:cuantos];
vsopXmas=vAux';
vsopXmenos=vAux';
for i=1:cuantos
%plot(XmasSoporte(1),XmasSoporte(2),'gx','LineWidth',4);
 plot(Xmas(I(i),1),Xmas(I(i),2),'g*','LineWidth',4);
```

```
 vsopXmas(i,1)=Xmas(I(i),1); vsopXmas(i,2)=Xmas(I(i),2);
end
%Vector de soporte para Xmenos segun getBorder
dmenorXmenos=0; dmenorXmenos=kernel(Xmenos(1,:),
Xmas(1,:)); vdXmenos=zeros(length(Xmenos),1);
for i=1:length(Xmenos)
   for j=1:length(Xmas)
      aux=kernel(Xmas(i,:),Xmenos(j,:));
      vdXmenos(i)=vdXmenos(i)+aux;
      if aux<dmenorXmenos
          dmenorXmenos=aux;
          XmenosSoporte=Xmenos(j,:);
      end
    end
end
[Y2, I2]=sort(vdXmenos);
TiempoFinal=cputime;
Tiempo=TiempoFinal-TiempoInicial;
fprintf('\nTiempo en GetBorder %3.2f',Tiempo);
set (txtTpo,'String',Tiempo)
XmenosSoporte; % ya no se usa for i=1:cuantos
plot
Xmenos(I2(i),1),Xmenos(I2(i),2),'gx','LineWidth',4);
vsopXmenos(i,1)=Xmenos(I2(i),1);
vsopXmenos(i,2)=Xmenos(I2(i),2);
end
```

Entrenamiento Chungking suave				
	Tiempo de entrenamiento			
	Con preprocesamineto			Entrenamiento sin preprocesamiento
Conjunto de datos	Preprocesamiento	Entrenamiento	Total	
A	0.18	0.22	0.4	1.51
B	0.15	0.28	0.43	1.52
C	0.15	0.96	1.11	3.19
D	0.25	0.96	1.21	9.02

Se aprecia una mejora entre 10 y 30% usando getborder.

5 Aplicación de Algoritmos genéticos en el preprocesamiento.

Nuestro planteamiento consiste en utilizar algoritmos genéticos para encontrar el primer pedazo, los algoritmos genéticos suelen ser una herramienta eficaz aunque no eficiente para encontrar los vectores de soporte, cuando no se tienen métodos establecidos, es una buena opción utilizar algoritmos genéticos, sin embargo si se disponen de otras herramientas, normalmente éstas convergen más rápido a la solución. La idea aquí es que podemos decir que los algoritmos genéticos dan pasos pequeños pero seguros y podemos iterar un número N (por determinar) de veces con algoritmos genéticos lo cual nos dará un conjunto con probabilidad más alta de tener

los vectores de soporte, a partir de ahí, se tomará éste como el primer pedazo para el método de pedaceo, y como éste tendrá una alta probabilidad de contener los vectores de soporte, aumentará (pensamos) la velocidad de convergencia del método de pedaceo.

Se toma una población inicial.

```
%fprintf('Generando poblacion inicial...\n');
%renglones de alfas = genes, columnas =
coromosomas alfas=zeros(tampobla,numgenes);
nuevasalfas=zeros(tampobla,numgenes);
for i=1:tampobla
   for j=1:numgenes
      factor=10^(numdi
      gitos-1);
      alfas(i,j)=0;
      for k=1:numdigitos+numdecimales
alfas(i,j)=alfas(i,j)+floor(random('unif',0,10))*factor;
         factor=factor*0.1;
      end
   end
end
```

para un numero determinado de generaciones se realiza:

```
while (numgenera < maxgenera)
   numgenera=numgenera+1;
   %se calcula la aptitud de la población
   aptitud=zeros(tampobla,1);
   desempate=zeros(tampobla,1);
   for i=1:tampobla
     aptitud(i)=-
0.5*alfas(i,:)*H*alfas(i,:)'+sum(alfas(i,:))-
castigo*(numgenera/maxgenera)*alfas(i,:)*Y;
   end
     % se aplica el operador de cruza
     % se realizan mutaciones
end
```

Entrenamiento Chungking suave				
	Tiempo de entrenamiento			
	Con preprocesamineto			Entrenamiento sin preprocesamiento
Conjunto de datos	Preprocesamiento	Entrenamiento	Total	
A	27.34	0.37	27.71	1.51
B	27.24	0.27	27.51	1.52
C	27.25	0.28	27.51	3.19
D	45.81	2.37	48.18	9.02

Se aprecia una mejora de entre un 8 y 26% en el entrenamiento, sin embargo el tiempo total (preprocesamiento + entrenamiento) es mayor que cuando no se usa preprocesamiento. Enseguida se muestra la distribución de datos para las muestras probadas. Estos algoritmos fueron escritos en MatLab y probados con algunos conjuntos de datos.

6 Conclusiones

Si bien los métodos de TransRed y GedBorder, nos suponen una mejora en el tiempo de entrenamiento, el tiempo total puede ser mayor que el necesario para entrenar sin preprocesamiento, el que más nos interesaba que es el de AG aporta una mejora en el entrenamiento pero tomando en cuenta el tiempo total éste puede ser mayor con preprocesamiento que sin él. Lo que podemos decir a favor del AG es que se adapta muy bien a cualquier conjunto de datos, no así los otros métodos cuya eficacia depende de la distribución de los datos. Podemos concluir el trabajo: a) Que los AG NO ofrecen buen rendimiento para conjuntos pequeños y medianos de datos, b) Los AG se ven rebasados en rendimiento por los métodos de GetBorder y TransRed pero para aplicar éstos se debe conocer la distribución gráfica de los datos; c) En trabajos futuros se puede aplicar otros métodos como las EE. Se puede hacer también una combinación de métodos de preprocesamiento como GetBorder y AG para ver su rendimiento.

References:

1. Efrén Mesura Montes, de la técnica Multiobjetivo NPGA para el manejo de Restricciones en algoritmos Genéticos.
2.

3.
2002.
4. Coello Coello Carlos A a la computación CINVESTAV-IPN archivo PDF. Enero 2004.
5. Thorsten Joachims, Learning to Classify Text Using Support Vector Machines., Dissertation, Kluwer, 2002.
6. T. Joachims, Estimating the Generalization Performance of a SVM Efficiently. Proceedings of the International Conference on Machine Learning, Morgan Kaufman, 2000.