Introducción al Procesamiento de Lenguaje Natural

Grupo de PLN – InCo

Aprendizaje Automático: Clasificación supervisada

Clasificación

Según Jorge Luis Borges (ensayo *El idioma analítico de John Wilkins*, 1952), los animales se clasifican en:
(a) pertenecientes al Emperador,
(b) embalsamados,
(c) amaestrados,
(d) lechones,
(e) sirenas,
(f) fabulosos,
(g) perros sueltos,
(h) incluidos en esta clasificación,
(i) que se agitan como locos,
(j) innumerables
(k) dibujados con un pincel finísimo de pelo de camello,
(l) etcétera,
(m) que acaban de romper el jarrón,
(n) que de lejos parecen moscas.

Clasificación en PLN

- En general, hablamos de clasificación de documentos:
  - artículos
  - noticias
  - oraciones
  - tuits
- Según un conjunto de clases:
  - conjunto de tópicos
  - positivo / negativo
  - es spam / no es spam
- También se hace clasificación a nivel de palabras u otros elementos dentro del texto:
  - clasificar punto como fin de oración o parte de sigla
  - clasificar palabras con su categoría gramatical (POS)
- Determinar la palabra que sigue a una secuencia también puede verse como clasificación (el conjunto de clases es el vocabulario completo).
  - modelos de lenguaje (se verán más adelante)

Clasificación en PLN

- Reglas escritas a mano (enfoques simbólicos)
  - si aparece la palabra “fantástico” es positivo
- Aprendizaje automático supervisado
  - a partir de un conjunto de ejemplos ya clasificados
- Enfoques híbridos
  - por ejemplo, incluir información obtenida mediante reglas como atributos

Representación de los textos

- Para aplicar métodos de aprendizaje automático necesitamos representar los textos como objetos matemáticos.
- En general los representamos como vectores de atributos.
- Usual: atributos de tipo Bag of Words (BoW):
  - el vector es del tamaño del vocabulario
  - con 0 y 1
  - o cantidad de ocurrencias
  - puedo eliminar stop words
  - puedo usar lemas o raíces (stem)
- Vector de atributos que representan características del texto: cantidad de palabras positivas/negativas, largo del texto, cantidad de adjetivos, ...
- Más adelante veremos word embeddings.

Aprendizaje automático supervisado

- Tenemos
  - un conjunto de clases $C = \{c1, \ldots, cM\}$
  - ejemplos anotados (conjunto de entrenamiento) $\{(d1,c1), \ldots, (dN,cN)\}$
- Queremos construir un clasificador que pueda mapear un nuevo documento d a su clase $c \in C$.
- Un método probabilista nos va a dar una distribución de probabilidad sobre todo el conjunto de clases.
- Típicamente, nos vamos a quedar con la de mayor probabilidad.
- O sea que queremos: $\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|d)$

Métodos generativos vs discriminativos

- Existen muchos algoritmos de AA para construir clasificadores.
- Pueden clasificarse en dos familias: Generativos y Discriminativos.
- Los algoritmos generativos, como **Naive Bayes**, intentan modelar cada clase. Dado un nuevo ejemplo buscan la clase que mejor lo modela. Podrían generar nuevos elementos de esa clase en base a lo aprendido.
- Los algoritmos discriminativos, como **Regresión Logística**, aprenden qué atributos (features) de la entrada son los más útiles para discriminar las diferentes clases.
- Aunque los modelos discriminativos suelen ser más precisos, y por lo tanto se usan más, los generativos se siguen usando.

Multinomial Naïve Bayes

Un clasificador bayesiano usa la regla de Bayes para calcular la probabilidad condicional *P(c|d)* en función de otras tres probabilidades.

Queremos: $\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|d)$

Regla de Bayes: $P(x|y) = \frac{P(y|x)P(x)}{P(y)}$

Calculamos: $\underset{c \in C}{\operatorname{argmax}} \frac{P(d|c)P(c)}{P(d)} = \underset{c \in C}{\operatorname{argmax}} P(d|c)P(c)$

La probabilidad del documento *d* es constante.

P(d|c): cómo sería *d* si supiéramos que su clase es *c* .

Multinomial Naïve Ba

Por ejemplo, las palabras incluidas en el documento (BoW).

Se modela cada documento como un conjunto de atributos, por lo que $P(d|c) = P(f1, ..., fN \mid c)$.

Y se hacen algunas simplificaciones (por eso es "ingenuo"): se asume que los atributos son independientes.

Funciona sorprendentemente bien.

$$P(f_1, f_2, ..., f_n|c) = P(f_1|c) \cdot P(f_2|c) \cdot ... \cdot P(f_n|c)$$

Finalmente, se calcula:

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{f \in F} P(f|c)$$

En realidad se aplica el logaritmo:
log P(c) +
sumatoria de log P(wi | c)

Generación del modelo

Se estiman las probabilidades en base a conteos en el corpus de entrenamiento:

Generación del modelo

Se estiman las probabilidades en base a conteos en el corpus de entrenamiento:

$$\hat{P}(c) = \frac{N_c}{N_{doc}}$$

$$\hat{P}(w_i|c) = \frac{count(w_i, c)}{\sum_{w \in V} count(w, c)}$$

Este valor puede ser 0, haciendo que la productoria dé 0.

V es el vocabulario de todo el conjunto de entrenamiento.

Generación del modelo

Se estiman las probabilidades en base a conteos en el corpus de entrenamiento:

$$\hat{P}(c) = \frac{N_c}{N_{doc}}$$

Suavizado de Laplace

$$\hat{P}(w_i|c) = \frac{count(w_i,c)+1}{\sum_{w \in V}(count(w,c)+1)} = \frac{count(w_i,c)+1}{\left(\sum_{w \in V} count(w,c)\right)+|V|}$$

function TRAIN NAIVE BAYES(D, C) **returns** V, log $P(c)$, log $P(w|c)$

for each class $c \in C$ # Calculate $P(c)$ terms
  N_{doc} = number of documents in D
  N_c = number of documents from D in class c
  $logprior[c] \leftarrow \log \frac{N_c}{N_{doc}}$
  $V \leftarrow$ vocabulary of D
  $bigdoc[c] \leftarrow$ **append**(d) **for** d $\in$ D **with** class c
  for each word w in V # Calculate $P(w|c)$ terms
    $count(w,c) \leftarrow$ # of occurrences of w in $bigdoc[c]$
    $loglikelihood[w,c] \leftarrow \log \frac{count(w,c) + 1}{\sum_{w' \text{ in } V} (count(w',c) + 1)}$
return *logprior*, *loglikelihood*, V

function TEST NAIVE BAYES(*testdoc*, *logprior*, *loglikelihood*, C, V) **returns** best c

for each class $c \in C$
  $sum[c] \leftarrow logprior[c]$
  for each position i in *testdoc*
    $word \leftarrow testdoc[i]$
    if $word \in V$
      $sum[c] \leftarrow sum[c] + loglikelihood[word,c]$
return $\text{argmax}_c\ sum[c]$

Figure 4.2 The naive Bayes algorithm, using add-1 smoothing. To use add-α smoothing instead, change the $+1$ to $+\alpha$ for loglikelihood counts in training.

Clasificación de documentos

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable with no fun

Se elimina "with", porque no pertenece a V.

$$P(-) = \frac{3}{5} \qquad P(+) = \frac{2}{5}$$

$$P(\text{"predictable"}|-) = \frac{1+1}{14+20} \qquad P(\text{"predictable"}|+) = \frac{0+1}{9+20}$$

$$P(\text{"no"}|-) = \frac{1+1}{14+20} \qquad P(\text{"no"}|+) = \frac{0+1}{9+20}$$

$$P(\text{"fun"}|-) = \frac{0+1}{14+20} \qquad P(\text{"fun"}|+) = \frac{1+1}{9+20}$$

Hay **14** palabras en los ejemplos positivos

Y **9** en los ejemplos negativos

$$P(-)P(S|-) = \frac{3}{5} \times \frac{2 \times 2 \times 1}{34^3} = 6.1 \times 10^{-5}$$

$$P(+)P(S|+) = \frac{2}{5} \times \frac{1 \times 1 \times 2}{29^3} = 3.2 \times 10^{-5}$$

El modelo predice la clase negativa

Features para Análisis de Sentimiento

Igual que antes, pero...

- No contar múltiples ocurrencias de una palabra en un mismo documento (Binary Naive Bayes).

- Manejo de la negación: analizar "didnt like this movie, but" como "didnt NOT_like NOT_this NOT_movie, but".

- Usar lexicones de sentimiento y usar como features la cantidad de palabras en un lexicón positivo y la cantidad de palabras en un lexicón negativo.

Features para POS tagging

Ejemplo: calcular POS tag, si tengo la palabra, y los POS tags de las palabras anterior y siguiente en el contexto

$$P(C = Adj|X_{ant} = Det, X_{sig} = Nom, X_s = blanco) = P(C = Adj)P(X_s = blanco|C = Adj)P(X_{ant} = Det|C = Adj)P(X_{sig} = Nom|C = Adj)$$

Regresión logística

- En PLN, la regresión logística es como una baseline para los algoritmos de clasificación supervisada.

- Es un modelo que está muy relacionado con redes neuronales.

Regresión logística

Redefinimos el problema de clasificación supervisada, incorporando algunos elementos nuevos:

- Un vector de features que representa la entrada.
- Una función de clasificación que estima la clase de una instancia calculando P(y | x) (por ejemplo, Sigmoide).
- Una función objetivo que queremos optimizar (por ejemplo, la función de pérdida cross-entropy)
- Un algoritmo para hacer la optimización (por ejemplo, descenso por gradiente estocástico).

Regresión logística

Tenemos dos fases

Entrenamiento: entrenamos el sistema, ajustando un vector de pesos w_i y un sesgo (bias) b.

Test: dado un ejemplo x calculamos $P(y|x)$ y retornamos la etiqueta y (clasificación binaria, y es 1 o 0) de mayor probabilidad.

Regresión logística: idea general

Se ajustan los pesos de: $$z = \left(\sum_{i=1}^{n} w_i x_i\right) + b$$

Expresamos lo anterior usando el producto interno (dot product): $$z = \mathbf{w} \cdot \mathbf{x} + b$$

La función sigmoide (o función logística) lleva los valores al intervalo (0,1):

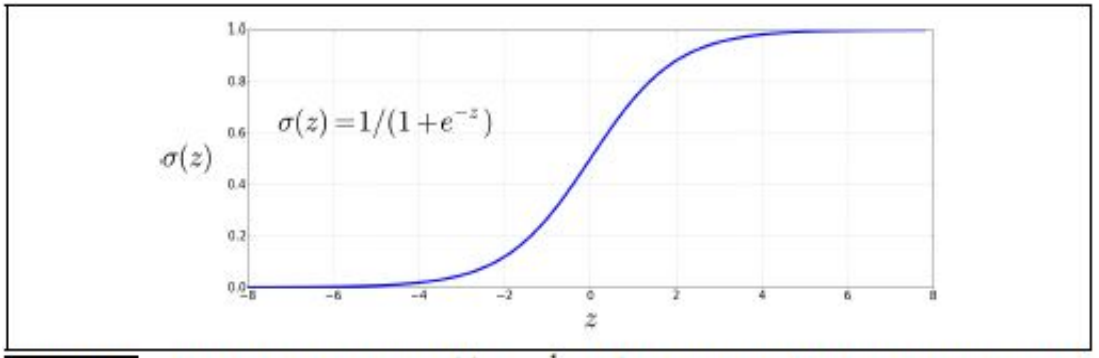
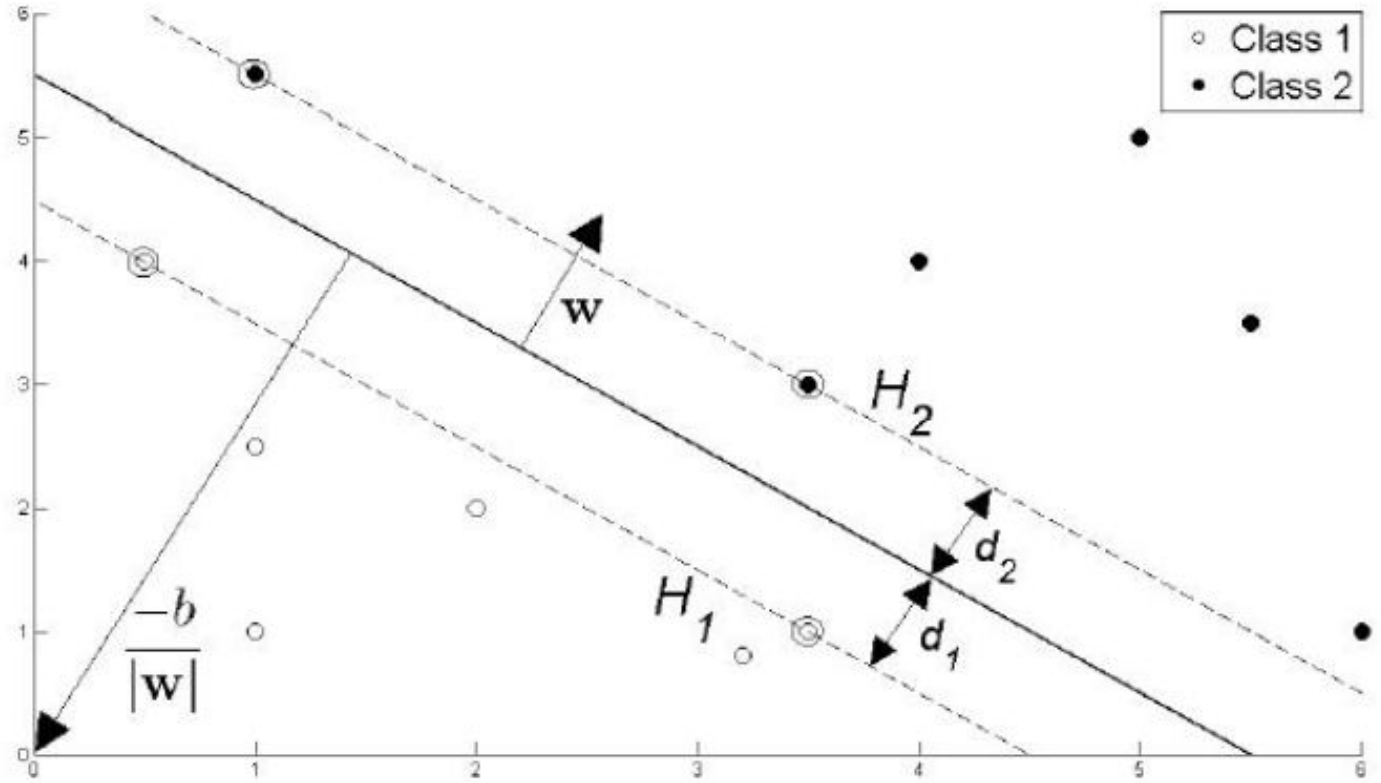


Figure 5.1 The sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}}$ takes a real value and maps it to the range $(0, 1)$. It is nearly linear around 0 but outlier values get squashed toward 0 or 1.

Hay que ajustar para que se cumpla *P(y=1) + P(Y=0) = 1* .

Otros métodos

Support Vector Machines



Otros métodos de clasificación

- knn – Vecinos más cercanos
    - Basados en memoria
    - "Lazy"
    - Más costo de clasificación
- Árboles de decisión – No muy usados en PLN
- Random Forests – Combinación de clasificadores
- etc...

Metodología

Corpus de aprendizaje

- Dividimos el corpus en dos partes:
  - Corpus de entrenamiento (80%?)
  - Corpus de evaluación (test) (20%?)
- O en tres partes:
  - Corpus de entrenamiento (80%?)
  - Corpus de validación (desarrollo/held-out) (10%?)
  - Corpus de evaluación (test) (10%?)
- O hacemos *cross-validation* (si el corpus es chico).
- Para evaluar, nuestro conjunto de evaluación debe ser *diferente* al de entrenamiento.
- … pero deberían tener la misma distribución.

Esquema general

1) Ingeniería de atributos
2) Selección de atributos
3) Selección de modelo
4) Aprendizaje
5) Evaluación

Ingeniería de atributos

- En general, los métodos de clasificación trabajan sobre conjuntos de pares atributo/valor
  - Atributos para palabras: forma de superficie, lema, terminación, POS, pertenencia a una lista, cantidad de letras, etc, etc.
  - Atributos para oraciones: incluye palabra de una lista, número de veces que aparece una palabra, incluye expresión, largo, etc, etc.
  - Atributos para documentos: largo, ocurrencias de cada palabra, etc, etc.
- Uno de los atributos es la clase objetivo (*target class*)
- La ingeniería de atributos es precisamente obtener esos atributos a partir de nuestra forma original.

Ingeniería de atributos

- Algunas estrategias
  - Para convertir categorías a números: one-hot-encoding
    - `idioma: {español, inglés, francés}`
      - `Es_español:{0,1}`
      - `Es_inglés:{0,1}`
      - `Es_francés:{0,1}`
  - Para convertir números a categorías:
    - dividir en rangos
      - iguales
      - según distribución en el corpus de entrenamiento

Ingeniería de atributos

- ¿Cómo…
  - extraer atributos de árboles?
    - Distancia a la raíz, secuencia de categorías desde la raíz a cierta hoja, componente padre de una hoja, profundidad del árbol, etc, etc.
  - extraer atributos de grafos?
    - Conectividad, número de relaciones de cierto tipo, etc., etc.
- Reglas de conocimiento
  - Intentan incorporar conocimiento experto como atributo.
  - Idea: a) construir una regla determinista que tome una instancia y prediga una clase. b) Incorporar el resultado de la regla como un nuevo atributo:

    Ej: Si un correo incluye las palabras “Estimado ganador” y menciona una cifra grande de dinero, entonces el correo es spam.

Selección de Atributos

- Encontrar el menor número de atributos que permitan caracterizar al corpus.

- Razones
  - A veces, demasiados atributos pueden producir sobreajuste.
  - Eficiencia.

Selección de Modelo

- Los modelos de aprendizaje generalmente incluyen *hiperparámetros.*

class sklearn.tree.DecisionTreeClassifier(*criterion='gini'*, *splitter='best'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features=None*, *random_state=None*, *max_leaf_nodes=None*, *class_weight=None*, *presort=False*)

Detalles

Selección de Modelo

- ¿Cómo ajustamos los hiperparámetros?
  - Corpus held-out
  - Cross Validation
- Corpus held-out
  - Separamos una parte del corpus de entrenamiento y lo utilizamos para evaluar configuraciones intermedias.
- Cross-Validation
  - Divido el corpus de entrenamiento en k partes.
  - Entreno sobre (k-1) partes y evalúo en la restante.
  - Repito para cada parte y calculo la media.

Medidas de evaluación

- Accuracy (Acierto)
  - Fracción de las instancias clasificadas correctamente
- Las usuales en el PLN son (similares a RI):
  - Precisión (Precision)
    - Instancias positivas que se clasificaron correctamente / Total de Instancias Positivas
    - TP / (TP + FP) [TP= True Positives, FP = False Positives]
  - Exhaustividad/Cobertura (Recall)
    - Fracción de las instancias positivas que se clasificaron correctamente / Fracción de las instancias que son positivas
    - TP/(TP + FN) [FN = False Negatives]
  - Medida-F
    - Media armónica entre Precisión y Recall: 2*Precision*Recall/ (Precision + Recall)
  - Matriz de confusión

Medidas de evaluación

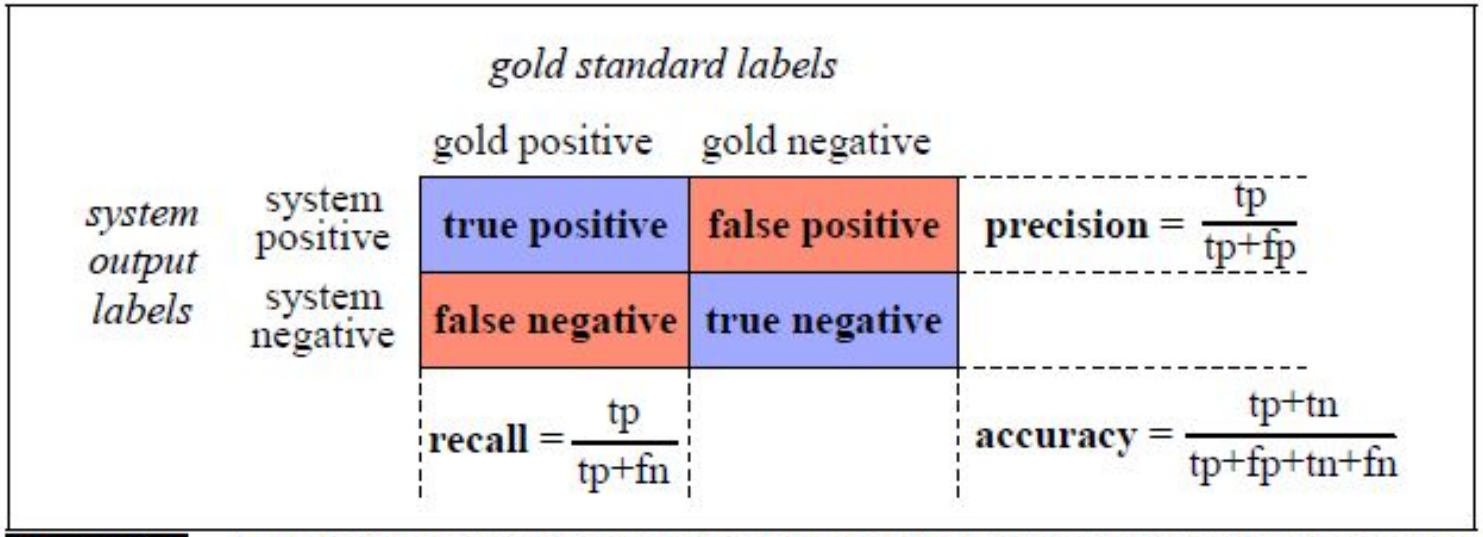


Figure 4.4 A confusion matrix for visualizing how well a binary classification system performs against gold standard labels.

Medidas de evaluación

Más de dos clases

- Por ejemplo: positivo/negativo/neutro
- Calculo precision/recall/medida F para cada clase.
- Luego:
  - micro precision / micro recall / micro F

    se calculan sobre el total
  - macro precision / macro recall / macro F

    es el promedio de las medidas de cada clase
  - variante: weighted

    ponderado por cantidad de ocurrencias

Medidas de evaluación

Secuencias (BIO):

En general se mide *exact match* (secuencias completadas correctamente detectadas), pero se pueden calcular medidas parciales.

Aprendizaje

- Utilizamos el corpus de entrenamiento para generar un *modelo* (función de clasificación).
- Utilizamos la función de clasificación para calcular la clase objetivo para cada instancia del corpus de evaluación.
- Evaluamos performance sobre el corpus de evaluación (¡que no vimos nunca antes!).

Mejoras

- ¿Cómo podríamos mejorar?
  - Cambiar el método
  - Elegir los atributos (son muy pocos…)
  - Ajustar hiperparámetros del modelo

Referencias

- Martin & Jurafsky, 3ª ed., draft agosto 2024, Capítulos 4 y 5.
- R. Garreta, G. Moncecchi, Learning Scikit-learn: Machine Learning in Python