Aprendizaje en Ensamble

ITAM

Menú

- Inspiración
  - Sabiduría de las masas
- Bagging
  - Una técnica exitosa: Random Forest
- Boosting
  - Una técnica exitosa: Adaboost

Inspiración

- Experimento de Galton
- Wisdom of Crowds (Surowiecki)
- Who wants to be a millionaire

Ensambles

- Estos métodos construyen múltiples modelos
  - Diversidad de modelos, diversos aspectos de los datos
- Combinan la predicción de los diversos modelos
  - Votaciones, promedios, etc.
- La diferencia entre las diversas técnicas se reduce a diferencias en estos dos puntos

Bagging

- Bagging (bootstrap aggregating)
- Este método toma diferentes muestras aleatorias de los datos con reemplazo (boostrap)
- Aquí asumimos que las diferencias entre las muestras llevaran a diferencias en los modelos resultantes
- La predicción final se toma usando alguna regla para combinar las predicciones individuales
  - Votación (mayoría, pluralidad, ponderada, suave..)
  - Promedio (simple y ponderada)

Bagging

- Como la diversidad es fundamental, otra idea para fomentarla es utilizar diferentes subconjuntos de variables para cada modelo (subspace sampling)
  - Fomenta diversidad
  - Aumenta velocidad
  - Ayuda un poco con la maldición de la dimensionalidad
- Los árboles de decisión son sensibles a variaciones en los datos y pueden aprovechar bien esta técnica
  - Pequeñas variaciones en los datos pueden provocar construcciones distintas
- Esto es el fundamento de los árboles aleatorios

Random Forest Entrenamiento

- Input: Datos D, número de árboles T, número de atributos p
- Output: Un conjunto de árboles
- for t=1 to T do
  - Crear muestra D_t a partir de D con reemplazo
  - Seleccionar p atributos al azar y eliminar el resto de D_t
  - Crear un árbol A_t usando D_t sin podar
- return $\{A_t \mid 1<=t<=T\}$
- Nota: Se recomienda que p=log(número de atributos en D) o bien p=sqrt(número de atributos en D)

Random Forest Predicción

- Input: Conjunto de Árboles, instancia a etiquetar x, bandera de si es clasificación o regresión
- Output: Predicción para x
- for t=1 to T do
  - y_t=At.predict(X)
- if classification
  - return vote({y_t|1<=t<=T})
- else
  - return mean({y_t|1<=t<=T})

Boosting

- Surge de la pregunta de si son equivalentes las clases de problemas weakly learnable y strongly learnable
  - La respuesta la dio Schapire en su artículo “The Stength of Weak Learnability”
  - La respuesta es que si son equivalentes y la demostración es por construcción. La construcción es boosting (bootstrap aggreagation)
- La idea general:
  - Generar secuencialmente un conjunto de modelos. Cada nuevo modelo es entrenado para corregir los errores de los modelos anteriores. La salida es una combinación de los modelos

AdaBoost

- Adaboost (adaptive boosting) es una implementación de esta idea. En particular define:
  - Cómo dar peso a los ejemplos de entrenamiento para reflejar los aciertos y errores de los otros modelos
  - Cómo dar peso a cada modelo que refleje lo importante que es para el ensamble final
- Define una función de costo exponencial de la cuál se derivan los valores anteriores
- Toma como entrada cualquier algoritmo de clasificación binaria y supone que las clases están etiquetadas con -1 y 1

Boosting

- Para cada iteración se calculan los nuevos pesos que debe tener cada ejemplo
  - Esto se puede implementar muestreando los datos de entrenamiento con la nueva distribución
- La idea es adjudicar la mitad del peso a los datos bien clasificados y la mitad a los datos mal clasificados.
  - Dados D datos al principio cada instancia tiene peso 1/|D|
  - Subsecuentemente dado el error de clasificación ε asignamos la mitad del peso a los ejemplos mal clasificado y la otra mitad a los bien clasificados
    - Por ejemplo si nos equivocamos en 25% de los ejemplos, para el siguiente modelo éstos tendrán el doble de peso, mientras que los bien clasificados se reducirán a 0.66

Boosting

Algorithm 11.3: Boosting($D, T, \mathscr{A}$) – train an ensemble of binary classifiers from reweighted training sets.

Input : data set D; ensemble size T; learning algorithm $\mathscr{A}$.
Output : weighted ensemble of models.

$w_{1i} \leftarrow 1/|D|$ for all $x_i \in D$; // start with uniform weights
for $t = 1$ to T **do**
  run $\mathscr{A}$ on D with weights w_{ti} to produce a model M_t;
  calculate weighted error ϵ_t;
  if $\epsilon_t \geq 1/2$ **then**
    set $T \leftarrow t-1$ and break
  end
  $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$; // confidence for this model
  $w_{(t+1)i} \leftarrow \frac{w_{ti}}{2\epsilon_t}$ for misclassified instances $x_i \in D$; // increase weight
  $w_{(t+1)j} \leftarrow \frac{w_{tj}}{2(1-\epsilon_t)}$ for correctly classified instances $x_j \in D$; // decrease weight
end
return $M(x) = \sum_{t=1}^{T} \alpha_t M_t(x)$

Algoritmo tomado del libro de Peter Flach (Machine Learning)

AdaBoost

- La elección de los pesos y de alfa está relacionada con minimizar la función de costo exponencial (pues esta simplifica el algoritmo)
- Queremos minimizar

$$error = \sum_{i=1}^{N} e^{-m_i}$$

$$m_i = y_i \sum_{t=1}^{T} \alpha_t M_t(x_i)$$

Aquí y_i es la clase real (1 o -1 y M_t es la clase asignada)

Otros métodos

- La ideas principales del aprendizaje en ensamble son
  - Contar con modelos que capturen diferentes aspectos de los datos
  - Contar con una manera de mezclarlos
- Dicho esto podemos pensar en hacer un ensamble de modelos heterogéneos (redes neuronales + regresiones + k-vecinos, etc.) con la idea de que provean diversidad
- Podemos mezclarlos utilizando otro modelo más, por ejemplo una regresión logísitica
  - Esto se conoce como stacking
  - Ahora el modelo mezclador tiene parámetros que aprender (a diferencia de adaboost) por lo que se necesitan tomar en cuenta esto en el proceso de aprendizaje

Ejercicio Opcional

- Baje un conjunto de datos de UCI
  - http://archive.ics.uci.edu/ml/
    - Sugiero Abalone
  - http://archive.ics.uci.edu/ml/datasets/Abalone?pagewanted=all
- Utilice los paquetes de Sklearn y compare el desempeño de un árbol de decisión y un random, forest (comparar con adaboost es opcional)