# Aprendizaje Automático sobre Grandes Volúmenes de Datos

## Clase 15

Pablo Ariel Duboue, PhD

Universidad Nacional de Córdoba,
Facultad de Matemática, Astronomía y Física

# Material de lectura

- Clase pasada:
  - Tackling the Poor Assumptions of Naive Bayes Text Classifiers
    - Rennie et al., ICML 2003
    - http://people.csail.mit.edu/jrennie/papers/icml03-nb.pdf
  - http://www.cs.columbia.edu/~smaskey/CS6998-0412/slides/week7_statnlp_web.pdf
  - http://spectrallyclustered.wordpress.com/2013/02/20/naive-bayes-on-hadoop/
  - http://machinomics.blogspot.com.ar/2013/11/naive-bayes-with-map-reduce.html
- Ésta clase:
  - Random Forests
    - Breiman, Machine Learning (2001)

# Preguntas

- Ejemplo en python de la clase pasada, ¿cuál es la plataforma subyacente?
  - Una plataforma de ejemplo del autor
  - Implementar MapReduce sin localidad de datos ni redundancia a fallas es muy sencillo
- ¿El método de aprendizaje que yo elija va a depender del problema o con todos puedo lograr resultados en cualquier escenario?
  - No Free Lunch Theorem!

# Recordatorio

- El sitio Web de la materia es http://aprendizajengrande.net
  - Allí está el material del curso (filminas, audio)
- Leer la cuenta de Twitter https://twitter.com/aprendengrande es obligatorio antes de venir a clase
  - Allí encontrarán anuncios como cambios de aula, etc
  - No necesitan tener cuenta de Twitter para ver los anuncios, simplemente visiten la página
- Suscribirse a la lista de mail en aprendizajengrande@librelist.com es optativo
  - Si están suscriptos a la lista no necesitan ver Twitter
- Feedback para alumnos de posgrado es obligatorio y firmado, incluyan si son alumnos de grado, posgrado u oyentes
  - El "resúmen" de la clase puede ser tan sencillo como un listado del título de los temas tratados

# Naive Bayes en MapReduce

- map(clave={features con nombres y valores}, valor: función objetivo)
  - devuelve clave=función objetivo:nombre del features, valor=valor del feature
- reduce(clave=función objetivo $y_0$:nombre del feature $F_i$, valor=valor del feature $v_{i,0}$):
  - calcular $P(F_i = v_{i,0}|y = y_0)$:
    - Calcular la suma sobre todos los valores posibles para el feature dado y usarla para normalizar
  - devuelve clave=función objetivo:nombre del feature, valor=valor del feature:probabilidad

# Algoritmos Actualizables

- Naive Bayes pertenece a una familia de algoritmos que pueden ser actualizados
  - Es posible agregar más datos una vez el algoritmo ya ha sido entrenado
- En muchos casos los algoritmos actualizables pueden ser paralelizados calculando las actualizaciones en paralelo y aplicandolas finalmente en el nodo central
- Entrenamiento de redes neuronales en modo batch

# Búsqueda Distribuída

- En el método de Hill Climbing con *restarts* se explora un espacio de búsqueda en la dirección de mayor mejora hasta que se estabiliza el valor
  - En ese momento se vuelve a retomar la búsqueda desde un punto al azar
  - Salirse de máximos locales
  - El algoritmo devuelve el mejor valor encontrado en todos los *restarts*
- Se puede considerar cada *restart* como corriendo en paralelo
  - Paralelización trivial

# Corrigiendo los errores de Naive Bayes

- Tackling the Poor Assumptions of Naive Bayes Text Classifiers
  - Rennie et al., ICML 2003
- Naive Bayes tiene problemas con:
  - Clases desbalanceadas
  - Mayoría de datos dependientes en una clase vs. en otra
  - Distribuciones específicas a datos textuales

# Naive Bayes Multinomial

- $m$ clases $c$, $n$ cantidad de *features* (palabras) $f_i$, necesitamos estimar los parámetros $\vec{\theta}_c = \{\theta_{c1}, \ldots, \theta_{cn}\}$ tales que $\sum_i \theta_{ci} = 1$

$$p(d|\vec{\theta}_c) = \frac{(\sum_i f_i)!}{\prod_i f_i!} \prod (\theta_{ci})^{f_i}$$

- Estimador MAP:

$$map_{NB}(d) = \text{argmax}_c \left[ \log p(\vec{\theta}_c) + \sum_i f_i \log \theta_{ci} \right] = \text{argmax}_c \left[ b_c + \sum_i f_i w_{ci} \right]$$

- Con smoothing de Dirichlet ($N_{ci}$, cantidad de veces la palabra $i$ aparece en documentos con la categoría $c$; $N_c$, total de palabras en la clase $c$; $\alpha_i$, valores de smoothing; $\alpha = \sum \alpha_i$):

$$\hat{\theta}_{ci} = \frac{N_{ci} + \alpha_i}{N_c + \alpha}$$

# Mejoras

- Cuando hay muchas más instancias para un valor de la función objetivo que para otras, el valor con menos instancias sufre
- En vez de estimar la probabilidad de que las *features* observadas correspondan con un cierto valor de la función objetivo, estimar que **no** correspondan
  - Naive Bayes Complementario

$$\hat{\theta}_{\bar{c}i} = \frac{N_{\bar{c}i} + \alpha_i}{N_c + \alpha}$$

$$map_{CNB}(d) = \text{argmax}_c \left[ \log p(\vec{\theta}_c) - \sum_i f_i \log \hat{\theta}_{\bar{c}i} \right]$$

- Normalizar vectores para *contrarrestar features* dependientes

$$\hat{w}_{ci} = \frac{\log \hat{\theta}_{ci}}{\sum_k \left| \log \hat{\theta}_{ck} \right|}$$

# Meta-learning y métodos ensemble

- Una técnica para obtener mejores resultados es combinar clasificadores
  - Votación
  - Entrenar un segundo clasificador
- Otras técnicas de meta-learning involucran cambiar el peso de cada instancia
  - De ciertos ejemplos se puede aprender más que otros
  - Parecido al concepto de *support vector*

# Random Forests

- Generar un gran conjunto de árboles de decisión repitiendo el siguiente algoritmo:
  - Tomar un subconjunto de los datos de entrenamiento al azar, con reemplazo
  - Para la construcción de los nodos de cada árbol, tomar un subconjunto de *features* al azar y realizar la división en el nodo usando sólo esas *features*
- Para predecir, usar todos los árboles a la vez y quedarse con el valor predicho más frequente
- Por la Ley de los Grandes Números, a partir de cierto punto agregar más árboles no cambia el resultado del clasificador
- Muy tolerantes a ruido y a *features* nocivas

# Midiendo el Error de Generalización

- Los Random Forests permiten medir el error de generalización utilizando árboles *out-of-bag:*
  - Medimos el error de generalización sobre el mismo conjunto de datos sobre el cual se entrenó el random forest
  - Para cada instancia de entrenamiento, calculamos la clase predicha por el random forest completo, y la comparamos con la clase predicha por los árboles que no utilizaron esa instancia para entrenarse
  - Es similar a cros-validación pero Breiman (2001) explica que es un estimador sin sesgo mientras que cros-validación comparte el sesgo del algoritmo original

# Paralelizando Random Forests

- Paralelizar Random Forests es trivial en la medida que podamos obtener un muestreo con reemplazo en cada nodo de manera eficiente
  - A priori ésto no parece posible en Hadoop a causa de la localidad de datos
  - Un mapper que asigna varios valores al azar para cada dato leido y reducers que computan los árboles en sí
- Apache Mahout (según su documentación Web) no utiliza muestreo con reemplazo si no que divide el conjunto de entrenamiento de manera completa
  - El número de árboles es el número de mappers

# Resúmen

- MapReduce
- Teorema CAP
- Operaciones matriciales distribuídas
  - Descomposición LU
- Descenso por el gradiente distribuido
- MPI y AMQ
- Paralelización de Naive Bayes y Árboles de Decisión

# MapReduce

- Modelo de cómputo que simplifica el uso de clusters
- Computa una función $f(\{(k_{in}, v_{in})\}) \rightarrow \{(k_{out}, list(v_{out})\}$
  - $map(k_{in}, v_{in}) \rightarrow list(k_{out}, v_{int})$
  - $reduce(k_{out}, list(v_{int})) \rightarrow list(v_{out})$

# Teorema CAP

- Cuando estamos en un ambiente distribuido, de estas tres características sólo podes escoger dos:
  - Consistencia
  - Disponibilidad
  - Tolerancia a particiones de la red

# Operaciones Matriciales Distribuidas

- Distribución de matrices a nodos
- Matriz por Vector
- Matriz por Matriz
- Inversión Matricial

# Temas Claves

- Identificación de la máxima tarea que puede hacerse por nodo
- Descomposición de la tarea normal en tareas por nodo
- Tareas globales, realizadas en un nodo (central) y tareas paralelas
- Comunicación entre tareas mediante HDFS y archivos bandera
- Tareas que sólo hacen Map
- Cálculos intermedios

# Descenso por el Gradiente

- Descenso por el Gradiente
  - Parallelized Stochastic Gradient Descent (Zinkevich et al., NIPS 2010)
- Dirección Alternada de Multiplicadores
  - Alternating Direction Method of Multipliers (Boyd et al., 2011)
- Búsqueda distribuida

# Otros Modelos

- MPI
- ActiveMQ

# Paralelizando Algoritmos

- Naive Bayes
- Random Forests