Aprendizaje Automático sobre Grandes Volúmenes de Datos

Clase 14

Pablo Ariel Duboue, PhD

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

Material de lectura

- Clase pasada:
  - Alternating Direction Method of Multipliers (Boyd et al. 2011)
    - web.stanford.edu/~boyd/papers/pdf/admm_distr_stats.pdf
  - MPI (http://en.wikipedia.org/wiki/Message_Passing_Interface)
    - http://www.mpich.org/
  - Colas de pasaje de mensajes (http://en.wikipedia.org/wiki/Advanced_Message_Queuing_Protocol)
    - http://activemq.apache.org/
- Ésta clase:
  - 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

Preguntas

- ¿Cuándo usar MR vs. MPI vs. AMQ?
  - MR: programadores sin experiencia en cálculo distribuido, hardware barato
  - MPI: hardware dedicado de buena calidad
  - AMQ: programadores con experiencia en cálculo distribuido, hardware con acceso restringido a Internet
- ¿Cómo calcular la tarea máxima?
  - Depende de la tarea y las máquinas
- ¿MPI con DFS?
  - Es posible, pero no dá el caso de uso
- Distribución de datos en descenso por el gradiente, ¿se realiza al comienzo?
  - Sí, y por eso se puede usar una implementación de descenso por el gradiente ya conocida

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

Filmínas de la defensa de Jingen Xiang

- Filmínas 28-36

Zinkevich et al., NIPS 2010

- Distribuir los datos al azar entre nodos
- Realizar descenso por el gradiente en cada nodo, por separado
- Unificar los resultados en un nodo central

Algoritmo en nodo worker

Algorithm 1 SGD($\{c^1, \ldots, c^m\}, T, \eta, w_0$)

for $t = 1$ **to** T **do**
  Draw $j \in \{1 \ldots m\}$ uniformly at random.
  $w_t \leftarrow w_{t-1} - \eta \partial_w c^j(w_{t-1})$.
end for
return w_T.

(adaptado de Zinkevich et al, 2010)

Algoritmo en nodo central

Algorithm 2 ParallelSGD$(\{c^1, \dots c^m\}, T, \eta, w_0, k)$

for all $i \in \{1, \dots k\}$ **parallel do**
  $v_i = \text{SGD}(\{c^1, \dots c^m\}, T, \eta, w_0)$ on client
end for
Aggregate from all computers $v = \frac{1}{k} \sum_{i=1}^{k} v_i$ and **return** v

(adaptado de Zinkevich et al, 2010)

Dirección Alternada de Multiplicadores

- Optimizar dos funciones a la vez
  - Problema similar a regresión logística y SVMs
  - Alternar la optimización de una función y la otra
- Clave: en ciertas condiciones, la primera optimización se puede hacer en paralelo mientras que la segunda es reducida y puede hacerse en una sola máquina

$$\begin{aligned} \text{minimizar} \quad & f(x) + g(z) \\ \text{dado que} \quad & Ax + Bz = c \end{aligned}$$

- Ver Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers (201) por Boyd, Parikh, Chu, Peleato, Eckstein

MPI

- Estándar de facto
  - Desde 1994
  - Mismo programa en todos los nodos, se pasan datos entre ellos
- Provee:
  - Topología virtual, Sincronización, Comunicación
- Funciones:
  - Comunicación punto-a-punto (rendez-vous), send/receive
  - Topología Cartesiana o de grafo general
  - Combinar resultados parciales (gather/reduce)
  - Sincronización de nodos (barreras)
  - Información general (número de nodos, número de nodo actual, vecinos, topología)

MPI vs. MapReduce

- MapReduce es un subconjunto del estándar MPI
  - Operaciones collectivas y operaciones de usuario
- MPI no es tolerante a fallas
- MPI provee muchas más primitivas
  - Requiere que el programador se ocupe de muchos más detalles
- En el caso de inversión de matrices, la implementación MPI es más eficiente para matrices pequeñas pero tiene mucho mayor overhead de comunicación
  - Escala de manera más pobre

Message Queues

- Message-Oriented Middleware (MOM)
  - Soporte por software o hardware
  - Envío y recepción de mensajes
  - Redes distribuidas heterogéneas
  - Normalmente asíncrono
    - Colas
    - Potencialmente persistentes (tolerancia a fallos)
  - Ruteo
  - Potencial transformación
- Requiere un message transfer
  - Talón de Aquiles

AMQP vs. MapReduce

- El broker es el eslabón más débil
- Topología explícita en AMQP
  - Más versátil
  - Más trabajo por parte del programador
- Tolerancia a fallas de AMQP es potencialmente mejor que MapReduce
  - Cómputos intermedios pueden ser mantenidos en colas persistentes

Naive Bayes

- Teorema de Bayes

$$P(y|\vec{f}) = \frac{P(\vec{f}|y)P(y)}{P(\vec{f})}$$

- Naive Bayes asume que las features son probabilisticamente independientes

$$y_{NB} = max_y P(f_1, ..., f_n|y)P(y) = max_y P(f_1|y)...P(f_n|y)P(y)$$

- Podemos estimar $P(f_i|y)$ a partir de conteos

NB: de conteos a probabilidades

- Conteos de instancias (N_i) vs. conteos de features (N_f)
- De la definición de probabilidad conjunta:
  $P(".com"|Arts)P(Arts) = P(".com", Arts) = \frac{N_f(".com", Arts)}{N_f(.)}$
- De la definición de probabilidad simple:
  $P(Arts) = \frac{N_i(Arts)}{N_i(.)}$
- Despejando para la probabilidad condicional:
  $P(".com"|Arts) = \frac{N_i(.)}{N_f(.)} \frac{N_f(".com", Arts)}{N_i(Arts)}$

Ejemplo: Contar Palabras

- map(input_key: DocName, input_value: DocText):
  - for each word w in input_value:
    - EmitIntermediate(w, 1)
- reduce(output_key: Word, intermediate_values: List[Int]):
  - int result = 0
  - for each v in intermediate_values:
    - result += v
  - Emit(result)

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

En python

- https://github.com/analyticbastard/myml/blob/master/myml/superv
- `def map(self, key, value):`
  - `for i, k in enumerate(key):`
    - `yield (value, i), k`
- `def reduce(self, key, values):`
  - `val = set(values)`
  - `N = len(values)`
  - `for newkey in val:`
    - `yield (key[0], key[1], newkey), 1.0*np.sum(values == newkey)/N`

Implementación Alternativa

- Necesitamos calcular:
  - **#(Y=*):** número total de instancias (*documentos*)
  - **#(Y=y):** número de instancias por valor de la función objetivo (*etiqueta*)
  - **#(Y=y,F=*):** número de valores de features (*palabras*) para un cierto valor de la función objetivo (*etiqueta*)
  - #(Y=y,F=v): numero veces un cierto valor de una cierta feature (*palabra*) aparece bajo un cierto valor de la función objetivo (*etiqueta*)
  - dom(X): cuántos valores posibles pueden tomar las features (*vocabulario*)
  - **dom(Y):** cuántos valores puede tomar la función objetivo (*etiquetas*)

Calculando valores independientes de las instancias

```java
public void map(LongWritable key, Text value, Context context)
        throws InterruptedException, IOException {
    String[] words = value.toString().split("\\s+");
    Vector<String> labels = tokenizeLabels(words[0]);
    Vector<String> text = tokenizeDoc(words);

    for (String label : labels) {
        context.write(new Text(label), new IntWritable(text.size()));
    }
}

public void reduce(Text key, Iterable<IntWritable> values, Context context)
        throws InterruptedException, IOException {

    // Each time this Reducer is invoked, that means we have another unique label.
    context.getCounter(NB_COUNTERS.UNIQUE_LABELS).increment(1);

    long pY = 0;
    long pYW = 0;
    for (IntWritable value : values) {
        // Increment the global counter (Y = *).
        context.getCounter(NB_COUNTERS.TOTAL_DOCS).increment(1);

        // Increment the number of documents with this label (Y = y).
        pY++;

        // Increment the number of words under this label.
        pYW += value.get();
    }

    // Write out the results in the format:
    // <label Y> {<# of documents with label Y>:<# of words under label Y>}
    context.write(key, new Text(String.format("%s:%s", pY, pYW)));
}
```

(Adaptado de http://spectrallyclustered.wordpress.com/2013/02/20/naive-bayes-on-hadoop/)

Calculando valores dependientes de las instancias

```java
public void map(LongWritable key, Text value, Context context) throws InterruptedException, IOException {
    String[] words = value.toString().split("\\s+");
    Vector<String> labels = tokenizeLabels(words[0]);
    Vector<String> text = tokenizeDoc(words);

    for (String label : labels) {
        for (String word : text) {
         // (Y = y, W = w)
         context.write(new Text(word), new Text(label));
        }
    }
}

public void reduce(Text key, Iterable<Text> values, Context context)
    throws InterruptedException, IOException {

    // Update the counter to indicate the size of the vocabulary.
    context.getCounter(NB_COUNTERS.VOCABULARY_SIZE).increment(1);

    // Loop through the labels.
    HashMap<String, Integer> counts = new HashMap<String, Integer>();
    for (Text label : values) {
        String labelKey = label.toString();
        counts.put(labelKey,
                new Integer(counts.containsKey(labelKey) ? counts.get(labelKey).intValue() + 1 : 1));
    }
    StringBuilder outKey = new StringBuilder();
    for (String label : counts.keySet()) {
        outKey.append(String.format("%s:%s ", label, counts.get(label).intValue()));
    }

    // Write out the Map associated with the word.
    context.write(key, new Text(outKey.toString().trim()));
}
```

(Adaptado de http://spectrallyclustered.wordpress.com/2013/02/20/naive-bayes-on-hadoop/)

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

Algoritmo mejorado

Table 4. Our new Naive Bayes procedure. Assignments are over all possible index values. Steps 1 through 3 distinguish TWCNB from WCNB.

- Let $\vec{d} = (\vec{d_1}, \ldots, \vec{d_n})$ be a set of documents; d_{ij} is the count of word i in document j.
- Let $\vec{y} = (y_1, \ldots, y_n)$ be the labels.
- TWCNB$(\vec{d}, \vec{y})$
  1. $d_{ij} = \log(d_{ij} + 1)$ (TF transform § 4.1)
  2. $d_{ij} = d_{ij} \log \frac{\sum_k 1}{\sum_k \delta_{ik}}$ (IDF transform § 4.2)
  3. $d_{ij} = \frac{d_{ij}}{\sqrt{\sum_k (d_{kj})^2}}$ (length norm. § 4.3)
  4. $\hat{\theta}_{ci} = \frac{\sum_{j:y_j \neq c} d_{ij} + \alpha_i}{\sum_{j:y_j \neq c} \sum_k d_{kj} + \alpha}$ (complement § 3.1)
  5. $w_{ci} = \log \hat{\theta}_{ci}$
  6. $w_{ci} = \frac{w_{ci}}{\sum_i w_{ci}}$ (weight normalization § 3.2)
  7. Let $t = (t_1, \ldots, t_n)$ be a test document; let t_i be the count of word i.
  8. Label the document according to

$$l(t) = \arg\min_c \sum_i t_i w_{ci}$$