Aprendizaje Automático sobre Grandes Volúmenes de Datos

Clase 5

Pablo Ariel Duboue, PhD

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

Material de lectura

- Clase pasada:
  - Gale, William A. (1995). "Good–Turing smoothing without tears". Journal of Quantitative Linguistics 2: 3. doi:10.1080/09296179508590051
  - Capítulo 5 del Marlsand (2009) "Machine Learning, an Algorithmic Perspective"
  - Capítulo 5 del Smola & Vishwanathan (2008) "Introduction to Machine Learning"
  - http://en.wikipedia.org/wiki/Logistic_regression
  - http://en.wikipedia.org/wiki/Linear_regression
- Ésta clase:
  - Capítulo 13 del Owen et al. (2012)
  - http://ufal.mff.cuni.cz/~zabokrtsky/courses/npfl104/html/feature_e

Preguntas

- ¿Más de dos clases?
  - Entrenar un clasificador binario
    - una clases vs. el resto
    - una clase vs. otra
  - En regresión logística, usar una clase como pivote y realizar una regresión binaria por cada uno de las otras clases

Regresión Logística: estimando la probabilidad de p_i

$$Y_i \mid x_{1,i}, \ldots, x_{m,i} \sim \text{Bernoulli}(p_i) \tag{1}$$

$$\mathbb{E}[Y_i \mid x_{1,i}, \ldots, x_{m,i}] = p_i \tag{2}$$

$$\Pr(Y_i = y_i \mid x_{1,i}, \ldots, x_{m,i}) = \begin{cases} p_i & \text{if } y_i = 1 \\ 1 - p_i & \text{if } y_i = 0 \end{cases} \tag{3}$$

$$\Pr(Y_i = y_i \mid x_{1,i}, \ldots, x_{m,i}) = p_i^{y_i}(1 - p_i)^{(1-y_i)} \tag{4}$$

Estimando p_i y los coeficientes

- Los p_i y los coeficientes de la combinación lineal de las features son todos valores no observados que deben estimarse como parte de la búsqueda

$$\text{logit}(p_i) = \ln\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \beta_1 x_{1,i} + \cdots + \beta_m x_{m,i}$$

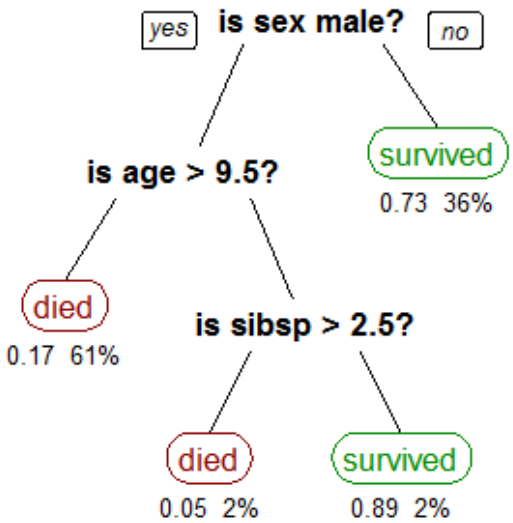
- Usar algún tipo de regularización para evitar soluciones triviales u ultra-complejas (overfitting) en la búsqueda

Algunos Comentarios

- 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 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

Revisión: Árboles de Decisión

- Dividir los datos según un feature solo y un predicado simple sobre ese feature



(CC-BY-SA Stephen Milborrow, from Wikipedia)

Ejemplo de DT

- Datos del censo de EEUU 1994
- clase objetivo: >50K, <=50K
- Features:
  - age: continuous.
  - workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
  - sex: male, female.
  - education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.
  - education-num: continuous.
  - marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
  - occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing,

Archivo ARFF

```
@relation adult
@attribute age real
@attribute workclass { Private, Self-emp-not-inc, Self-emp-inc,
Federal-gov, Local-gov, State-gov, Without-pay, Never-worked }
@attribute education { Bachelors, Some-college, 11th, HS-grad,
Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters,
1st-4th, 10th, Doctorate, 5th-6th, Preschool }
% ...
@data
28, Private, 338409, Bachelors, 13, Married-civ-spouse, Prof-specialty,
Wife, Black, Female, 0, 0, 40, Cuba, <=50K
37, Private, 284582, Masters, 14, Married-civ-spouse, Exec-managerial,
Wife, White, Female, 0, 0, 40, United-States, <=50K
52, Self-emp-not-inc, 209642, HS-grad, 9, Married-civ-spouse,
Exec-managerial, Husband, White, Male, 0, 0, 45, United-States, >50K
49, Private, 160187, 9th, 5, Married-spouse-absent, Other-service,
```

Usando Weka línea de comando

1

$ java -cp weka.jar weka.classifiers.trees.Id3 -t adult.arff
weka.core.UnsupportedAttributeTypeException:
weka.classifiers.trees.Id3: Cannot handle numeric attributes!
$ java -cp weka.jar weka.classifiers.trees.Id3 -t adult-no-num.arff
weka.core.NoSupportForMissingValuesException:
weka.classifiers.trees.Id3: Cannot handle missing values!
$ java -cp weka.jar weka.classifiers.trees.Id3 -t adult-no-missing.arff

```
relationship = Wife
| occupation = Tech-support
| | education = Bachelors: >50K
| | education = Some-college
| | | workclass = Private
| | | | race = White: >50K
| | | | race = Black: <=50K
| | | workclass = Federal-gov: <=50K (...)
Correctly Classified Instances 23471 77.8165%
Incorrectly Classified Instances 5297 17.5618%
```

Usando Weka línea de comando

2

$ java -cp weka.jar weka.classifiers.trees.J48 -t adult-no-missing.arff

```
marital-status = Married-civ-spouse
| education = Bachelors
| | relationship = Wife: >50K (269.0/84.0)
| | relationship = Own-child: <=50K (11.0/4.0)
| | relationship = Husband: >50K (2298.0/718.0)
| | relationship = Not-in-family: <=50K (1.0)
| | relationship = Other-relative: <=50K (20.0/3.0)
| | relationship = Unmarried: >50K (0.0)
| education = Some-college
| | occupation = Tech-support: >50K (109.0/43.0)
| | occupation = Craft-repair: <=50K (527.0/214.0)
| | occupation = Other-service: <=50K (123.0/17.0) (...)
Correctly Classified Instances 24803 82.2326%
Incorrectly Classified Instances 5359 17.7674%
```

Information Gain

- Podemos definir Information Gain a partir de la entropía (qué tan "ruidosa" es la partición)
- En caso de una partición binaria:

$$Entropy(S) = -p_+ log_2(p_+) - p_- log_2(p_-)$$

- Múltiples categorías (m):

$$Entropy(S) = \sum_{i=1}^{m} -p_i \log_2 p_i$$

- Ganancia de información entonces es:

$$Gain(S, A) = Entropy(S) - \sum_{v \in valores(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Ejemplo

Full set entropy: 0.809565832961416
Attribute: workclass Gain: 0.017104479622990 5
Attribute: education Gain: 0.0933939854773693
Attribute: marital-status Gain: 0.1581659232038
Attribute: occupation Gain: 0.093344624527962 4
Attribute: relationship Gain: 0.203664907965332
Attribute: race Gain: 0.0603373246980876
Attribute: sex Gain: 0.64386930448363
Attribute: native-country Gain: 0.0093290140743348 4
Attribute: target Gain: 0.809565832961416

keywords4bytecodes

- Proyecto personal: http://keywords4bytecodes.org
- Presentado en RECon, Montreal, 2012 (http://recon.cx)
- 1 millón de métodos Java desensamblados más texto asociado (JavaDocs)

Java Bytecodes

- La JVM es una máquina a pila
- El conjunto de opcodes (~200) es pequeño, para simplificar hacer el port a nuevas arquitecturas.
- Hay seis categorías de opcodes :
  - Almacenamiento en memoria (e.g. aaload, bastore)
  - Aritmético/lógico (e.g. iadd, fcmpg)
  - Conversión de tipos (e.g. i2b, f2d)
  - Construcción de objetos y manipulación (new, putfield)
  - Manipulacion de operandos y pila (e.g. swap, dup2_x1)
  - Flujo de control (e.g. if_icmpgt,goto)
  - Invocación de métodos y retorno (e.g. invokedynamic, lreturn)

Reverse Engineering Example

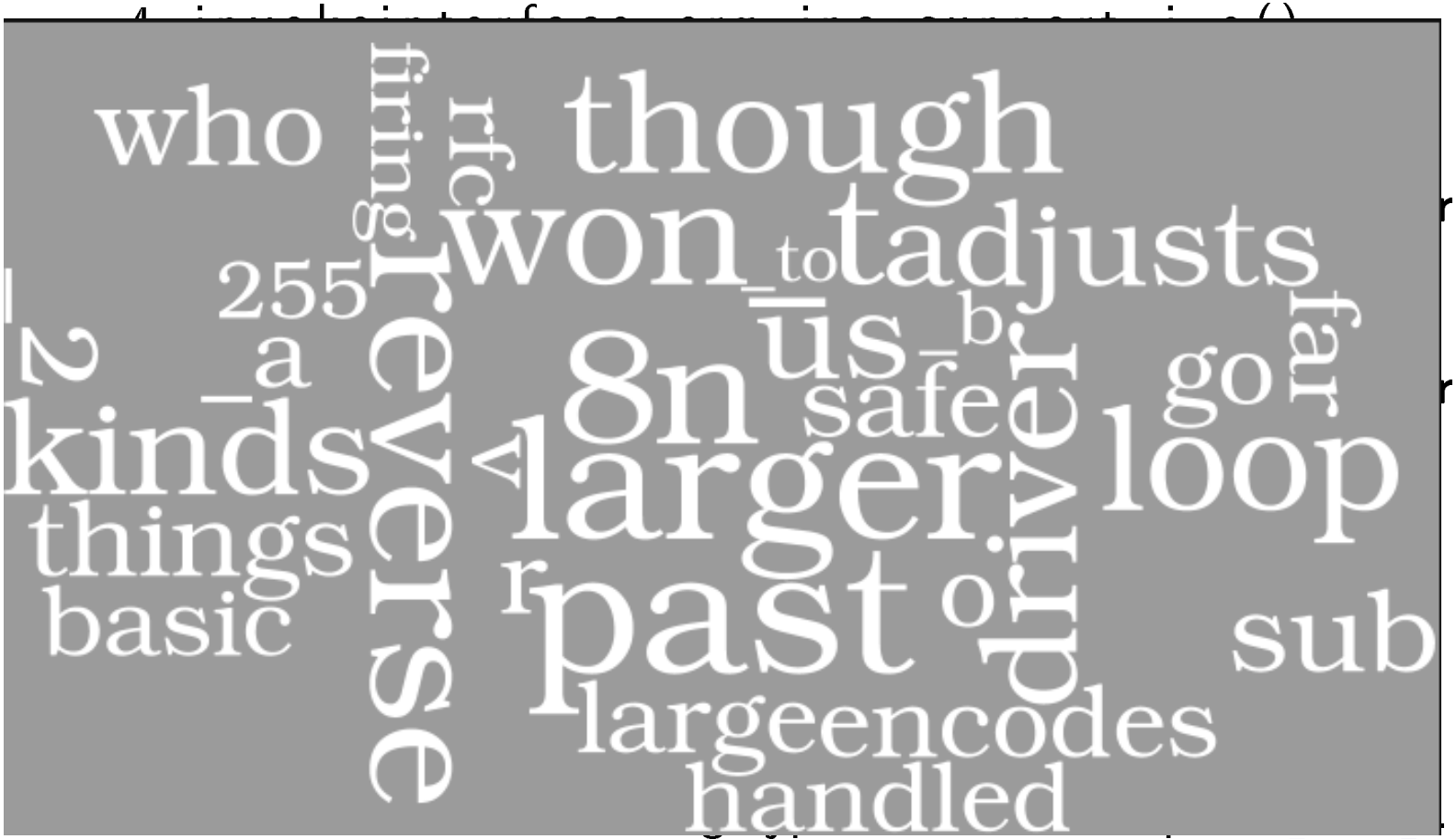
```
private final int c(int) {
    0 aload_0
    1 getfield org.jpc.emulator.f.v
    4 invokeinterface org.jpc.support.j.e()
    9 aload_0
    10 getfield org.jpc.emulator.f.i
    13 invokevirtual org.jpc.emulator.motherboard.q.e()
    16 aload_0
    17 getfield org.jpc.emulator.f.j
    20 invokevirtual org.jpc.emulator.motherboard.q.e()
    23 iconst_0
    24 istore_2
    25 iload_1
    26 ifle 128
    29 aload_0
    30 getfield org.jpc.emulator.f.b
    33 invokevirtual org.jpc.emulator.processor.t.w()
```

Reverse Engineering Example

```
private final int c(int) {
    0 aload_0
    1 getfield org.jpc.emulator.f.v
    4 invokeinterface org.jpc.support.j.e()
    9 aload_0
    10 getfield org.jpc.emulator.f.i
    13 invokevirtual org.jpc.emulator.motherboard.q.e()
    16 aload_0
    17 getfield org.jpc.emulator.f.j
    20 invokevirtual org.jpc.emulator.motherboard.q.e()
    23 iconst_0
    24 istore_2
    25 iload_1
    26 ifle 128
    29 aload_0
    30 getfield org.jpc.emulator.f.b
    33 invokevirtual org.jpc.emulator.processor.t.w()
```

Reverse Engineering Example

```
private final int c(int) {
    0 aload_0
    1 getfield org.jpc.emulator.f.v
```



Reverse Engineering Example

```
private final int c(int) {
    0 aload_0
    1 getfield org.jpc.emulator.f.v
    4 invokeinterface org.jpc.support.j.e()
    9 aload_0
    10 getfield org.jpc.emulator.f.i
    13 invokevirtual org.jpc.emulator.motherboard.q.e()
    16 aload_0
    17 getfield org.jpc.emulator.f.j
    20 invokevirtual org.jpc.emulator.motherboard.q.e()
    23 iconst_0
    24 istore_2
    25 iload_1
    26 ifle 128
    29 aload_0
    30 getfield org.jpc.emulator.f.b
    33 invokevirtual org.jpc.emulator.processor.t.w()
```

Reverse Engineering Example

```
private final int c(int) {
    0 aload_0
    1 getfield org.jpc.emulator.f.v
```

k4w: Debian

- A partir del archivo de paquetes Debian
  - `apt-file search --package-only .jar`
    - 1,400+ paquetes
  - `dpkg-query -p` *package name*
    - Buscar el campo `Source`
  - `dpkg-source -x` *source .dsc*
    - Buscar archivos fuentes Java.
  - `dpkg -x` *binary .deb*
    - Buscar los jars y desensamblar los métodos.
- Armando el corpus de Bytecodes / Javadoc
  - Desensamblar usando `jclassinfo --disasm`
  - Extraer los comentarios JavaDoc usando qdox.
    - Una librería de análisis de código fuente Java fácil de usar.
  - Hacer un matching heurístico de los métodos en el código fuente con los métodos compilados.
    - Normalizar las signaturas en el código fuente con las signaturas en el código binario.

k4w: Datos

- Corpus final:
  - 1M de métodos.
  - 35M palabras.
  - 24M instrucciones de JVM.

Ciclo de aprendizaje

- Tomamos el enunciado del problema y lo planteamos como un problema de aprendizaje
  - Normalmente clasificación, o bien simple o como una serie de clasificadores
    - Por ejemplo, tres clasificadores (uno por cada primer tercio del programa) más un clasificador final que toma la salida de los otros tres clasificadores como features
  - Diseñamos un conjunto inicial de features a extraer de los datos
  - Dividimos el conjunto de datos en entrenamiento y testeo (o utilizamos cross-validation)
- En el ciclo de aprendizaje, se varía el algoritmo de aprendizaje y sus parámetros o también la cantidad y tipo de features
  - Se espera que el error decrezca con estos cambios
  - Las features tienen mayor impacto que el algoritmo en sí

Features patrón

- Podemos empezar a escribir features del estilo "hay un iadd en el método" y cosas así
- Ésto se vuelve aburrido muy rápido
- *Feature templates* son recetas para extraer *features* a partir de los datos, generando gran cantidad de ellas
- Por ejemplo:
  - Tomar cada instrucción (e.g., "getfield org.jpc.emulator.f.i") y generar tres features:
    - instruccion ("getfield")
    - instrucción más operando ("getfield_org.jpc.emulator.f.i")
    - instrucción más operando abreviado ("getfield_org.jpc.emulator")

Filtrado de features

- Estimador Maximum Likelihood
  - Qué tan bien se predice la clase objetivo usando sólo esta *feature*
- Chi-square
  - Correlación χ^2 entre la *feature* y la clase objetivo
- TP-rate, FP-rate
  - El ratio de de true-positivos o false-positivos
- Firing rate
  - Cuantas veces está definida

Heurísticas de filtrado

- Tomar el subconjunto de features que mejor predice es NP-hard
- Proceso iterativo:
  - empezar con todas las features e ir tirando las que no ayudan (ablation study)
  - empezar de cero e ir agregando las features que más ayudan (incremental construction)
    - Usando las métricas de la filmina anterior para decidir que features agregar o re-entrenando en cada paso

Valores Faltantes

- Reemplazar con la media
- Reemplazar con la mediana
- Estimar del resto de los valores
  - Usando por ejemplo k-NN
    - Buscar otras instancias parecidas (según una métrica apropiada) y utilizar el valor más frecuente del atributo faltante
- Valores Bandera
  - Si se estiman valores faltantes, es útil incluir features que indiquen que el valor estaba indefinido originariamente

Normalización

- Algunos algoritmos de aprendizaje funcionan mejor si los valores de las features de entrada tienen media cero y dispersión 1
- Muchas veces se agrega la versión normalizada del feature como un feature extra y se deja al algoritmo de aprendizaje que decida qué feature es más útil.
    - El valor absoluto de un feature muchas veces contiene información relevante

Binning

- Transformar un valor numérico en una serie de valores categóricos
- Por ejemplo, en los saltos (e.g, "ifle 128"), encontrar valores umbral para decidir si es un salto "cercano, mediano o lejano"
- Se pueden usar valores umbral uniformes (e.g., 10 valores en el rango 0-2000) o utilizar programas específicos para encontrar los valores umbrales más informativos
    - Python Environment for Bayesian Learning: https://github.com/abhik/pebl

Combinaciones Aritméticas

- Si una cierta combinación entre features tiene sentido en el dominio del problema, puede ser conveniente agregarla
- Por ejemplo, si hay features como largo y ancho, es posible que el área (largo x ancho) sea informativa
- Debe haber alguna intuición detrás, agregar cualquier combinación aritmética es un despropósito

Combinaciones entre Instancias

- Si el problema de aprendizaje involucra un clasificador que se aplica a un conjunto de instancias (por ejemplo, para decidir qué respuesta responder a una pregunta), es posible utilizar dicho conjunto para generar features
  - Para un feature dado, agregar la distancia al valor medio de dicho feature en todo el conjunto de instancias
  - Reemplazar el conjunto de instancias por todos los pares de instancias y el problema de clasificación como "instancia 1 es mejor que instancia 2"

Binarización / thresholding

- Un caso extremo de binning es la binarización, en la cual la feature se transforma en dos valores, 0 y 1
- Algunos algoritmos de aprendizaje sólo pueden utilizar features binarias
- Con un valor de umbral adecuado es posible aumentar la señal en los datos mientras que se reduce el ruido

Limitaciones Particulares

- Algoritmo no puede utilizar valores de tipo categórico
  - Una de muchos