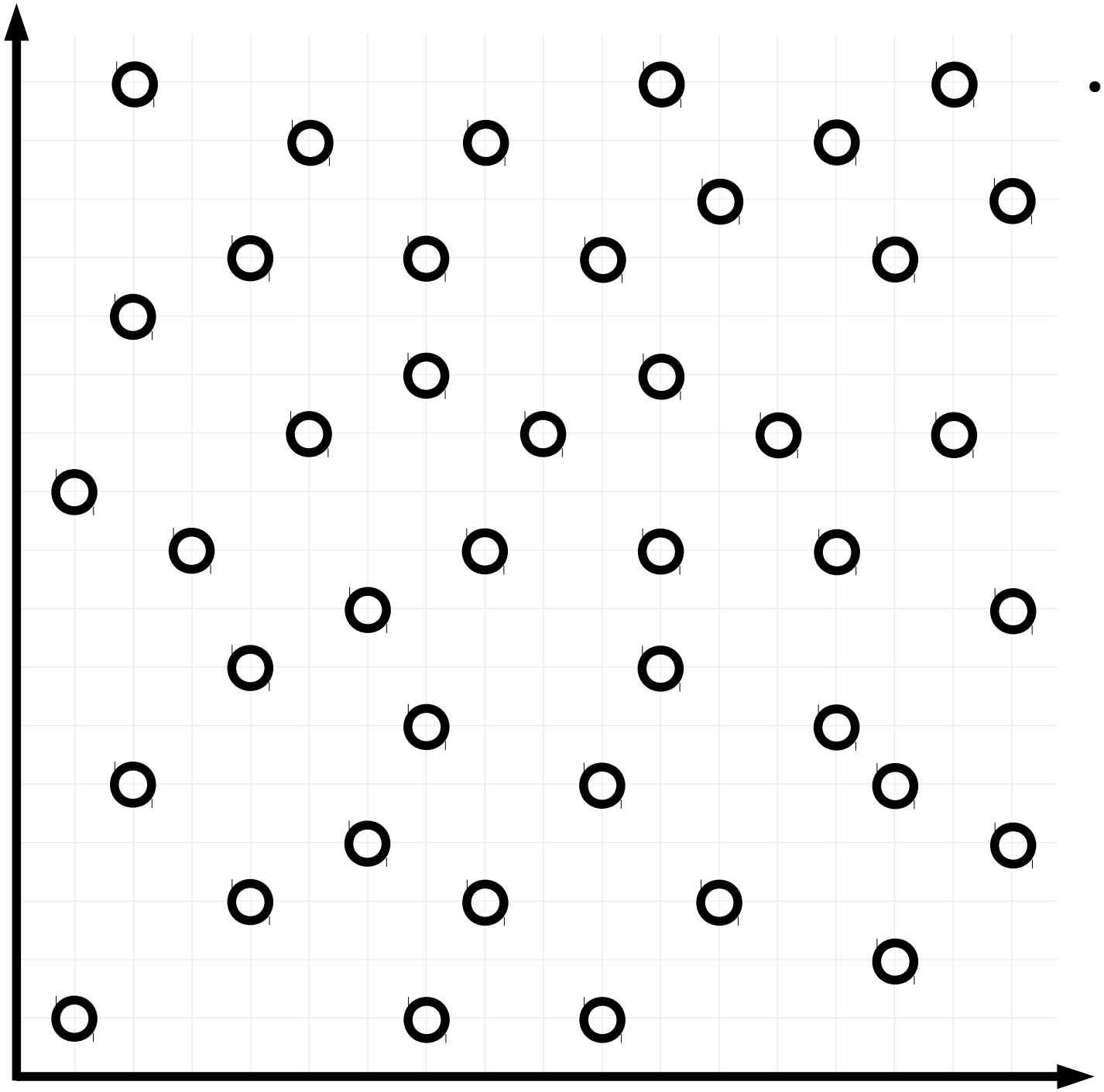
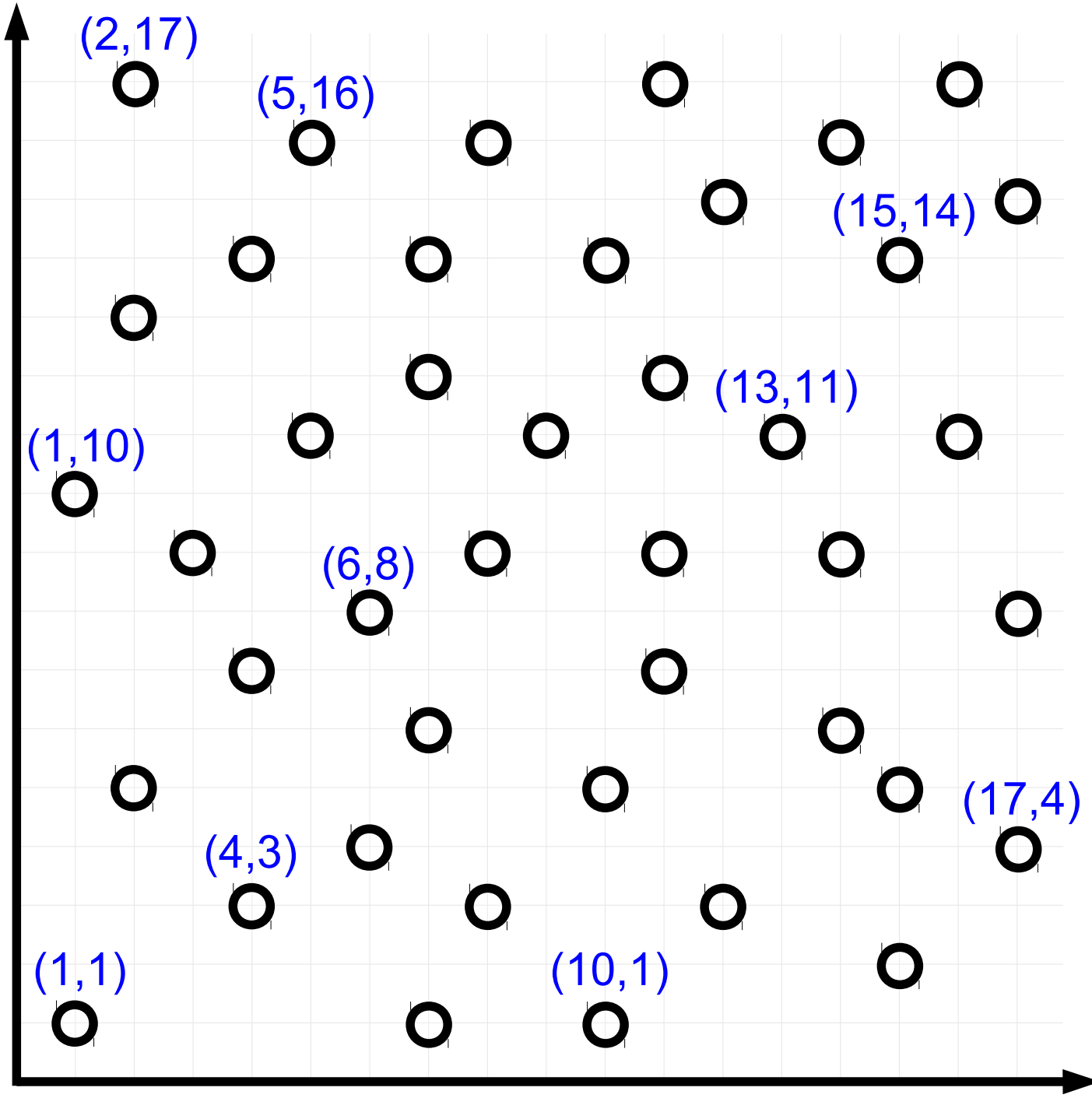
Aprendizaje Automático (Machine Learning)

Introducción a las Tecnologías del Habla

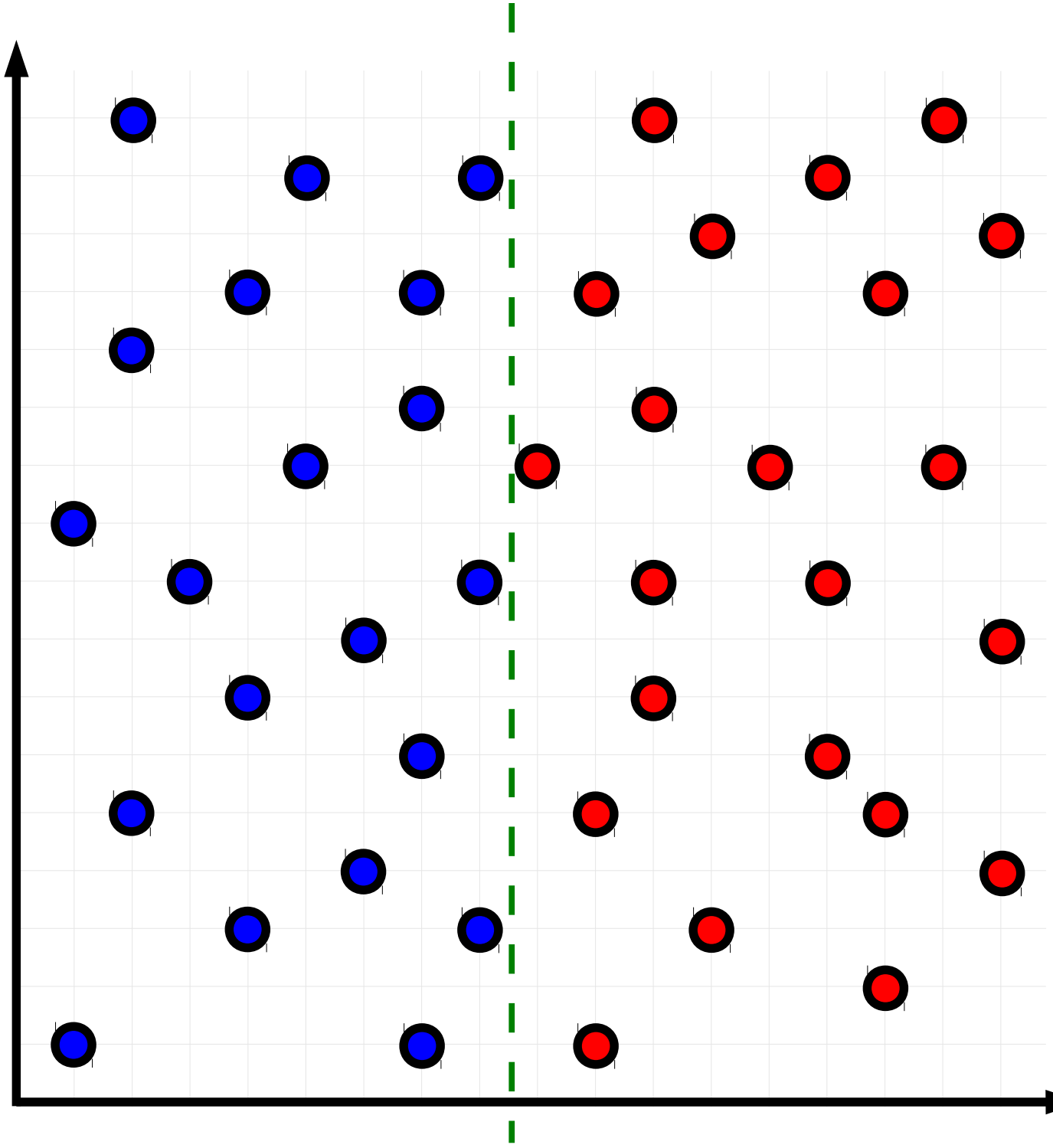
2º cuatrimestre 2014 – Agustín Gravano



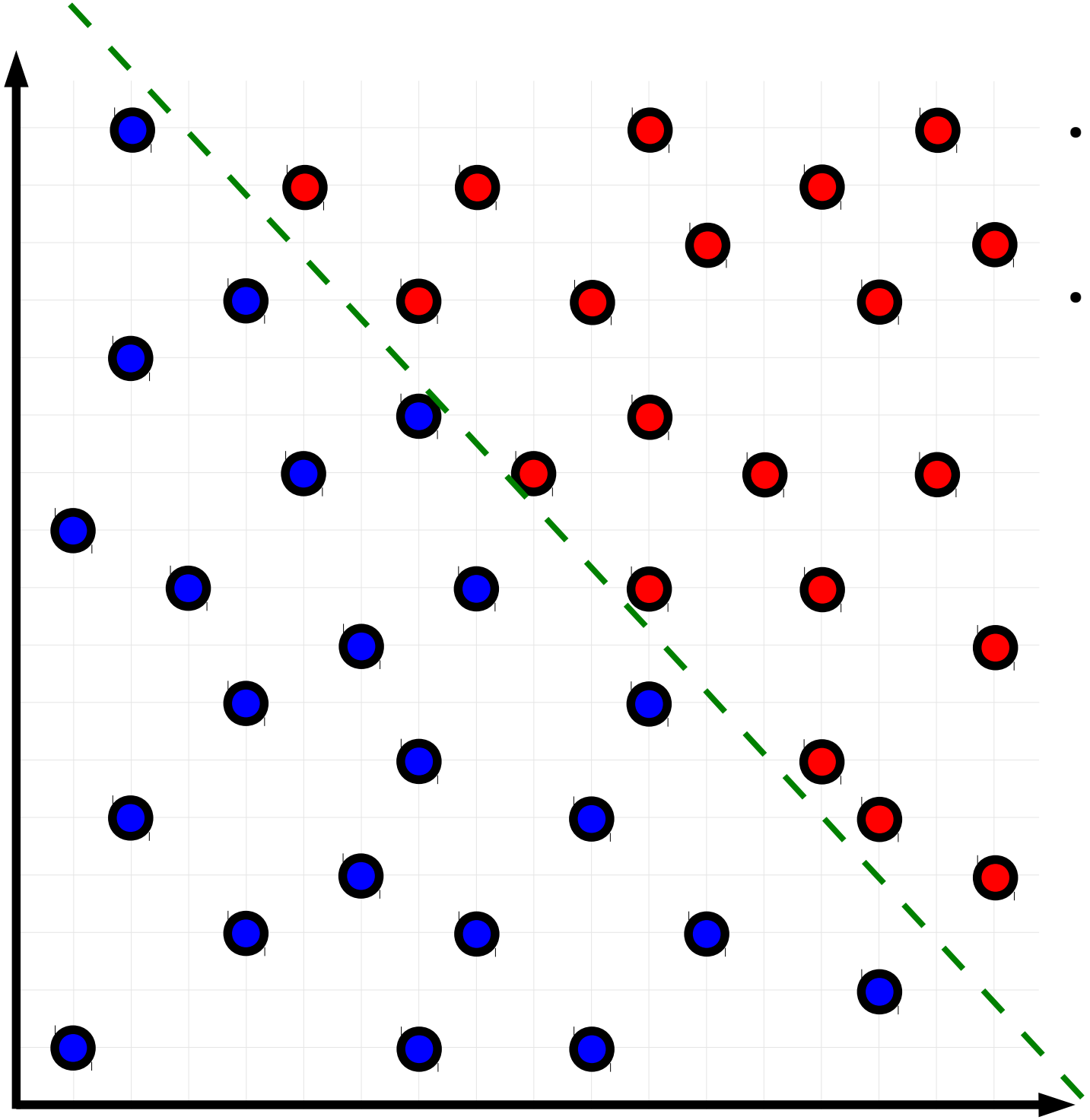
- Tenemos *N* puntos en el plano.



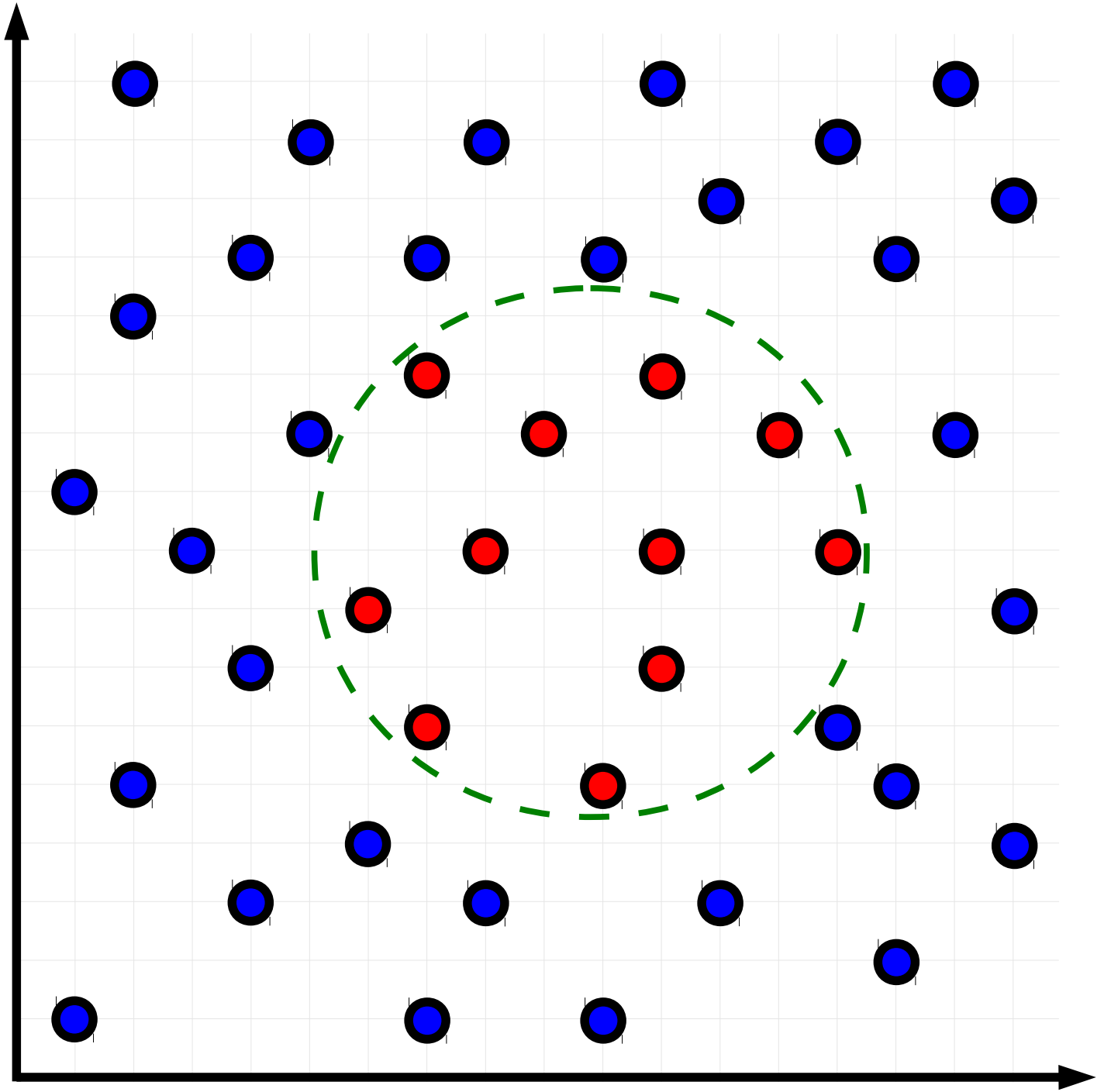
- Tenemos *N* puntos en el plano.
- Cada punto tiene dos coordenadas: (*x*, *y*).

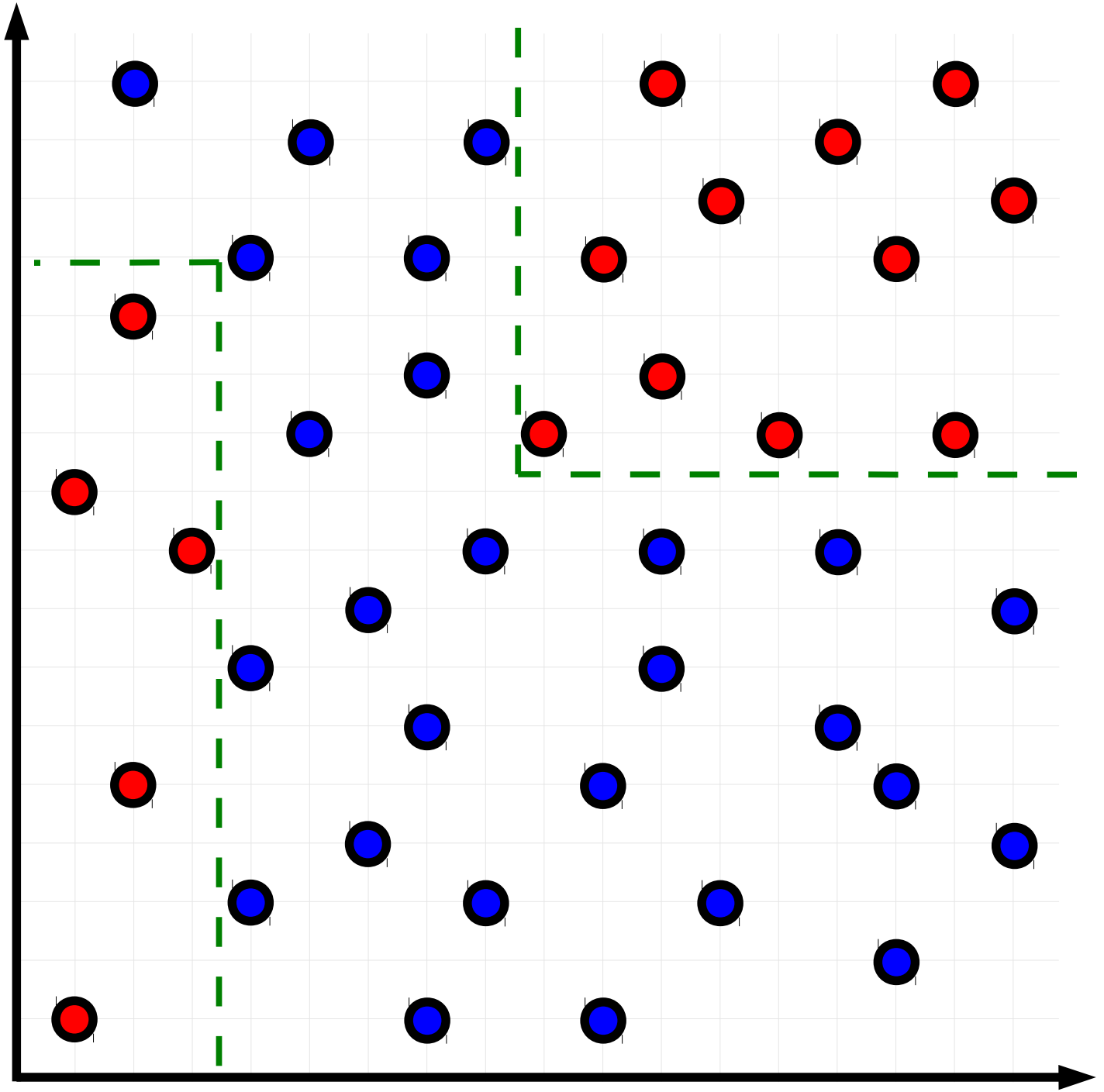


- Tenemos N puntos en el plano.
- Cada punto tiene dos coordenadas: (x, y).
- Cada punto tiene un color: **azul** o **rojo**.
- Queremos hallar una forma de determinar el color de un punto dado.
- Función $f(x, y) \rightarrow$ color
- $f(x, y) =$
  rojo si $x > 8$
  azul en caso contrario



- $f(x, y)$ =
  rojo si $y > m\,x + b$
  azul en caso contrario
- ***m*** y ***b*** son parámetros del modelo que deben ajustarse a los datos.





Humanos vs. Máquinas

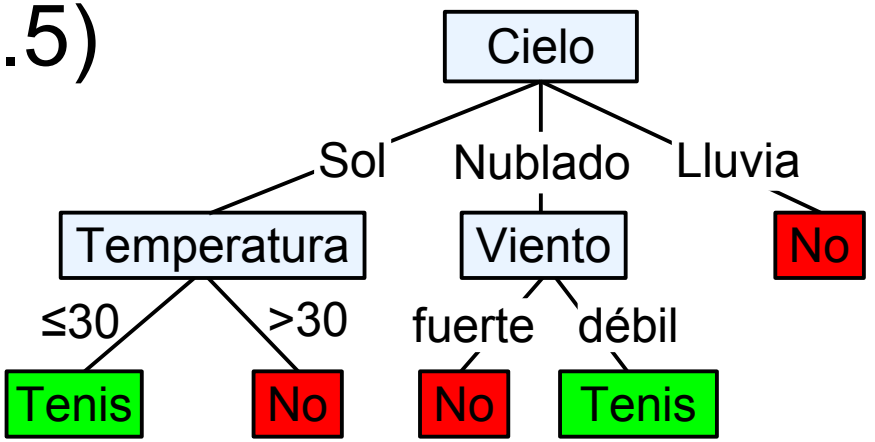
- Los humanos somos buenos encontrando estas reglas en 2D.
- Pero, ¿qué pasa si los puntos tienen decenas/ cientos/miles de coordenadas?
- Ahí es donde ML se hace indispensable.

Terminología

- Puntos → Instancias.
- *x*, *y* → Atributos de las instancias.
- Color → Clase de las instancias.
- Encontrar una función → Entrenar un modelo.

Algoritmos de Machine Learning

- Árboles de decisión (C4.5)



- Reglas (Ripper)

  IF (Cielo=Sol $\wedge$ Temperatura>30) THEN Tenis=No

  IF (Cielo=Nublado $\wedge$ Viento=Débil) THEN Tenis=Sí

  IF (Cielo=Lluvia) THEN Tenis=No

- Naive Bayes
- Support Vector Machines
- ...

Aplicaciones en PLN y Habla

- Segmentación en oraciones.
- Etiquetado de clases de palabra (*POS tagging*).
- Desambiguación del sentido (ej: abreviaturas, expresiones numéricas).
- Asignación de prosodia (TTS front-end).
- Detección del {sexo, id, edad} del hablante.
- Detección del idioma o dialecto.
- **Pronunciación de siglas.**
- ...

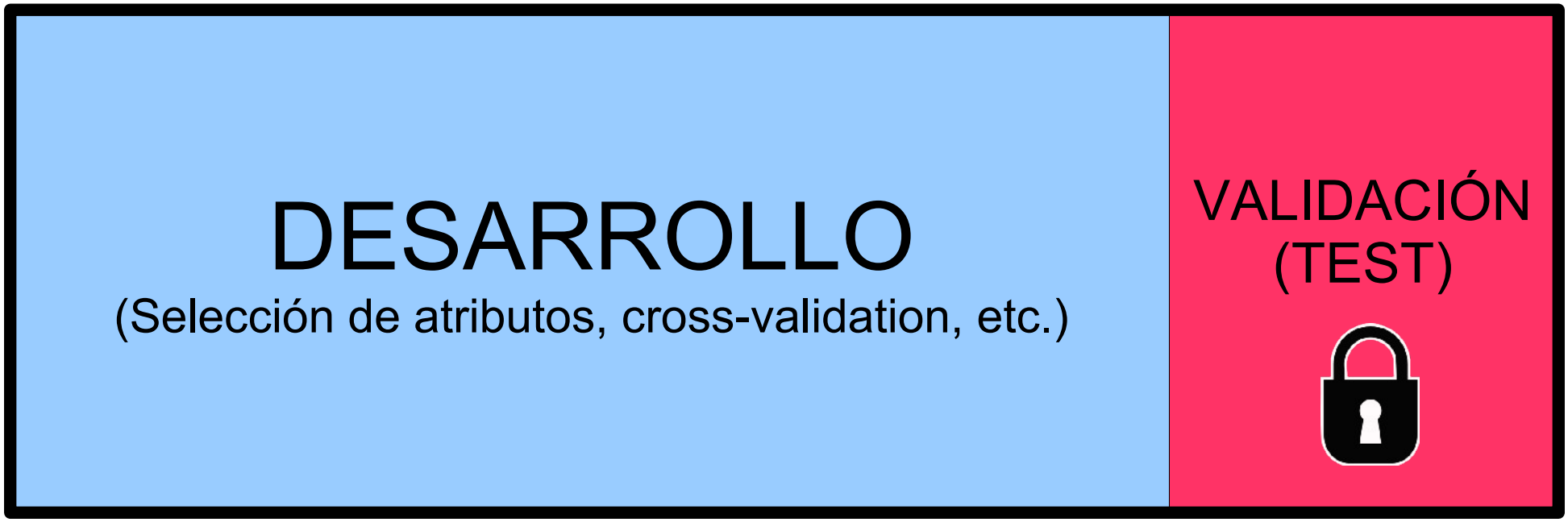
Pronunciación de Siglas

- Tarea: Determinar cómo pronunciar siglas en español.
  - **Input:** sigla.
    - Ejemplos: DGI, IBM, FBI, FMI, ATP, UBA, ALUAR, CONADUH, CONADEP, APTRA, AFA, FIFA.
  - **Output:** decidir si debe deletrearse, o leerse como un acrónimo.
- Este clasificador puede ser útil cuando encontramos una sigla desconocida en el texto a sintetizar.
- Por simplicidad, excluimos siglas con pronunciación especial: MIT (*emaití*), CNEA (*conea*), FCEN (*efecen*).
- siglas-listado.csv

Esquema General de ML

- Datos
  - Separar datos de desarrollo y validación.
  - Definir instancias, clases y atributos.
- Experimentación
  - Selección de atributos.
  - Medidas de performance.
  - Validación cruzada.
- Validación de los modelos

Datos de Desarrollo y Validación



- Lo antes posible, hay que separar un conjunto de datos de validación.
- Todas las pruebas y ajustes se hacen sobre el conjunto de desarrollo.
- Cuando termina el desarrollo, se evalúa sobre los datos separados.
- /home/ith50/clase09/siglas-listado-{dev,test}.csv

Instancias, clases y atributos

- Tenemos que definir:
  - cuáles son las instancias de nuestra tarea;
  - cuáles son sus clases y sus atributos.
- En nuestro ejemplo:
  - Instancias: Strings reconocidos como siglas en una etapa anterior del front-end.
  - Clases: Deletrear vs. Leer como un acrónimo.
  - Atributos: longitud de la sigla; #consonantes; #vocales; #consonantes consecutivas; … ???
  - extraer-atributos.py, siglas-{dev,test}.csv

Experimentación en Weka

- http://www.cs.waikato.ac.nz/ml/weka/
- Herramienta de ML basada en Java.
- Implementa ***muchos*** algoritmos.
- 3 modos de operación
  - GUI
  - Línea de comandos
  - Java: API, y código abierto (licencia GNU-GPL).
- Cómo ejecutarlo en Linux: **`weka -m 1024M &`**

  (o más memoria si hace falta.)

Weka

1) Iniciar GUI de Weka.
   - **`weka -m 1024M &`**
   - Presionar “Explorer”.
2) Solapa Preprocess: manipular archivos de datos.
   i. “Open file...” → /home/ith50/clase09/siglas-dev.csv
   ii. Abajo a la derecha elegir como clase al campo “clase”.
   iii. Presionar “Visualize All”:
      - Cuán bien discrimina cada atributo respecto de la clase.
      - Estaría bueno poder quedarnos sólo con los mejores...

Selección de Atributos

- ¿Demasiados atributos?
  - Aprendizaje muy lento (ej, para SVM).
  - Riesgo de sobreajuste (*overfitting*).
- Selección de atributos: usar sólo un subconjunto útil.
- Ejemplo: Ranking según su ganancia de información.

Selección de Atributos en Weka

- Solapa *“Select attributes”.*
  - Como *“Search Method”,* elegir *Ranker.*
  - Como *“Attribute Evaluator”*, elegir *InfoGainAttributeEval*.
- Click en el nombre del método elegido: abre una ventana con opciones y más información.
- En el menú desplegable, elegir “(Nom) clase” como clase.
- Presionar “Start”.
- Esto rankea los atributos según su ganancia de información con respecto a la clase.
- **Obs.:** Esta solapa es para experimentar. No altera los datos.

Selección de Atributos en Weka

- Solapa “Preprocess”.
- Click en *Filter* → *Choose.*
  - weka.filters.unsupervised.attribute.Remove
  - *attributeIndices*: ingresar los *k* mejores (según la selección de atributos). Ej: “2,4,5,9,12,10,6,7,14,1,40” (no olvidar la clase)
  - invertSelection: True (para quedarnos con estos atributos)
- Click “ok” para cerrar el diálogo. Click en *Apply.*
- Esto deja solo los atributos seleccionados.
- Esto también se puede hacer desde el shell de Linux:

```
java  weka.filters.unsupervised.attribute.Remove \
  -V -R "2,4,5,9,12,10,6,7,14,1,40" -i input.arff -o output.arff
```

Selección de Atributos

- Otras técnicas de selección de atributos:
  - Greedy forward selection
    1) $S \leftarrow \emptyset$ (S = conjunto de atributos)
    2) Para cada atributo a_i no usado, evaluar $S \cup \{a_i\}$.
    3) Si ningún $S \cup \{a_i\}$ produce una mejora, devolver S.
    4) En caso contrario, $S \leftarrow S \cup \{a_i\}$ y volver a 2).
  - Greedy backward elimination
  - Búsqueda exhaustiva
- Evaluación: Performance con algoritmos de ML rápidos como Ripper o C4.5.

Experimentación con Clasificadores

- Ya elegimos los mejores atributos.
- Siguiente paso: elegir un clasificador para entrenar un modelo.
- Para ello, experimentamos con diferentes clasificadores y configuraciones, siempre **sobre los datos de desarrollo**.
- Una vez elegido el mejor clasificador, entrenamos el modelo usando todos los datos de desarrollo.

Experimentación con Clasificadores en Weka

- Ir a la solapa 'Classify'.
  - Elegir un clasificador: Trees → J48.
  - Elegir Percentage split: 80%
    - o sea: 80% de los datos para training, 20% test.
  - Click en "**`J48 -C 0.25 -M 2`**" para ver opciones.
  - Click en *"Start"* y esperar...
  - Click derecho en la lista de resultados:
    - *Save result buffer.*
    - *Save model.*
    - *Visualize tree.*

Medidas de Performance

- Ejemplo: detector de *spam*. Clase = {*mail, spam*}
- Matriz de confusión
  - 100 datos de test: 50 spam y 50 mails.

		Valor real: *mail*	Valor real: *spam*
Valor predicho	*mail*	40 (tn)	5 (fn)
	spam	10 (fp)	45 (tp)

tp = *true positive*
fp = *false positive*
tn = *true negative*
fn = *false negative*

- Accuracy (% de aciertos) = (tp + tn) / total = 0.85
- Precisión = tp / (tp + fp) = 45 / (45 + 10) = 0.82
- Recall = tp / (tp + fn) = 45 / (45 + 5) = 0.90
- *F*-measure = (2 · precision · recall) / (precision + recall) = 0.86

Experimentación con Clasificadores en Weka

- Algunos clasificadores para probar:
  - weka.classifiers.trees.J48 (Arboles de decisión: C4.5)
  - weka.classifiers.bayes.NaiveBayes
  - weka.classifiers.rules.JRip (Aprendizaje de reglas: Ripper)
  - weka.classifiers.functions.SMO (Support Vector Machines)
    - 1) Transforma los datos a un espacio de dimensión superior.
    - 2) Clasifica los datos mediante un hiperplano en esa dimensión.
    - Entrenamiento muy costoso. Muy buenos resultados.
    - http://www.youtube.com/watch?v=3liCbRZPrZA
- Usar cross-validation: 5 fold, 10 fold.

Validación Cruzada

- 90% → datos de entrenamiento (*training data*)
- 10% → datos de validación (*validation / test data*)
- Datos de validación: **no** se usan en el entrenamiento.
- ¿Qué pasa si justo el 10% es muy “difícil” o “fácil”?
  - Sub o sobreestimación de la performance.
- Validación cruzada (*cross validation*)
  - Dividir la muestra en *k* conjuntos (*k folds*, ej: k=10).
  - Para *i* = 1..*k*: entrenar en todos menos *i*-ésimo conjunto, validar sobre el *i*-ésimo conjunto.
  - Performance: promedio de las *k* iteraciones.

Validación del Modelo Final

- Una vez terminado el desarrollo (seleccionamos los atributos, elegimos y configuramos el clasificador con mejores resultados con validación cruzada sobre los datos de desarrollo, y entrenamos el clasificador usando todos los datos de desarrollo), podemos **evaluar** el modelo final sobre **los datos de validación**.
- Esto nos da una **estimación realista** de la performance del modelo.
- Una vez usados los datos de validación, **no se debe volver atrás**.

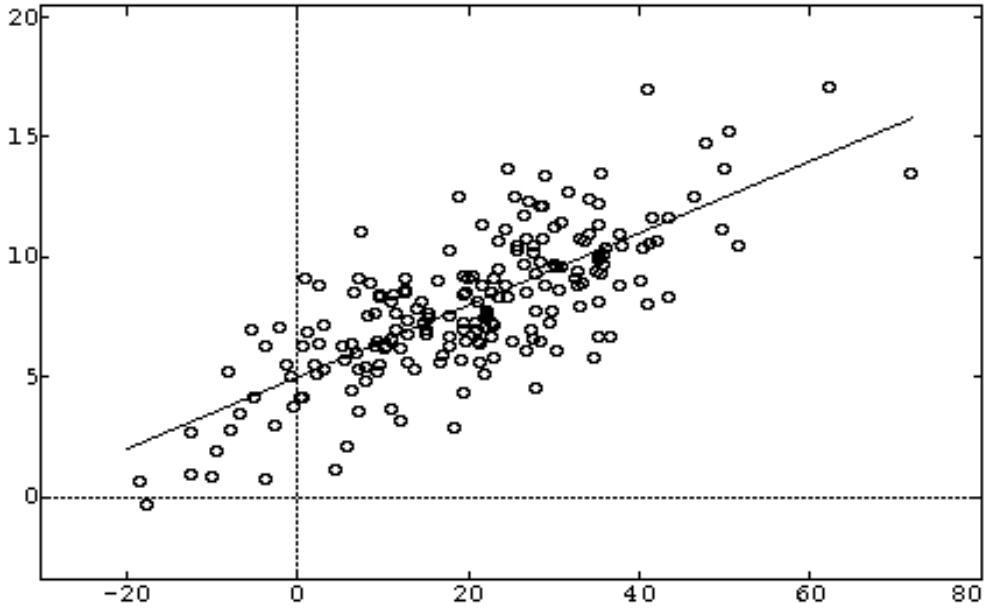
siglas-test.csv

Aprendizaje Automático

- Aprendizaje supervisado
  - Los datos de entrenamiento ya están clasificados.
  - Clasificación
    - Cada instancia pertenece a una clase:
      Pronunciación = {deletreo, acrónimo}
      ClasePalabra = {verbo, sustantivo, adjetivo, adverbio, ...}

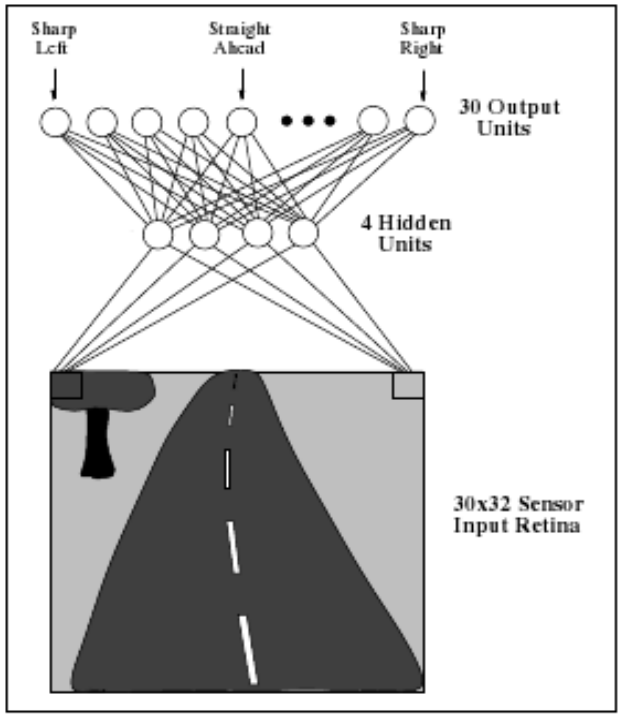
Aprendizaje supervisado: Regresión

- Las instancias no tienen una clase objetivo, sino un valor numérico objetivo.
  - *SpamFilter* : *mensaje* → probabilidad [0,1] de ser spam
- Ejemplo: Regresión lineal
  - $y = \beta_0 + \beta_1 x_1 + \ldots + \beta_n x_n$
  - y : variable dependiente (variable objetivo)
  - x_i: vars independientes (atributos)
  - β_i: coeficientes a ajustar
  - Ej: $y = 5 + 0.15\,x$

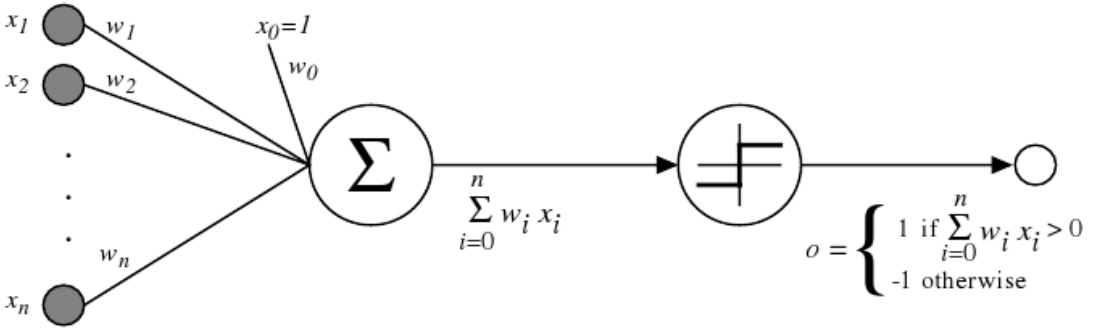


Aprendizaje supervisado: Redes neuronales

- Output: Vector de valores.
- Función objetivo tiene forma desconocida (y no importa).
- Ejemplo: conductor autónomo.



- Perceptron:

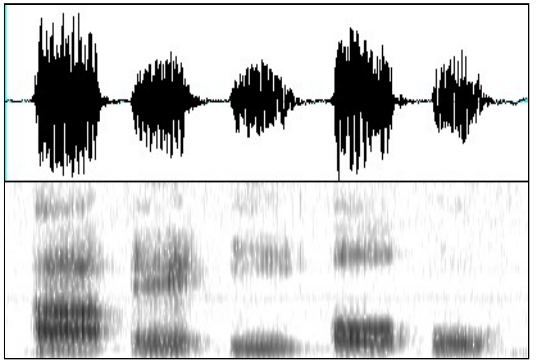


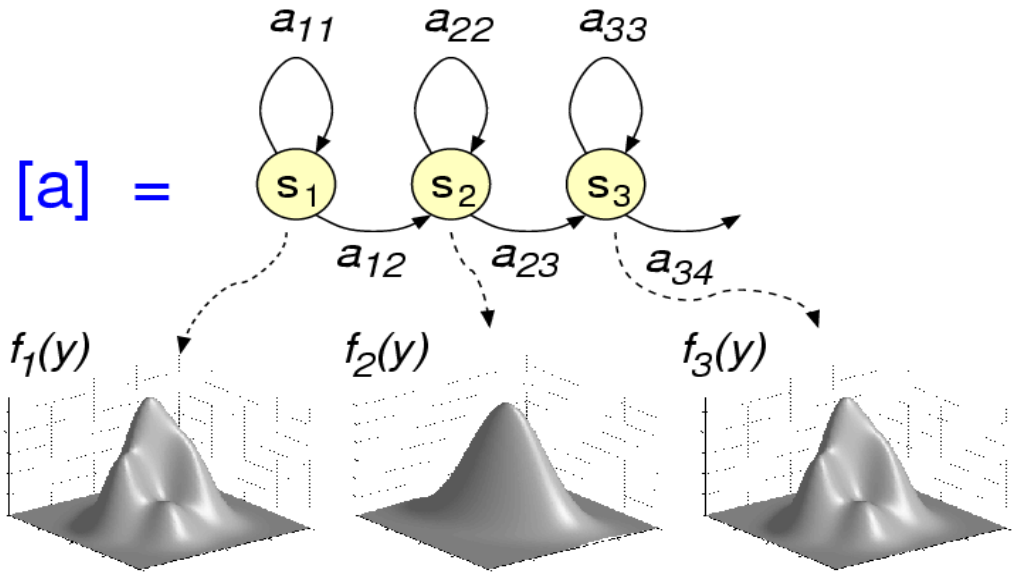
$$o(x_1, \ldots, x_n) = \begin{cases} 1 \text{ if } w_0 + w_1 x_1 + \cdots + w_n x_n > 0 \\ -1 \text{ otherwise.} \end{cases}$$

- Algoritmo de *Backpropagation*.

Aprendizaje supervisado: Hidden Markov Models

- Modelo gráfico probabilístico.
- Input: secuencia de observaciones
  - Ejemplo: valores espectrales.
- Output: secuencia de estados ocultos
  - Ejemplo: fonemas.



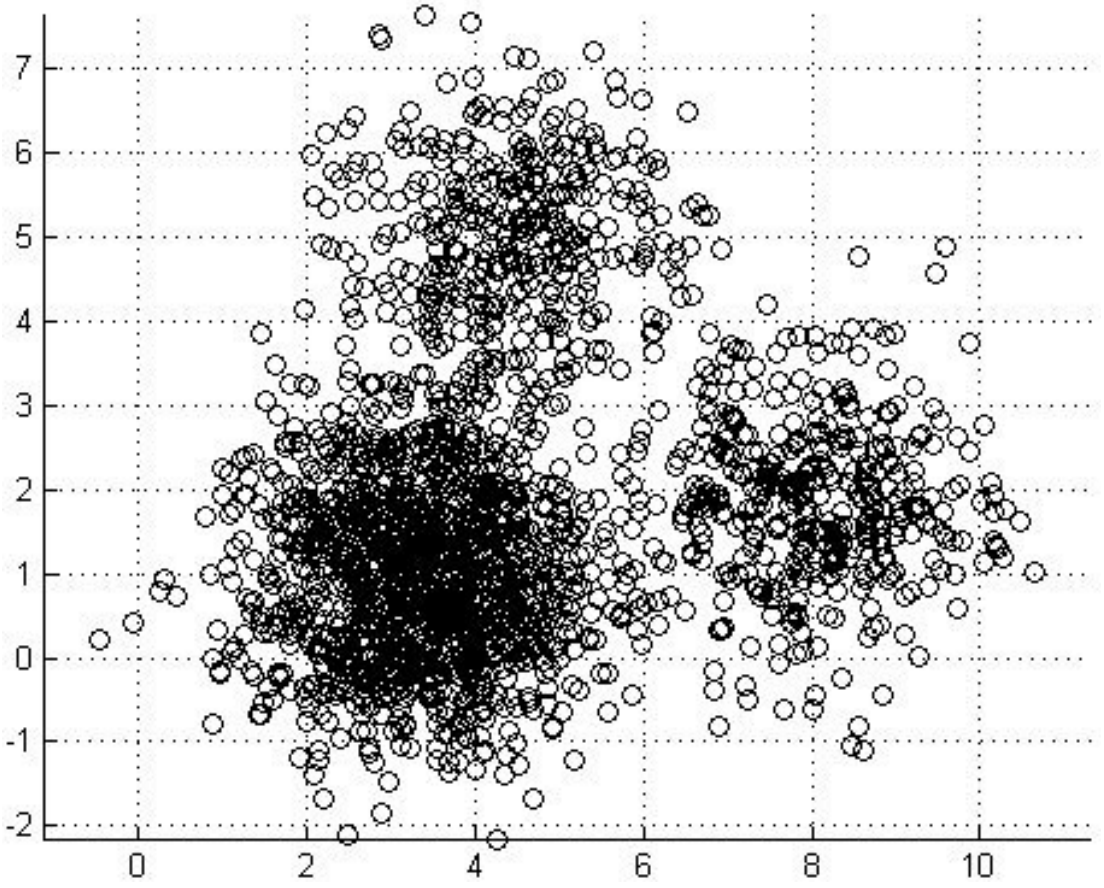


Aprendizaje Automático

- Aprendizaje supervisado
  - Los datos de entrenamiento ya están clasificados.
  - Clasificación
    - Cada instancia pertenece a una clase:
      Pronunciación = {deletreo, acrónimo}
      ClasePalabra = {verbo, sustantivo, adjetivo, adverbio, ...}
- Aprendizaje no supervisado
  - Los datos de entrenamiento no están clasificados.
    - Ya sea porque no existe o no se conoce una clasificación.

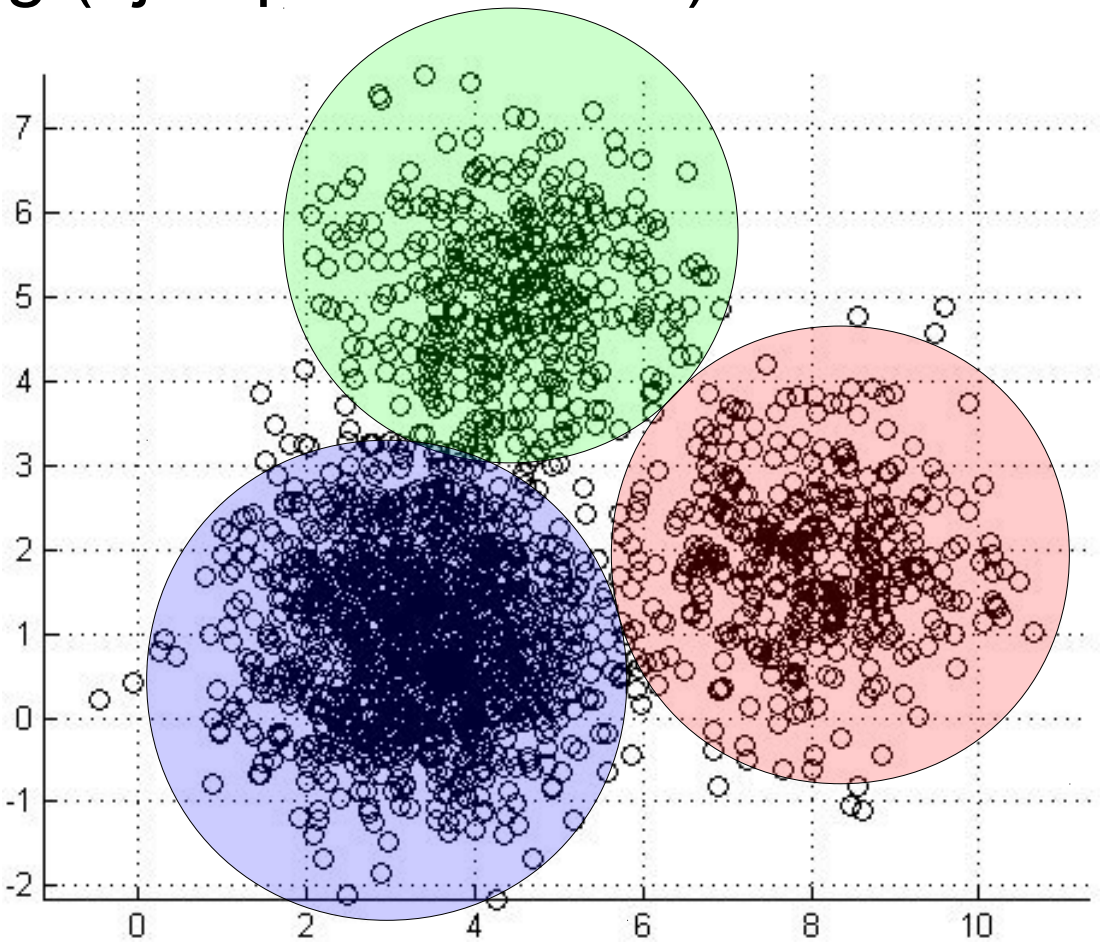
Aprendizaje no supervisado

- Los datos de entrenamiento no están clasificados.
- Clustering (ejemplo: *k-means*).



Aprendizaje no supervisado

- Los datos de entrenamiento no están clasificados.
- Clustering (ejemplo: *k-means*).



Aprendizaje Automático

- Aprendizaje supervisado
  - Los datos de entrenamiento ya están clasificados.
  - Clasificación
    - Cada instancia pertenece a una clase:
      Pronunciación = {deletreo, acrónimo}
      ClasePalabra = {verbo, sustantivo, adjetivo, adverbio, ...}
- Aprendizaje no supervisado
  - Los datos de entrenamiento no están clasificados.
    - Ya sea porque no existe o no se conoce una clasificación.
- Aprendizaje por refuerzos
  - Aprendizaje gradual, en base a premios y castigos.

Aprendizaje por refuerzos

- Conjunto de estados que definen el medio.
- Conjunto de acciones que el agente puede realizar.
- Reglas de transiciones entre estados.
- Reglas que asignan una recompensa inmediata (+ o -) a cada transición.
- Reglas que determinan qué observa el agente.
- Ejemplos: control de robots, programa de ascensores, ruteo de paquetes, juego del Go.

Formato .arff de Weka

- http://www.cs.waikato.ac.nz/~ml/weka/arff.html

```
% 1. Title: Iris Plants Database %
@RELATION iris
@ATTRIBUTE sepallength NUMERIC
@ATTRIBUTE sepalwidth NUMERIC
@ATTRIBUTE petallength NUMERIC
@ATTRIBUTE petalwidth NUMERIC
@ATTRIBUTE class {Iris-setosa,Iris-versicolor,Iris-virginica}

@DATA
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
...
```

Formato .arff de Weka

`@attribute attrName {numeric, string, nominal, date}`

- `numeric`: un número
- `nominal`: un conjunto (finito) de strings, e.g.
  `{Iris-setosa,Iris-versicolor,Iris-virginica}`
- `string`: string arbitrario
- `date`: (default ISO-8601) **`yyyy-MM-dd'T'HH:mm:ss`**

Weka desde la línea de comandos

- N-fold cross validation con NaiveBayes:

```
java -cp /usr/share/java/weka.jar
      weka.classifiers.bayes.NaiveBayes
      -t training_data.arff -x N -i
```

  Ejemplo:

```
java -cp /usr/share/java/weka.jar
      weka.classifiers.bayes.NaiveBayes
      -t zoo.arff -x 5 -i
```

- Cómo usar un conjunto de test predefinido:

```
java -cp /usr/share/java/weka.jar
      weka.classifiers.bayes.NaiveBayes
      -t training_data.arff -T test_data.arff
```

Weka desde la línea de comandos

- Cómo guardar un modelo entrenado:

```
java -cp /usr/share/java/weka.jar
      weka.classifiers.bayes.NaiveBayes
      -t trainingdata.arff -d output.model
```

- Cómo clasificar usar un conjunto de test predefinido:

```
java -cp /usr/share/java/weka.jar
      weka.classifiers.bayes.NaiveBayes
      -l input.model -T testingdata.arff
```

Resumen de la clase de hoy

- Aprendizaje supervisado. Clasificación.
  - Tarea, instancia, clase, atributos.
  - Datos de desarrollo y validación.
  - Selección de atributos.
  - Clasificadores.
  - Validación cruzada.
  - Weka.
- Clase que viene: extracción de atributos para tareas de ML de habla.