Retos del Aprendizaje Automático

"Malos datos" y/o "Malos algoritmos"

Dr.C. Eduardo Concepción Morales
Universidad de Cienfuegos

Problemática general

- Identificación de los problemas y planteamiento de las preguntas

  - ¿Podremos predecir el desempeño esperado de un estudiante al culminar el primer año a partir de las características de su perfil?
  - ¿Podríamos obtener un estimado de la demanda de electricidad para las próximas semanas a partir de determinadas métricas de consumo?
  - ¿Podremos identificar tumores de manera automática en imágenes del cerebro?
  - ¿Se podría recomendar un libro, posiblemente de interés, a una persona a partir de sus lecturas anteriores?

Problemática general (cont.)

- Los pasos que se deben acometer son los siguientes:
  - El estudio de los datos
  - La selección de un modelo
  - El entrenamiento del modelo con los datos
  - Realización de predicciones para nuevos casos (inferencia), con la esperanza que el modelo generalice bien.

Sobre los datos

- **Insuficiente cantidad de datos de entrenamiento**
  - La mayoría de los algoritmos de AA requieren de muchos datos de entrenamiento (miles y hasta millones, según el caso).
- **Datos de entrenamiento no representativos**
  - Con el fin de lograr una buena generalización, los datos de entrenamiento deben ser representativos de los nuevos casos.
  - Si la muestra es muy pequeña, puede existir el **ruido muestral** (es decir, datos no representativos por el efecto del azar)
  - Es **sesgo muestral** puede ocurrir si el método de muestreo e imperfecto.

Sobre los datos (cont.)

- **Datos de entrenamiento de mala calidad**
  - Errores, valores atípicos, datos ruidosos (producto de mediciones de baja calidad) hacen más difícil la tarea del sistema de detectar los patrones subyacentes.
  - Vale la pena realizar una limpieza de los datos
  - Por ejemplo:
    - Si hay valores atípicos *(outliers)*, se pudieran desechar, o corregirlos manualmente
    - Si algunas instancias no poseen datos *(missing data)* en algún rasgo, se debe decir si se desecha la instancia completa, si se desecha el rasgo completo, o si se sustituye el dato ausente por algún valor (por ejemplo, la mediana)

Sobre los datos (cont.)

- **Rasgos irrelevantes**
  - Si entra basura, sale basura (*garbage in, garbage out*). Es necesario utilizar rasgos relevantes suficientes para que el sistema sea capaz de aprender.
  - Este proceso se denomina **ingeniería de rasgos (*feature engineering*)**, e involucra los siguientes pasos:
    - **Selección de rasgos (*feature selection*)**: escoger los rasgos más útiles para el entrenamiento.
    - **Extracción de rasgos (*feature extraction*)**: combinar rasgos existentes para la obtención de otros posiblemente más útiles (por ejemplo, los algoritmos de reducción de la dimensionalidad)
    - **Creación de nuevos rasgos**: adquirir nuevos datos.

Sobre los algoritmos

- **Regresión logística**: Se utiliza en la clasificación binaria (y también multi clase).
- **Árboles de decisión (***desicion trees***)**: El modelo de entrenamiento aprende a predecir una variable de salida mediante el ajuste de reglas de decisión y de una representación arbórea.
- **Bosques aleatorios (***random forests***)**: Es una combinación de árboles de decisión.
- ***Boosting***: La idea general consiste en entrenar los predictores de manera secuencial, cada uno tratando de corregir a su predecesor. Por ejemplo**, XGBoost** (biblioteca)**.**

Sobre los algoritmos. Retos.

- **Ajuste en exceso** (*overfitting*): el modelo se desempeña muy bien en los datos de entrenamiento; pero no generaliza bien, es decir, su desempeño con los nuevos datos no es bueno.
- El ajuste en exceso ocurre cuando el modelo es muy complejo respecto a la cantidad y el ruido de los datos. Las soluciones en estos casos pueden ser:
  - Simplificar el modelo seleccionando uno con **menos parámetros** (por ejemplo, un modelo lineal en lugar de uno polinómico), **reduciendo la cantidad de atributos** en los datos de entrenamiento, o **restringiendo el modelo** (regularización).
  - Obtener más datos de entrenamiento.
  - Reducir el ruido en los datos (eliminar valores atípicos, corregir errores)

Sobre los algoritmos. Retos (cont.).

- **Ajuste deficiente** (*underfitting*): ocurre cuando el modelo es tan simple que no aprende la estructura subyacente en los datos de entrenamiento.
- Las principales opciones para corregir este problema son:
  - Seleccionar un modelo más poderoso, con más parámetros.
  - Incluir mejores rasgos (ingeniería de rasgos)
  - Reducir las restricciones del modelo (por ejemplo, reducir el hiper-parámetro de regularización)
  - Obtener más datos

Resumiendo

- Los pasos que se deben acometer son los siguientes:
  - El estudio de los datos
  - La selección de un modelo
  - El entrenamiento del modelo con los datos
  - Realización de predicciones para nuevos casos (inferencia), con la esperanza que el modelo generalice bien.

Resumiendo (cont.)

- En un proyecto de Aprendizaje Automático se seleccionan los datos de entrenamiento, los cuales se le suministran a un algoritmo de aprendizaje.
- Si el algoritmo es basado en un modelo, se afinan ciertos parámetros para ajustar el modelo a los datos de entrenamiento (es decir, para obtener buenos resultados en los datos de entrenamiento), y con la esperanza de que también se obtengan buenas predicciones para los datos nuevos.
- El sistema no tendrá buen desempeño si los datos de entrenamiento son escasos, o si los datos no son representativos, o son ruidosos, o si están contaminados con rasgos irrelevantes (GIGO).
- El modelo no debe ser muy simple (para evitar un ajuste deficiente – *underfitting*, ni muy complejo para evitar el ajuste en exceso (*overfitting*).

Prueba y validación

- La única manera de saber si un modelo generaliza bien es probarlo con nuevos casos.
- Un (mala) manera sería probarlo en producción y monitorear su desempeño.
- Una mejor opción consiste en dividir los datos en dos conjuntos: **conjunto de entrenamiento** (***training set***) y un **conjunto de pruebas** (***test set***).

Prueba y validación (cont.)

- La tasa de error ante los nuevos casos se denomina **error de generalización**.
- Evaluando el modelo con los datos de pruebas (no con los de entrenamiento) se puede obtener una estimación de este error.
- Si el error del entrenamiento es bajo, pero el error de generalización es alto, eso significa que el modelo está ajustándose en exceso a los datos de entrenamiento.
- Es común tomar alrededor de un 80% de los datos para el entrenamiento y un 20% para las pruebas.

Prueba y validación (cont.)

- Verdaderos positivos, falsos negativos, falsos positivos, verdaderos negativos.
- Precisión (*precision*), exactitud (*accuracy*), sensibilidad (*sensitivity, recall, true positive rate*), especificidad (*specificity, true negative rate*)

Algoritmo/Realidad	Condición Negativa	Condición Positiva	
Resultado Negativo	Verdadero Negativo	Falso Negativo	
Resultado Positivo	Falso Positivo	Verdadero Positivo	Precisión=VP/RP
	Especificidad=VN/CN	Sensibilidad=VP/CP	Exactitud=(VN+VP)/Total

Ajuste de hiperparámetros y selección de modelos

- La evaluación de un modelo es simple: utilizamos el conjunto de pruebas.
- ¿Cómo proceder para decidir entre dos (o más) modelos?
- Una opción es entrenar ambos modelos y comparar su desempeño en los datos de pruebas.

Ajuste de hiperparámetros y selección de modelos

- Supongamos ahora que se escogió el mejor modelo, pero se quieren refinar sus parámetros (por ejemplo, un hiperparámetro de regularización).
- Si se entrena el modelo para diferentes valores de los hiperparámetros, y se escoge el valor que produzca el menor error de generalización, en general, no se garantiza un buen desempeño.
- El problema estriba en que el error de generalización se mide varias veces sobre el mismo conjunto de pruebas de manera que el modelo y el hiperparámetro se **ajustaron para ese conjunto en particular**.

Ajuste de hiperparámetros y selección de modelos (cont.)

- Una solución común a este problema es la validación con retención (*holdout validation*): simplemente se separa una parte del conjunto de entrenamiento para evaluar varios modelos candidatos y se selecciona el mejor.
- El nuevo conjunto se denomina conjunto de validación (*validation set*, *development set*, *dev set*).

Ajuste de hiperparámetros y selección de modelos (cont.)

- Se entrenan múltiples modelos con varios hiperparámetros en el conjunto de entrenamiento reducido (el conjunto original de entrenamiento menos el conjunto de validación), y se selecciona el de mejor desempeño en el conjunto de validación.
- Después de este proceso, se entrena de nuevo el modelo seleccionado con el conjunto de entrenamiento completo (incluyendo el conjunto de validación), y este es el modelo final.

Ajuste de hiperparámetros y selección de modelos (cont.)

- Finalmente, se evalúa el modelo entrenado en el conjunto de pruebas para obtener una estimación del error de generalización.
- Esto puede generar problemas tanto en el caso de que el conjunto de validación sea pequeño, como en el caso si el conjunto de entrenamiento es pequeño.
- Una solución consiste en la realizar validaciones cruzadas *(cross-validation)* utilizando muchos conjuntos de validación pequeños.

Ajuste de hiperparámetros y selección de modelos (cont.)

- El modelo se evalúa una vez por cada conjunto de validación, mientras es entrenado en el resto de los datos.
- Promediando todas las evaluaciones de un modelo se obtiene una medida más exacta de su desempeño.
- Esto tiene un inconveniente: el tiempo de entrenamiento se multiplica por la cantidad de conjuntos de validación utilizados.

Validación	Entrenamiento	Entrenamiento	Entrenamiento
Entrenamiento	**Validación**	Entrenamiento	Entrenamiento

Ajuste de hiperparámetros y selección de modelos (cont.)

- Búsqueda en rejilla (Grid Search)
- Búsqueda aleatorizada (Randomized Search)
- Combinación de modelos


mirror_mod.use_x = False
mirror_mod.use_y = True
mirror_mod.use_z = False
elif operation == "MIRROR_Z":
mirror_mod.use_x = False
mirror_mod.use_y = False
mirror_mod.use_z = True
#selection at the end -add back the deselected mirror modifier object
mirror_ob.select= 1
modifier_ob.select=1
bpy.context.scene.objects.active = modifier_ob
print("Selected" + str(modifier_ob)) # modifier ob is the active ob
#mirror_ob.select = 0

¡Gracias!