KNN y balanceo de clases

Jocelyn Dunstan Escudero

jdunstan@uc.cl

Departamento de Ciencia de la Computación
& Instituto de Matemática Computacional
Pontificia Universidad Católica de Chile

Santiago, Chile

Objetivos

- Identificar las características fundamentales y el funcionamiento del algoritmo KNN
- Analizar la importancia del preprocesamiento de datos, especialmente el escalado de características
- Explorar estrategias para manejar desafíos comunes en modelado de datos como el desequilibrio de clases y la imputación de valores faltantes.

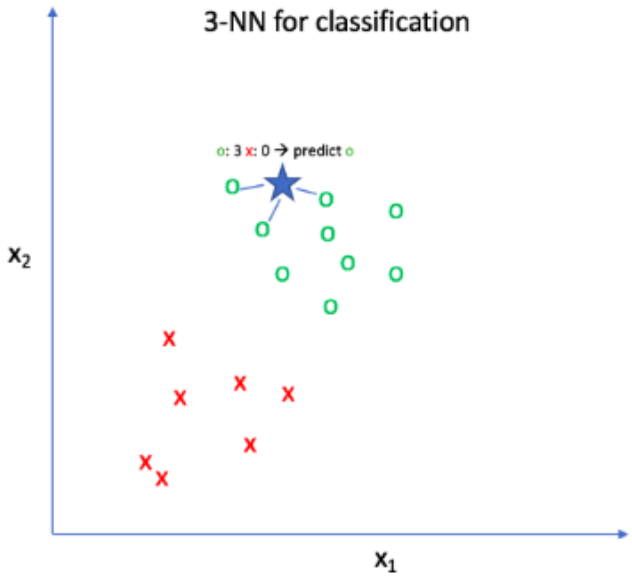
K-nearest neighbors

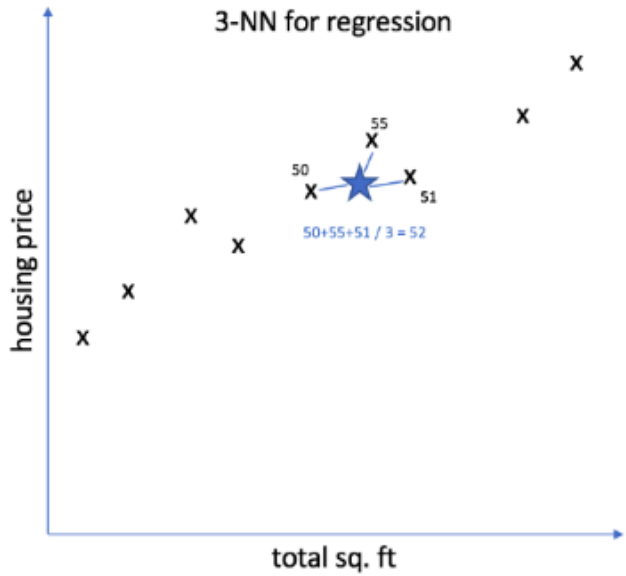
Idea general

- Es uno de los algoritmos de ML más simple que existen
- Siempre necesita el conjunto de entrenamiento y no hay un modelo para probar en un conjunto de pruebas.
- Tienden a funcionar bien en datos pequeños con pocas características.
- Primero calcula la distancia entre los puntos.
- Elige la distancia a los puntos de datos más cercanos.
- Realiza una votación por mayoría para elegir la clase a la que pertenece.

https://towardsdatascience.com/introduction-to-k-nearest-neighbors-3b534bb11d26

K-nearest neighbors





https://www.jeremyjordan.me/k-nearest-neighbors/

Una nota acerca de trabajar con pocos datos

- No todo es deep learning... Especialmente si tienes pocos datos.
- Hay métodos que funcionan bien con pocas filas o de pocas columnas.
- Los modelos de baja complejidad funcionarán mejor (pocos parámetros).
- Los modelos lineales podrían ofrecer una buena solución.
- Los clasificadores, como el de k-vecinos más cercanos, también son buenas alternativas. Los métodos basados en árboles también son buenas opciones.

Magnitudes muy distintas entre características afecta el cálculo de distancias

Data Scaling

- Si el método calcula distancias, entonces es sensible a las mezclas de magnitudes altas y bajas. Existen los siguientes tipos de escalamiento:
  - Standardization (redistribución con media y desviación standard σ).
  - Mean normalization (devuelve valores entre -1 y 1 con media μ)
  - Min-max scaling (¡muy común! Lleva todos los valores entre 0 y 1)

https://medium.com/greyatom/why-how-and-when-to-scale-your-features-4b30ab09db5e

Data Scaling

scaling model to do trainig and testing
scaler = MinMaxScaler()
X_normalised= scaler.fit_transform(X)

new data comes in
new_test_point_normalised =
scaler.transform(new_test_point)

- Tenga en cuenta que los nuevos datos pueden estar fuera de [0,1], ¡y eso está bien! Solo necesitamos estar más o menos en la misma escala.
- Si una nueva magnitud entra en el modelo (y es relevante), entonces el modelo debe ser calculado de nuevo.

¿Cuándo escalar?

- K-nearest neighbors calcula distancias Euclidianas, hay que escalar.
- PCA se basa en la varianza de los datos, y una magnitud elevada implica una varianza elevada..
- El descenso por gradiente es rápido en las magnitudes pequeñas y lento en las altas, por lo que puede oscilar ineficientemente con diferentes magnitudes.
- Los modelos basados en árboles se basan en decisiones de un solo predictor, por lo que son independientes del escalado. No obstante, debe comparar diferentes métodos, así que empiece por el escalado y compare los árboles con otros algoritmos.
- El análisis lineal discriminante (LDA) y el Naive Bayes están preparados para manejar diferentes escalas, por lo que no es necesario un escalado previo.

https://medium.com/greyatom/why-how-and-when-to-scale-your-features-4b30ab09db5e

Balanceo de clases

Class imbalance

Podemos tener desequilibrio de clases (muy común). Puedes tener un clasificador con un rendimiento excelente que sólo predice la clase predominante.
Podemos proceder como sigue:

1. Si es posible, consiga más ejemplos de la clase subrepresentada.
2. Prueba una métrica diferente: la precisión no es una buena métrica en estos casos. Compruebe la matriz de confusión, calcule la precisión (exactitud) - recall (exhaustividad) - F1 (una combinación entre precisión y recall). También puedes utilizar la kappa de Cohen (clasificación normalizada por desequilibrio) y la curva ROC.

Class imbalance

Podemos proceder como sigue:

3. Si tiene muchos datos, es posible que quiera reducir la clase predominante.
4. Generar muestras sintéticas (SMOTE: Synthetic Minority Over-sampling Techinique)

https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/

Imputación de valores (esto no lo vamos a ver en profundidad)

Tipos de datos faltantes

- **Missing completely at random (MCAR):** independendiente de los datos observados y no observados. Reduce poder estadístico pero no introduce sesgo. Ej. Pacientes con resultados de examen faltantes debido a que no se procesaron correctamente.
- **Missing at random (MAR):** es sistemático a la data observada. Por ejemplo, si pacientes hombres son más propensos a no-rankear la severidad de su depresión vs. pacientes mujeres.
- **Not missing at random (NMAR):** sistemático en variables no medidas en el estudio. Siguiendo con el ejemplo anterior, es NMAR si pacientes con depresión severa no contestan la encuesta.

https://www.ncbi.nlm.nih.gov/books/NBK493614/

¿Qué hacer frente a datos faltantes?

1. **Nada:** Hay algoritmos que funcionan con datos faltantes (RF no pero XGBoost si)
2. **Imputación por la mediana/promedio:** es fácil pero no toma en cuenta correlación entre variables. Y no funciona en variables categóricas.
3. **Imputación por el valor más frecuente:** funciona con variables categóricas pero no toma en cuenta correlaciones.
4. **Usar k-NN:** muy útil porque usa similaridad entre observaciones. Es más trabajo que los anteriores y tiende a ser sensible a outliers.

¿Qué hacer frente a datos faltantes?

5. **Imputation Using Multivariate Imputation by Chained Equation (MICE):** hace la imputación usando ecuaciones en cadena y puede ser usado con variables continuas o categóricas.

```
from impyute.imputation.cs import mice

imputed_training=mice(train.values)
```

6. **Usar deep learning (datawig):** puede ser usado con variables continuas o categóricas.

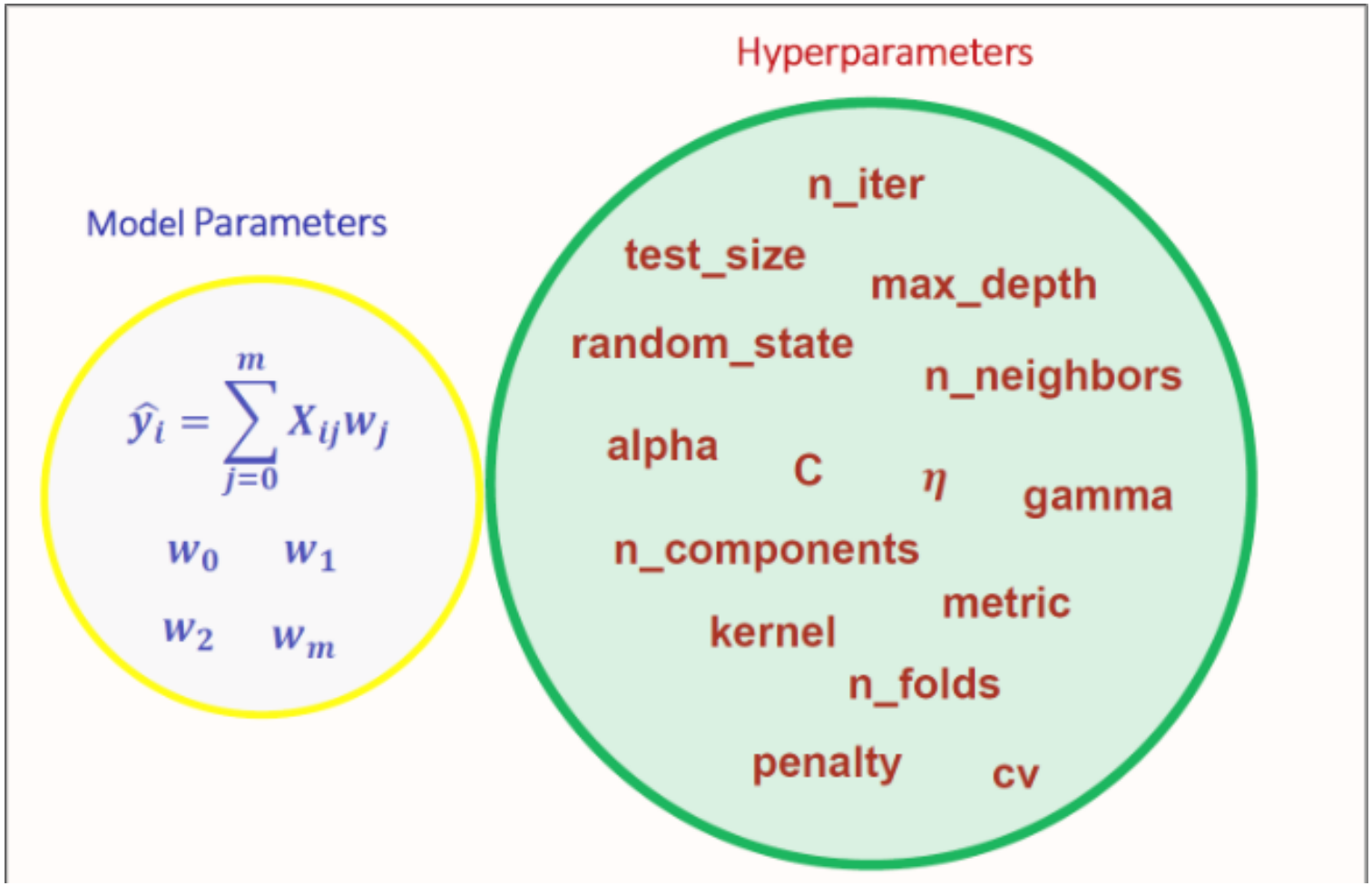
```
import datawig

imputer = datawig.SimpleImputer(
   input_columns=['1','2','3','4','5','6','7', 'target'], # column(s) containing information about the column we
want to impute
   output_column= '0', # the column we'd like to impute values for
   output_path = 'imputer_model' # stores model data and metrics
   )
```

https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/

Parametros vs. Hyperparametros

Parameters vs. hyperparameters



https://towardsdatascience.com/model-parameters-and-hyperparameters-in-machine-learning-what-is-the-difference-702d30970f6

Parameters vs. hyperparameters

- **Model Parameters:** These are the parameters in the model that must be determined using the training data set. These are the fitted parameters.
- **Hyperparameters:** These are adjustable parameters that must be tuned in order to obtain a model with optimal performance.

https://towardsdatascience.com/model-parameters-and-hyperparameters-in-machine-learning-what-is-the-difference-702d30970f6

Parameters and hyperparameters in DL

(Model Design + Hyperparameters) → Model Parameters

The building blocks:
- # Layers
- Activations
- Optimizers

...

The knobs that you can turn:
- Learning Rate
- Dropout

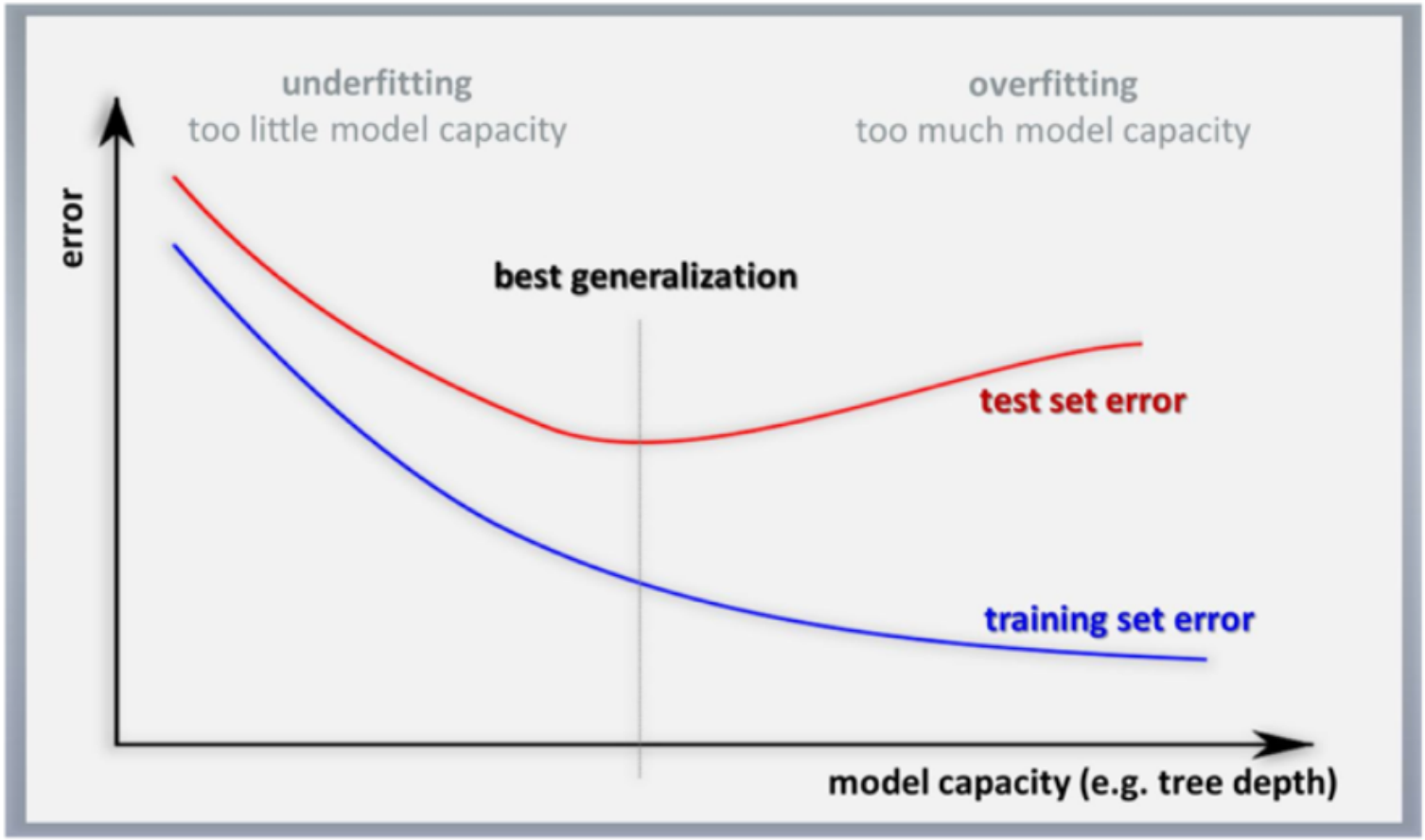
...

The variables learned from the data:
- weights

...

Model Design Variables + Hyperparameters → Model Parameters

https://www.programmersought.com/article/80954979866/


underfitting
too little model capacity
overfitting
too much model capacity
error
best generalization
test set error
training set error
model capacity (e.g. tree depth)

Objetivos

- Identificar las características fundamentales y el funcionamiento del algoritmo KNN
- Analizar la importancia del preprocesamiento de datos, especialmente el escalado de características
- Explorar estrategias para manejar desafíos comunes en modelado de datos como el desequilibrio de clases y la imputación de valores faltantes.