Aprendizaje Supervisado

Facundo Carrillo
fcarrillo@udesa.edu.ar

Taller III

Hoy: Taller III

Temas:

- K - vecinos más cercanos
- Recursos en sklearn:
  - Otros clasificadores
  - Selección de modelos: grid search
  - Preprocesamiento
  - Feature extraction
  - Reducción de dimensionalidad
  - Pipeline
  - Multi clase - Multi label

Bibliografía:

- Mitchell Machine Learning: website libro , pdf
- Scikit-learn: https://scikit-learn.org/

K - vecinos más cercanos (aka k-nearest neighbors)

Caracteristicas

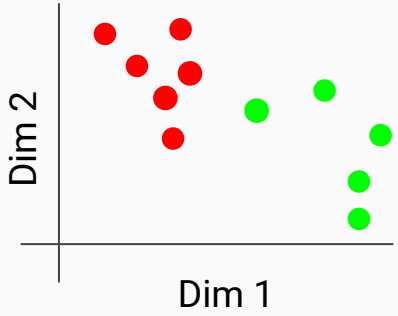
- Idea bastante trivial
- Tengo que definir un *K*
- La etapa de entrenamiento no hace mucho
- Necesito elegir/definir una medida de distancia (distancia euclidiana por ejemplo)
- ¿Atributos faltantes?

Pseudo-código de predict

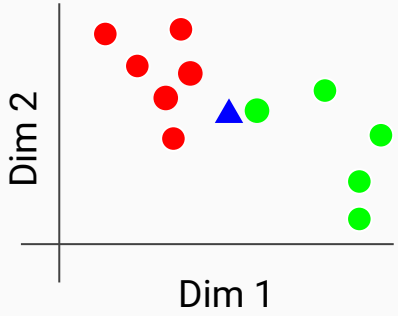
- Tomo el vector de *N* dimensiones que quiero predecir
- Me fijo las *K* instancias de entrenamiento más cercanas
- Devuelvo la clase con más frecuencia en los K vecinos más cerca

K - vecinos más cercanos (aka k-nearest neighbors)

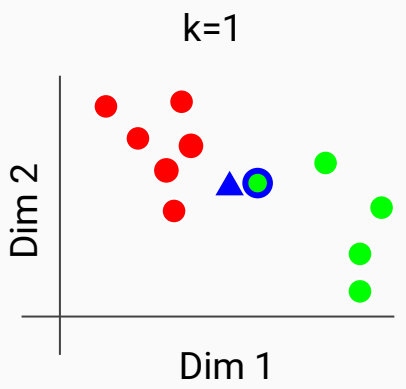
Entrenamiento



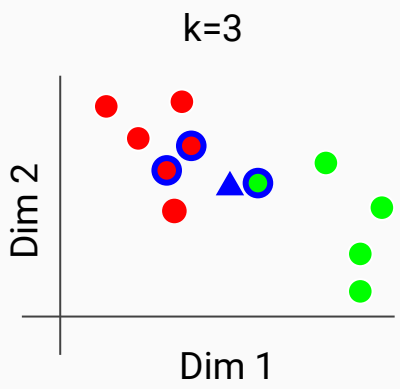
Predicción nueva muestra



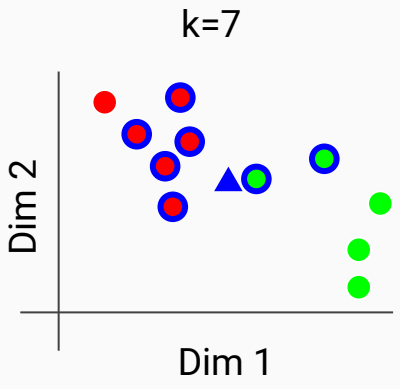
K - vecinos más cercanos (aka k-nearest neighbors)



El modelo predice ●



El modelo predice ●



El modelo predice ●

Preguntas/Ideas:

- ¿Qué pasa si empatan? ¿Quién gana?
- Podríamos pesar diferente según la distancia
- Podríamos pesar diferente con algún prior por clase (para pesar más un error frente a otro)
- Otras distancias (ej: coseno) en vez de distancia euclidea

Recursos Sklearn

Sklearn tiene muchas cosas, algunas relevantes:

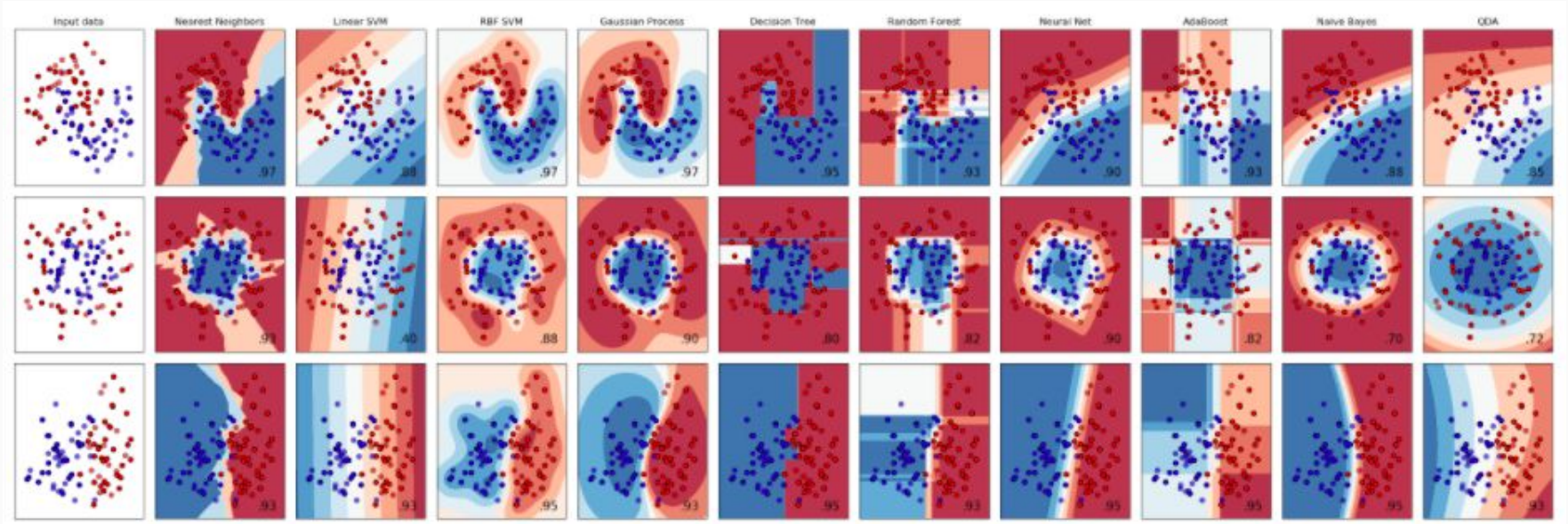
- Otros clasificadores
- Selección de modelos: grid search
- Preprocesamiento
- Feature extraction
- Reducción de dimensionalidad
- Pipeline
- Multi clase - Multi label

Recursos Sklearn: otros clasificadores

Clasificadores:

- Sklearn tiene muchos implementados, cada uno con muchos parámetros
- No hay ninguno necesariamente mejor

https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html



Recursos Sklearn: otros clasificadores

sklearn.neural_network.MLPClassifier

Perceptron multicapa (redes neuronales)

- Sklearn no se focaliza mucho en esto
- https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html#sklearn.neural_network.MLPClassifier

Tiene lo básico que podemos pedirle a un perceptrón multicapa:

- Cantidad de capas ocultas
- Elección de función de activación
- Solver para backpropagation
- Learning rate
- Etc

Más (y de verdad) de esto en materia **Redes Neuronales**

Recursos Sklearn: otros clasificadores

sklearn.neighbors.KNeighborsClassifier: K - vecinos más cercanos

- Cantidad de vecinos
- Función de peso de los vecinos
- Algoritmo para optimizar búsquedas en espacios grandes
- Métrica
- Etc

https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier

sklearn.svm.LinearSVC: Support Vector Machines

- Normal de penalidad (l1,l2)
- Loss function
- Parámetro de regularización
- Peso de las clases

https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC

Recursos Sklearn: Ensemble

Los ensambles permiten unir varios modelos con diferentes estrategias

- sklearn.ensemble.VotingClassifier

  Toma diferentes modelos y el resultado es un voto entre las predicciones de los diferentes modelos.

```
>>> clf1 = LogisticRegression(multi_class='multinomial', random_state=1)
>>> clf2 = RandomForestClassifier(n_estimators=50, random_state=1)
>>> clf3 = GaussianNB()
>>> X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])
>>> y = np.array([1, 1, 1, 2, 2, 2])
>>> eclf1 = VotingClassifier(estimators=[
...         ('lr', clf1), ('rf', clf2), ('gnb', clf3)], voting='hard')
>>> eclf1 = eclf1.fit(X, y)
>>> print(eclf1.predict(X))
```

Recursos Sklearn: Ensemble

Los ensambles permiten unir varios modelos con diferentes estrategias

- sklearn.ensemble.StackingClassifier

  Toma diferentes modelos, la predicción de cada uno de estos suele ser el input de otro modelo que es el que toma la decisión

```
>>> estimators = [
...     ('rf', RandomForestClassifier(n_estimators=10, random_state=42)),
...     (gnb, GaussianNB())
... ]
>>> clf = StackingClassifier(estimators=estimators, final_estimator=LogisticRegression())
```

Recursos Sklearn: Ensemble

Los ensambles permiten unir varios modelos con diferentes estrategias

- sklearn.ensemble.BaggingClassifier

  Entrena los clasificadores de base con un subset random de instancias de entrenamiento. Luego para una predicción nueva: vota

```
>>> clf = BaggingClassifier(base_estimator=SVC(),
...                         n_estimators=10, random_state=0).fit(X, y)
```

Recursos Sklearn: Grid Search

- Ya conocemos un montón de clasificadores (y aún hay más)
- Cada uno tiene varios y diferentes parámetros
- Tenemos ensambles que nos permiten combinar varios…

Osea, las pruebas que podemos hacer son exponenciales. ¿Cómo exploramos ordenadamente esto? Grid Search!

Recursos Sklearn: Grid Search

Exhaustive Grid Search: Dado un clasificador, definimos exhaustivamente diferentes atributos para que se prueben todos

```
>>> parameters = {'kernel':('linear', 'rbf'), 'C':[1, 10]}
>>> svc = svm.SVC()
>>> clf = GridSearchCV(svc, parameters)
>>> clf.fit(iris.data, iris.target)
GridSearchCV(estimator=SVC(),
             param_grid={'C': [1, 10], 'kernel': ('linear', 'rbf')})
```

Randomized Grid Search: Dado un clasificador, definimos “rangos” donde se mueven los atributos y se van eligiendo de manera random para explorar el espacio

```
>>> logistic = LogisticRegression(solver='saga', tol=1e-2, max_iter=200,
...                               random_state=0)
>>> distributions = dict(C=uniform(loc=0, scale=4),
...                      penalty=['l2', 'l1'])
>>> clf = RandomizedSearchCV(logistic, distributions, random_state=0)
```

Recursos Sklearn: Grid Search

Más opciones:

- Multiple clasificadores: Sklearn no lo ofrece pero podemos implementarlo combinando lo de arriba o usando librerías que lo hagan: https://gist.github.com/DimaK415/428bbeb0e79551f780bb990e7c26f813

- Podemos entrenar un modelo que se vaya moviendo por el espacio de parámetros para explorarlo inteligentemente (esto suena al típico backpropagation de una NN)

Recursos Sklearn: Preprocesamiento

Modelo que nos ayuda a hacer diferentes tipos de preprocesamiento:
https://scikit-learn.org/stable/modules/preprocessing.html

- Escalado: MinMaxScaler, MaxAbsScaler, StanderScale
- Mapeos a distribuciones: uniforme, gausiana,
- Normalizaciones: l1, l2
- Encodeo de categoricas: Ordinal encoder, One Hot Encoder
- Discretizaciones
- Binarización
- Imputation of missing values
- PolynomialFeatures Transformation

Y sigue la lista, si tienen que hacer un pre-procesamiento, sospechen si no está listado...

Recursos Sklearn: Reducción de dimensionalidad

Muy util!
https://scikit-learn.org/stable/modules/classes.html#module-sklearn.decomposition

- Factor Analysis (FA)
- FastICA: A fast algorithm for Independent Component Analysis.
- PCA y SparsePCA: Principal component analysis
- LDA: Latent Dirichlet Allocation
- NMF: Non-Negative Matrix Factorization
- SVD:TruncatedSVD

OJO con usar información para reducir dimensionalidad de los test de fold!

Recursos Sklearn: Feature extraction

No es el fuerte de Sklearn, pero por ahi sirve
https://scikit-learn.org/stable/modules/feature_extraction.html

- Datos de diccionarios a features
- Extracción sobre texto
  - tokenizing
  - counting
  - Normalizing
  - CountVectorizer
  - Tf–idf
- Extracción sobre imágenes
  - Dos o tres cosas, no mucho mas
  - Las NN ya funcionan demasiado bien
  - Sino quieren usar NN, miren opencv para procesamiento de imagen (old school)

Recursos Sklearn: Pipeline

Sklearn ofrece esta herramienta para encapsular una sucesión de transformaciones/modelos
https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC

pipe = Pipeline([('scaler', StandardScaler()), ('svc', SVC())])
pipe.fit(X_train,y_train)
```

También se le pueden agregar parámetros

```
pipe = Pipeline(steps=[('pca', pca), ('logistic', logistic)])

param_grid = {
    'pca__n_components': [5, 15, 30, 45, 64],
    'logistic__C': np.logspace(-4, 4, 4),
}
search = GridSearchCV(pipe, param_grid, n_jobs=-1)
```

No solo en GridSearch: `pipe.set_params` para agregar parámetros al modelo, más.

Recursos Sklearn: Multi clase - Multi label

Clasificación Multi-Clase:

- Clasificación no binaria, la predicción puede ir a más de 2 clases (pero solo a una)
- Ej: Un triage medico, el paciente puede ser *verde, amarillo y rojo*
- No todos los algoritmos son fácilmente llevados a multiclase
- No todas las métricas/scores tienen sentido en multiclase

Clasificación Multi-Label:

- Clasificación de una instancia con más de una clase
- Ej: Tengo que clasificar un texto, podría ser *novela* y *ciencia ficción*, o *novela* y *terror*

Estrategias:

- One-Vs-The-Rest: Un clasificador binario por clase, la clase real vs el conjunto de todas las demas (#clases# = #modelos)
- One-vs-One: Un clasificador binario por pares de clases (ojo: #modelos es del orden de #clases^2)

Recursos Sklearn: ¿Dónde está el taller?

https://colab.research.google.com/drive/18igZpCFVW341pSv8jPOJrTBFXeT1Yabk