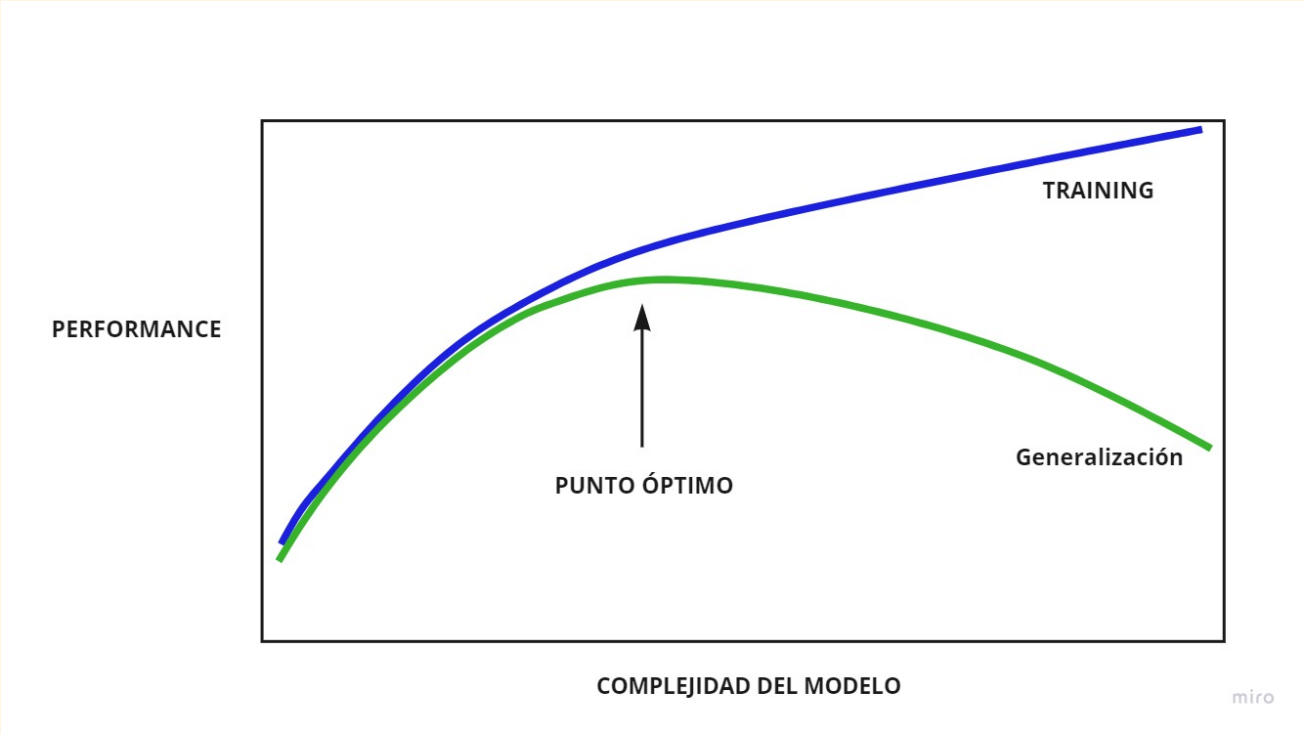
Regularización

Alfonso Tobar Arancibia
Data Scientist
26-10-2020


github.com/Rcubes/clases-ml/Slides

Complejidad de los modelos



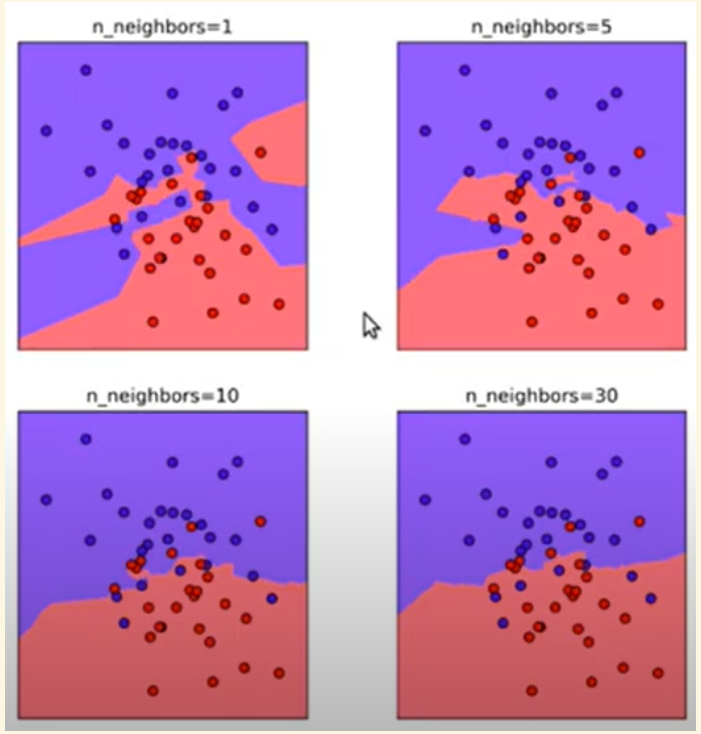
La complejidad de un modelo se controlará

- Alterando la data
  - Más Observaciones
  - Más variables
- Alterando el modelo
  - Hiperparámetros

Qué es un Hiperparámetro?

Corresponde a un valor qué varía la complejidad de un modelo y que no **puede ser aprendido por el modelo por sí mismo**. Debe buscarse por el modelador mediante GridSearch y Cross Validation.

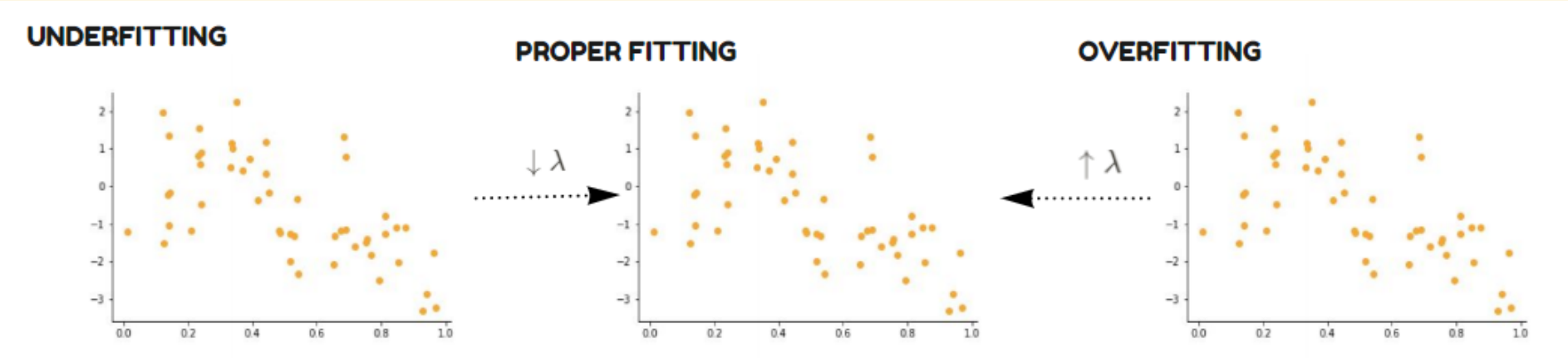
- **n_neighbors**: Número de Vecinos



Se llama **Regularización** el proceso de evitar que el modelo se complejice de modo tal que pueda generalizar mejor.

Todo Hiperparámetro tiene una componente de Ajuste y de regularización.

Regularización L1 y L2



L2

$$\beta_{Ridge} = argmin_{\beta} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

L1

$$\beta_{Lasso} = argmin_{\beta} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Implementación en Scikit Learn

```
from sklearn.linear_model import Lasso, LassoCV, Ridge, RidgeCV
```

Lasso

```
lasso = Lasso(alpha = 1, random_state = 123)
lasso.fit(X,y)
y_pred = lasso.predict(X,y)
```

Hiperparámetros

alpha: Equivalente a lambda. Parámetro regularizador, más grande más regularización. Default: 1.

Implementación en Scikit Learn

```
lassocv = LassoCV(eps = 1e-3, n_alphas = 100, random_state = 123, n_jobs = -1)
lassocv.fit(X,y)
y_pred = lassocv.predict(X,y)
```

Hiperparámetros

- **n_alphas**: Número de alphas distintos a probar. Default: 100.
- **eps**: Ratio alpha_min / alpha_max. Default: 1e-3

Atributos

- **.alpha_** devuelve el valor del alpha óptimo.
- **.alphas_** Grilla de alphas usados.

Ver docs para más detalles.

Implementación en Scikit Learn

Ridge

```
ridge = Ridge(alpha = 1, random_state = 123)
ridge.fit(X,y)
y_pred = ridge.predict(X,y)
```

Hiperparámetros

alpha: Equivalente a lambda. Parámetro regularizador, más grande más regularización. Default: 1.

Implementación en Scikit Learn

```
ridgecv = RidgeCV(alphas = (0.1, 1.0, 10.0))
ridgecv.fit(X,y)
y_pred = ridgecv.predict(X,y)
```

Hiperparámetros

alphas: Arreglo de valores de alpha a probar. Default: (0.1, 1.0, 10.0).

> NOTA: RidgeCV no tiene random_state.

Atributos

- **.alpha_** devuelve el valor del alpha óptimo.

Ver docs para más detalles.

Implementación en Scikit Learn

ElasticNet

```
from sklearn.linear_model import ElasticNet, ElasticNetCV
elastic = ElasticNet(alpha = 1, l1_ratio = 0.5, random_state = 123)
elastic.fit(X,y)
y_pred = elastic.predict(X,y)
```

Hiperparámetros

alpha: Equivalente a lambda. Parámetro regularizador, más grande más regularización. Default: 1. **l1_ratio**: Corresponde a la proporción de l1 y l2 a utilizar. Más cercano a 1 es más Lasso, más cercano a 0 es Ridge.

$$a * L1 + b * L2$$

donde $alpha = a + b$ y $l1\_ratio = a/(a + b)$

Implementación en Scikit Learn

```
elasticCV = ElasticNetCV(l1_ratio=0.5, eps=0.001, n_alphas=100, random_state = 123)
elasticcv.fit(X,y)
y_pred = elasticcv.predict(X,y)
```

Hiperparámetros

l1_ratio: Corresponde a la proporción de l1 y l2 a utilizar. Más cercano a 1 es más Lasso, más cercano a 0 es Ridge.

- **n_alphas**: Número de alphas distintos a probar. Default: 100.
- **eps**: Ratio alpha_min / alpha_max. Default: 1e-3

Ver docs para más detalles.