{desafío}
latam_

Unidad 2:
Algoritmo Maximización de Esperanzas_

{desafío}
latam_

Alcances de la lectura asignada

- Conocer el algoritmo de Maximización de Esperanzas como una técnica estimación paramétrica
- Identificar cuáles son los casos donde el algoritmo puede solucionar problemas de datos incompletos
- Aplicar el algoritmo para la extracción de clases latentes.
- Conocer otras aplicaciones el Algoritmo

Activación de Conceptos

- En la unidad anterior aprendimos sobre Máquinas de Soporte Vectorial.
- ¡Pongamos a prueba nuestros conocimientos!
- Las preguntas van en subslides.

¿Cuál es el principio de clasificación para SVM?

- Implementamos un clasificador discriminante que se base en la combinación lineal de parámetros.
- Implementamos un clasificador generativo que encuentre las clases en función a la distribución conjunta.
- Buscamos maximizar el margen donde es posible separar las clases

¿Qué significa "Soporte Vectorial" en Máquinas de Soporte Vectorial?

- Observaciones que permiten establecer los límites positivos y negativos del margen.
- Observaciones que se posicionan dentro de la clase opuesta.
- Observaciones inferidas a partir del clasificador lineal

¿Cuál es una de las principales desventajas de SVM?

- SVM es un modelo de caja negra.
- SVM es sensible a la escala de los atributos.
- SVM es ineficiente en espacios $\mathbb{N}$-dimensionales.

¿Cuál es el objetivo del kernelizar?

- Estimar la probabilidad de clase de una observación.
- Reducir la dimensionalidad de la matriz de atributos.
- Reexpresar la matriz de atributos en un nuevo espacio que permita la separación.

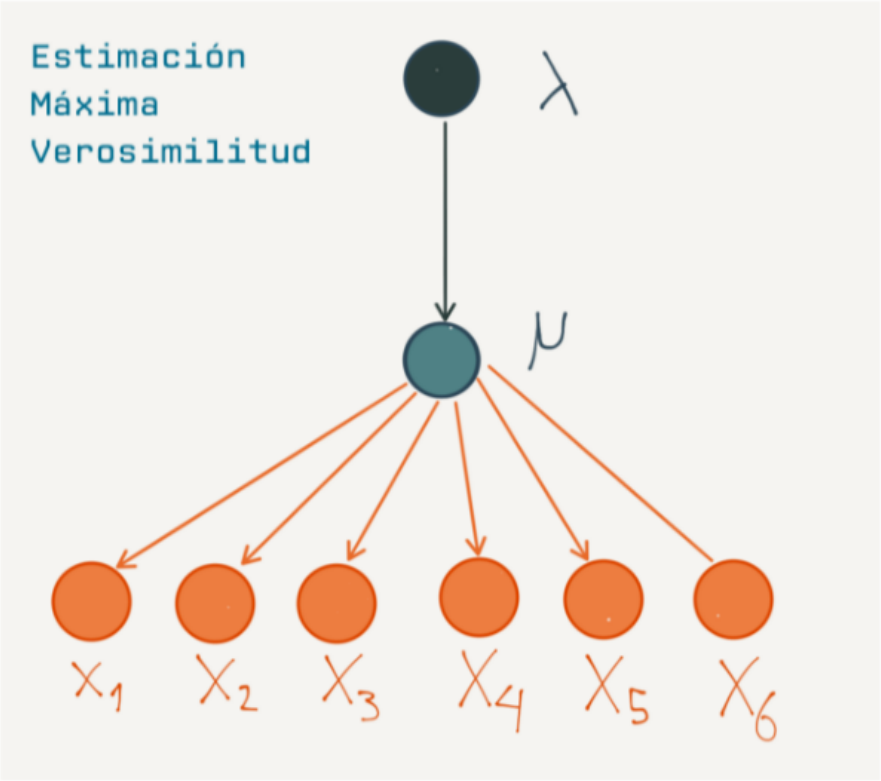
¿Cuál es el efecto de $\mathbf{C}$ y $\boldsymbol{\gamma}$?

- C Penaliza el costo y γ determina el ancho del kernel cuando es lineal.
- C determina el ancho del kernel y γ penaliza la función de costo.
- C Penaliza el costo y γ determina el ancho del kernel cuando es radial basis function.

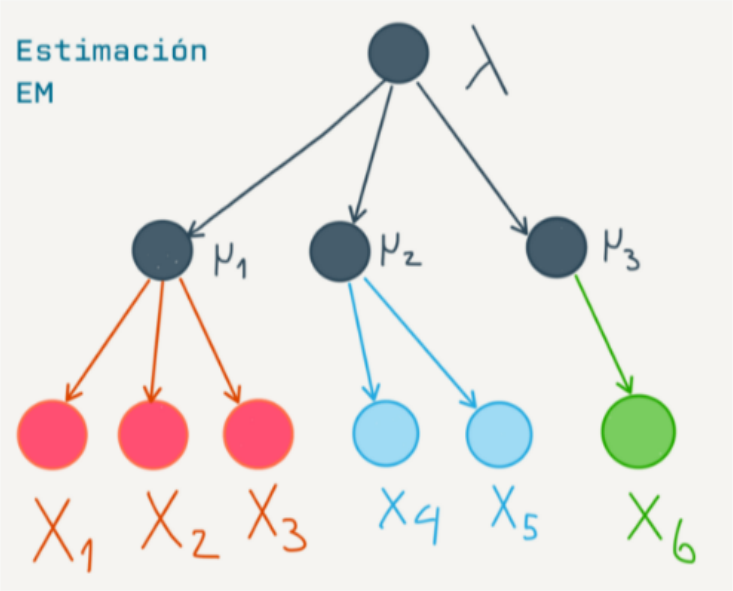
Motivación

{desafío}
latam_

Estimación por Máxima Verosimilitud



Estimación por Maximización de Esperanzas



El proceso algorítmico

Algorithm 8.2 *The EM Algorithm.*

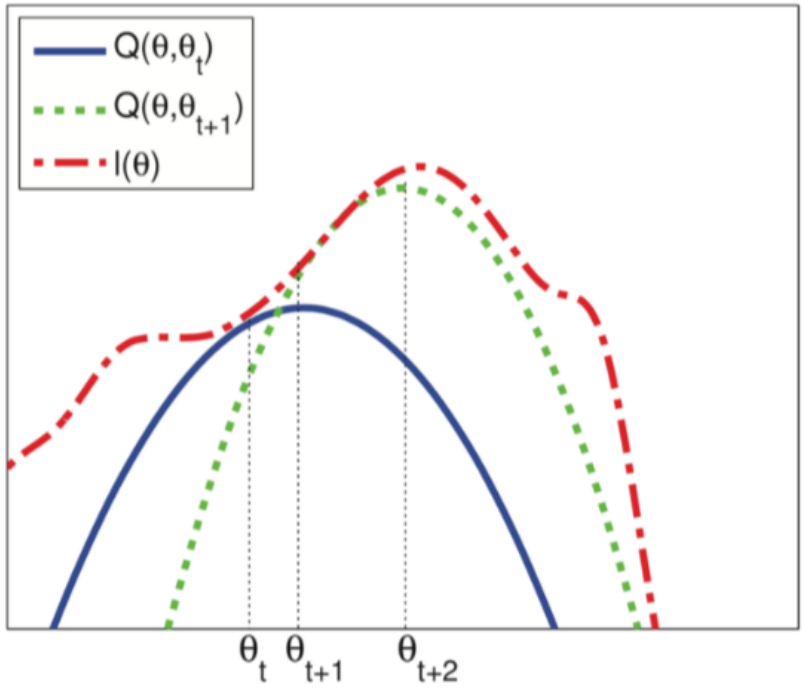
1. Start with initial guesses for the parameters $\hat{\theta}^{(0)}$.
2. *Expectation Step*: at the jth step, compute

$$Q(\theta', \hat{\theta}^{(j)}) = \mathrm{E}(\ell_0(\theta'; \mathbf{T})|\mathbf{Z}, \hat{\theta}^{(j)}) \tag{8.43}$$

   as a function of the dummy argument θ'.
3. *Maximization Step*: determine the new estimate $\hat{\theta}^{(j+1)}$ as the maximizer of $Q(\theta', \hat{\theta}^{(j)})$ over θ'.
4. Iterate steps 2 and 3 until convergence.

Hastie et al. 2009. The Elements of Statistical Learning.

EM en una imagen



Murphy, K. 2012. Machine Learning: a probabilistic perspective

Modelo de Mezcla de Gaussianas

{desafío}
latam_

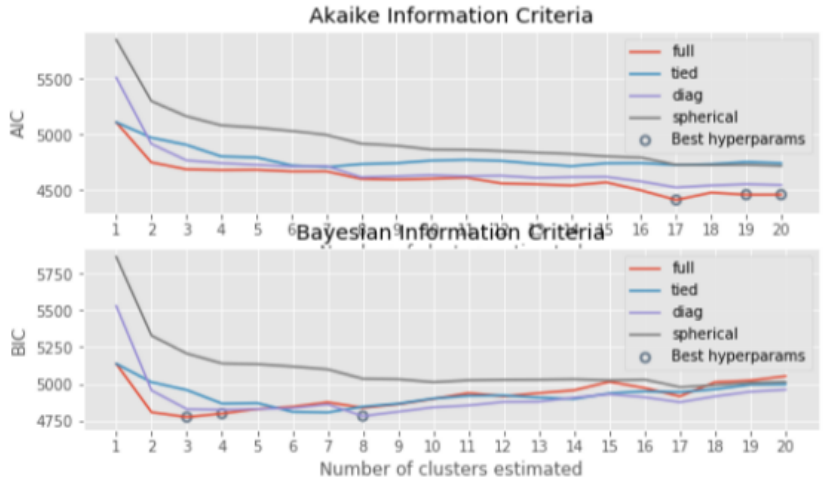
Objetivo

- Generar una aproximación informada sobre el número óptimo de grupos en una matriz de atributos, dado que **no tenemos información sobre los grupos**.
- Implementación con `sklearn`.

In [6]:
```
from sklearn.mixture import GaussianMixture
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import classification_report
```

Identificando óptimo de clusters

In [7]:
```
afx.gmm_information_criteria_report(df.loc[:,'glucose':'sspg'], k=np.arange(1, 21))
```



{desafío}
latam_

Criterios de Información

- Origen: Desviación del modelo respecto a los datos "verdaderos"
- (A/B)IC: Logverosimilitud penalizada

Criterio de Información de Akaike

$$\text{AIC} = -2(\mathcal{L}_{\text{modelo}} - \#\text{params})$$

Criterio de Información Bayesiano

$$\text{BIC} = \ln(\#\text{params})(\mathcal{L}_{\text{modelo}} - \#\text{params})$$

Implementación en `sklearn`

In [9]:
```
# Implementaremos un modelo GMM con tres componentes
# acorde a lo informado por el criterio BIC
gmm = GaussianMixture(n_components=3, covariance_type='full', random_state=323).fit(X_tra
in)
predict_classes_gmm = gmm.predict(X_test)
```

In [10]:
```
print(classification_report(y_test,
                            predict_classes_gmm))
```

```
              precision    recall  f1-score   support

           0       1.00      0.15      0.27        13
           1       0.67      1.00      0.80        24
           2       1.00      0.91      0.95        11

   micro avg       0.75      0.75      0.75        48
   macro avg       0.89      0.69      0.67        48
weighted avg       0.83      0.75      0.69        48
```

Densidades inferidas

In [11]:
```
print("Densidades de cada clase inferida: ", gmm.weights_.round(3))
print("Suma de densidades: ", np.sum(gmm.weights_).round(3))
```

```
Densidades de cada clase inferida:  [0.073 0.721 0.206]
Suma de densidades:  1.0
```

Caracterización de densidades

In [12]:
```
print("Ponderador para clase {}: {}".format(0,gmm.weights_[0]))
for j, k in enumerate(x_mat.columns):
    # dentro de cada clase extraemos la media de los atributos
    print("Media {}: {}".format(k, gmm.means_[0][j]))
```

```
Ponderador para clase 0: 0.07294484820645235
Media glucose: -0.3185897814663789
Media insulin: -0.32399625945329596
Media sspg: 2.9350650096526896
```

Mezcla probabilística

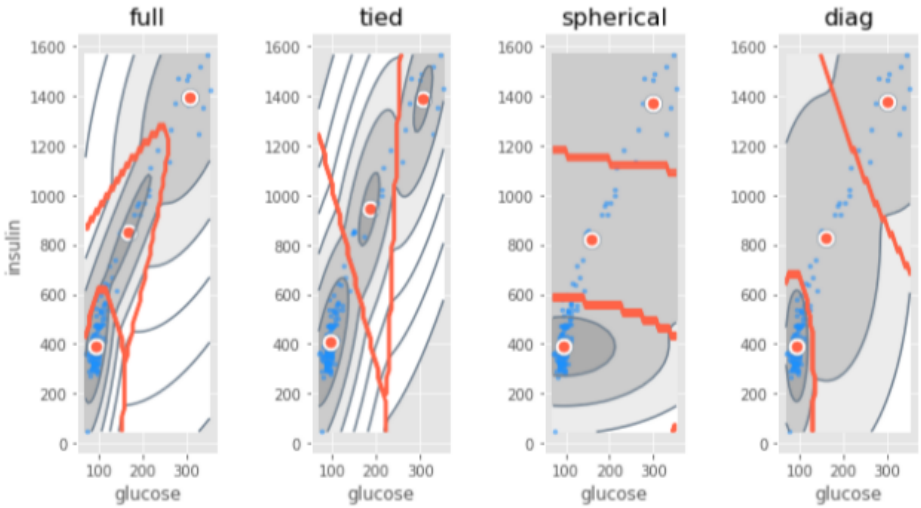
In [13]:
```
pd.DataFrame(gmm.predict_proba(X).round(3)[110:115])
```

Out[13]:

	0	1	2
0	0.0	0.555	0.445
1	0.0	0.878	0.122
2	0.0	0.000	1.000
3	0.0	0.000	1.000
4	0.0	0.996	0.003

Efecto de la covarianza

In [16]:
```
covariances()
```



Covarianza `full`

In [17]:
```
gmm_full = GaussianMixture(n_components=3,covariance_type='full',random_state=323).fit(X_
train);
print(classification_report(y_test, gmm_full.predict(X_test)))
```

```
              precision    recall  f1-score   support

           0       1.00      0.15      0.27        13
           1       0.67      1.00      0.80        24
           2       1.00      0.91      0.95        11

   micro avg       0.75      0.75      0.75        48
   macro avg       0.89      0.69      0.67        48
weighted avg       0.83      0.75      0.69        48
```

Covarianza `diag`

In [18]:
```
gmm_diag = GaussianMixture(n_components=3,covariance_type='diag',random_state=323).fit(X_
train);
print(classification_report(y_test, gmm_diag.predict(X_test)))
```

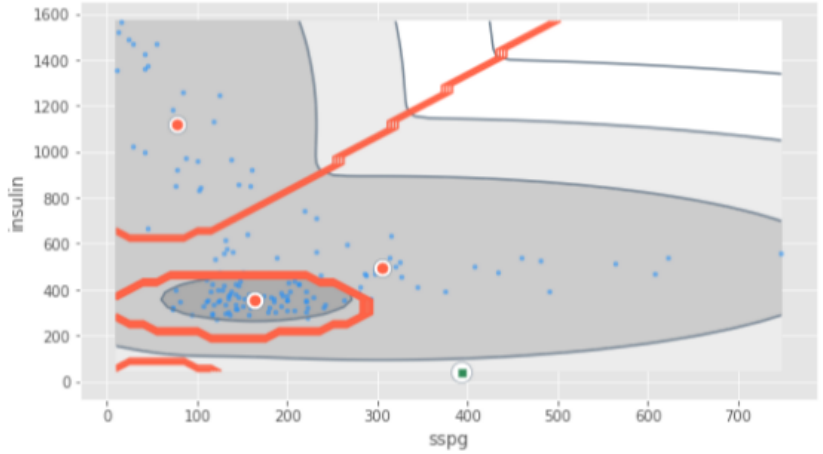
```
              precision    recall  f1-score   support

           0       0.73      0.62      0.67        13
           1       0.81      0.92      0.86        24
           2       1.00      0.91      0.95        11

   micro avg       0.83      0.83      0.83        48
   macro avg       0.85      0.81      0.83        48
weighted avg       0.83      0.83      0.83        48
```

Detección de anomalías

In [20]:
```
gmm_anomalies()
```



{desafío}
latam_

Otras aplicaciones

Imputación de Datos Perdidos (Little y Rubin, 1997)

- Por defecto la comunidad de Machine Learning no trabaja muy bien con datos perdidos. Sus estrategias incluyen:
  - Imputación por moda y media.
  - Interpolación de datos.

- **Problema:** puede replicar sesgos subyacentes de los datos $\rightsquigarrow$ Pr(Missing) $\not\perp \mathbf{X}_i\beta$

- Estrategia EM $\rightsquigarrow$ Completar de forma iterativa la logverosimilitud de los datos no observados.
- Iterar el proceso de generación K veces.
- Estimar un modelo en cada proceso de generación de datos y resumir los puntos estimados.

- EM es más conveniente dado que preserva la relación entre los datos.

Modelación concomitante de clases latentes (Rabe-Hesketh y Skrondal, 2004)

- Híbrido entre un modelo de mezclas gausianas y un modelo generalizado multinomial.

- Permite identificar la probabilidad de pertenencia a una de las densidades inferidas **en función de un conjunto de covariables**.
- ¿Por qué no etiquetar las observaciones con $\underset{\theta \in \Theta}{\text{argmax}}$?

Ignoramos la existencia de una mezcla de probabilidades generadas para cada una observación

Inferencia de tópicos (Blei, Ng y Jordan, 2003)

- Asignación Latente Dirichlet (*Latent Dirichlet Allocation*) ⇝ Modelo generativo probabilístico.
- Implementa una variante del EM (Variational EM) para inferir tópicos de un conjunto de documentos **no clasificados**.

- Tenemos $\mathcal{K}$ tópicos inferidos en un vocabulario **fijo**.
- Cada documento $d \in \mathcal{D}$ es una colección de palabras.
- Cada palabras $p \in \mathcal{P}$ tiene una probabilidad de ocurrencia $p \sim \text{Dirichlet}(\beta_{i,\mathcal{K}=k})$