Linear Classification

Shyue Ping Ong

Aiiso Yufeng Li Family Department of Chemical and Nano Engineering
University of California, San Diego
http://materialsvirtuallab.org

Overview

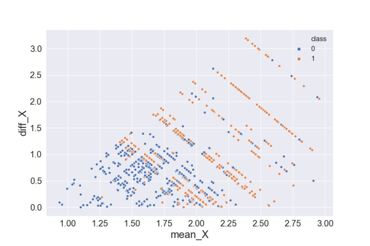
Preliminaries

Preliminaries

- Linear methods can also be used for classification, i.e., decision boundaries are linear.
- These methods are surprisingly effective across a large spectrum of datasets, even compared to more complex ML models.

Metal vs Insulator Dataset

- To demonstrate the use of these methods, we will first discuss the “toy” dataset.
- 2000+ binary (A_xB_y) compounds with experimental band gaps.
- Class 0: metals; Class 1: insulators.
- Using pymatgen, we can generate some simple features. Here, we will create simply features based on the mean and absolute difference in electronegativity between A and B (why?).



Creating the features and classes

```
from __future__ import annotations

import numpy as np
import pandas as pd

from pymatgen.core import Composition

binaries = pd.read_csv("binary_band_gap.csv")
# We create a column holding the Composition object.
# Note the use of list comprehension in Python.
binaries["composition"] = [Composition(c) for c in binaries["Formula"]]
electronegs = [[el.X for el in c] for c in binaries["composition"]]
# Create the features mean and difference between electronegativities
binaries["mean_X"] = [np.mean(e) for e in electronegs]
binaries["diff_X"] = [max(e) - min(e) for e in electronegs]
# Label metals (band gap of 0. 1e-5 is used as numerical tolerance) as class 0
# Insulators are labelled as class 1.
binaries["class"] = [0 if eg < 1e-5 else 1 for eg in binaries["Eg (eV)"]]
```

Basic concepts

Basic concepts

- If there are K classes, we have a $N \times K$ indicator response matrix. Each row is a vector $Y = (Y_1, Y_2, ..., Y_K)$ where $Y_k = 1$ if the instance belongs to the kth class and all other Ys are 0.

$$\mathrm{Y} = \begin{pmatrix} 0 & 0 & ... & 1 \\ 1 & 0 & ... & 0 \\ ... & & & \\ 0 & 1 & ... & 0 \end{pmatrix}$$

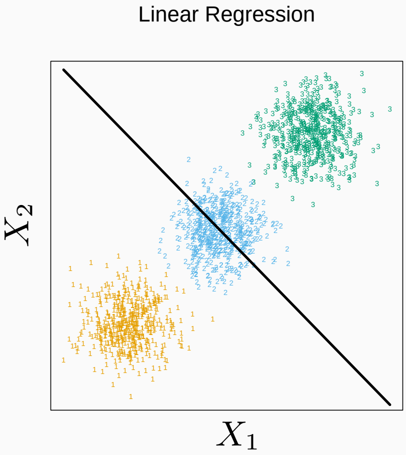
- For the kth response variable, the fitted $\hat{f}_k(x) = \hat{\beta}_{k0} + \hat{\beta}_k^T x$.
- Decision boundary between k and l class is given by $\hat{f}_k(x) = \hat{f}_l(x)$.
- Input is divided into regions.
- Similar to linear regression, we can augment the input space with polynomial (e.g., X_1^2, X_2^S, X_1X_2) and other basis functions, leading to boundaries that are non-linear.

Linear regression of indicator matrix

- Treat each column of Y as a target. Least squares solution:

$$\hat{Y} = X(X^TX)^{-1}X^TY$$

- For each new observation x, we compute $\hat{f}_k(x) = (1, x^T)(X^TX)^{-1}X^TY$.
- Find the largest component, and that will result in the classification k, $G(x) = \text{argmax}_{k \in G} \hat{f}_k(x)$.
- Major issue: some categories may be masked for $K \geq 3$.



Discriminant Analysis

Discriminant Analysis

- From Bayes rule, we have:

$$P(G = k|X = x) = \frac{f_k(x)\pi_k}{\sum_{l=1}^{K} f_l(x)\pi_l}$$

- where $f_k(x)$ are the class conditional probability densities ($P(X = x|G = k)$) and π_k are the prior probabilities of being in class k.
- Most common approach - assume Gaussian class densities.

$$f_k(x) = \frac{1}{(2\pi)^{p/2}|\Sigma_k|^{1/2}} \exp -\frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1}(x - \mu_k)$$

Linear Discriminant Analysis

- Assume all classes have a common covariance matrix, i.e., $\Sigma_k = \Sigma$.
- To compare two classes k and l, we can compare the log ratios.

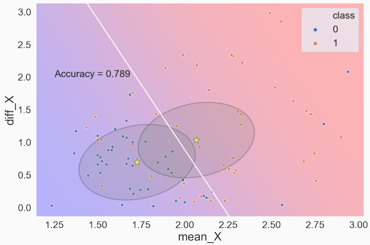
$$\begin{aligned}\log \frac{P(G=k|X=x)}{P(G=l|X=x)} &= \log \frac{f_k(x)}{f_l(x)} + \log \frac{\pi_k}{\pi_l} \\ &= \log \frac{\pi_k}{\pi_l} - \frac{1}{2}(\mu_k + \mu_l)^T \Sigma^{-1} (\mu_k - \mu_l) \\ &\quad + x^T \Sigma^{-1} (\mu_k - \mu_l)\end{aligned}$$

- At the decision boundary, $P(G=k|X=x) = P(G=l|X=x)$, which leads to a linear equation in x.
- Equivalently, we have

$$G(x) = \underset{k}{\operatorname{argmax}} \left\{ \log \pi_k - \frac{1}{2}\mu_k^T \Sigma^{-1} \mu_k + x^T \Sigma^{-1} \mu_k \right\}$$

Linear Discriminant Analysis, contd.

- In general, we do not know the prior distributions and covariance matrix. These are estimated from the data.
  - $\hat{\pi}_k = N_k/N$
  - $\hat{\mu}_k = \sum_{g_i=k} x_i/N$
  - $\hat{\Sigma} = \sum_{k=1}^{K} \sum_{g_i=k} (x_i - \hat{\mu}_k)^T (x_i - \hat{\mu}_k)/(N - K)$
- Avoids masking problem of linear regression classification.
- For the example data,

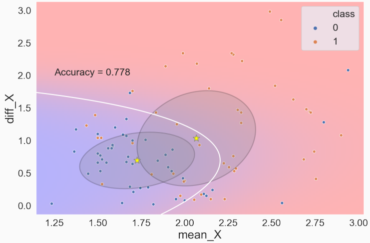


Quadratic Discriminant Analysis

- Covariances are not assumed equal.

$$G(x) = \underset{k}{\operatorname{argmax}} \left\{ \log \pi_k - \frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) - \frac{1}{2} \log |\Sigma_k| \right\}$$

- No cancellation of terms and decision boundaries are quadratic.
- Covariances must be estimated for each category.
- For the same metal-insulator example,



Discriminant analysis in scikit-learn

```
from __future__ import annotations

from sklearn.discriminant_analysis import LinearDiscriminantAnalysis, QuadraticDiscrimina

lda = LinearDiscriminantAnalysis(solver="svd", store_covariance=True)
X = binaries[["mean_X", "diff_X"]]
y = binaries["class"]
model = lda.fit(X, y)
y_pred = model.predict(X)

qda = QuadraticDiscriminantAnalysis(store_covariance=True)
y_pred = qda.fit(X, y).predict(X)
```

Logistic regression

- Model posterior probabilities with linear function.

$$
\begin{aligned}
\log \frac{P(G=1|X=x)}{P(G=K|X=x)} &= \beta_{10} + \beta_1^T x \\
\log \frac{P(G=2|X=x)}{P(G=K|X=x)} &= \beta_{20} + \beta_2^T x \\
&\dots \\
\log \frac{P(G=K-1|X=x)}{P(G=K|X=x)} &= \beta_{(k-1)0} + \beta_{k-1}^T x
\end{aligned}
$$

- Results in the following posterior probabilities:

$$
\begin{aligned}
P(G=1|X=x) &= \frac{\exp\left(\beta_{10} + \beta_1^T x\right)}{1 + \sum_{l=1}^{K-1} \exp\left(\beta_{l0} + \beta_l^T x\right)} \\
P(G=K|X=x) &= \frac{1}{1 + \sum_{l=1}^{K-1} \exp\left(\beta_{l0} + \beta_l^T x\right)}
\end{aligned}
$$

Solving for the Logistic Regression Coefficients

- Typically fitted using *maximum likelihood*.

$$l(\beta) = \sum_{i=1}^{N} \log P(G = k|X = xi; \beta)$$

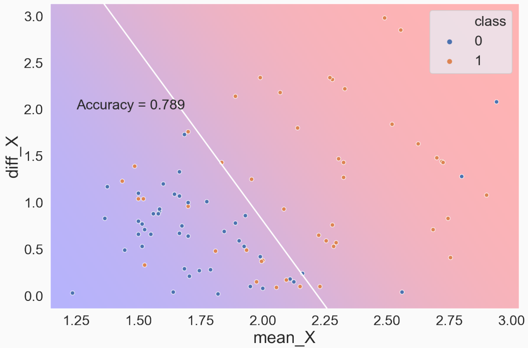
- Differentiation and setting $\frac{\partial l}{\partial \beta} = 0$ leads to equations that are non-linear in β.
- These equations are solved using some optimization algorithm (e.g., Newton-Raphson, BFGS, etc.).

Logistic regression on metal/insulator dataset

```python
from __future__ import annotations

from sklearn.linear_model import LogisticRegression

clf = LogisticRegression(penalty="none", random_state=0)
model = clf.fit(X, y)
y_pred = model.predict(X)
```



Loss functions for binary classification

- Consider a simple binary classification with two levels (-1, 1). The decision boundary is at 0.
- Using the square error does not make sense, since we only care about whether it is > 0 or < 0.
- *Margin* $yf(x)$ is positive when prediction and actual value is in the same class, and negative if they are in opposite classes.
- Need a loss that penalizes negative values much more than positive values for margins, i.e., monotone decreasing function.
- Exponential loss: $L(y, f(x)) = e^{-yf(x)}$
- Binomial/multinomial loss (can be used for K-classes):

$$L(y, p(x)) = -\sum_{k=1}^{K} I(y = G_k) f_k(x) + \log\left(\sum_{l=1}^{K} e^{f_l(x)}\right)$$

Loss functions for binary classification

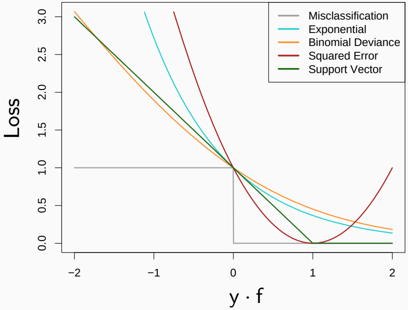


Figure 1: Loss functions for binary classification. Response: $y = \pm 1$. X-axis is the margin $y \cdot f$. Misclassification : $I(\mathrm{sign}(f) \neq y)$; exponential: e^{-yf}; binomial deviance: $\log(1 + e^{-2yf})$; squared error: $(y - f)^2$; and support vector: $(1 - yf)_+$. Source: [?]

The End