UIC
BUSINESS

Business Data Mining

IDS 472 (Spring 2024)

Instructor: Wenxin Zhou

Linear Models

UIC BUSINESS

- Linear models are an important class of ML models
- Their popularity in various applications is attributed to
  1) simple structure and training efficiency
  2) interpretability
  3) matching their complexity with that of problems that have been historically available
  4) a "sacrificing" approach to decision-making
- This chapter covers the most widely used linear models
  - linear discriminant analysis
  - logistic regression
  - multiple linear regression
  - shrinkage methods: ridge, lasso, elastic-net

Optimal Classification

UIC BUSINESS

- We start our discussion from linear classifiers
- Suppose we can find c functions $g_i(\mathbf{x}), i = 0, \dots, c-1$, where c is the number of classes. Define a classifier as

$$\psi(\mathbf{x}) = \underset{i}{\operatorname{argmax}}\ g_i(\mathbf{x})$$

- $\psi(\mathbf{x})$ assigns label i to $\mathbf{x} \in \mathbb{R}^p$ if $g_i(\mathbf{x}) > g_j(\mathbf{x})$ for $i \neq j$
- The functions $g_i(\mathbf{x})$ are known as *discriminant functions*
- The classifier *partitions* the feature space $\mathbb{R}^p$ into c *decision regions*, $\mathcal{R}_1, \dots, \mathcal{R}_c$, such as $\mathcal{R}_1 \cup \mathcal{R}_2 \cup \cdots \cup \mathcal{R}_c = \mathbb{R}^p$
- A decision boundary is the boundary between two decision regions $\mathcal{R}_i$ and $\mathcal{R}_j$: $\{\, \mathbf{x} \mid g_i(\mathbf{x}) = g_j(\mathbf{x}), i \neq j \}$
- Discriminant functions for standard kNN classifier are

$$g_i(\mathbf{x}) = \sum_{j=1}^{k} \frac{1}{k} I\{Y_{(j)}(\mathbf{x}) = i\}, i = 0, \dots, c-1$$

Bayes Classifier

UIC BUSINESS

- Recall the misclassification error rate $\varepsilon = P(\psi(\mathbf{X}) \neq Y)$
- Bayes classifier is the classifier that has the lowest error
- For binary classification, set $g_i(\mathbf{x}) = P(Y = i|\mathbf{x}), i = 0, 1$, known as the *posterior probability*. The classifier is

$$\psi(\mathbf{x}) = \begin{cases} 1 & \text{if } P(Y = 1|\mathbf{x}) > P(Y = 0|\mathbf{x}) \\ 0 & \text{otherwise} \end{cases} = \begin{cases} 1 & \text{if } P(Y = 1|\mathbf{x}) > \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

- Using Bayes rule,

$$P(Y = i|\mathbf{x}) = \frac{p(\mathbf{x}|Y = i)P(Y = i)}{p(\mathbf{x})}$$

- Equivalently,

$$\psi(\mathbf{x}) = \begin{cases} 1 & \text{if } p(\mathbf{x}|Y = 1)P(Y = 1) > p(\mathbf{x}|Y = 0)P(Y = 0) \\ 0 & \text{otherwise} \end{cases}$$

$$= \begin{cases} 1 & \text{if } \log\left(\frac{p(\mathbf{x}|Y=1)}{p(\mathbf{x}|Y=0)}\right) > \log\left(\frac{P(Y=0)}{P(Y=1)}\right) \\ 0 & \text{otherwise} \end{cases}$$

Linear Models for Classification

UIC BUSINESS

- Linear models for classification are models for which the *decision boundaries are linear functions of the feature vector* $\mathbf{x}$
  - linear discriminant analysis
  - logistic regression
  - perceptron
  - linear support vector machine
- In this chapter, we discuss the first two models
  - understand the working principle of each model
  - understand some important parameters used in scikit-learn implementation of each model

Linear Discriminant Analysis

UIC BUSINESS

- To develop LDA classifier, the first assumption is that class-conditional probability densities are Gaussian:

$$p(\mathbf{x}|Y=i) = \frac{1}{(2\pi)^{p/2}|\boldsymbol{\Sigma}_i|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)}, \; i = 0, 1, \ldots, c-1.$$

  - c: number of classes
  - $\boldsymbol{\mu}_i$: class-specific mean vector
  - $\boldsymbol{\Sigma}_i$: class-specific covariance matrix
- The second assumption: $\boldsymbol{\Sigma}_i = \boldsymbol{\Sigma}, \forall i$
- For binary classification, some algebraic simplifications yield

$$\psi(\mathbf{x}) = \begin{cases} 1 & \text{if } \left(\mathbf{x} - \frac{\boldsymbol{\mu}_0+\boldsymbol{\mu}_1}{2}\right)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) + \log\left(\frac{P(Y=1)}{1-P(Y=1)}\right) > 0 \\ 0 & \text{otherwise} \end{cases}$$

- Equivalently,

$$\psi(\mathbf{x}) = \begin{cases} 1 & \text{if } g(\mathbf{x}) > 0 \\ 0 & \text{otherwise} \end{cases} \qquad g(\mathbf{x}) = \left(\mathbf{x} - \frac{\boldsymbol{\mu}_0 + \boldsymbol{\mu}_1}{2}\right)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) + \log\left(\frac{P(Y=1)}{1-P(Y=1)}\right)$$

Linear Discriminant Analysis

UIC BUSINESS

- Decision boundary of LDA classifier is $\{\mathbf{x} \mid \mathbf{a}^T\mathbf{x} + b = 0\}$, where $\mathbf{a} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)$,

$$b = -\left(\frac{\boldsymbol{\mu}_0 + \boldsymbol{\mu}_1}{2}\right)^T \boldsymbol{\Sigma}^{-1}\left(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0\right) + \log\left(\frac{P(Y=1)}{1-P(Y=1)}\right)$$

- Given a training data $\mathbf{S}_{tr} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_\mathbf{n}, y_n)\}$, the sample LDA classifier is

$$\psi(\mathbf{x}) = \begin{cases} 1 & \text{if } \left(\mathbf{x} - \frac{\hat{\boldsymbol{\mu}}_0 + \hat{\boldsymbol{\mu}}_1}{2}\right)^T \hat{\boldsymbol{\Sigma}}^{-1}\left(\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0\right) + \log\left(\frac{P(Y=1)}{1-P(Y=1)}\right) > 0 \\ 0 & \text{otherwise} \end{cases}$$

where

$$\hat{\boldsymbol{\mu}}_i = \frac{1}{n_i}\sum_{j=1}^{n_i} \mathbf{x}_j \qquad \hat{\boldsymbol{\Sigma}} = \frac{1}{n_0 + n_1 - 2}\sum_{i=0}^{1}\sum_{j=1}^{n}(\mathbf{x}_j - \hat{\boldsymbol{\mu}}_i)(\mathbf{x}_j - \hat{\boldsymbol{\mu}}_i)^T I_{\{y_j = i\}}$$

- The LDA classifier is the Bayes classifier when class-conditional densities are Gaussian with a common covariance matrix

A Toy Example

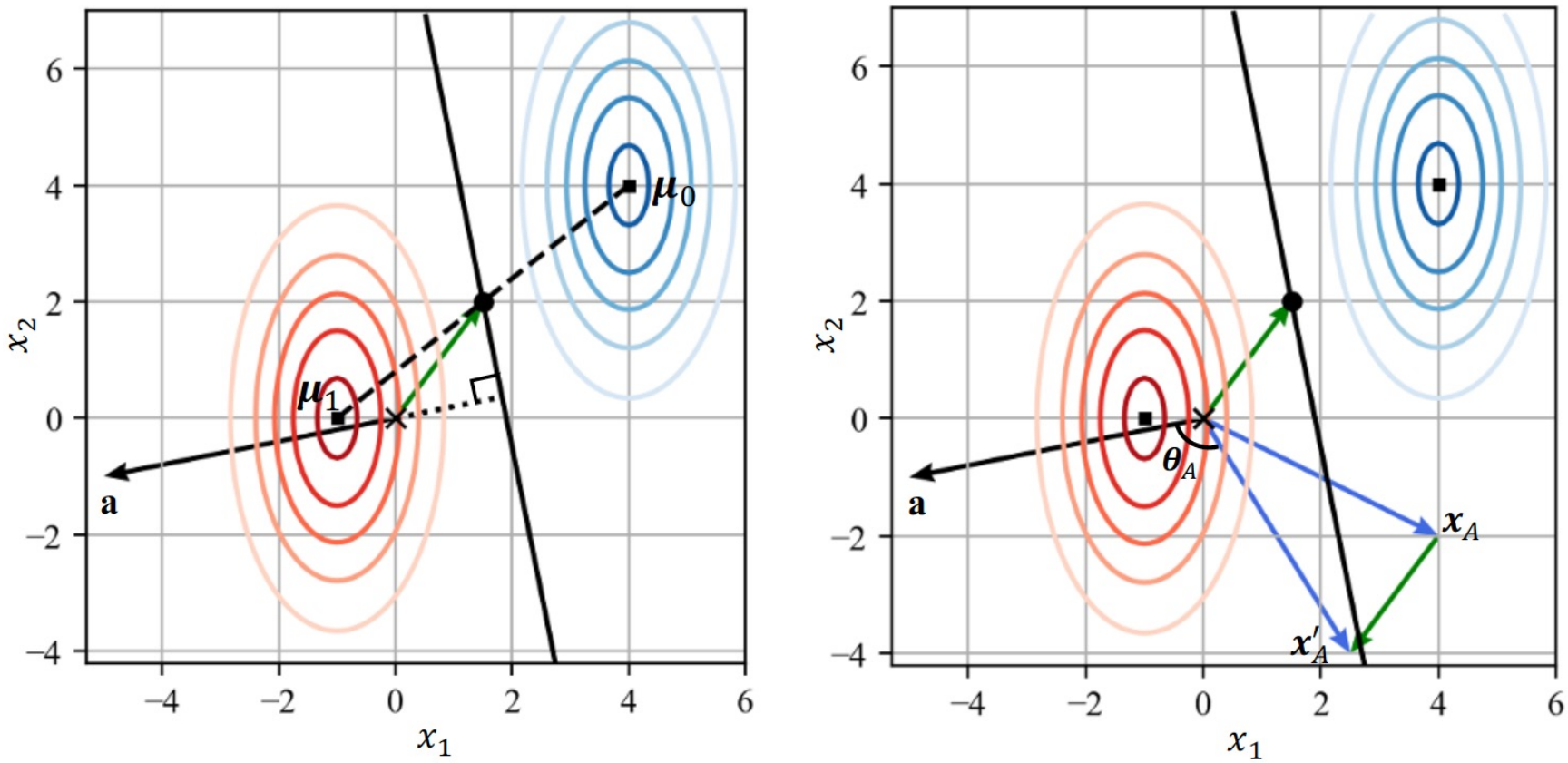
UIC BUSINESS

- Two mean vectors $\boldsymbol{\mu}_0 = [4, 4]^T, \boldsymbol{\mu}_1 = [-1, 0]^T$, common covariance matrix

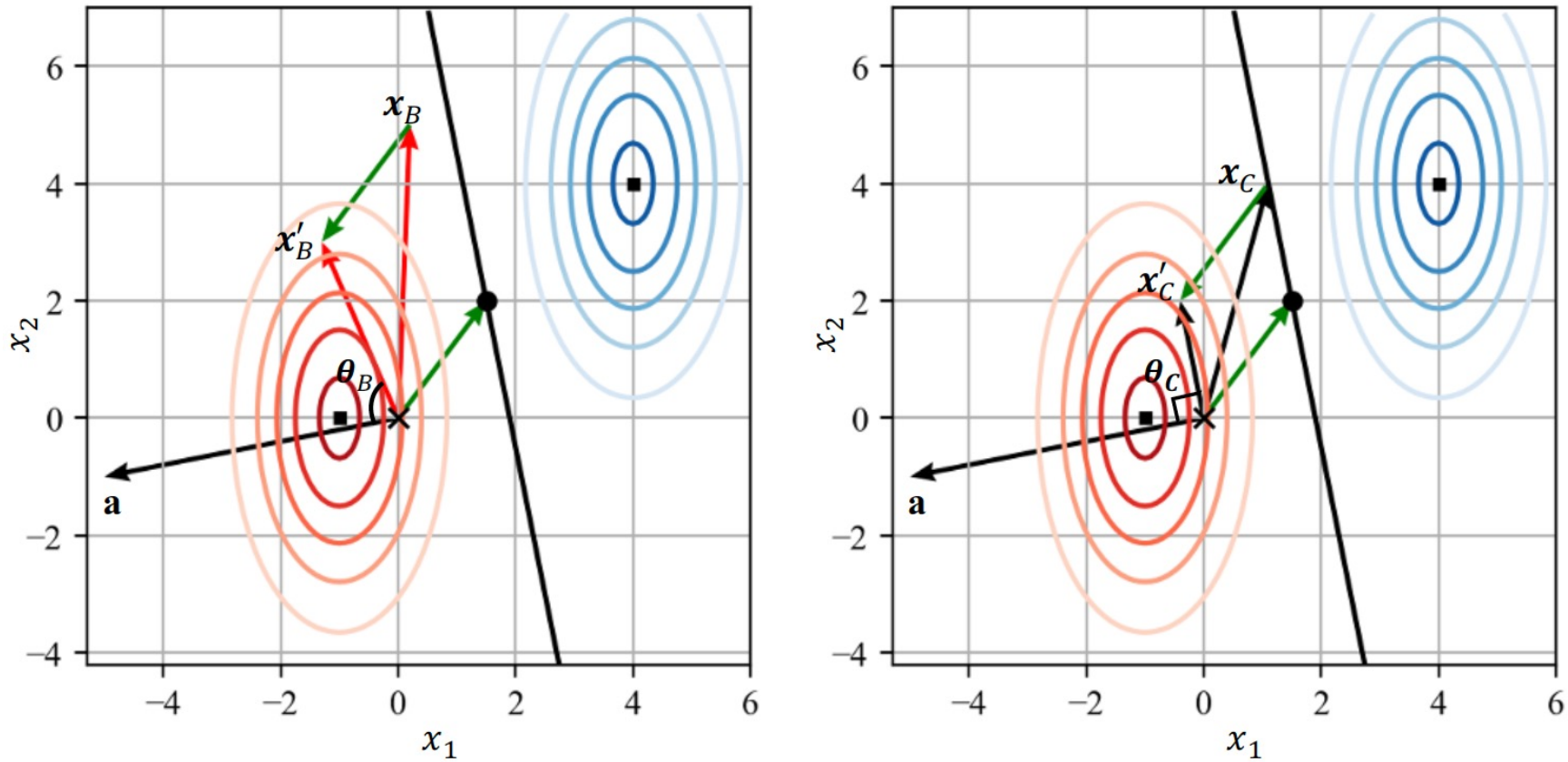
$$\boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix}$$

  and $P(Y = 1) = 0.5$
- Decision boundary is $\{\mathbf{x} = [x_1, x_2]^T \mid -5x_1 - x_2 + 9.5 = 0\}$
- $\mathbf{a} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) = [-5, -1]^T$ is perpendicular to the linear decision boundary
- Classify three observations: $\mathbf{x}_A = [4, -2]^T, \mathbf{x}_B = [0.2, 5]^T$ and $\mathbf{x}_C = [1.1, 4]^T$
- Replacing $\mathbf{x}_A$, $\mathbf{x}_B$ and $\mathbf{x}_C$ in the discriminant function gives -8.5, 3.5, 0, respectively
- $\mathbf{x}_A$ is classified as 0, $\mathbf{x}_B$ is classified as 1
- $\mathbf{x}_C$ lies on the boundary and could be randomly assigned to one of the two classes

A Toy Example



A Toy Example



Linear Discriminant Analysis

UIC BUSINESS

- If the prior probabilities $P(Y = i), i = 0, 1$ are unknown, they can be estimated by the sampling ratio $\frac{n_i}{n}, i = 0, 1$
- Extension to *multiclass classification*: $n = \Sigma_{i=0}^{c-1} n_i$ is total sample size, the discriminant functions are

$$g_i(\mathbf{x}) = \hat{\boldsymbol{\mu}}_i^T \hat{\boldsymbol{\Sigma}}^{-1} \mathbf{x} - \frac{1}{2} \hat{\boldsymbol{\mu}}_i^T \hat{\boldsymbol{\Sigma}}^{-1} \hat{\boldsymbol{\mu}}_i + \log P(Y = i)$$

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{n-c} \sum_{i=0}^{c-1} \sum_{j=1}^{n} (\mathbf{x}_j - \hat{\boldsymbol{\mu}}_i)(\mathbf{x}_j - \hat{\boldsymbol{\mu}}_i)^T I_{\{y_j = i\}}$$

- In scikit-learn, the LDA classifier is implemented by the `LinearDiscriminantAnalysis` class from the `sklearn.discriminant_analysis` module

Linear Discriminant Analysis

UIC BUSINESS

```python
arrays = np.load('data/iris_train_scaled.npz')
X_train = arrays['X']
y_train = arrays['y']
arrays = np.load('data/iris_test_scaled.npz')
X_test = arrays['X']
y_test = arrays['y']

print('X shape = {}'.format(X_train.shape) + '\ny shape = {}'.format(y_train.shape))
```

✓ 0.0s Python

```
X shape = (120, 4)
y shape = (120,)
```

```python
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

lda = LinearDiscriminantAnalysis(solver='lsqr')
lda.fit(X_train, y_train)
```

✓ 0.6s Python

```
▾ LinearDiscriminantAnalysis
LinearDiscriminantAnalysis(solver='lsqr')
```

Logistic Regression

UIC BUSINESS

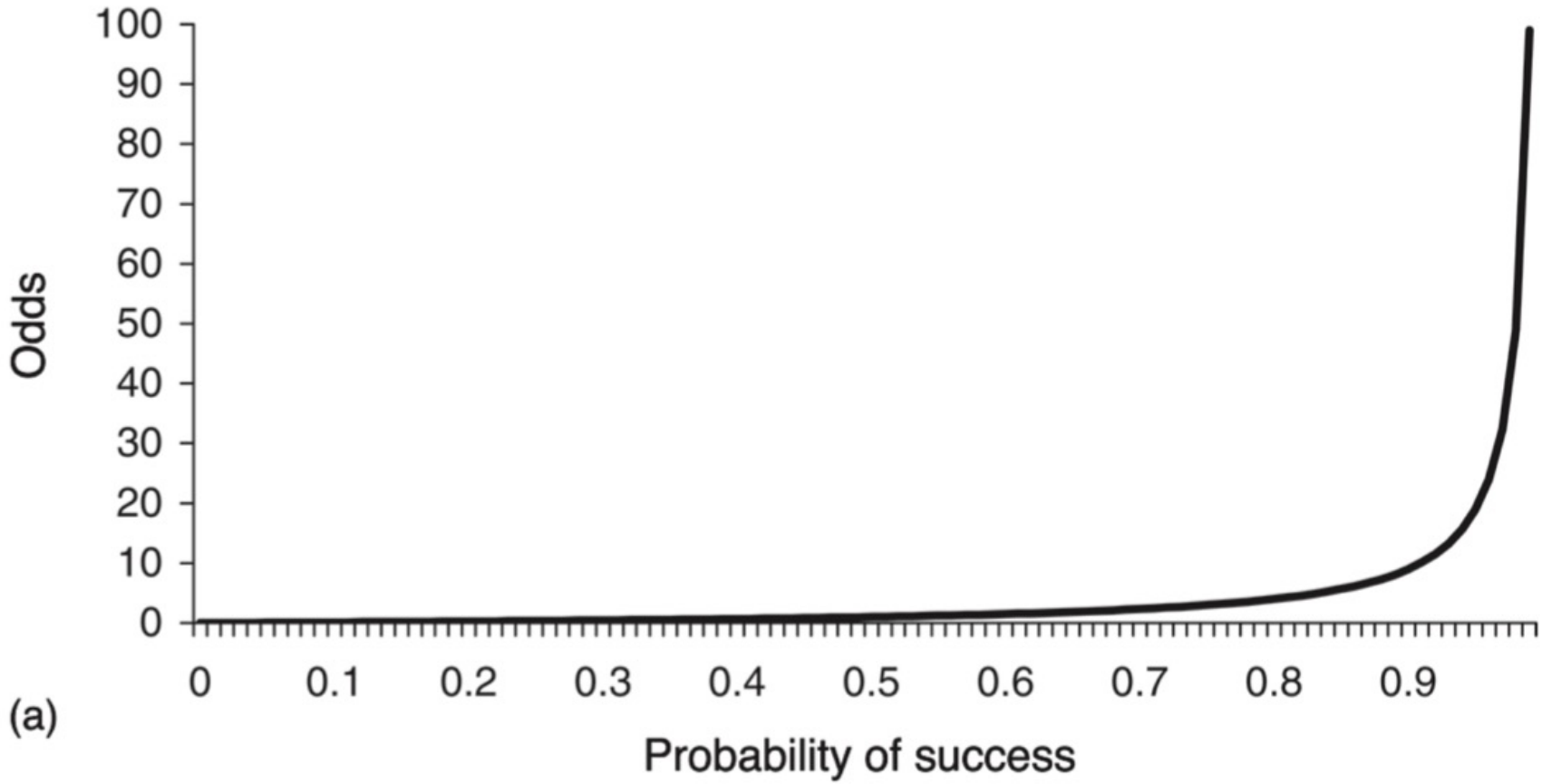
- The linearity of decision boundaries in LDA is a direct consequence of the Gaussian assumption
- To achieve linear decision boundary, alternatively assume some monotonic transformation of *posterior* $P(Y = i \mid \mathbf{x})$ is linear
- A well-known transformation is the *logit* function: for $p \in (0, 1)$,

$$\text{logit}(p) = \log\underbrace{\left(\frac{p}{1-p}\right)}_{\text{odds}}$$

- For binary classification, if the prob of having an observation from one class is $p = 0.8$, then odds of having an observation from that class is $\frac{0.8}{0.2} = 4$

Odds function

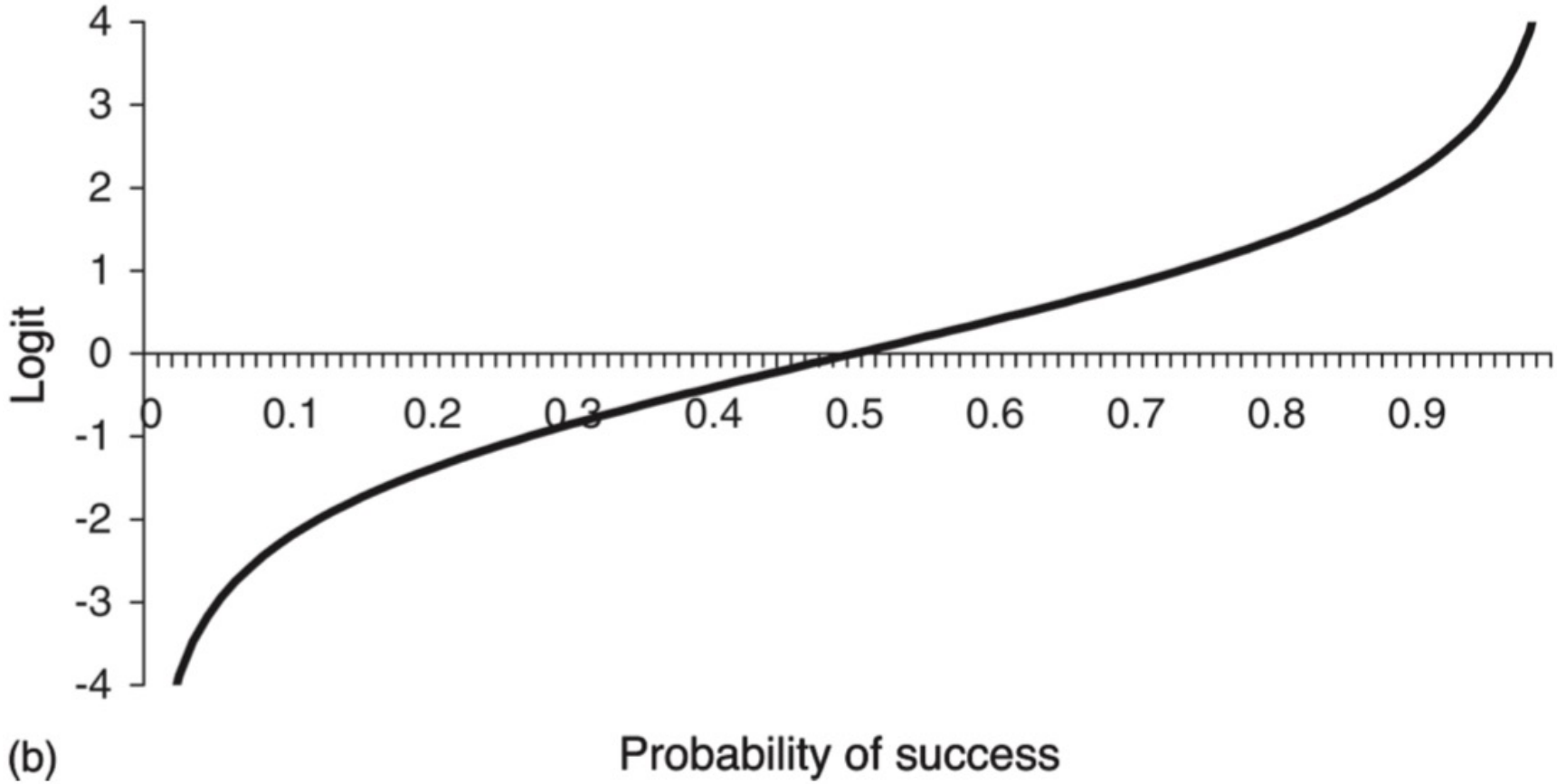
UIC BUSINESS



(a)

Logit function

UIC BUSINESS



Logistic Regression

UIC BUSINESS

- *Logistic (sigmoid)* function: $\sigma(x) = \dfrac{1}{1+e^{-x}}$
- Using logit(p) as the argument of $\sigma(x)$ leads to

$$\sigma\left(\text{logit}(p)\right) = \frac{1}{1+e^{-\text{logit}(p)}} = \frac{1}{1+e^{-\log\left(\frac{p}{1-p}\right)}} = p,$$

that is, logistic (sigmoid) function is the inverse of logit function

- In logistic regression, assume the logit of posteriors are linear:

$$\text{logit}\big(P(Y=1|\mathbf{x})\big) = \log\left(\frac{P(Y=1|\mathbf{x})}{1-P(Y=1|\mathbf{x})}\right) = \mathbf{a}^T\mathbf{x} + b$$

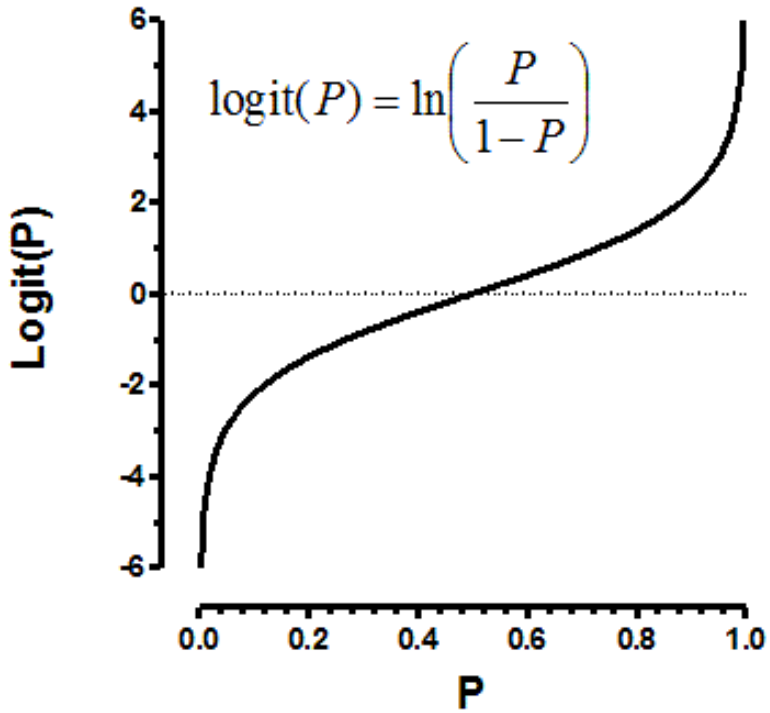
or equivalently,

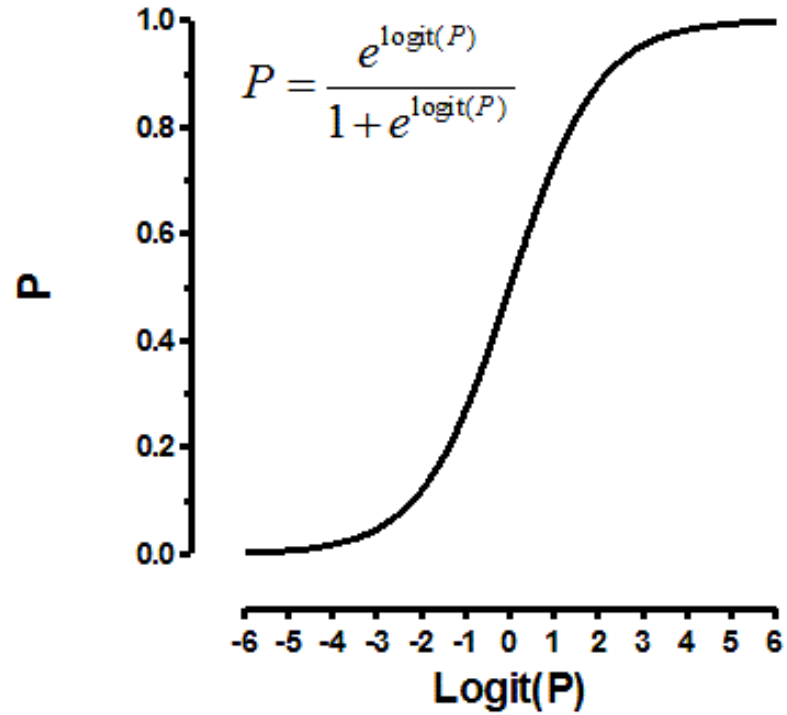
$$P(Y=0|\mathbf{x}) = \frac{e^{-(\mathbf{a}^T\mathbf{x}+b)}}{1+e^{-(\mathbf{a}^T\mathbf{x}+b)}} = \frac{1}{1+e^{(\mathbf{a}^T\mathbf{x}+b)}}$$

$$P(Y=1|\mathbf{x}) = \frac{1}{1+e^{-(\mathbf{a}^T\mathbf{x}+b)}} = \frac{e^{(\mathbf{a}^T\mathbf{x}+b)}}{1+e^{(\mathbf{a}^T\mathbf{x}+b)}}$$

Logit vs Sigmoid

UIC BUSINESS





Logistic Regression

UIC BUSINESS

- A compact expression is

$$P(Y=i|\mathbf{x}) = \frac{1}{1+e^{(-1)^i(\mathbf{a}^T\mathbf{x}+b)}} = P(Y=1|\mathbf{x})^i\left(1-P(Y=1|\mathbf{x})\right)^{(1-i)},\ i=0,1$$

- The classifier based on logistic regression is

$$\psi(\mathbf{x}) = \begin{cases} 1 & \text{if} \quad \frac{1}{1+e^{-(\mathbf{a}^T\mathbf{x}+b)}} - \frac{e^{-(\mathbf{a}^T\mathbf{x}+b)}}{1+e^{-(\mathbf{a}^T\mathbf{x}+b)}} > 0 \\ 0 & \text{otherwise} \end{cases}$$

default cutoff value

$P(Y=1|\mathbf{x})$

$$= \begin{cases} 1 & \text{if} \quad \frac{1}{1+e^{-(\mathbf{a}^T\mathbf{x}+b)}} > \frac{1}{2} \\ 0 & \text{otherwise} \end{cases} = \begin{cases} 1 & \text{if} \quad \mathbf{a}^T\mathbf{x}+b > 0 \\ 0 & \text{otherwise} \end{cases}$$

- To estimate the unknown parameters $\mathbf{a}$ and b, a common approach is to *maximize the log likelihood of observing the labels given observations* as a function of $\mathbf{a}$ and b

Logistic Regression

UIC BUSINESS

- Let $\boldsymbol{\beta} = [b, \mathbf{a}^T]^T$ be the $(p+1)$-dimensional (column) vector of unknown parameters
- For *independent* data,

$$\log\big(P(Y_1 = y_1, \ldots, Y_n = y_n | \mathbf{x}_1, \ldots, \mathbf{x}_n; \boldsymbol{\beta})\big)$$

$$= \log\Big(\prod_{j=1}^{n} P(Y_j = 1|\mathbf{x}_j;\boldsymbol{\beta})^{y_j}\big(1 - P(Y_j = 1|\mathbf{x}_j;\boldsymbol{\beta})\big)^{(1-y_j)}\Big)$$

$$\log\left(\frac{1}{1+e^{-b-\mathbf{a}^T\mathbf{x}}}\right) \qquad \log\left(\frac{e^{-b-\mathbf{a}^T\mathbf{x}}}{1+e^{-b-\mathbf{a}^T\mathbf{x}}}\right)$$

$$= \sum_{j=1}^{n} y_j \log\big(P(Y_j = 1|\mathbf{x}_j;\boldsymbol{\beta})\big) + (1-y_j)\log\big(1 - P(Y_j = 1|\mathbf{x}_j;\boldsymbol{\beta})\big)$$

- Here $P(Y_1 = y_1, \ldots, Y_n = y_n \mid \mathbf{x}_1, \ldots, \mathbf{x}_n; \boldsymbol{\beta})$ shows that this probability is a function of unknown parameters
- This approach is an example of *maximum likelihood estimation*: estimate the model parameter by maximizing the (log) likelihood of observing labels for the given observations

Logistic Regression

UIC BUSINESS

- Multiplying the log likelihood function by -1 results in a *loss function* (error function):

$$e(\boldsymbol{\beta}) = -\frac{1}{n}\sum_{j=1}^{n} y_j \log\big(P(Y_j = 1|\mathbf{x}_j;\boldsymbol{\beta})\big) + (1-y_j)\log\big(1 - P(Y_j = 1|\mathbf{x}_j;\boldsymbol{\beta})\big)$$

- $e(\boldsymbol{\beta})$ is a convex function of $\boldsymbol{\beta}$ with a unique minimum
  - setting the derivative (wrt $\boldsymbol{\beta}$) to zero leads to the minimum
  - this equation is non-linear, no closed-form solution exists
  - we reply on iterative optimization methods to estimate $\boldsymbol{\beta}$
- Plugging-in the estimator $\widehat{\boldsymbol{\beta}} = \left[\hat{b}, \hat{\mathbf{a}}^T\right]^T$ results in the classifier

$$\psi(\mathbf{x}) = \begin{cases} 1 & \text{if} \quad \hat{\mathbf{a}}^T\mathbf{x} + \hat{b} > 0 \\ 0 & \text{otherwise} \end{cases}$$

Logistic Regression

UIC BUSINESS

- The previous loss function is used when labels are 0 and 1
- When the labels are $y_j \in \{-1, 1\}$, we have

$$P(Y_j = y_j | \mathbf{x}_j) = \frac{1}{1 + e^{-y_j(\mathbf{a}^T \mathbf{x}_j + b)}}, \quad y_j = -1, 1$$

- In this case, the loss function becomes

$$e(\boldsymbol{\beta}) = \frac{1}{n} \sum_{j=1}^{n} \log\left(1 + e^{-y_j(\mathbf{a}^T \mathbf{x}_j + b)}\right)$$

- $p + 1$ parameters need to be estimated in logistic regression, whereas for LDA, we need $2p$ (for means) + $p(p+1)/2$ (for the pooled sample covariance matrix) + 1 (for prior prob)

Multiclass Classification

UIC BUSINESS

- In multiclass case, assume

$$P(Y = i|\mathbf{x}) = \frac{e^{(\mathbf{a}_i^T \mathbf{x} + b_i)}}{1 + \sum_{i=1}^{c-2} e^{(\mathbf{a}_i^T \mathbf{x} + b_i)}}, \quad i = 0, 1, \ldots, c-2$$

$$P(Y = c-1|\mathbf{x}) = \frac{1 \longrightarrow e^0}{1 + \sum_{i=1}^{c-2} e^{(\mathbf{a}_i^T \mathbf{x} + b_i)}}$$

- The discriminant functions are $g_i(\mathbf{x}) = \mathbf{a}_i^T \mathbf{x} + b_i, i = 0, \ldots, c-2$, and $g_{c-1}(\mathbf{x}) = 0$
- The negative log likelihood is the well-known *cross-entropy loss*:

$$-\sum_{j=1}^{n} \sum_{i=0}^{c-1} I\big(Y_j = i\big) \log\big(P(Y_j = i)\big)$$

- This is known as *multinomial logistic regression*

Regularization

UIC BUSINESS

- In practice, it is helpful to consider a *penalized* form of the loss function: penalize large coefficients to end up with smaller coefficients
- This approach is known as *regularization* or *shrinkage*: *regularize* or *shrink* estimated coefficients
- l_2-regularization is to minimize

$$e(\boldsymbol{\beta}) = C \sum_{j=1}^{n} \log\left(1 + e^{-y_j(\mathbf{a}^T \mathbf{x}_j + b)}\right) + \frac{1}{2}||\mathbf{a}||_2^2$$

where $\| \mathbf{a} \|_2 = \sqrt{\mathbf{a}^T \mathbf{a}} = \sqrt{\sum_{k=1}^{p} a_k^2}$ and C is a tuning parameter

- The higher C, the less regularization we expect
- Logistic regression trained using this objective function is known as logistic regression with l_2-*penalty* (or *ridge penalty*)

Regularization

UIC BUSINESS

- Another two useful shrinkage methods are *l_1-regularization* and *elastic-net regularization*
- l_1-regularized logistic regression minimizes

$$e(\boldsymbol{\beta}) = C\sum_{j=1}^{n}\log\left(1+e^{-y_j(\mathbf{a}^T\mathbf{x}_j+b)}\right)+||\mathbf{a}||_1$$

where $\|\mathbf{a}\|_1 = \sum_{k=1}^{p}|a_k|$

- Elastic-net regularization is to minimize

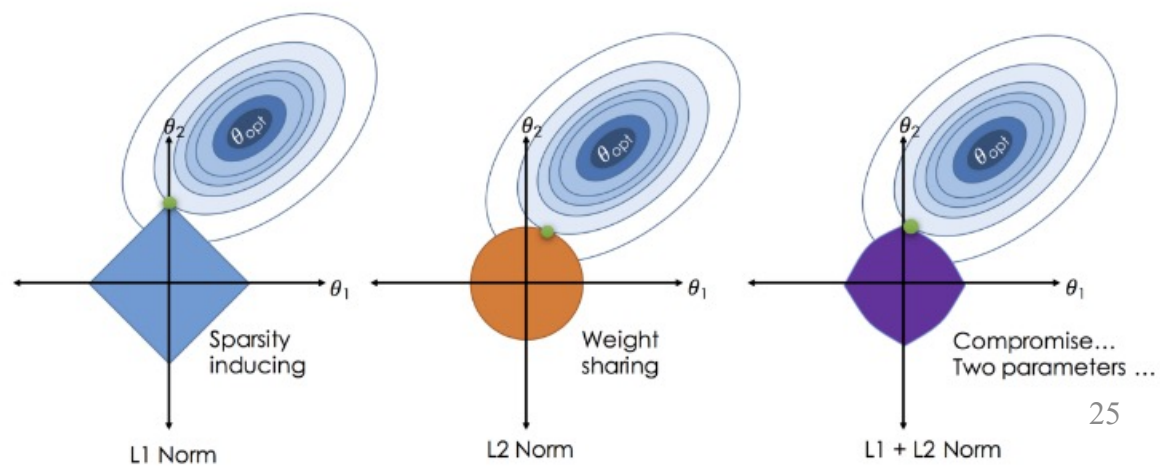
$$e(\boldsymbol{\beta}) = C\sum_{j=1}^{n}\log\left(1+e^{-y_j(\mathbf{a}^T\mathbf{x}_j+b)}\right)+\nu||\mathbf{a}||_1+\frac{1-\nu}{2}||\mathbf{a}||_2^2$$

where $\nu \in [0,1]$ is another tuning parameter

Properties of l_1 Penalty

UIC BUSINESS

- If C is sufficiently small, the solution contains some coefficients (elements of $\mathbf{a}$) that are exactly zero and, therefore, are not selected to be part of the discriminant function
- l_1 penalty possesses a *feature selection* mechanism: choosing "important" features and discarding the rest
- l_2 penalty shrinks the magnitude of coefficients to zero, but does not set them to exact zero
- l_1 penalty is also known as *lasso* – least absolute shrinkage & selection operator



Scikit-learn Implementation

UIC BUSINESS

- Logistic regression classifier is implemented by the `LogisticRegression` class from `sklearn.linear_model`
  - by default, scikit-learn uses l_2 regularization with $C = 1.0$
  - possible options for `penalty` parameter are `'l1'`, `'l2'`, `'elasticnet'`
  - when `penalty='elasticnet'`, one can set $\nu \in [0, 1]$ by setting the value of `l1_ratio`, and change the default value of `solver` from `'lbfgs'` to `'saga'`
  - when `penalty='l1'`, `'liblinear'` solver can be used

- `'lbfgs'`: limited-memory BFGS (Broyden–Fletcher–Goldfarb–Shanno)
- `'sag'`: stochastic average gradient; `'saga'` is a variant of `'sag'` that supports the non-smooth l_1 penalty

Iris Flower Classification

UIC BUSINESS

- For illustration purpose, we train a logistic regression with l_2 regularization for Iris flower classification using two features: sepal width and petal length

```
arrays = np.load('data/iris_train_scaled.npz')
X_train = arrays['X']
y_train = arrays['y']

arrays = np.load('data/iris_test_scaled.npz')
X_test = arrays['X']
y_test = arrays['y']

print('X shape = {}'.format(X_train.shape) + '\ny shape = {}'.format(y_train.shape))
print('X shape = {}'.format(X_test.shape) + '\ny shape = {}'.format(y_test.shape))
```

✓ 0.0s Python

```
X shape = (120, 4)
y shape = (120,)
X shape = (30, 4)
y shape = (30,)
```

```
X_train = X_train[:,[1,2]]
X_test = X_test[:,[1,2]]
X_train.shape
```

✓ 0.0s Python

```
(120, 2)
```

Iris Flower Classification

UIC BUSINESS

```
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn.linear_model import LogisticRegression as LRR

color = ['aquamarine', 'bisque', 'lightgrey']
cmap = ListedColormap(color)
mins = X_train.min(axis=0) - 0.1
maxs = X_train.max(axis=0) + 0.1
x = np.arange(mins[0], maxs[0], 0.01)
y = np.arange(mins[1], maxs[1], 0.01)
X, Y = np.meshgrid(x, y)
coordinates = np.array([X.ravel(), Y.ravel()]).T
fig, axs = plt.subplots(1, 2, figsize=(6, 2), dpi = 150)
fig.tight_layout()

C_val = [0.01, 100]
```

Iris Flower Classification

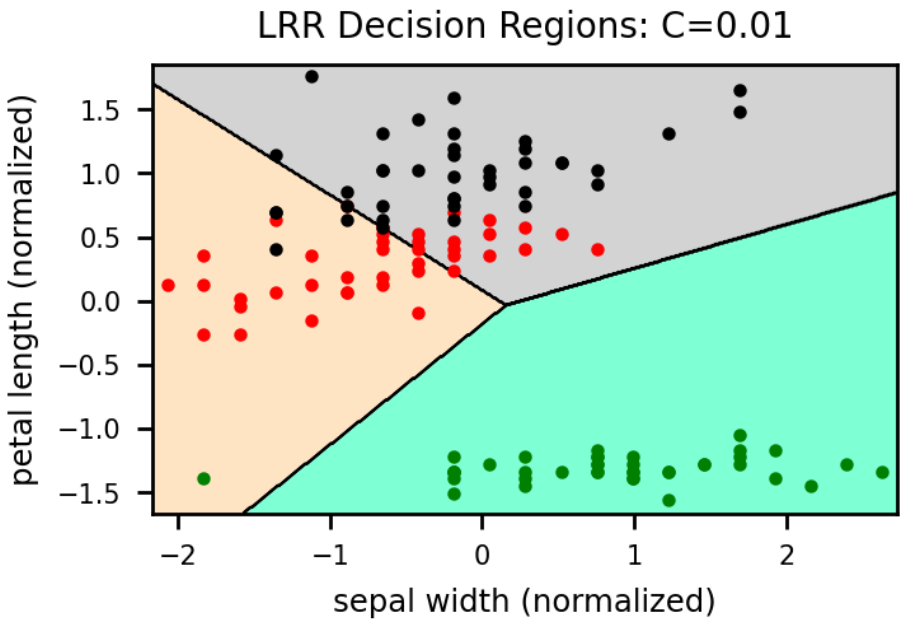
UIC BUSINESS

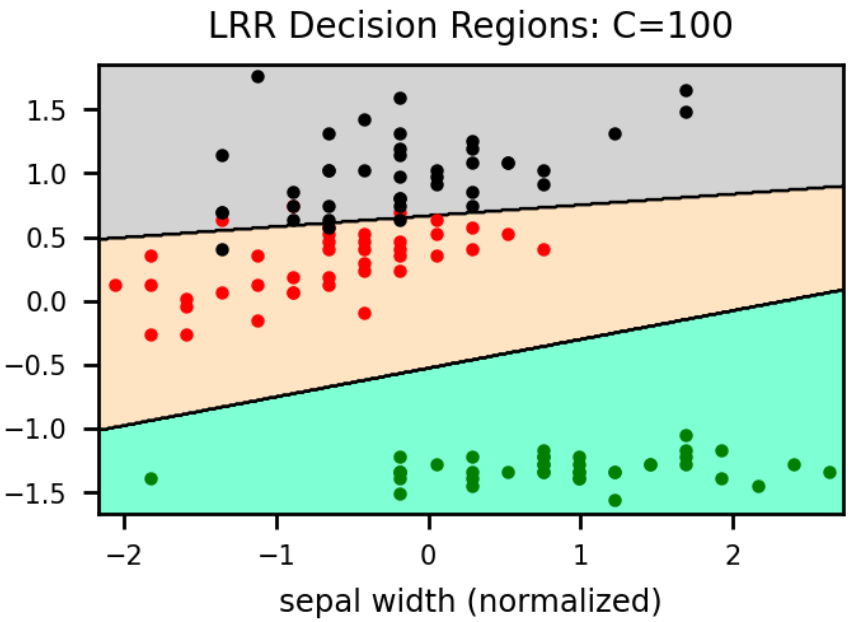
```
for ax, C in zip(axs.ravel(), C_val):
    lrr = LRR(C=C)
    lrr.fit(X_train, y_train)
    Z = lrr.predict(coordinates)
    Z = Z.reshape(X.shape)
    ax.tick_params(axis='both', labelsize=6)
    ax.set_title('LRR Decision Regions: C=' + str(C), fontsize=8)
    ax.pcolormesh(X, Y, Z, cmap = cmap, shading='nearest')
    ax.contour(X ,Y, Z, colors='black', linewidths=0.5)
    ax.plot(X_train[y_train==0, 0], X_train[y_train==0, 1],
            'g.', markersize=4)
    ax.plot(X_train[y_train==1, 0], X_train[y_train==1, 1],
            'r.', markersize=4)
    ax.plot(X_train[y_train==2, 0], X_train[y_train==2, 1],
            'k.', markersize=4)
    if (C==C_val[0]):
        ax.set_ylabel('petal length (normalized)', fontsize=7)
    ax.set_xlabel('sepal width (normalized)', fontsize=7)
    print('The accuracy for C={} on the training data is {:.3f}'.format(C, lrr.score(X_train, y_train)))
    print('The accuracy for C={} on the test data is {:.3f}'.format(C, lrr.score(X_test, y_test)))
```

```
The accuracy for C=0.01 on the training data is 0.817
The accuracy for C=0.01 on the test data is 0.833
The accuracy for C=100 on the training data is 0.950
The accuracy for C=100 on the test data is 0.967
```

Iris Flower Classification

UIC BUSINESS





Quantifying Odds Ratio

UIC BUSINESS

- When LR classifier is used, an important question is the interpretation of its coefficients
- Consider the odds of $Y = 1$ (versus $Y = 0$)

$$r_0 \triangleq \text{odds} = \frac{P(Y=1|\mathbf{x})}{P(Y=0|\mathbf{x})} = e^{\mathbf{a}^T\mathbf{x}+b} = e^{a_1x_1+a_2x_2+\ldots+a_px_p+b}$$

- When there is one unit of increase in x_1,

$$r_1 \triangleq \text{odds} = \frac{P(Y=1|\tilde{\mathbf{x}}_1)}{P(Y=0|\tilde{\mathbf{x}}_1)} = e^{a_1(x_1+1)+a_2x_2+\ldots+a_px_p+b}$$

where $\tilde{\mathbf{x}}_1 = [x_1+1, x_2, \ldots, x_p]^T$

- The *odds ratio* becomes

$$\text{odds ratio} = \frac{r_1}{r_0} = \frac{e^{a_1(x_1+1)+a_2x_2+\ldots+a_px_p+b}}{e^{a_1x_1+a_2x_2+\ldots+a_px_p+b}} = e^{a_1}$$

Relative Chance in Odds

UIC BUSINESS

- An increase of one unit in x_1 leads to multiplying odds of $Y = 1$ by e^{a_1}
- Define the *relative change in odds* (RCO) as a function of one unit in x_i:

$$\text{RCO\%} = \frac{r_i - r_0}{r_0} \times 100 = \frac{r_0 e^{a_i} - r_0}{r_0} \times 100 = (e^{a_i} - 1) \times 100$$

- For example, suppose we have trained an LR classifier with $a_2 = -0.2$. This means a one unit increase in x_2 leads to an RCO of -18.1% (decreases odds of $Y = 1$ by 18.1%)
- A k unit increase in the value of x_i leads to

$$\text{RCO\%} = \frac{r_i - r_0}{r_0} \times 100 = (e^{k a_i} - 1) \times 100$$

Oral Cancer Classification

UIC BUSINESS

- In this example, we work with a genomic data (gene expressions) taken from patients affected by oral leukoplakia
  - data was obtained from Gene Expression Omnibus (GEO) with accession #GSE26549
  - the data includes 19897 features (19894 genes, 3 binary clinical variables), and 86 patients with a median follow-up of 7.11 years
  - 35 individuals (35/86, 40.7%) developed oral cancer
- The goal is to build a classifier to classify those who developed OC from those who did not
- We use logistic regression with l_1 regularization with a $C = 16$

Oral Cancer Classification

UIC BUSINESS

```python
# load the oral cancer dataset
data = pd.read_csv('data/GenomicData_OralCancer.txt', sep=" ", header=None)
data.head()
```

✓ 0.6s Python

	0	1	2	3	4	5	6	7	8	9	...	19888	19889	19890	19891	19892	19893	19894	19895	19896	19897
0	4.44	8.98	5.58	6.89	6.40	6.35	7.12	6.87	7.18	7.81	...	6.08	5.49	12.59	11.72	8.99	10.87	1	0	1	1
1	4.59	8.57	6.57	7.25	6.44	6.34	7.40	6.91	7.18	8.12	...	6.17	6.08	13.04	11.36	8.96	11.03	1	0	0	1
2	4.74	8.80	6.22	7.13	6.79	6.08	7.42	6.93	7.48	8.82	...	6.39	5.99	13.29	11.87	8.63	10.87	1	1	0	-1
3	4.62	8.77	6.32	7.34	6.29	5.65	7.12	6.89	7.27	7.18	...	6.32	5.69	13.33	12.02	8.86	11.08	0	1	1	1
4	4.84	8.81	6.51	7.16	6.12	5.99	7.13	6.85	7.21	7.97	...	6.57	5.59	13.22	11.87	8.89	11.15	1	1	0	1

5 rows × 19898 columns

```python
# load the variable names
header = pd.read_csv('data/GenomicData_OralCancer_var_names.txt', header=None)
data.columns=header.iloc[:,0]
data.head()
```

✓ 0.0s Python

	OR4F17	SEPT14	OR4F16	GPAM	LOC100287934	LOC643837	SAMD11	KLHL17	PLEKHN1	ISG15	...	MRGPRX3	OR8G1	SPRR2F	NME2	GRI
0	4.44	8.98	5.58	6.89	6.40	6.35	7.12	6.87	7.18	7.81	...	6.08	5.49	12.59	11.72	
1	4.59	8.57	6.57	7.25	6.44	6.34	7.40	6.91	7.18	8.12	...	6.17	6.08	13.04	11.36	
2	4.74	8.80	6.22	7.13	6.79	6.08	7.42	6.93	7.48	8.82	...	6.39	5.99	13.29	11.87	
3	4.62	8.77	6.32	7.34	6.29	5.65	7.12	6.89	7.27	7.18	...	6.32	5.69	13.33	12.02	
4	4.84	8.81	6.51	7.16	6.12	5.99	7.13	6.85	7.21	7.97	...	6.57	5.59	13.22	11.87	

5 rows × 19898 columns

Oral Cancer Classification

UIC BUSINESS

```python
data.describe()
```

✓ 11.5s Python

	OR4F17	SEPT14	OR4F16	GPAM	LOC100287934	LOC643837	SAMD11	KLHL17	PLEKHN1	ISG15	...	MRGPRX3	86
count	86.000000	86.000000	86.000000	86.000000	86.000000	86.000000	86.000000	86.000000	86.000000	86.000000	...	86.000000	
mean	4.627209	8.522442	6.120465	7.143721	6.363953	6.102907	7.246512	6.842558	7.323953	7.893023	...	6.419535	
std	0.295446	0.393856	0.447600	0.312086	0.265703	0.207539	0.195140	0.155746	0.135258	0.828175	...	0.280243	(
min	4.260000	7.640000	5.090000	6.590000	5.810000	5.650000	6.830000	6.440000	6.950000	6.820000	...	5.840000	4
25%	4.470000	8.212500	5.835000	6.970000	6.180000	5.960000	7.092500	6.742500	7.222500	7.232500	...	6.222500	5
50%	4.575000	8.495000	6.090000	7.125000	6.330000	6.080000	7.240000	6.855000	7.315000	7.695000	...	6.415000	5
75%	4.685000	8.800000	6.452500	7.317500	6.515000	6.237500	7.390000	6.927500	7.420000	8.397500	...	6.550000	5
max	6.140000	9.300000	7.770000	8.930000	6.960000	6.700000	7.780000	7.170000	7.790000	10.140000	...	7.570000	6

8 rows × 19898 columns

```python
print('(sample size, dimension) = {}'.format(data.shape))
```

✓ 0.0s Python

```
(sample size, dimension) = (86, 19898)
```

Oral Cancer Classification

UIC BUSINESS

```python
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression as LRR

y_train = data.oral_cancer_output
X_train = data.drop('oral_cancer_output', axis=1)
scaler = StandardScaler().fit(X_train)
X_train = scaler.transform(X_train)

lrr = LRR(penalty='l1', C=16, solver='liblinear', random_state=42)
lrr.fit(X_train, y_train)
```

✓ 0.1s Python

```
LogisticRegression
LogisticRegression(C=16, penalty='l1', random_state=42, solver='liblinear')
```

```python
# check the number of non-zero coefficients
coeffs = lrr.coef_.ravel()
np.sum(coeffs != 0)
```

✓ 0.0s Python

```
292
```

Oral Cancer Classification

UIC BUSINESS

- There are 292 features (here all are genes) with non-zero coefficients in our *linear logistic regression* (LLR) classifier
- Sort the identified 292 features based on their odds ratio:

```python
non_zero_coeffs = coeffs[coeffs!= 0]

# odds ratios
ORs = np.exp(non_zero_coeffs)
sorted_args = ORs.argsort()
ORs_sorted=ORs[sorted_args]
ORs_sorted[:10]
```

✓ 0.0s Python

```
array([0.74175367, 0.76221689, 0.77267586, 0.77451542, 0.78316466,
       0.8163162 , 0.83429172, 0.83761203, 0.84076963, 0.84205156])
```

```python
feature_names = header.iloc[:-1,0].values
selected_features = feature_names[coeffs!= 0]
selected_features_sorted = selected_features[sorted_args]
selected_features_sorted[0:10]
```

✓ 0.0s Python

```
array(['DGCR6', 'ZNF609', 'CNO', 'RNASE13', 'BRD7P3', 'HHAT', 'UBXN1',
       'C15orf62', 'ORAI2', 'C1orf151'], dtype=object)
```

Visualization of RCO

UIC BUSINESS

- Plot the RCOs for 10 genes that led to the highest decrease and 10 genes that led to the highest increase in RCO by one unit increase in their expressions

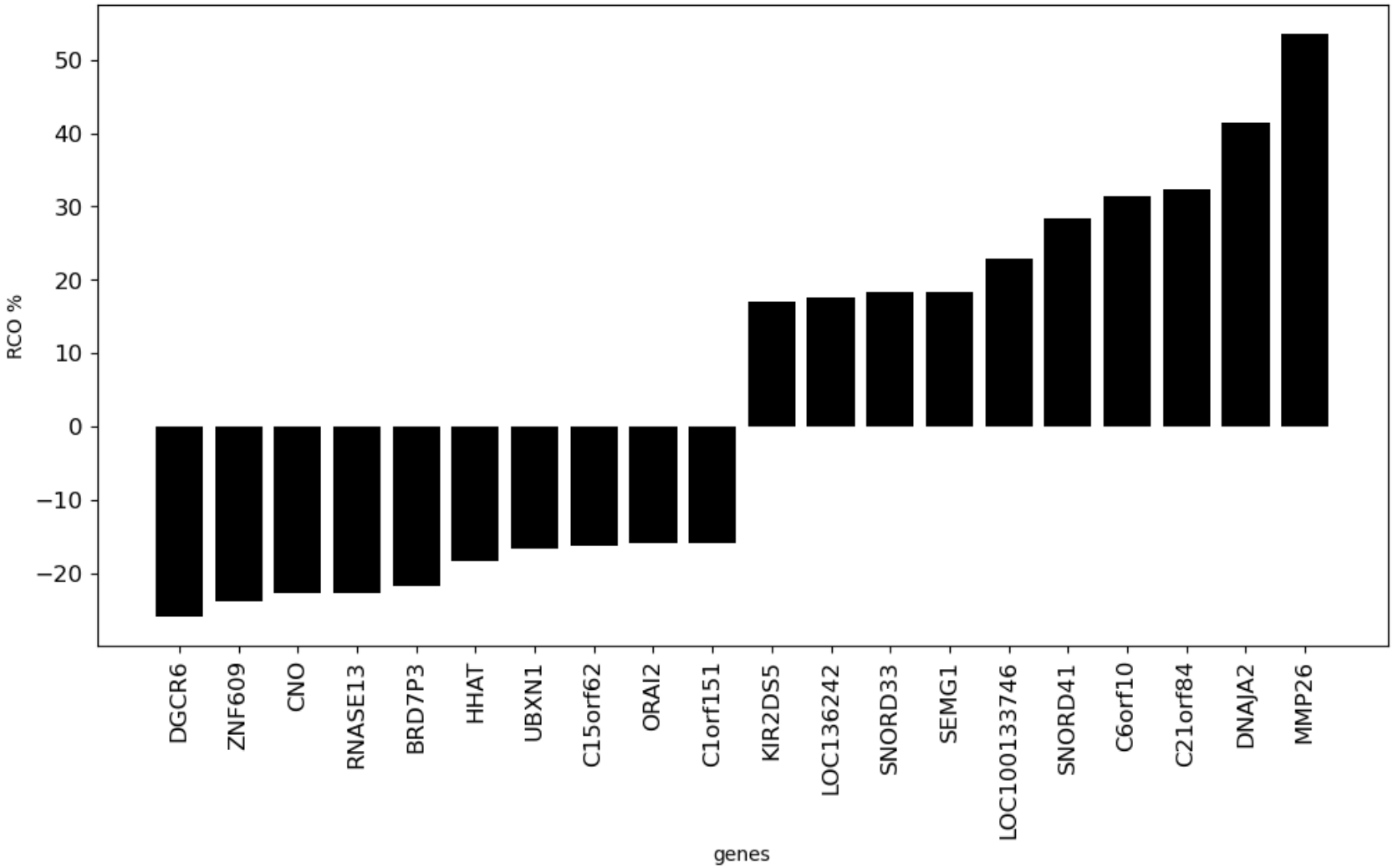
```python
from matplotlib import pyplot
import matplotlib.pyplot as plt

plt.figure(figsize=(12,6))
pyplot.bar(features_selected_RCOs, selected_RCOs, color='black')
plt.xlabel('genes')
plt.ylabel('RCO %')
plt.xticks(rotation=90, fontsize=12)
plt.yticks(fontsize=12)
```

✓ 0.1s Python

RCO vs Genes

UIC BUSINESS



Linear Models for Regression

UIC BUSINESS

- In linear regression models, the *estimate of the response variable is a linear function of parameters*: $f(\mathbf{x}) = \mathbf{a}^T\mathbf{x} + b$
  - when $p = 1$, this is *simple linear regression*
  - when $p > 1$, it is *multiple linear regression*
  - underlying assumption: $\mathbb{E}[Y|\mathbf{X} = \mathbf{x}] = f(\mathbf{x}) = \mathbf{a}^T\mathbf{x} + b$
- Given training data $\mathbf{S}_{tr} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)\}$, the most common approach to estimate $\boldsymbol{\beta} = [b, \mathbf{a}^T]^T$ is the *least squares method*
- The goal is to minimize the *residual sum of squares* (RSS), and the solution is called the *ordinary least squares* (OLS)

$$\hat{\boldsymbol{\beta}}_{\text{ols}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\ \text{RSS}(\boldsymbol{\beta}) \quad \text{RSS}(\boldsymbol{\beta}) = \sum_{j=1}^{n} \left(y_j - f(\mathbf{x}_j)\right)^2 = \sum_{j=1}^{n} (y_j - \boldsymbol{\beta}^T \tilde{\mathbf{x}}_j)^2$$

$$\tilde{\mathbf{x}}_j = \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix}, \ \boldsymbol{\beta} = \begin{bmatrix} b \\ \mathbf{a} \end{bmatrix}, \ \boldsymbol{\beta}^T \tilde{\mathbf{x}}_j = b + \mathbf{a}^T \mathbf{x}_j$$

Linear Models for Regression

UIC BUSINESS

- Matrix form: $\mathrm{RSS}(\boldsymbol{\beta}) = (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) = ||\mathbf{y} - \mathbf{X}\beta||_2^2$

$$\mathbf{X} = \begin{bmatrix} 1 & \mathbf{x}_1^T \\ \vdots & \vdots \\ 1 & \mathbf{x}_n^T \end{bmatrix}$$ is the $n \times (p+1)$ feature matrix

bivariate function: $f(x_1, x_2)$

gradient: $\begin{bmatrix} \frac{d}{dx_1} f(x_1, x_2) \\ \frac{d}{dx_2} f(x_1, x_2) \end{bmatrix}$

- To find its minimum, set the gradient to zero:

$$\frac{\partial \mathrm{RSS}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} = \mathbf{0}$$

- For two vectors $\mathbf{w}$ and $\boldsymbol{\beta}$, and a symmetric matrix $\mathbf{W}$:

$$\frac{\partial \mathbf{w}^T\boldsymbol{\beta}}{\partial \boldsymbol{\beta}} = \frac{\partial \boldsymbol{\beta}^T\mathbf{w}}{\partial \boldsymbol{\beta}} = \mathbf{w}$$

$$\frac{\partial \boldsymbol{\beta}^T\mathbf{W}\boldsymbol{\beta}}{\partial \boldsymbol{\beta}} = 2\mathbf{W}\boldsymbol{\beta}$$

quadratic function: $x_1^2 + x_2^2$

gradient: $\begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix}$

Linear Models for Regression

UIC BUSINESS

- Assume $\mathbf{X}^T\mathbf{X}$ is invertible ($\mathbf{X}$ has full column rank $p+1 \leq n$)
- The well-known *normal equation* provides a closed-form solution for the parameters that minimize the RSS

$$\hat{\boldsymbol{\beta}}_{\text{ols}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

- l_2-regularized least squares regression (ridge regression):

$$\hat{\boldsymbol{\beta}}_{\text{ridge}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\left[\text{RSS}(\boldsymbol{\beta}) + \alpha||\mathbf{a}||_2^2\right]$$

Equivalently,

$$\hat{\boldsymbol{\beta}}_{\text{ridge}} = (\mathbf{X}^T\mathbf{X} + \alpha\mathbf{I}_{p+1})^{-1}\mathbf{X}^T\mathbf{y}$$

- l_1-regularized least squares regression (lasso):

$$\hat{\boldsymbol{\beta}}_{\text{lasso}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\left[\text{RSS}(\boldsymbol{\beta}) + \alpha||\mathbf{a}||_1\right]$$

- Elastic-net least squares regression:

$$\hat{\boldsymbol{\beta}}_{\text{elastic-net}} = \underset{\boldsymbol{\beta}}{\operatorname{argmax}}\left[\text{RSS}(\boldsymbol{\beta}) + \alpha\,\nu||\mathbf{a}||_1 + \alpha\frac{1-\nu}{2}||\mathbf{a}||_2^2\right]$$

A Toy Example

UIC BUSINESS

x	-1	2	-1	0	0
y	0	2	2	0	-1

$$\mathbf{X} = \begin{bmatrix} 1 & -1 \\ 1 & 2 \\ 1 & -1 \\ 1 & 0 \\ 1 & 0 \end{bmatrix}, \quad y = \begin{bmatrix} 0 \\ 2 \\ 2 \\ 0 \\ -1 \end{bmatrix}$$

$$\hat{\boldsymbol{\beta}}_{\text{ols}} = \left(\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ -1 & 2 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 1 & 2 \\ 1 & -1 \\ 1 & 0 \\ 1 & 0 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ -1 & 2 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \\ 2 \\ 0 \\ -1 \end{bmatrix} = \begin{bmatrix} \frac{3}{5} \\ \frac{1}{3} \end{bmatrix}$$

A Toy Example

UIC BUSINESS

$$\hat{y}_1 = \frac{3}{5} - \frac{1}{3} = \frac{4}{15}, \quad \hat{y}_2 = \frac{3}{5} + \frac{2}{3} = \frac{19}{15}$$

$$\hat{y}_3 = \hat{y}_1, \quad \hat{y}_4 = \frac{3}{5}, \quad \hat{y}_5 = \hat{y}_4$$

$$\text{RSS} = \left(0 - \frac{4}{15}\right)^2 + \left(2 - \frac{19}{15}\right)^2 + \left(2 - \frac{4}{15}\right)^2 + \left(0 - \frac{3}{5}\right)^2 + \left(-1 - \frac{3}{5}\right)^2 \approx 6.53$$

$$\text{TSS} = \left(0 - \frac{3}{5}\right)^2 + \left(2 - \frac{3}{5}\right)^2 + \left(2 - \frac{3}{5}\right)^2 + \left(0 - \frac{3}{5}\right)^2 + \left(-1 - \frac{3}{5}\right)^2 = \frac{36}{5} = 7.2$$

$$\hat{R}^2 = 1 - \frac{\text{RSS}}{\text{TSS}} = 1 - \frac{6.53}{7.2} = 0.092$$

Boston Housing Data

UIC BUSINESS

- ols, ridge, lasso and elasticnet are implemented in `LinearRegression`, `Ridge`, `Lasso` and `ElasticNet` classes from `sklearn.linear_model` module
- Consider Boston Housing dataset from `BostonHousing.csv`
  - this dataset records `medv` (median house value) for 506 neighborhoods around Boston
  - build a regression model to predict `medv` using 13 features such as `rmvar` (average num of rooms per house), `age` (proportion of owner-occupied units built prior to 1940), `lstat` (precent of households with low socioeconomic status)

```python
Boston = pd.read_csv('data/BostonHousing.csv')
Boston = Boston.rename(columns={'CAT. MEDV': 'CAT_MEDV'})
print(Boston.columns)
```

✓ 0.0s Python

```
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'LSTAT', 'MEDV', 'CAT_MEDV'],
      dtype='object')
```

Boston Housing Data

UIC BUSINESS

```python
Boston.head()
```

✓ 0.0s Python

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	lstat	medv
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	5.33	36.2

```python
Boston.medv.describe()
```

✓ 0.0s Python

```
count    506.000000
mean      22.532806
std        9.197104
min        5.000000
25%       17.025000
50%       21.200000
75%       25.000000
max       50.000000
Name: medv, dtype: float64
```

Boston Housing Data

UIC BUSINESS

```python
from sklearn.linear_model import LinearRegression as LR
```

✓ 0.0s Python

```python
X = pd.DataFrame({'intercept': np.ones(Boston.shape[0]),
                  'lstat': Boston['lstat']})
X[:4]
```

✓ 0.0s Python

	intercept	lstat
0	1.0	4.98
1	1.0	9.14
2	1.0	4.03
3	1.0	2.94

```python
y = Boston['medv']
model = LR(fit_intercept=False)
results = model.fit(X, y)
```

✓ 0.0s Python

```python
results.coef_
```

✓ 0.0s Python

```
array([34.55384088, -0.95004935])
```

Boston Housing Data

UIC BUSINESS

```python
plt.scatter(Boston['lstat'], Boston['medv'], facecolors='blue',
            edgecolors='blue', marker='o', s=10, alpha=0.5)
plt.plot(Boston['lstat'],
         results.coef_[0] + results.coef_[1] * Boston['lstat'],
         color='red')
plt.xlabel('lstat')
plt.ylabel('medv')
```

✓ 0.0s Python