SCIENCE
ACADEMY

Linear Regression: Find Optimal Model

- You want to minimize $E_{in}(\underline{w}) = (\underline{\underline{X}}\underline{w} - \underline{y})^T(\underline{\underline{X}}\underline{w} - \underline{y})$ with respect to $\underline{w}$

$$\nabla E_{in}(\underline{w}^*) = \underline{0}$$
$$\frac{2}{N}\underline{\underline{X}}^T(\underline{\underline{X}}\underline{w}^* - \underline{y}) = \underline{0}$$
$$\underline{\underline{X}}^T\underline{\underline{X}}\underline{w}^* = \underline{\underline{X}}^T\underline{y}$$

- If the square matrix $\underline{\underline{X}}^T\underline{\underline{X}}$ is invertible:

$$\underline{w}^* = (\underline{\underline{X}}^T\underline{\underline{X}})^{-1}\underline{\underline{X}}^T\underline{y} = \underline{\underline{X}}^\dagger\underline{y}$$

- The matrix $\underline{\underline{X}}^\dagger \triangleq (\underline{\underline{X}}^T\underline{\underline{X}})^{-1}\underline{\underline{X}}^T$ is called **pseudo-inverse**
  - It generalizes the inverse for non-square matrices, in fact:
    - $\underline{\underline{X}}^\dagger\underline{\underline{X}} = \underline{\underline{I}}$
    - If $\underline{\underline{X}}$ is square and invertible, then $\underline{\underline{X}}^\dagger = \underline{\underline{X}}^{-1}$

SCIENCE ACADEMY

Complexity of One-Step Learning

- Learning with the pseudo-inverse is **one-step learning**
  - Contrasts with iterative methods, e.g., gradient descent
- Inverting a square matrix of size P is related to the number of parameters, not examples N
  - Complexity of one-step learning is $O(P^3)$

Linear Models Are Linear in What?

- A **model is linear** when the signal $s = \sum_{i=0}^{P} w_i x_i = \underline{\boldsymbol{w}}^T \underline{\boldsymbol{x}}$ is linear **with variables**
  - Unknown variables: weights w_i
  - Inputs x_i are fixed
- Applying a **non-linear transform to inputs** $z_i = \Phi(x_i)$ keeps the model linear, e.g.,
  - Positive/negative part (e.g., $z_i = RELU(x_i), RELU(-x_i)$)
  - Waterfall (conditioning model to different feature ranges)
  - Thresholding (e.g., $z_i = \min(x_i, T)$)
  - Indicator variables (e.g., $z_i = I(x_i > 0)$)
  - Winsorizing (replace outliers with a large constant value)
- Applying a **non-linear transform to weights** $z_i = \Phi(w_i)$ makes the model non-linear

Non-Linear Transformations in Linear Models

- **Transform variables**
  - Use $\Phi : \mathcal{X} \rightarrow \mathcal{Z}$
  - Transform each point $\underline{\boldsymbol{x}}_n \in \mathcal{X} = \mathbb{R}^d$ into a point in feature space $\underline{\boldsymbol{z}}_n = \Phi(\underline{\boldsymbol{x}}_n) \in \mathcal{Z} = \mathbb{R}^{\tilde{d}}$ with $d \neq \tilde{d}$
- **Learn**
  - Learn linear model in $\mathcal{Z}$, obtaining $\underline{\tilde{\boldsymbol{w}}}$ for separating hyperplane
- **Predict**
  - Evaluate model on new point in $\mathcal{Z}$:

$$y = \text{sign}(\underline{\tilde{\boldsymbol{w}}}^T \Phi(\underline{\boldsymbol{x}})) \text{or} \quad y = \underline{\tilde{\boldsymbol{w}}}^T \Phi(\underline{\boldsymbol{x}})$$

- Compute **decision boundary**
  - In $\mathcal{X}$ if Φ is invertible; or
  - By classifying any $\underline{\boldsymbol{x}} \in \mathcal{X}$ in $\mathcal{Z}$

- ***Perceptron***
- Logistic Regression
- LDA, QDA
- Kernel Methods

Example of Classification Problems

- **Binary classification problem**
  - $y \in \{0, 1\}$
    - Typically assign 1 to what you want to detect
  - Email: `spam`, `not_spam`
  - Online transaction: `fraudulent`, `valid`
  - Tumor: `malignant`, `benign`
- **Multi-class classification problem**
  - $y \in \{0, 1, 2, ..., K\}$
  - Email tagging: `work`, `family`, `friends`
  - Medical diagnosis: `not_ill`, `cold`, `flu`
  - Weather: `sunny`, `rainy`, `cloudy`, `snow`

Linear Regression for Classification

- Use **linear regression for classification**
  - Transform outputs into $\{+1, -1\} \in \mathbb{R}$
  - Learn $\underline{\boldsymbol{w}}^T \underline{\boldsymbol{x}}_n \approx y_n = \pm 1$
  - Use sign($\underline{\boldsymbol{w}}^T \underline{\boldsymbol{x}}_n$) as model (perceptron)
- **Not optimal**: outliers influence fit due to square distance metric
  - Use weights from linear regression to initialize a learning algorithm for classification (e.g., PLA)

Perceptron Learning Algorithm (PLA)

- First machine learning algorithm discovered
- **Algorithm**
  - Training set $\mathcal{D} = \{(\underline{\boldsymbol{x}}_1, y_1), ..., (\underline{\boldsymbol{x}}_n, y_n)\}$
  - Initialize weights $\underline{\boldsymbol{w}}$
    - Random values
    - Use linear regression for classification as seed
  - Pick a misclassified point $\text{sign}(\underline{\boldsymbol{w}}^T \underline{\boldsymbol{x}}_i) \neq y_i$ from training set $\mathcal{D}$
  - Update weights: $\underline{\boldsymbol{w}}(t+1) = \underline{\boldsymbol{w}}(t) + y_i \underline{\boldsymbol{x}}_i$
    - Like stochastic gradient descent
  - Iterate until no misclassified points
- Algorithm **converges** (slowly) for linearly separable data
- **Pocket version of PLA**
  - Idea
    - Continuously update solution
    - Keep best solution "in the back pocket"
  - Have a solution if stopping after max iterations

Non-Linear Transformations for Classifications

- Classification problems have **varying degrees of non-linear boundaries**:
  1. Non-linearly separable data
     - E.g., $+$ in center, $-$ in corners in a 2-feature scatter plot
  2. Mostly linear classes with few outliers
  3. Higher-order decision boundary
     - E.g., quadratic
  4. Non-linear data-feature relationship
     - E.g., variable threshold

- Perceptron
- ***Logistic Regression***
- LDA, QDA
- Kernel Methods

Logistic Regression Is a Probabilistic Classifier

- **Logistic regression** learns:
  - The probability of each class $\Pr(y|\underline{\boldsymbol{x}})$ given input $\underline{\boldsymbol{x}}$
  - Instead of predicting class y directly
- **Parametric approach**: assume $\Pr(y|\underline{\boldsymbol{x}}; \underline{\boldsymbol{w}})$ has a known functional form

$$\Pr(y = 1|\underline{\boldsymbol{x}}; \underline{\boldsymbol{w}}) = \text{logit}(\underline{\boldsymbol{w}}^T \underline{\boldsymbol{x}})$$

- **Optimize parameters** $\underline{\boldsymbol{w}}$ using maximum likelihood

$$\underline{\boldsymbol{w}}^* = \text{argmax}_{\underline{\boldsymbol{w}}} \Pr(\mathcal{D}; \underline{\boldsymbol{w}})$$

- **Predict** by outputting class with highest probability

$$h_{\underline{\boldsymbol{w}}}(\underline{\boldsymbol{x}}) = \begin{cases} +1 & \Pr(y = 1|\underline{\boldsymbol{x}}; \underline{\boldsymbol{w}}) \geq 0.5 \\ -1 & \Pr(y = 1|\underline{\boldsymbol{x}}; \underline{\boldsymbol{w}}) < 0.5 \end{cases}$$

Logistic Regression: Example

- Assume y is:
  - The event y_i *"patient with characteristics* $\underline{\boldsymbol{x}}$ *had heart attack"*
  - Function of parameters $\underline{\boldsymbol{x}}$ (e.g., age, gender, diet)
- In data set $\mathcal{D}$:
  - No samples of $\Pr(y|\underline{\boldsymbol{x}})$
  - Have realizations:
    - *"Patient with* $\underline{\boldsymbol{x}}_1$ *had a heart attack"*
    - *"Patient with* $\underline{\boldsymbol{x}}_2$ *didn't"*
    - ...
- Learn $\Pr(y|\underline{\boldsymbol{x}})$
  - Find best parameters $\underline{\boldsymbol{w}}$ for logistic regression model to explain data $\mathcal{D}$

Logistic Function

- Aka "sigmoid"
- **Logistic function** logit(s) is defined as

$$\text{logit}(s) \triangleq \frac{e^s}{1+e^s} = \frac{1}{1+e^{-s}}$$

  - Varies in [0, 1]
  - Crosses the origin at 0.5
  - Asymptotes at 0 and 1
  - It is a soft version of sign()

Logistic Regression vs Linear Classifier

- **Functional form is similar**
  - **Logistic regression**: $h(\underline{\boldsymbol{w}}) = \text{logit}(\underline{\boldsymbol{w}}^T \underline{\boldsymbol{x}})$
  - **Linear classifier** (perceptron): $h(\underline{\boldsymbol{w}}) = \text{sign}(\underline{\boldsymbol{w}}^T \underline{\boldsymbol{x}})$
- **Difference** in probabilistic interpretation and fitting method
  - Logistic regression** lacks samples of probability function to interpolate
    - Uses realizations of random variable
    - Seeks model parameters maximizing data likelihood
  - **Linear classification** assumes class value is linear function of inputs

Error for Probabilistic Binary Classifiers

- For probabilistic binary classification, use **log-probability error** as point-wise error

$$e(h(\underline{\boldsymbol{x}}), y) \triangleq -\log(\Pr(y = h(\underline{\boldsymbol{x}})|\underline{\boldsymbol{x}}; \underline{\boldsymbol{w}}))$$

  - Negate for positive errors: $\log([0, 1]) \in [-\infty, 0)$
- Log probability generalizes 0-1 error
  - Case $y = 1$
    - Output $h(\underline{\boldsymbol{x}})$ close to 1 $\implies$ probability 1 $\implies$ $\log(1) = 0$ $\implies$ $e(\cdot) = 0$
    - Output close to 0 $\implies$ $e() = -\log(0) \to +\infty$
  - Similar behavior for $y = 0$

One-Liner Error for Probabilistic Binary Classifiers

- **Point-wise error** for example $(\underline{\boldsymbol{x}}, y)$ for probabilistic binary classifiers is defined as:

$$
\begin{aligned}
e(h(\underline{\boldsymbol{x}}), y) &\triangleq -\log(\Pr(h(\underline{\boldsymbol{x}}) = y|\underline{\boldsymbol{x}})) \\
&= \begin{cases} -\log(\Pr(y = 1|\underline{\boldsymbol{x}})) & y = 1 \\ -\log(\Pr(y = 0|\underline{\boldsymbol{x}})) & y = 0 \end{cases} \\
&= \begin{cases} -\log(\Pr(y = 1|\underline{\boldsymbol{x}})) & y = 1 \\ -\log(1 - \Pr(y = 1|\underline{\boldsymbol{x}})) & y = 0 \end{cases}
\end{aligned}
$$

- Any function of a binary variable:

$$
y = \begin{cases} a & x = 1 \\ b & x = 0 \end{cases}
$$

  can be written as one-liner: $y = x \cdot a + (1 - x) \cdot b$

- Point-wise error can be written independently of $\Pr(y = 1|\underline{\boldsymbol{x}})$:

$$
e(h(\underline{\boldsymbol{x}}), y) = -y \log(\Pr(y = 1|\underline{\boldsymbol{x}}) - (1 - y) \log(1 - \Pr(y = 1|\underline{\boldsymbol{x}}))
$$

One-Liner Error for Logistic Regression

- The point-wise error for a binary classifier is:

$$e(h(\underline{\boldsymbol{x}}), y) = -y \log(\Pr(y = 1|\underline{\boldsymbol{x}}) - (1 - y) \log(1 - \Pr(y = 1|\underline{\boldsymbol{x}}))$$

- Simplify further **with logit function**:

$$e(h(\underline{\boldsymbol{x}}), y) \triangleq -\log \Pr(h(\underline{\boldsymbol{x}}) = y)$$

... a bit of math manipulation ...

$$= -\log \text{logit}(y \underline{\boldsymbol{w}}^T \underline{\boldsymbol{x}}) \qquad \text{since } \text{logit}(s) = \frac{1}{1 + e^{-s}}$$

$$= \log(1 + \exp(-y \underline{\boldsymbol{w}}^T \underline{\boldsymbol{x}}))$$

- Point-wise error for logistic regression equals **cross-entropy error**

Cross-Entropy Error

- **Point-wise error for logistic regression** has expression:

$$e(h(\underline{\boldsymbol{x}}), y) = \log(1 + \exp(-y \cdot \underline{\boldsymbol{w}}^T \underline{\boldsymbol{x}}))$$

  - It is called **cross-entropy error**
  - Note: no $-$ before $\log(\cdot)$ but before $y \cdot \underline{\boldsymbol{w}}^T \underline{\boldsymbol{x}}$
- Cross-entropy error **generalizes 0-1 error**
  - If $\underline{\boldsymbol{w}}^T \underline{\boldsymbol{x}}$ agrees with y in sign and $|\underline{\boldsymbol{w}}^T \underline{\boldsymbol{x}}|$ is large $\implies$ error goes to 0
  - If they disagree in sign $\implies$ error goes towards ∞
- Define **in-sample error on training set** as average of point-wise errors:

$$E_{in} \triangleq \frac{1}{N} \sum_n e(h(\underline{\boldsymbol{x}}_n), y_n)$$

Fitting Logistic Regression

- A plausible error measure of a hypothesis is based on **likelihood of data**

$$\Pr(\mathcal{D}|h = f)$$

  - *"How likely is the data $\mathcal{D}$ under the assumption that $h = f$?"*
  - *"How likely is that the data $\mathcal{D}$ was generated by h?"*

- **Maximize likelihood** $\mathcal{D}$ generated from logistic regression

$$\Pr(y = 1|\underline{\boldsymbol{x}}; \underline{\boldsymbol{w}})$$

- It can be proved that this is equivalent to **minimizing in-sample error** on training set **using cross-entropy error**

Fitting Logistic Regression (Optional)

- Find $\underline{\boldsymbol{w}}$ that maximizes likelihood for data set $\mathcal{D} = \{(\underline{\boldsymbol{x}}_1, y_1), ..., (\underline{\boldsymbol{x}}_N, y_N)\}$ generated by model $h(\underline{\boldsymbol{x}})$:

  $$\Pr(D|\underline{\boldsymbol{w}}) = \Pr(y_1 = h(\underline{\boldsymbol{x}}_1) \wedge ... \wedge y_N = h(\underline{\boldsymbol{x}}_N)) = \Pr(y_1 = y_1' \wedge ... \wedge y_N = y_N')$$

- Model form:

  $$y' = h(\underline{\boldsymbol{x}}) = \begin{cases} +1 & \text{if } \text{logit}(\underline{\boldsymbol{w}}^T \underline{\boldsymbol{x}}) > 0.5 \\ -1 & \text{otherwise} \end{cases}$$

- Assuming independence among training examples:

  $$\Pr(D|\underline{\boldsymbol{w}}) = \prod_{i=1}^{N} \Pr(y_i = y_i'|\underline{\boldsymbol{x}}_i)$$

- Fold y_n in expression:
  - When $y_n = 1$, $\Pr(y_n = y_n') = \text{logit}(\underline{\boldsymbol{w}}^T \underline{\boldsymbol{x}}_n)$
  - When $y_n = -1$, $\Pr(y_n = y_n') = \text{logit}(-\underline{\boldsymbol{w}}^T \underline{\boldsymbol{x}}_n)$
  - Thus, $\Pr(y_n = y_n') = \text{logit}(y_n \underline{\boldsymbol{w}}^T \underline{\boldsymbol{x}}_n)$

Fitting Logistic Regression (Optional)

- Given:

$$\Pr(D|\underline{\boldsymbol{w}}) = \prod_{i=1}^{N} \text{logit}(y_n \underline{\boldsymbol{w}}^T \underline{\boldsymbol{x}}_n)$$

- Re-write optimization as minimizing sum of point-wise errors $E_{in} = \sum e(h(\underline{\boldsymbol{x}}_n), y_n)$
  - Maximize log(...) with respect to $\underline{\boldsymbol{w}}$ since log argument > 0 and log() is monotone
- Equivalently, minimize:

$$\begin{aligned} -\frac{1}{N}\log(...) &= -\frac{1}{N}\log(\prod(...)) = -\frac{1}{N}\sum(\log(...)) \\ &= \frac{1}{N}\sum \log(\frac{1}{\text{logit}(y_n \underline{\boldsymbol{w}}^T \underline{\boldsymbol{x}}_n)}) \\ &= \frac{1}{N}\sum \log(1+\exp(-y_n \underline{\boldsymbol{w}}^T \underline{\boldsymbol{x}}_n)) = \frac{1}{N}\sum e(h(\underline{\boldsymbol{x}}_n), y_n) = E_{in}(\underline{\boldsymbol{w}}) \end{aligned}$$

Gradient Descent for Logistic Regression

- Gradient descent requires two inputs:
  - Gradient of the cost function $\frac{\partial E}{w_j}$ for all j
  - Cost function $E_{in}(\underline{\boldsymbol{w}})$
- The cost function is:

$$E_{in}(\underline{\boldsymbol{w}}) = \frac{1}{N}\sum_i e(h(\underline{\boldsymbol{x}}_i; \underline{\boldsymbol{w}}), y_i)$$

- The cost function for logistic regression:

$$E_{in}(\underline{\boldsymbol{w}}) = \frac{1}{N}\sum_i \log(1 + \exp(-y_i \underline{\boldsymbol{w}}^T \underline{\boldsymbol{x}}_i))$$

- Thus gradient descent converges to global minimum
  - It can be shown that $E_{in}(\underline{\boldsymbol{w}})$ is convex in $\underline{\boldsymbol{w}}$
  - In fact sum of exponentials and flipped exponentials is convex and log is monotone

One-Vs-All Multi-Class Classification

- Aka "one-vs-rest" classifier
- Assume we have n classes $c_1, ..., c_n$ to distinguish given $\underline{\boldsymbol{x}}$
- Learn
  - Create n binary classification problems where we classify c_i vs c_{-i} (everything but i)
  - Learn n classifiers with optimal $\underline{\boldsymbol{w}}_i$, each estimating $\Pr(y = i|\underline{\boldsymbol{x}}; \underline{\boldsymbol{w}}_i)$
- Predict
  - Evaluate the n classifiers
  - Pick the class $y = i$ with the maximum $\Pr(y = i|\underline{\boldsymbol{x}})$

Cost Function for Multi-Class Probabilistic Classification

- The cost function for logistic regression is:

$$E_{in}(\underline{\boldsymbol{w}}) = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log \Pr(y=1|\underline{\boldsymbol{x}}_i) + (1-y_i)\log(1-\Pr(y=1|\underline{\boldsymbol{x}}_i)))$$

- Encode the expected outputs $\underline{\boldsymbol{y}}_i$ one-hot
  - I.e., the j-th element $\underline{\boldsymbol{y}}_i|_j$ is 1 iff the correct class is the j-th
  - E.g., for $k = 4$ 1000
- Using $\underline{\boldsymbol{h}}(\underline{\boldsymbol{x}}))$ as the outputs from the model and $\Pr(y=1|\underline{\boldsymbol{x}}) = p(\underline{\boldsymbol{x}})$:

$$E_{in}(\underline{\boldsymbol{w}}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K}\left(\underline{\boldsymbol{y}}_i \log(\underline{\boldsymbol{p}}(\underline{\boldsymbol{x}}_i)) + (1-\underline{\boldsymbol{y}}_i)\log(1-\underline{\boldsymbol{p}}(\underline{\boldsymbol{x}}_i))\right)|_k$$

- In the innermost summation we consider the error on each class / digit
  - E.g., for $k = 4$ 1000 vs 0100
  - The digits that are equal don't contribute to the error
  - The different digits give a positive contribution

- Perceptron
- Logistic Regression
- ***LDA, QDA***
- Kernel Methods

Basic Idea of Parametric Models

- We assume a model is generating the data
  - The functional form of the model is known
  - The model is parametrized with unknown parameters to estimate
- **Pros**
  - Utilize structure in the data
  - Models are easy to fit: few parameters to estimate
  - Accurate predictions if model assumptions are correct
- **Cons**
  - Strong assumptions about the data
  - Low accuracy if model assumptions are incorrect

Linear and Quadratic Discriminant Analysis

- Aka LDA and QDA
- Parametric models
  - Assume each class generating process is multivariate Gaussian
  - Classifiers with linear and quadratic decision surface
- **Pros**
  - Closed-form solutions easy to compute (sample mean and covariance)
  - Inherently multiclass
  - No hyperparams to tune
- **Cons**
  - Strong assumptions about the data

LDA / QDA: Model Form

- Both LDA and QDA assume that the class generating process $f_k(\underline{\boldsymbol{x}}; \underline{\boldsymbol{\mu_k}}, \underline{\underline{\boldsymbol{\Sigma_k}}})$ is from a multivariate Gaussian
- Linear discriminant analysis has a model:

$$f_k(\underline{\boldsymbol{x}}; \underline{\boldsymbol{\mu_k}}, \underline{\underline{\boldsymbol{\Sigma_k}}}) \sim \mathcal{N}(\underline{\boldsymbol{\mu_k}}, \underline{\underline{\boldsymbol{\Sigma}}}))$$

  where:

  - Means $\underline{\boldsymbol{\mu_k}}$ are different for all k classes
  - The covariance matrix $\underline{\underline{\boldsymbol{\Sigma}}}$ is the same for all k classes
  - It can be proven that the classes are separated by linear decision boundaries

- Quadratic discriminant analysis has a model:

  - Classes k have different covariance matrix $\underline{\underline{\boldsymbol{\Sigma_k}}}$
  - It can be proved that the classes are separated by quadratic boundaries

Bayes Theorem for LDA / QDA

- Consider a classification setup with multi-class output $Y \in \{1, ..., K\}$
- We need to build a parametric model for the conditional distribution:

$$\Pr(Y = k | X = \underline{\boldsymbol{x}})$$

- Use Bayes theorem:

$$\Pr(Y = k | X = \underline{\boldsymbol{x}}) = \frac{\Pr(X = \underline{\boldsymbol{x}} | Y = k) \cdot \Pr(Y = k)}{\Pr(X = \underline{\boldsymbol{x}})}$$

  where:

  - $\Pr(X = \underline{\boldsymbol{x}} | Y = k)$: given a class, estimate the probability of $\underline{\boldsymbol{x}}$
  - $\Pr(Y = k) = \pi_k$: estimate the probability of each class (prior)
  - $\Pr(X = \underline{\boldsymbol{x}})$: estimate the probability of each input

- Estimate probabilities from data

LDA / QDA: Boundary Decision (*)

- Consider the ratio between the probabilities of $Y = k$ vs $Y = j$:

$$r = \frac{\Pr(Y = k|X = \underline{\boldsymbol{x}})}{\Pr(Y = j|X = \underline{\boldsymbol{x}})}$$

- Using the model assumption and Bayes theorem:

$$\begin{aligned}\Pr(Y = k|X = \underline{\boldsymbol{x}}) &\propto \Pr(X = \underline{\boldsymbol{x}}|Y = k) \cdot \Pr(Y = k) \\ &= f_k(\underline{\boldsymbol{x}}; \underline{\boldsymbol{\mu_k}}) \cdot \pi_k\end{aligned}$$

where:

$$f_k(\underline{\boldsymbol{x}}) = c \cdot \exp\left(-\frac{1}{2}(\underline{\boldsymbol{x}} - \underline{\boldsymbol{\mu_k}})^T \underline{\underline{\boldsymbol{\Sigma}}}^{-1}(\underline{\boldsymbol{x}} - \underline{\boldsymbol{\mu_k}})\right)$$

- We can apply a $\log(\cdot)$ since it is a monotone transformation

$$r = \log \frac{f_k(\underline{\boldsymbol{x}})}{f_j(\underline{\boldsymbol{x}})} + \log \frac{\pi_k}{\pi_j}$$

LDA / QDA: Boundary Decision (*)

- We can apply a $\log(\cdot)$ since it is a monotone transformation

$$r = \log \frac{f_k(\underline{x})}{f_j(\underline{x})} + \log \frac{\pi_k}{\pi_j}$$

- The second term does not depend on $\underline{x}$
- The first term is proportional to:

$$(\underline{x} - \underline{\mu_k})^T \underline{\underline{\Sigma}}^{-1} (\underline{x} - \underline{\mu_k}) - (\underline{x} - \underline{\mu_j})^T \underline{\underline{\Sigma}}^{-1} (\underline{x} - \underline{\mu_j})$$

- By expanding the expression, simplifying $\underline{x}^T \underline{\underline{\Sigma}}^{-1} \underline{x}$ noticing that $\underline{x}^T \underline{\underline{\Sigma}}^{-1} (\underline{\mu_k} - \underline{\mu_j})$ plus its transposed value (because $\underline{\underline{\Sigma}}$ is symmetrical) is equal to $2\underline{x}^T \underline{\underline{\Sigma}}^{-1} (\underline{\mu_k} - \underline{\mu_j})$ we get:

$$-\frac{1}{2} (\underline{\mu_k} + \underline{\mu_j})^T \underline{\underline{\Sigma}}^{-1} (\underline{\mu_k} - \underline{\mu_j}) + \underline{x}^T \underline{\underline{\Sigma}}^{-1} (\underline{\mu_k} - \underline{\mu_j})$$

which is a term linear in $\underline{x}$

LDA / QDA: Learn Model

- In practice:
  - We don't care about $\Pr(X = \underline{x})$ since this is common for all classes
  - We assume to know the prior $\Pr(Y = k) = \pi_k$ or we estimate it from the data
  - We need to estimate the conditional probability $\Pr(X = \underline{x} | Y = k)$
- The model assumes that the conditional probability has a Gaussian distribution:

$$\Pr(X = \underline{x} | Y = k) = f_k(\underline{x}; \underline{\mu_k}, \underline{\underline{\Sigma_k}})$$

where:

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

- We can estimate the parameters $\underline{\mu_k}$, $\underline{\underline{\Sigma_k}}$ from the data using sample mean and covariance

Evaluating LDA / QDA

- When we get a new $\underline{\boldsymbol{x}} = \underline{\boldsymbol{x}}'$ we compute for each class $Y = k$

$$\Pr(Y = k | X = \underline{\boldsymbol{x}}) \propto f_k(\underline{\boldsymbol{x}}; \underline{\boldsymbol{\mu_k}}, \underline{\underline{\boldsymbol{\Sigma_k}}}) \cdot \pi_k$$

  and choose the k that maximizes the posterior probability

- Sometimes $f_i(\underline{\boldsymbol{x}})$ is from a multivariate Gaussian distribution where $\underline{\underline{\boldsymbol{\Sigma}}}$ is diagonal because of feature independence
  - Then the expressions can be simplified even more

- Perceptron
- Logistic Regression
- LDA, QDA
- ***Kernel Methods***