summary

linear regression

1. lasso regression-ℓ_1 regularization: tend to get sparse matrix W
2. ridge regression-ℓ_2 regularization: tend to punish large value in the matrix W
3. regularization: idea is to minimize $\sum w$ or $\sum w^2$
4. when d is large, we tend to use gradient descent method; when d is small, we use closed form method
5. decision boundary of lasso and ridge: non-linear or linear?

logistic regression

1. what is the cost function of logistic function? how to prove the cost function is convex
2. $J(w) = -\frac{1}{m}\sum_{i=1}^{m}[y_i \log(f_{w,b}(x_i)) + (1-y_i)\log(1-f_{w,b}(x_i))]$
3. $\nabla_w J(w) = \frac{1}{m}\sum_{i=1}^{m} y_i[f_{w,b}(x_i) - 1]x_i$
4. softmax function derivative: similar to sigmoid function, multiclass logistic regression

decision tree

1. how use information gain to determine the attribute: $I(A;B) = H(A) - H(A|B)$
2. gini index: $\sum_{i=1}^{K} p_i(1-p_i)$
3. misclassification error: $1 - \max\{p_i\}_{i=1}^{K}$
4. entropy: -$\sum_{i=1}^{K} p_i \log p_i$

neural network and CNN

1. formula in CNN: $P = \frac{F-1}{2}$; $W_2 = (W_1 - F + 2P)/S + 1$; $H_2 = (H_1 - F + 2P)/S + 1$;
2. number of parameters: $F^2CK + K$

bias and variance; performance

1. derive the decomposition
2. k-fold cross validation
3. increasing classification thresholds: FNR increase, FPR decrease
4. ROC curve: y-axis is TPR; x-axis is FPR; the higher the ROC the better the model
5. confusion matrix; accuracy; precision; recall

k-means

1. two lemmas for acclerated k means
2. silhouette score
3. rand index

probability

information theory

1. information: $I(x_k) = \log\frac{1}{p_k} = -\log(p_k)$
2. entropy: $H_p(\chi) = E[I(x_k)] = \sum_{x_k\in\chi} p_k \cdot I(x_k) = -\sum_{x_k\in\chi} p_k \cdot \log(p_k)$
3. cross-entropy: $H_{P,Q}(\chi) = -\sum_{x_k\in\chi} P(X=x_k)\cdot\log(Q(X=x_k))$
   cross entropy is nonnegative and is not smaller than entropy: $H_{P,Q}(\chi) \geq H_P(\chi)$
4. KL-divergence: $D_{P,Q}(\chi) = \int_{x\in\mathcal{X}} p_X(x)\log\frac{p_X(x)}{q_X(x)}$

linear regression

vector-matrix differentiation formulas

1. $\frac{d(X^\top w)}{dw} = X$
2. $\frac{d(\omega^\top X)}{d\omega} = X$
3. $\frac{d(wX)}{dw} = X^\top$
4. $\frac{d(w^\top Xw)}{dw} = (X + X^\top)w$

least square regression

let $e = Xw - y$,

$$\begin{aligned} J(w) &= e^\top e = (Xw-y)\top(Xw-y) \\ &= (w\top X^\top - y^\top)(Xw-y) \\ &= w^\top X^\top Xw - w\top X^\top y - y^\top Xw + y^\top y \\ &= w^\top X^\top Xw - 2y^\top Xw + y^\top y \end{aligned}$$

differentiating $J(w)$ w.r.t. w and setting the result to 0:

$$\frac{\partial}{\partial w}J(w) = 0$$

$$\frac{\partial}{\partial w}(w^\top X^\top Xw - 2y^\top Xw + y^\top y) = 0$$
$$\Rightarrow 2X^\top Xw - 2X^\top y = 0$$
$$\Rightarrow 2X^\top Xw = 2X^\top y$$

linear regression solved by gradient descent

the optimization problem is:

$$\bar{w}^* = argmin_{\bar{w}} J(w), J(\bar{w}) = \frac{1}{2}\sum_{i=1}^{m}(x_i^\top w + b - y_i)^2 = \frac{1}{2}(X\bar{w} - y)^2$$

where $X = [(1, x_1^\top), ..., (1, x_m^\top)] \in \mathbb{R}^{m\times(d+1)}$, and $\bar{w} = [b; w] \in \mathbb{R}^{(d+1)\times 1}$ we update w by gradient descent algorithm:

$$w \leftarrow w - \alpha\frac{\partial J(w)}{\partial w}, \frac{\partial J(w)}{\partial w} = X^\top(Xw - y)$$

where α is called step-size or learning rate.

Remark 0.1.

1. *the complexity of closed-form solution is $O(d^3 + md)$ and the complexity of gradient descent is $O(T \times md^2)$*
2. $f_{w,b}(X) = Xw = \begin{bmatrix} x_1^\top w \\ \vdots \\ x_m^\top w \end{bmatrix}$, *where* $x_i^\top w = [1, x_{i,1}, ..., x_{1,d}] \begin{bmatrix} b \\ w_1 \\ \vdots \\ w_d \end{bmatrix}$

linear regression: probabilistic perspective

1. *we assume that: $y = w^\top x + e$, where $e \sim \mathcal{N}(0, \sigma^2)$ is called observation noise or residual error*
2. *y is a random variable and its conditional probability is*
$$p(y|x, w) = \mathcal{N}(w^\top x, \sigma^2)$$
3. *maximum log-likelihood estimation:*
$$\begin{aligned} w_{MLE} &= \arg max_w \log \mathcal{L}(w|D) = \arg\max_w \log(\prod_i^m p(y_i|w_i, w)) \\ &= \arg\max_w \log\left(\frac{1}{\sigma^m (2\pi)^{\frac{m}{2}}} e^{-\frac{1}{2\sigma^2}\sum_i^m (y_i - w^\top x_i)^2}\right) \\ &= \arg\max_w - \log(\sigma^m (2\pi)^{\frac{m}{2}}) - \frac{1}{2\sigma^2}\sum_i^m (y_i - w^\top x_i)^2 \\ &\equiv \arg\min_w \frac{1}{2}\sum_i^m (y_i - w^\top x_i)^2 \end{aligned}$$

linear regression for classification

ridge regression

1. originally we need to solve the problem:
$$\min_{w,b} \sum_{i=1}^m (f_{w,b}(x_i) - y_i)^2$$
the problem is that the solution
$$\hat{w} = (X^\top X)^{-1} X^\top y$$
maybe not exist since $X^\top X$ may be not invertible
2. we add one term $\lambda_{\bar{w}^\top \bar{w}}$:
$$\min_{w,b} \sum_{i=1}^m (f_{w,b}(x_i) - y_i)^2 + \lambda \bar{w}^\top \bar{w}$$
where $\bar{w} = \hat{I}_d w = [0, w_1, w_2, ..., w_d]^\top$ and $\hat{I}_d \in \mathbb{R}^{(d+1)\times(d+1)}$ is defined by setting the $(1,1)$ entry in the $d+1$ dimensional identity matrix $\hat{I}_{d+1}$ as 0.
3. solve the problem:
$$\begin{gathered} \min_{w,b} (Xw - y)^\top (Xw - y) + \lambda \bar{w}^\top \bar{w} \\ \frac{\partial}{\partial w}(Xw - y)^\top (Xw - y) + \lambda \bar{w}^\top \bar{w} = 0 \\ \rightarrow 2X^\top X w - 2X^\top y + 2\lambda \hat{I}_d w = 0 \\ \rightarrow X^\top X w + \lambda \hat{I}_d w = x^\top y \\ \rightarrow (X^\top x + \lambda \hat{I}_d) w = X^\top y \\ \rightarrow w = (X^\top X + \lambda \hat{I}_d)^{-1} X^\top y \end{gathered}$$
here we have $(X^T X + \lambda \hat{I}_d)$ is positive definite since $X^T X$ is semi-positive definite.
4. we observe that when we set a larger λ, the resulting curve will be more smoother.

polynomial regression

1. some data are not linearly separated such as the XOR data. we could design a novel linear regression model to make the data linearly separated:
$$f_{w,b}(x) = w_1 x_1 + w_2 x_2 + w_{12} x_1 x_{w11} x_1^2 + w_{22} x_2^2$$
where $w_0 = b$
2. the linear model $f_{w,b}(x) = x^\top w + b$ can be written as
$$f_{w,b}(x) = \sum_{i=0}^d x_i w_i = w_0 + \sum_{i=1}^d x_i w_i$$
by including terms involving the products of pairs of components of x we obtain a quadratic model:
$$f_{w,b}(x) = w_0 + \sum_{i=1}^d w_i x_i + \sum_{i=1}^d \sum_{j=1}^d w_{ij} x_i x_j$$
we can also get $f_{w,b}(x)$ with higher order ...
3. for high dimensional d and high polynomial order, the number of polynomial terms becomes explosive. polynomials of order larger than 3 is seldom used.
4. we write the polynomial regression as:
$$f_{w,b}(x) = \phi(x)^\top w$$
where
$$\begin{gathered} \phi(x) = [1, x_1, ..., x_d, ..., x_i x_j, ..., x_i x_j x_k, ...]^\top \\ w = [w_0, w_1, ..., w_d, ..., w_{ij}, ..., w_{ijk}]^\top \end{gathered}$$
5. $f_{w,b}(x)$ is still a linear function with respect to w, so it is still called a linear model.
6. we extend to the case of m data point, i.e. $X = [x_1^\top, x_2^\top, ..., x_m^\top] \in \mathbb{R}^{m\times(d+1)}$, the basis expansion is presented by
$$P(X) = [\phi(x_1)^\top; , , , ; \phi(x_m)^\top] \in \mathbb{R}^{m\times|w|}$$

lasso regression

1. we get w by solving the problem:
$$\min_{w,b} \sum_{i=1}^m (f_{w,b}(x_i) - y_i)^2 + \lambda\alpha|w|$$
2. this is called Lasso regression, and the regularization is called ℓ_1 regularization. It will encourage the sparse parameters.

robust linear regression

1. we do not want outlier data influence the learned parameters w too much.
2. the loss curve of the residual sum of squares:

$$J(w) = \frac{1}{2}\sum_{i=1}^{m}(x_i^\top w - y_i)^2$$

we adopt the ℓ_1 loss to replace the ℓ_2 loss:

$$J(w) = \sum_{i=1}^{m}|x_i^\top w - y_i|$$

using ℓ_1 loss, we can alleviate the influence of outliers since when the residual is large, the ℓ_1 loss is much smaller than ℓ_2 loss.

3. the ℓ_1 loss function is non-differentiable, so we cannot apply the gradient descent algorithm.
4. we utilize the equation:

$$|a| = \min_{\mu>0}\frac{1}{2}(\frac{a^2}{\mu}+\mu)$$

the ℓ_1 minimization problem can be reformulated as

$$\min_{w}\min_{\mu_1,\ldots,\mu_m>0}\frac{1}{2}\left(\frac{(x_i^\top w - y_i)^2}{\mu_i}+\mu_i\right)$$

it can be iteratively and alternatively optimized as follows:

(a) given $w, \mu_i = |x_i^\top - y_i|, i = 1, \ldots, m$

(b) given $\mu, w = \arg\min_w \sum_{i=1}^{m}\frac{1}{2}(x_i^\top w - y_i)^2$

logistic regression

classification and representation

1. a desired hypothesis unction for this task should be $f_{w,b}(x) \in [0,1]$
2. feature scaling does not help if linear regression does not work well.
3. even the training set satisfies that all $y_i \in [0,1]$ for all points (x_i, y_i), the linear y hypothesis function $f_{w,b}(x) \in [0,1]$ may not classify for all values of x_i
4. hypothesis representation $f_{w,b}(x) = g(w^\top x) \in [0,1], g(z) = \frac{1}{1+\exp(-z)}$, where $g(\cdot)$ is called sigmoid function or logistic function.
5. $g'(z) = \frac{\exp(-z)}{[1+\exp(-z)]^2} = g(1-g)$

logistic regression

1. training set: m training examples $\{(x_i, y_i)_{i=1}^m\}$
2. hypothesis function: $f_{w,b}(x) = g(w^\top x + b) = \frac{1}{1+\exp(-w^\top x - b)}$
3. cost function: ℓ_2 loss or residual sum of squares: $J(w) = \frac{1}{2m}\sum_i^m (g(w^\top x_i) - y_i)^2$, which is non-convex w.r.t. w while the cost function for linear regression: $J(w) = \frac{1}{2m}\sum_{i=1}^{m}(f_{w,b}(x_i) - y_i)^2 = \frac{1}{2m}\|Xw - y\|^2$
4. we need a convex cost function
5. cross-entropy: $H(p,q) = -\int_x p(x)\log(q(x))dx$ or $-\sum_x p(x)\log(q(x))$ where $p(x)$, $q(x)$ are probability density functions of x if x is a continuous random variable, or, probability mass functions if x is a discrete random variable.
6. cross-entropy loss: $cost(y(x), f_{w,b}(x)) = H(y(x), f_{w,b}(x)) = -P(y=1|x)\cdot\log P(y = 0|x)\cdot\log P(y=0|x;w) = \begin{cases} -\log(f_{w,b}(x)) & \text{if } y(x) = 1 \\ -\log(1 - f_{w,b}(x)) & \text{if } y(x) = 0 \end{cases}$
7. cost function of logistic regression:

$$J(w) = \frac{1}{m}\sum_{i=1}^{m} cost(y_i, f_{w,b}(x_i))$$

which can be simplified as

$$J(w) = -\frac{1}{m}\sum_{i=1}^{m}[y_i\log(f_{w,b}(x_i)) + (1-y)\log(1 - f_{w,b}(x_i))]$$

we can prove that $J(w)$ is convex w.r.t. w

8. learning w by minimize $J(w)$, i.e.

$$w^* = \arg\min_w J(w)$$

$$= \arg\min_w -\frac{1}{m}\sum_{i=1}^{m}[y_i\log(f_{w,b}(x_i)) + (1-y_i)\log(1 - f_{w,b}(x_i))]$$

we utilize gradient descent method:

$$w \leftarrow w - \alpha\nabla_w J(w)$$

$$\nabla_w J(w) = \frac{1}{m}\sum_{i=1}^{m}[f_{w,b}(x_i) - y_i]x_i$$

9. multi-classification: different from binary classification $y \in \{0,1\}$, we have $y \in \{0, 1, \ldots, C\}$
10. softmax regression:

$$f_{w,b}^{(i)}(x) = \frac{\exp(w_j^\top x + b_j)}{\sum_{c=1}^{C}}\exp(w_c^\top x + b_c) = P(y = j|x, W, b)$$

where $W = [w_1, w_2, \ldots, w_C]$, $b = [b_1, b_2, \ldots, b_C]$ with C being the number of classes. we write $f_{W,b}^{(j)}$ as $f_{W_j,b_j}(\cdot)$

11. cost function for multi-classification:

$$J(w) = -\frac{1}{m}\sum_i^m\sum_j^C[\mathbb{I}(y_i = j)\log(f_{w_j,b_j}(x_i))]$$

where $\mathbb{I}(a) = 1$ if a is true, otherwise $\mathbb{I}(a) = 0$

12. gradient descent: $w_j \leftarrow w_j - \alpha\frac{\partial J(W)}{\partial w_j}$,

$$\frac{\partial J(W)}{\partial w_j} = -\frac{1}{m}\sum_i^m[\frac{\mathbb{I}(y_i = j)}{f_{w_j,b_j}(x_i)}\cdot\frac{\nabla f_{w_j,b_j}(x_i)}{\nabla w_j}$$

$$+\sum_{c\neq j}^{C}\frac{\mathbb{I}(y_i = j)}{f_{w_c,b_c}(x_i)}\cdot\frac{\nabla f_{w_c,b_c}(x_i)}{\nabla w_j}]$$

$$\rightarrow \frac{\partial J(W)}{\partial w_j} = \frac{1}{m}\sum_i^m(f_{w_j,b_j}(x_i) - \mathbb{I}(y_i = j))x_i$$

regularized logistic regression

1. by keeping all features, but reduce magnitude/value of each parameter, such that each feature contributes a bit to predict y. In this way we can address the overfitting problem.
2. the objective function: $\bar{J}(w) = J(w) + \frac{\lambda}{2m}\sum_{j=1}^{d} w_j^2$

$$= -\frac{1}{m}\sum_{i}^{m}[y_i \log(f_{w,b}(x_i)) + (1-y_i)\log(1-f_{w,b}(x_i))] + \frac{\lambda}{2m}\sum_{j=1}^{d} w_j^2$$

the bias parameter w_0b is not regularized

3. solve the objective function by gradient descent:

$$w_j = w_j - \frac{\alpha}{m}[\sum_{i=1}^{m}(f_{w,b}(x_i) - y_i)\cdot x_i(j) + \lambda w_j]$$

$$w_0 \leftarrow w_0 - \frac{\alpha}{m}\sum_{i=1}^{m}(f_{w,b}(x_i) - y_i)\cdot x_i(0)$$

where $w_i(j)$ denotes the j-th entry of x_i, and $j = 0, ..., d$

probabilistic perspective of logistic regression

summary: linear regression vs. logistic regression

Support vector machine

Motivation

1. given training data set $D = \{(x_i, y_i)\}_{i=1}^{m}$, and $x_i \in \mathbb{R}^n$, $y_i \in \{-1, +1\}$
2. we adopt the sign hypothesis function $y = sign(f_w(x)) = sign(w^\top x)$
3. we require that if $y_i = 1$, then $w^\top x_i > 0$; if $y_i = -1$, then $w^\top x_i < 0$
4. we introduce the concept margin: the distance from the closest point of positive and negative classes to the decision boundary
5. the intuition is to choose the decision boundary with large margin, which is called large margin classifier, also called support vector machine.

Derivation I: large margin

1. we need to derive the following problem:

$$\min_{w,b} \frac{1}{2}\|w\|^2$$

$$\text{s.t. } y_i(w^\top x_i + b) \geq 1, \forall i$$

2. we want to maximize the margin over all training data points
3. lemma: x has distance $\frac{|f_w(x)|}{||w||}$ to the hyperplane $f_w(x) = w^\top x = 0$
4. lemma: x has distance $\frac{|f_{w,b}(x)|}{\|w\|}$ to the hyperplane $f_{w,b}(x) = w^\top x + b = 0$
5. the margin over all training data points:

$$\gamma = \min_i \frac{|f_{w,b}(x_i)|}{\|w\|}$$

recall $y_i \in \{1, -1\}$, we have

$$\gamma = \min_i \frac{y_i f_{w,b}(x_i)}{\|w\|}$$

maximize margin over all training data points:

$$\max_{w,b} \gamma = \max_{w,b}\min_i \frac{y_i f_{w,b}(x_i)}{\|w\|} = \max_{w,b}\min_i \frac{y_i(w^\top x_i + b)}{\|w\|}$$

6. when (w, b) scaled by a factor C, the margin unchanged

$$\frac{y_i(cw^\top x_i + cb)}{\|cw\|} = \frac{y_i(w^\top x_i + b)}{\|w\|}$$

consider a fixed scale such that

$$y_{i^*}(w^\top x_{i^*} + b) = 1$$

where x_{i^*} is the point closest to the hyperplane, so we have

$$y_i(w^\top x_i + b) \geq 1$$

and at least for one i the equality holds, then the margin is $\frac{1}{\|w\|}$ so the optimization is what we see in 1.

7. training: solving the above optimization problem is called training or learning of the large margin classifier, and we obtain the solution w^*, b^*
8. prediction: given the solution w^*, b^*, for a new test data x_t, we predict it as $+1$ if $(w^*)^\top x_t + b^* > 0$, otherwise -1

Derivation II: hinge loss

1. recall in logistic regression, the hypothesis function is :

$$f_{w,b}(x) = \frac{1}{1 + \exp(-w\top x)} = g(z)$$

where $z = w^\top x$. the objective function of logistic regression

$$J(w) = -\delta_{y=1}\log(f_{w,b}(x)) - \delta_{y=-1}\log(1 - f_{w,b}(x))$$

where $\delta_a = 1$ if a is true, otherwise 0

2. objective function of regularized logistic regression

$$\frac{1}{m}\sum_{i}^{m}[\delta_{i=1}(-\log(f_{w.b}(x_i))) + \delta_{y_i=-1}(-\log(1 - f_{w,b}(x_i)))] + \frac{\lambda}{2m}\sum_{j=1}^{n}\omega_j^2$$

so we have that the objective function of support vector machine

$$\frac{1}{m}\sum_{i}^{m}[\delta_{y_i=1}cost_1(w^\top x_i + b) + \delta_{y_i=-1}cost_{-1}(w^\top x_i + b)] + \frac{\lambda}{2m}\sum_{j=1}^{n}\omega_j^2$$

$$\equiv C\sum_{i}^{m}[\delta_{y_i=1}cost_1(w^\top x_i + b) + \delta_{y_i=-1}cost_{-1}(w^\top x_i + b)] + \frac{1}{2}\sum_{j=1}^{n}\omega_j^2$$

where $C = \frac{1}{\lambda}$

3. we observe that if $y_i = 1$, we need $w^\top x_i + b \geq 1$: we need $cost_1(w^\top x_i + b) = 0$ if $w^\top x_i + b \geq 1$; if $y_i = -1$, we need $w^\top x_i + b \leq -1$: we need $cost_{-1}(w^\top x_i + b) = 0$ if $w^\top x_i + b \leq -1$. based on that, we can define the hinge loss:
$$\max(0, 1 - y_i(w^\top x_i + b))$$
4. hinge loss is non-smooth, so we transform the objective function of support vector machine to the following:
$$\min_{w,b} \frac{1}{2}\sum_{j=1}^{n} \omega_j^2$$
$$\text{s.t. } w^\top x_i + b \geq 1 \text{ if } y_i = 1; w^\top x_i + b < -1 \text{ if } y_i = -1$$
which is the same as the objective function derived above.

Lagrange duality and KKT conditions (review)

1. given a general minimization problem:
$$\min_{x\in\mathbb{R}^n} f(x)$$
$$\text{subject to } h_i(x) \leq 0, \quad i = 1, ..., m$$
$$\ell_j(x) = 0, \quad j = 1, ..., r$$
the Lagrangian function is
$$L(x, u, v) = f(x) + \sum_{i=1}^{m} u_i h_i(x) + \sum_{j=1}^{r} v_j \ell_j(x)$$
the Lagrangian dual function is
$$g(u, v) = \min_{x\in\mathbb{R}^n} L(x, u, v)$$
the dual problem is
$$\max_{u\in\mathbb{R}^m, v\in\mathbb{R}^r} g(u, v)$$
$$\text{subject to } u \geq 0$$
2. for the above general minimization problem, the KKT conditions are
$$\partial f(x) + \sum_{i=1}^{m} u_i \partial h_i(x) + \sum_{j=1}^{r} v_j \partial \ell_j(x) = 0$$
$$u_i \cdot h_i(x) = 0 \text{ for all } i, j$$
$$h_i(x) \leq 0, \ell_j(x) = 0 \text{ for all } i, j$$
$$u_i \geq 0 \text{ for all } i$$
3. the objective function of support vector machine is
$$\min_{w,b} \frac{1}{2}\|w\|^2$$
$$\text{s.t. } y_i(w^\top x_i + b) \geq 1, \forall i$$
it can be transformed to
$$\min_{w,b} \frac{1}{2}\|w\|^2$$
$$\text{s.t. } 1 - y_i(w^\top x_i + b) \leq 0, \forall i$$
the Lagrangian function is
$$\mathcal{L}(w, b, \alpha) = \frac{1}{2}\|w\|^2 + \sum_{i}^{m} \alpha_i(1 - y_i(w^\top x_i + b))$$
the primal and dual optimal solutions should satisfy KKT conditions
$$\text{stationarity: } \frac{\partial L}{\partial w} = 0 \rightarrow w = \sum_{i}^{m} \alpha_i y_i x_i, \quad \frac{\partial L}{\partial b} = 0 \rightarrow \sum_{i}^{m} \alpha_i y_i = 0$$
$$\text{feasibility: } \alpha_i \geq 0, 1 - y_i(w^\top x_i + b) \leq 0, \forall i$$
$$\text{complementary slackness: } \alpha_i(1 - y_i(w^\top x_i + b)) = 0, \forall i$$

Optimizing SVM by Lagrange duality

1. recall the Lagrangian function is
$$\mathcal{L}(w, b, \alpha) = \frac{1}{2}\|w\|^2 + \sum_{i}^{m} \alpha_i(1 - y_i(w^\top x_i + b))$$
replacing the stationary condition into Lagrangian function:
$$\mathcal{L}(w, b, \alpha) = \frac{1}{2}\|w\|^2 + \sum_{i}^{m} \alpha_i - \sum_{i}^{m} \alpha_i y_i (\sum_{j}^{m} \alpha_j y_j x_j)^\top x_i - \sum_{i}^{m} \alpha_i y_i b$$
$$= \frac{1}{2}\|w\|^2 + \sum_{i}^{m} \alpha_i - \sum_{i,j}^{m} \alpha_i \alpha_j y_i y_j x_i^\top x_j - b\sum_{i}^{m} \alpha_i y_i$$
$$= \sum_{i}^{m} \alpha_i - \frac{1}{2}\|w\|^2 = \sum_{i}^{m} \alpha_i - \frac{1}{2}\sum_{i,j}^{m} \alpha_i \alpha_j y_i y_j x_i^\top x_j$$
the dual problem is
$$\max_{\alpha} \sum_{i}^{m} \alpha_i - \frac{1}{2}\sum_{i,j}^{m} \alpha_i \alpha_j y_i y_j x_i^\top x_j$$
$$\text{s.t. } \sum_{i}^{m} \alpha_i y_i = 0, \alpha_i \geq 0, \forall i$$
it can be solved by any off-the-shelf optimization solver. then we replace the solved α back into the stationary condition, thus we obtain the primal solution w
$$w = \sum_{i}^{m} \alpha_i y_i x_i$$
2. to determine the bias parameter b, for any support vector x_j, $j \in S$, we have
$$y_i(w^\top x_j + b) = 1, \forall j \in S$$
$$\rightarrow y_j(\sum_{i}^{m} \alpha_i y_i x_i^\top + b) = 1, \forall j \in S$$
product y_j for both sides of teh above equation, and utilizing $y_j \cdot y_j = 1$, we have
$$\sum_{i}^{m} \alpha_i y_i x_i^\top x_j + b = y_j, \forall j \in S$$
$$\rightarrow b = \frac{1}{|S|}\sum_{j\in S}(y_j - \sum_{i}^{m} \alpha_i y_i x_i^\top x_j)$$
3. solution interpretation:
   (a) the primal solution w and the dual solution α should also satisfy other KKT conditions
      i. feasibility: $\alpha_i \geq 0, 1 - y_i(w^\top x_i + b) \leq 0, \forall i$

ii. complementary slackness: $\alpha_i(1 - y_i(w^\top x_i + b)) = 0, \forall i$

(b) If $\alpha i = 0$, then it means that x_i doesn't contribute to w, i.e., the SVM classifier

(c) The data points with $\alpha i > 0$ construct the classifier, and they are called support vectors, which locate at the hyperplanes $y_i(w > x_i + b) = 1$. And, we define the support set as $S = \{i|\alpha i > 0\}$ This is why we call it support vector machine.

4. prediction:

   (a) given the optimized parameters $\{\alpha, w, b\}$, given a new data x, its prediction is

$$w^\top x + b = \sum_i^m \alpha_i y_i x_i^\top x + \frac{1}{|S|}\sum_{j\in S}(y_j - \sum_i^m \alpha_i y_i x_i^\top x_j)$$

   (b) if $w^\top x + b > 0$, then the predicted class of x is $+1$, otherwise -1. if and only if $y(w^\top x_b) > 0$, then your prediction is correct. Note that the prediction of new data depends on inner product with existing training data, which is important to derive kernel SVM.

SVM with slack variables

1. in above derivation, we assume that all primal constraints $y_i(w^\top x_i + b) \geq 1, \forall i$ can be satisfied, implying that the training data is separable. however, sometimes samples of different classes are overlapped, i.e., non-separable. consequently, some constraints will be violated, and we can not obtain the feasible solution.

2. we introduce slack variable $\xi_i \geq 0$, we allow some errors for training data, i.e., $y_i(w^\top x_i + b) \geq 1 - \xi_i.\forall i$,rather than $y_i(w^\top x_i + b) \geq 1, \forall i$. we hope that such errors $\xi_i, \forall i$ are small

3. in this case, the SVM is formulated as follows:

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_i^m \xi_i$$

$$\text{s.t. } 1 - \xi_i - y_i(w^\top x_i + b) \leq 0, -\xi_i \leq 0, \forall i$$

   the Lagrangian function is

$$\mathcal{L}(w,b,\xi,\alpha,\mu) = \frac{1}{2}\|w\|^2 + C\sum_i^m \xi_i + \sum_i^m[\alpha_i(1 - \xi - y_i(w^\top x_i + b)) + \mu_i(-\xi_i)], \alpha_i, \mu_i \geq 0, \forall i$$

4. the KKT conditions:

$$\text{stationarity: } \frac{\partial \mathcal{L}}{\partial w} = 0 \to w = \sum_i^m \alpha_i y_i x_i, \quad \frac{\partial \mathcal{L}}{\partial b} = 0 \to \sum_i^m \alpha_i y_i = 0$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = 0 \to \alpha_i = C - \mu_i, \forall i$$

$$\text{feasibility: } \alpha_i \geq 0, 1 - \xi_i - y_i(w^\top x_i + b) \leq, \xi_i \geq 0, \mu_i \geq 0, \forall i$$

$$\text{complementary slackness: } \alpha_i(1 - \xi_i - y_i(w^\top x_i + b)) = 0, \mu_i\xi_i = 0, \forall i$$

5. replacing all stationary conditions into Lagrangian function to eliminate primal variables, we have

$$\mathcal{L}(\alpha,\mu) = \frac{1}{2}\|w\|^2 + \sum_i^m[\alpha_i(1 - y_i(w^\top x_i + b))] + \sum_i^m(C - \alpha_i - \mu_i)\xi_i$$

$$\sum_i^m \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j x_i^\top x_j$$

6. we obtain the following dual problem:

$$\max_{\alpha,\mu} \sum_i^m \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j x_i^\top x_j$$

$$\text{s.t. } \sum_i^m \alpha_i y_i = 0, 0 \leq \alpha_i \leq C, \mu_i \geq 0, \alpha_i = C - \mu_i, \forall i$$

7. utilizing $\alpha_i = C - \mu_i$, we obtain a simpler dual problem:

$$\max_{\alpha} \sum_i^m \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j x_i^\top x_j$$

$$\text{s.t. } \sum_i^m \alpha_i y_i = 0, 0 \leq \alpha_i \leq C, \forall i$$

8. Solution interpretation

   (a) The solution α_i has three cases: $\alpha_i = 0, 0 < \alpha_i < C, \alpha_i = C$

   (b) $\alpha_i = 0$: the corresponding data are correctly classified and doesn't contribute to the classifier, locating outside of the margin $0 < \alpha_i < C$: in this case, $\mu_i > 0$ due to $\alpha_i = C - \mu_i$; Since $\mu_i\xi_i = 0$, then we have $\xi_i = 0$. The corresponding data are correctly classified and contributes to the classifier, locating on the margin

   (c) $\alpha_i = C$: in this case, $\mu_i = 0$; then we have $\xi_i > 0$. The corresponding data contributes to the classifier, locating inside the margin

   (d) If $\xi_i \leq 1$, then the data is still correctly classified, not crossing decision boundary

   (e) If $\xi_i > 1$, then the data is incorrectly classified, crossing decision boundary

9. How to determine the bias parameter b ?

   (a) We define $\mathcal{M} = \{i \mid 0 < \alpha_i < C\}$

   (b) Since $0 < \alpha_i < C$, we have $\xi_i = 0$

   (c) Then, for any support vector $\mathbf{x}_j, j \in \mathcal{M}$, we have

$$y_j\left(\mathbf{w}^\top \mathbf{x}_j + b\right) = 1, \forall j \in \mathcal{M}$$

$$\Rightarrow y_j\left(\sum_i^m \alpha_i y_i \mathbf{x}_i^\top \mathbf{x}_j + b\right) = 1, \forall j \in \mathcal{M}$$

   (d) Utilizing $y_j \cdot y_j = 1$, we have

$$\sum_i^m \alpha_i y_i \mathbf{x}_i^\top \mathbf{x}_j + b = y_j, \forall j \in \mathcal{M}$$

$$\Rightarrow b = \frac{1}{|\mathcal{M}|}\sum_{j\in\mathcal{M}}\left(y_j - \sum_i^m \alpha_i y_i \mathbf{x}_i^\top \mathbf{x}_j\right)$$

   (e) Note that using the average of all support vectors, rather than one single support vector, could make the solution of b more numerically stable.

10. Why do we prefer to optimize the dual problem, rather than directly optimizing the primal problem? There are two main advantages:

   (a) By examining the dual form of the optimization problem, we gained significant insight into the structure of the problem

   (b) The entire algorithm can be written in terms of only inner products between input feature vectors. In the following, we will exploit this property to apply the kernels to classification problem. The resulting algorithm, support vector machines, will be able to efficiently learn in very high dimensional spaces.

SVM with kernels

1. in many real problems, such as image classification, the dimensionality of original features $|x|$ is already very high. Consequently, the dimensionality of high-order polynomial function will be too high, causing high computational cost or overfitting. To tackle this difficulty, we will introduce kernel.
2. we can obtain a non-linear decision boundary with some kernel functions:
$$k(x_i, x_j) = \phi(x_i)^\top \phi(x_j)$$
utilizing this kernel to replacing $x_i^\top x_j$, we have the following dual problem
$$\max_\alpha \sum_i^m \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j k(x_i, x_j)$$
$$\text{s.t.} \sum_i^m \alpha_i y_i = 0, \alpha_i \geq 0, \forall i$$
the solution of b becomes
$$b = \frac{1}{|S|}\sum_{j \in S}(y_j - \sum_i^m \alpha_i y_i k(x_i, x_j))$$
the prediction of new data x becomes
$$w^\top x + b = \sum_i^m \alpha_i y_i k(x_i, x) + b$$
since α is sparse, the above classifier is also called sparse kernel classifier.
3. widely used kernels:
$$\text{polynomial kernel: } k(x, x_i) = (1 + \frac{x^\top x_i}{\sigma^2}^p, p > 0)$$
$$\text{radial basis function kernel: } k(x, x_i) = \exp\{-\frac{||x - x_i||^2}{2\sigma^2}\}$$
$$\text{sigmoidal kernel: } k(x, x_i) = \frac{1}{1 + \exp(-\frac{x^\top x_i + b}{\sigma^2})}$$

Others

decision tree

univariate trees

1. in a univariate tree, in each internal node, the test uses only one of the input dimensions
2. tree induction is the construction of the tree given a training set. we are interested in finding the smallest tree that code the given training set with no error, where tree size is measured as the number of nodes in the tree and the complexity of the decision nodes.
3. procedure of building a tree:
   (a) select an attribute and split the data into its children in a tree, continue splitting with available attributes
   (b) until leaf nodes are only one class remains; or a maximum depth is reached; or a performance metric is achieved
4. impurity: every step we choose the attribute that has the largest reduction of impurity. this attribute is believed to be the best one for the step.
5. if a node is pure, we do not need to split and we add a leaf node labeled with probability 1
6. we care about how to measure the impurity of one node: using
   (a) classification error: $\phi(1, 1-p) = 1 - \max(p, 1-p)$
   (b) entropy: $\phi(p, 1-p) = -p\log_2 p - (1-p)\log_2(1-p)$
   (c) gini index: $\phi(p, 1-p) = 2p(1-p)$
7. for multicategory problem, we have:
   (a) entropy: $\phi(p) = -\sum_{i=1}^K p_i \log_2 p_i$
   (b) gini index: $\phi(p) = \sum_{i=1}^K p_i(1-p_i) = 1 - \sum_{i=1}^K p_i^2$
8. a regression tree is constructed in almost the same manner as a classification tree, except that the impurity measure that is appropriate for classification is replaced by a measure appropriate for regression. we use mse to measure he goodness of a split: for each leave, $MSE = \frac{\sum_{i=1}^n (y_i - \bar{y})^2}{n}$, the total MSE is $\mathcal{S} = \sum_{m \in leaves(Tree)} e_m$
9. pruning is an approach to alleviate overfitting. the general procedure is:
   (a) split training data further into training and validation sets
   (b) grow a deep tree based on training set
   (c) evaluate impact on validation set of pruning each possible node
   (d) greedily remove the node htat most improves validation set accuracy
   (e) repeat the last two steps until further pruning is harmful.
10. parameters of decision trees:
   (a) minimum samples for a node split: prevent overfitting, too high values can lead to under-fitting
   (b) minimum samples for a terminal node: control over-fitting
   (c) maximum attributes to consider for split: higher values can lead to over-fitting
   (d) maximum depth of tree: higher values lead to overfitting
11. advantages of decision tree:
   (a) easy to understand
   (b) useful in data exploration
   (c) less data cleaning required
   (d) data type is not a constraint
   (e) non-parametric method
12. disadvantages
   (a) overfitting: can be solved by setting constraints on model parameters and pruning
   (b) continuous variables: decision tree loses information when it categorizes variables into categories for continuous numerical variables

ensemble model

the idea is constructing many diverse decision trees, then combine their predictions as the final prediction

1. bagging:
   (a) Given a standard training set D of size n, bagging generates m new training sets D_i, each of size n, by sampling from D uniformly and with replacement.
   (b) Fit an overgrown tree to each resampled training data set, and we obtain several diverse decision trees
   (c) since there are many shared samples between two resampled training data sets, Bagging is likely to produce many correlated trees.
2. random forest
   (a) Each time a split is to be performed, the search for the split attribute is limited to a random subset of m of the N attributes
3. Bagging introduces randomness into the data-level and Random forests introduces randomness into both the data-level and attribute level

NN

computational graph

back propagation

Example 0.2. *compute $J = \cos(\sin(x^2) + 3x^2)$ on the forward pass and teh derivative $\frac{dJ}{dx}$ on the backward pass:*
let

$$J = cos(u), u = u_1 + u_2, u1 = \sin(t), u_2 = 3t, t = x^2$$

we have the backward process:

$$\begin{aligned}
&\frac{dJ}{du} + = -\sin(u) \\
&\frac{dJ}{du_1} + = \frac{dJ}{du}\frac{du}{du_1}, \quad \frac{du}{du_1} = 1 \quad \frac{dJ}{du_2} + = \frac{dJ}{du}\frac{du}{du_2}, \quad \frac{du}{du_2} = 1 \\
&\frac{dJ}{dt} + = \frac{dJ}{du_1}\frac{du_1}{dt}, \quad \frac{du_1}{dt} = \cos(t) \\
&\frac{dJ}{dt} + = \frac{dJ}{du_2}\frac{du_2}{dt}, \quad \frac{du_2}{dt} = 3 \\
&\frac{dJ}{dx} + = \frac{dJ}{dt}\frac{dt}{dx}, \quad \frac{dt}{dx} = 2x
\end{aligned}$$

computational cost

CNN

1. convolutional layer:
   assume the input is $W_1 \times H_1 \times C$ convolution layer needs 4 hyperparameters:
   (a) number of filters K
   (b) the filter size F
   (c) the stride S
   (d) the zero padding $P = \frac{F-1}{2}$

   this will produce an output of $W_2 \times H_2 \times K$ where:

   (a) $W_2 = (W_1 - F + 2P)/S + 1$
   (b) $H_2 = (H_1 - F + 2P)/S + 1$
   (c) number of parameters: $F^2CK + K$, K biases

   Example 0.3. *input volume: $32 \times 32 \times 3$ and 10 5×5 filters with stride 1, pad 2.*

   (a) number of parameters in this layer is

   $$10 \times (5 \times 5 \times 3 + 1) = 760 \quad \textit{+1 is for bias}$$

   (b) output volume size:

   $$(32 + 2 \times 2 - 5)/1 + 1 = 32$$

   so the volume size is

   $$32 \times 32 \times 10$$

2. max pooling: using a max pool with 2×2 filters and stride 2, we have

$$\begin{bmatrix} 1 & 1 & 2 & 4 \\ 5 & 6 & 7 & 8 \\ 3 & 2 & 1 & 0 \\ 1 & 2 & 3 & 4 \end{bmatrix} \rightarrow \begin{bmatrix} 6 & 8 \\ 3 & 4 \end{bmatrix}$$

3. fully connected layer:
   (a) shape of activation map: $1 \times 1 \times K$
   (b) number of parameters; $W \times H \times C \times K + K$

bias variance decomposition

over fitting, under fitting and model complexity

1. under-fitting is the inability of the model to predict well the labels of the data it was trained on. reasons for under-fitting can be:
   (a) your model is too simple for the data
   (b) the features you engineered are not informative enough
2. over-fitting predicts very well on the training data but poorly on the testing data. reasons for over-fitting can be:
   (a) your model is too complex for the data
   (b) you have too many features but a small number of training examples

bias-variance trade-off

1. in general, the training error decreases with the model complexity increasing
2. in terms of the testing error, we have the following observations:
   (a) when the model complexity is low, the test prediction error is high, and it has high bias and low variance
   (b) when the model complexity is high, the test prediction error is low, and it has low bias and high variance
   (c) the test prediction error does not decrease after certain point

   the observation is called bias-variance trade-off
3. we are provided by a training data set $D = \{(x_i, y_i)\}_{i=1^n}$, which is drawn i.i.d. from some distribution $P(\mathcal{X}, \mathcal{Y})$
4. the relationship between the input features x and the output y is
$$y = t(x) + e, e \sim \mathcal{N}(0, \sigma^2)$$
$$p(y|x) = \mathcal{N}(t(x), \sigma^2 I)$$
   where $t(x)$ can be seen as the unknown target function and the mean of $p(y|x)$
5. the goal of machine learning is to learn a hypothesis function based on the training dataset D using some learning algorithm $\mathcal{A}$, i.e. $h_D = \mathcal{A}(D)$
6. expected hypothesis function $\bar{h} = E_{D\sim P^n}[h_D] = \int_D h_D p(D) dD$
7. given the test pair $(x, y) \sim P(\mathcal{X}, \mathcal{Y})$ and h_D, the expected test error is defined as
$$E_{(x,y)\sim P}[(h_D(x) - y)^2] = \int_x \int_y (h_D(x) - y)^2 p(x, y) dx dy$$
8. given the test pair $(x, y) \sim P(\mathcal{X}, \mathcal{Y})$, and $\mathcal{A}$, the expected test error is defined as
$$E_{(x,y)\sim P, D\sim P^n}[(h_D(x) - y)^2] = \int_D \int_x \int_y (h_D(x) - y)^2 p(x, y) p(D) dx dy dD$$
9. we are interested in evaluating the quality of a machine learning algorithm $\mathcal{A}$ with respect to a data distribution $P(\mathcal{X}, \mathcal{Y})$. we show that this expression decomposes into three meaningful terms.
10. The expected test error can be decomposed as follows
$$E_{(\mathbf{x},y),D}\left[(h_D(\mathbf{x}) - y)^2\right] = E_{(\mathbf{x},y),D}\left[\left[h_D(\mathbf{x}) - \bar{h}(\mathbf{x})) + (\bar{h}(\mathbf{x}) - y)\right]^2\right]$$
$$= E_{(\mathbf{x},y),D}\left[\left(h_D(\mathbf{x}) - \bar{h}(\mathbf{x})\right)^2\right] + 2E_{(\mathbf{x},y),D}\left[\left(h_D(\mathbf{x}) - \bar{h}(\mathbf{x})\right)\left(\bar{h}(\mathbf{x}) - y\right)\right]$$
$$+ E_{(\mathbf{x},y),D}\left[(\bar{h}(\mathbf{x}) - y)^2\right]$$
11. We have
$$E_{(\mathbf{x},y),D}\left[\left(h_D(\mathbf{x}) - \bar{h}(\mathbf{x})\right)\cdot\left(\bar{h}(\mathbf{x}) - y\right)\right]$$
$$= E_{(\mathbf{x},y)}\left[E_D\left[h_D(\mathbf{x}) - \bar{h}(\mathbf{x})\right]\cdot\left(\bar{h}(\mathbf{x}) - y\right)\right]$$
$$= E_{(\mathbf{x},y)}\left[\left(E_D\left[h_D(\mathbf{x})\right] - \bar{h}(\mathbf{x})\right)\cdot\left(\bar{h}(\mathbf{x}) - y\right)\right] = 0$$
   Replace the above equation to the expected test error, then we have
$$E_{(\mathbf{x},y),D}\left[(h_D(\mathbf{x}) - y)^2\right] = E_{(\mathbf{x},y),D}\left[\left(h_D(\mathbf{x}) - \bar{h}(\mathbf{x})\right)^2\right] + E_{(\mathbf{x},y),D}\left[(\bar{h}(\mathbf{x}) - y)^2\right]$$
   We also have
$$E_{(\mathbf{x},y),D}\left[(\bar{h}(\mathbf{x}) - y)^2\right] = E_{(\mathbf{x},y),D}\left[[(\bar{h}(\mathbf{x}) - t(\mathbf{x})) + (t(\mathbf{x}) - y)]^2\right]$$
$$= E_{(\mathbf{x},y)}\left[(t(\mathbf{x}) - y)^2\right] + E_{(\mathbf{x},y)}[\bar{h}(\mathbf{x}) - t(\mathbf{x}))^2] + 2E_{(\mathbf{x},y)}[(t(\mathbf{x}) - y)(\bar{h}(\mathbf{x}) - t(\mathbf{x}))]$$
$$= E_{(\mathbf{x},y)}\left[(t(\mathbf{x}) - y)^2\right] + E_{(\mathbf{x},y)}[\bar{h}(\mathbf{x}) - t(\mathbf{x}))^2],$$
   where we utilize
$$E_{(\mathbf{x},y)}[(t(\mathbf{x}) - y)(\bar{h}(\mathbf{x}) - t(\mathbf{x}))] = E_{\mathbf{x}}\left[E_{y|\mathbf{x}}[(t(\mathbf{x}) - y)(\bar{h}(\mathbf{x}) - t(\mathbf{x}))]\right]$$
$$= E_{\mathbf{x}}\left[\left(t(\mathbf{x}) - E_{y|\mathbf{x}}(y)\right)\left(\bar{h}(\mathbf{x}) - t(\mathbf{x})\right)\right]$$
$$= E_{\mathbf{x}}[(t(\mathbf{x}) - t(\mathbf{x}))(\bar{h}(\mathbf{x}) - t(\mathbf{x}))] = 0$$
   Finally, we have
$$E_{(\mathbf{x},y),D}\left[(h_D(\mathbf{x}) - y)^2\right]$$
$$= E_{(\mathbf{x},y),D}\left[\left(h_D(\mathbf{x}) - \bar{h}(\mathbf{x})\right)^2\right] + E_{(\mathbf{x},y)}\left[(\bar{h}(\mathbf{x}) - t(\mathbf{x}))^2\right] + E_{(\mathbf{x},y)}\left[(t(\mathbf{x}) - y)^2\right]$$
   Above three terms are variance, bias ^{2}, noise, respectively.
12. variance: captures how much your classifier changes if you train on a different training set. variance is due to your fixed training set.
13. *bias*^{2}: your classifier being biased to a particular kind of solution. bias is inherent to your model.
14. noise: this error measures ambiguity due to your data distribution and feature representation.
15. when the model complexity increases, the variance increases and the bias decreases. the total test error will firstly decrease then increase along with the increase of model complexity.

performance evaluation

1. After learning, we certainly want to know about how the algorithm/the learned model performs.
2. Essentially, we want to know about the accuracy of the algorithm in terms of predicting novel data based on the limited data we have.
3. To answer this question, we need a way to evaluate the algorithm using the limited data we have.
4. we have introduced the bias-variance trade-off, which is related to overfitting/underfitting:
5. When the model complexity is low: the test prediction error is high, with high bias and low variance, corresponding to underfitting;
6. When the model complexity is high: the test prediction error is high, with low bias and high variance, corresponding to overfitting.
7. How to find a model with suitable complexity? This is called model selection problem.

cross-validation

1. split the train data into K folds
2. try each fold as validation, while other folds as train
3. train the model on the train folds, and evaluate the performance on the validation fold
4. calculate the average results on all validation folds across all trials, pick the hyper-parameters with the best average result

this is called K-fold cross validation. it is a widely used method to tune the hyper-parameters.

evaluation metrics for regression

after the evaluation protocol setup, we need an evaluation metric to provide a measure of the accuracy. for regression, the main metrics for evaluation are the MSE and the MAE

1. MSE $= \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}$
2. (mean absolute error) MAE $= \frac{\sum i{=}1^n |y_i - \hat{y}_i|}{n}$

where y_i denotes the target output and $\hat{y}_i$ denotes the predicted output for sample i

evaluation metrics for classification

1. a confusion matrix consists of 4 entries:
   (a) true positive (TP): a positive sample is classified as positive
   (b) true negative (TN): a negative is classified as negative sample
   (c) false Negative (FN): a positive sample is classified as negative
   (d) false positive: a negative sample is classified as positive
2. recall = TP/(TP+FN)
3. Precision = TP/(TP+FP)
4. Accuracy = (TP+TN)/(TP+TN+FP+FN): the total number of correctly classified samples over all samples under evaluation
5. (True Positive Rate) TPR = TP/(TP+FN)
6. (False Negative Rate) FNR = FN/(TP+FN)
7. (True Negative Rate) TNR = TN/(FP+TN)
8. (False Positive Rate) FPR = FP/(FP+TN)
9. we have: TPR + FNR = 1 and TNR + FPR = 1 and Accuracy = $\frac{TPR+TNR}{2} = 1 - \frac{FPR+FNR}{2}$
10. equal error rate (EER) is the point where the curves FPR and FNR intersect at, i.e. FPR = FNR

next we introduce ROC curve and AUC

1. when TPR-curve is plotted as y-axis and the FPR-curve is plotted as x-axis, the plot is called the ROC curve (receiver operating characteristic curve). the higher the ROC (bending to the top-left corner), the better is the classification accuracy
2. the area under the ROC (AUC) can also be used to gauge the classifier's accuracy
3. AUC ranges in value from 0 (100% wrong prediction) to 1 (100% correct prediction). it has the following properties:
   (a) scale-invariant: changing the rangee of the output will not change the AUC score
   (b) classification-threshold-invariant: given any threshold, AUC score will not be changed.

unsupervised learning

kernel density estimation

K-means

basic K-means clustering

1. first you choose K, the number of clusters. then you randomly put K feature vectors, called centroids, to the feature space.
2. next, compute the distance from each example x to each centroid c using some metric, like the Euclidean distance. then we assign the closest centroid to each example
3. for each centroid, we calculate the average feature vector of the examples labeled with it. these average feature vectors become the new locations of the centroids.
4. we recompute the distance from each example to each centroid, modify the assignment and repeat the procedure until the assignments do not change after the centroid locations are recomputed.
5. finally we conclude the clustering with a list of assignments of centroids IDs to the examples.

optimization perspective of K-means clustering

1. given the data set $\{x_i\}_{i=1}^n$, K-means aims to find cluster centers $c = \{c_j\}_{j=1}^K$ and assignments r, by minimizing the sum of squared distances of data points to their assigned cluster centers. K-means will minimize the within-cluster variance as follows:
$$\min_{c,r} J(c,r) = \min_{c,r} \sum_i^n \sum_k^K r_{ik}(x_i - c_k)^2$$
$$s.t. r \in \{0,1\}^{n\times K}, \sum_k^K = 1$$
where $r_{ik} = 1$ denotes x_i is assigned to cluster k.
Remark 0.4. *this is not a convex program since the conditions are not linear (in)equalities, so the optimization may be stuck in local minimum.*
2. the procedure of optimization:
   (a) initialization: set K cluster centers c to random values
   (b) assignment: given the cluster centers c, update the assignments r by solving the following sub-problem
   $$\min_r \sum_i^n \sum_k^K r_{ik}(x_i - c_k)^2, \text{ s.t. } r \in \{0,1\}^{n\times K}, \sum_k^K r_{ik} = 1$$
   Note that the assignment for each data x_i can be solved independently:
   $$\min_{r_i} \sum_k^K r_{ik}(x_i - c_k)^2, \text{ s.t. } r_i \in \{0,1\}^{1\times K}, \sum_i^K r_{ik} = 1$$
   the solution is
   $$k^* = argmin\{(x_i - c_k)^2\}_{k=1}^K, r_{ik}^* = 1$$
   we assign x_i to the closest cluster, exactly same with the assignment step in basic K-means algorithm.
   (c) refitting: given $\{r_{ik}\}$ we can update the cluster centers c by taking derivative w.r.t. c_k as 0:
   $$\min_{c_k} \sum_i^n r_{ik}(x_i - c_k)^2 \rightarrow \sum_i^n 2r_{ik}(x_i - c_k) = 0$$
   $$c_k = \frac{\sum_i^n r_{ik}x_i}{\sum_i^n r_{ik}}$$
   (d) repeat assignment and refitting until convergence
3. the convergence of K-means is guaranteed since whenever an assignment is changed, the sum squared distances J of data points from their assigned cluster centers is reduced. but the objective function J is non-convex, so the coordinate descent on J is not guaranteed to converge to the global minimum. what we could do is running K-means with multiple random initializations and picking the one with the lowest objective value as the final clustering result.

Fuzzy C-mean clustering

1. each data point can belong to more than one cluster
2. the objective function is formulated as follows:

$$\min_{c,r} J(c,r) = \min_{c,r} \sum_i^n \sum_k^K (r_{ik})^m (x_i - c_k)^2$$

$$s.t. r_{ik} \in [0,1]^{n\times K}, \sum_k^K r_{ik} = 1$$

where r_{ik} denotes the degree to which a sample x_i is assigned to cluster c_k.

3. the hyper-parameter $m > 1$ is called fuzzifier, and it defines the level of cluster fuzziness. a value of m close to 1 gives a cluster solution which becomes increasingly similar to the solution of hard clustering such as k-means; whereas a value of m close to infinite leads to complete fuzziness.
4. we still use coordinate descent algorithm to update r and c:
5. sub-problem of r: the constraints $r \in [0,1]^{n\times K}, \sum_k^K r_{ik}$ are equivalent to $r \geq 0, \sum_k^K r_{ik} = 1$, the lagrangian function is

$$\mathcal{L}(r,\alpha,\beta) = J(r) + \sum_i^n \alpha_i(1 - \sum_k^K r_{ik}) + \sum_i^n \sum_k^K \beta_{ik}(-r_{ik})$$

the KKT conditions are

$$\text{stationary: } \frac{\partial \mathcal{L}}{\partial r_{ik}} = 0 \to r_{ik} = (\frac{\alpha_i + \beta_{ik}}{m})^{\frac{1}{m-1}} \cdot (\frac{1}{d_{ik}^2})^{\frac{1}{m-1}}$$

$$\text{primal feasibility: } \sum_k^K r_{ik} = 1, r_{ik} \geq 0$$

$$\text{dual feasibility: } \beta_{ik} \geq 0, \forall i,k$$

$$\text{complementary: } \beta_{ik} \cdot r_{ik} = 0, \forall i,k$$

where $d_{ik}^2 = (x_i - c_k)^2$
if $\alpha_i + \beta_{ik} = 0$, then $r_{ik} = 0$, which violates the primal feasibility constraint $\sum_k^K r_{ik} = 1$, so we have $\alpha_i + \beta_{ik} \neq 0$, then $r_{ik} > 0$ and $\beta_{ik} = 0$, then

$$r_{ik} = (\frac{\alpha_i}{m})^{\frac{1}{m-1}} \cdot (\frac{1}{d_{ik}^2})^{\frac{1}{m-1}}$$

combine it with primal feasibility, we have

$$(\frac{\alpha_i}{m})^{\frac{1}{m-1}} = \frac{1}{\sum_k^K (\frac{1}{d_{ik}^2})^{\frac{1}{m-1}}}$$

$$r_{ik} = \frac{1}{\sum_j^K (\frac{1}{d_{ij}^2})^{\frac{1}{m-1}}} \cdot (\frac{1}{d_{ik}^2})^{\frac{1}{m-1}} = \frac{1}{\sum_j^K (\frac{d_{ik}^2}{d_{ij}^2})^{\frac{1}{m-1}}}$$

we see the effect of m than if $m \to 1$, then r_i is close to one-hot vector like hard assignment; if $m \to \infty$, r_i is close to uniform vector.

6. sub-problem for c: given r, the centroid c is updated by optimizing the following sub-problem:

$$\min_c J(c) = \min_c \sum_i^n \sum_k^K (r_{ik})^m (x_i - c_k)^2$$

by setting the derivative to 0, we obtain

$$\frac{\partial J(c)}{\partial c_k} = 0 \to c = \frac{\sum_i^n [(r_{ik})^m x_i]}{\sum_i^n (r_{ik})^m}$$

constrained K-means clustering

1. in practice, we have additional constraints like must-link constraints and cannot-link constraints, to deal with these conditions, we add a violate-constraints check to ensure that all constraints are satisfied. for each point, it is firstly assigned to the closest cluster, and check all constraints it involves: if all constraints are satisfied, then this assignment is accepted; if any constraint is violated, then assign it to the next closest cluster and repeat the violate-constraints check; if no legal cluster can be found, then the whole clustering fails.

accelerated K-means Clustering

1. For the assignment stage, you have to compute the distance between every pair of data and cluster center, i.e., $(\mathbf{x}_i, \mathbf{c}_k)$. Thus, the cost is $O(n \times K \times d)$, with n being the number of points, K being the number of clusters, and d being the feature dimension

2. For the center calculation stage, you have to calculate the center of every cluster, then the cost is $O\left(\sum_{i=1}^K n_i \times d\right) = O(n \times d)$

3. The total cost is $O(T \times (n \times (K+1) \times d))$, with T being the number of iterations.

4. For large scale and high dimensional data, the cost is high.

5. with triangle inequality we have two lemmas: 1: if $d(b,c) \geq 2d(x,b)$, then $d(x,c) > d(x,b)$;
2: $d(x,c) > \max\{0, d(x,b) - d(b,c)\}$

6. usage of lemma 1: let x be any data point, and c be the center to which x is currently assigned, and let c' be any other center, if $d(c,c') \geq 2d(x,c)$, then $d(x,c') \geq d(x,c)$. in this case, it is unnecessary to compute $d(x,c')$, leading to the cost reduction.

7. usage of lemma 2: we can define lower bound and upper bound of distance between a data point x and the center at the t iteration b_t.

8. lower bound: Let x be any data point, b^{t+1} be any center at the $t+1$ iteration, and b^t be the previous version of the same center. For example, suppose the centers are numbered 1 through k, and b^{t+1} is the center number j, then b^t is the center number j in the previous iteration. Suppose that in the previous iteration t, we knew a lower bound $l\left(x, b^t\right)$ such that $d\left(x, b^t\right) \geq l\left(x, b^t\right)$, then we can update a new lower bound $l\left(x, b^{t+1}\right)$ for the current iteration $t+1$. we have: $d(x, b^{t+1}) \geq \max\left\{0, d\left(x, b^t\right) - d\left(b^{t+1}, b^t\right)\right\} \geq \max\left\{0, l\left(x, b^t\right) - d\left(b^{t+1}, b^t\right)\right\} = l\left(x, b^{t+1}\right)$

9. Upper bound: Suppose $u(x) \geq d\left(x, c^t\right)$ is an upper bound on the distance between x and its currently assigned centroid c^t (the centroid at iteration t). And, suppose $l\left(x, (c')^t\right) \leq d\left(x, (c')^t\right)$ is a lower bound on the distance between x and some other center $(c')^t$. If $u(x) \leq l\left(x, (c')^t\right)$, then $d\left(x, c^t\right) \leq u(x) \leq l\left(x, (c')^t\right) \leq d\left(x, (c')^t\right)$. Thus, it is necessary to calculate neither $d\left(x, c^t\right)$ nor $d\left(x, (c')^t\right)$, leading to cost reduction. Note that it will never be necessary to calculate $d\left(x, (c')^t\right)$ in the current iteration, but it may be necessary to calculate $d\left(x, c^t\right)$, as $u(x) \leq l\left(x, (c'')^t\right)$ may not true for some other center c''.
Update the upper bound $u(x) : u(x) = u(x) + d\left(c^t, c^{t+1}\right) > d\left(x, c^{t+1}\right)$

performance evaluation of clustering

1. internal evaluation metrics: Silhouette coefficient: Given a clustering, we define
   (a) a : The mean distance between a point and all other points in the same cluster.
   (b) b : The mean distance between a point and all other points in the next nearest cluster.
   (c) Silhouette coefficient s for a single sample is formulated as:
   $$s = \frac{b-a}{\max(a,b)} \Rightarrow s = \begin{cases} 1-\frac{a}{b} & \text{if } a<b \\ 0 & \text{if } a=b \\ \frac{b}{a}-1 & \text{if } a>b \end{cases}$$
   (d) It is easy to know that $s \in (-1,1)$, and larger s value indicates better clustering performance.
   (e) Silhouette coefficient s for a set of samples is defined as the mean of the Silhouette Coefficient for each sample.
2. external evaluation metrics: Rand index
   Given a set of n samples $S = \{o_1, o_2, \ldots, o_n\}$, there are two clusterings/partitions of S to compare, including:
   (a) $X = \{X_1, X_2, \ldots, X_r\}$ with r clusters
   (b) $Y = \{Y_1, Y_2, \ldots, Y_s\}$ with s clusters

   We can calculate the following values:
   (a) a: The number of pairs of elements in S that are in the same subset in X and in the same subset in Y
   (b) b: The number of pairs of elements in S that are in the different subset in X and in the different subset in Y
   (c) c : The number of pairs of elements in S that are in the same subset in X and in the different subset in Y
   (d) d : The number of pairs of elements in S that are in the different subset in X and in the same subset in Y

   The rand index (RI) can be computed as follows:
   $$\text{RI} = \frac{a+b}{a+b+c+d} = \frac{a+b}{\frac{n(n-1)}{2}}$$
   Note that RI $\in [0,1]$, and higher score corresponds higher similarity.

Gaussian mixture model

we model the joint distribution over (x,z) as follows
$$p(x,z) = p(x|z)p(z)$$
where x denotes a feature variable, and z denotes the class label variable. we can model the marginal distribution over x as follows:
$$p(x) = \sum_z p(x,z) = \sum_z p(x|z)p(z)$$
this is called as mixture models.
a GMM represents a distribution as
$$p(x) = \mathbf{\Sigma}_{k=1}^{k} \pi_k \mathcal{N}(x|\mu_k, \mathbf{\Sigma}_k)$$
recall the Gaussian distribution is
$$\mathcal{N}(x|\mu_k, \mathbf{\Sigma}_k) = \frac{1}{\sqrt{(2\pi)^d |\mathbf{\Sigma}_k|}} \exp\left(-\frac{1}{2}(x-\mu_k)^\top \mathbf{\Sigma}_k^{-1}(x-\mu_k)\right)$$
with $|\mathbf{\Sigma}_k| = \det(\mathbf{\Sigma}_k)$ denotes the determinant of $\mathbf{\Sigma}_k$, d indicates the dimension of x.
The log likelihood is
$$\log \mathcal{L}(\mathbf{\Theta}) = \ln p(\mathbf{X} \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \mathbf{\Sigma}) = \sum_{n=1}^{N} \ln \left(\sum_{k=1}^{K} \pi_k \mathcal{N}\left(\mathbf{x}^{(n)} \mid \boldsymbol{\mu}_k, \mathbf{\Sigma}_k\right) \right)$$
where $\mathbf{X} = \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\right\}, \mathbf{\Theta} = \{\boldsymbol{\pi}, \boldsymbol{\mu}, \mathbf{\Sigma}\}, \boldsymbol{\pi} = \{\pi_1, \ldots, \pi_K\}, \boldsymbol{\mu} = \{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K\}, \mathbf{\Sigma} = \{\mathbf{\Sigma}_1, \ldots, \mathbf{\Sigma}_K\}$

1. We aim to learn the parameters Θ by maximizing the above log likelihood.
2. Due to the log-sum-exp operation, we cannot obtain a closed-form solution by setting the derivative to zero.

Latent variable

we introduce a hidden latent variable z, indicating which Gaussian component generates the observation x, with some probability. let $z \sim$Categorical(π), where $\pi > 0$, $\sum_k \pi_k = 1$. A latent variable model is a statistical model that relates a set of observable variables to a set of latent variables. variables which are always unobserved are called latent variables.

GMM

1. a gaussian mixture distribution:
$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \mathbf{\Sigma}_k)$$
2. we had: $z \sim$ Categorical(π): $p(z=k|\pi) = \pi_k$ where $\pi \geq 0$, $\sum_k \pi_k = 1$
3. the log-likelihood:
$$\begin{aligned} \ell(\boldsymbol{\pi}, \boldsymbol{\mu}, \mathbf{\Sigma}) = \ln p(\mathbf{X} \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \mathbf{\Sigma}) &= \sum_{n=1}^{N} \ln p\left(\mathbf{x}^{(n)} \mid \pi, \boldsymbol{\mu}, \mathbf{\Sigma}\right) \\ &= \sum_{n=1}^{N} \ln \sum_{k=1}^{K} p\left(\mathbf{x}^{(n)}, z^{(n)} = k \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \mathbf{\Sigma}\right) \\ &= \sum_{n=1}^{N} \ln \sum_{k=1}^{K} p\left(\mathbf{x}^{(n)} \mid z^{(n)} = k; \boldsymbol{\mu}, \mathbf{\Sigma}\right) p\left(z^{(n)} = k \mid \boldsymbol{\pi}\right) \end{aligned}$$
   with the constraint $1 - \sum_{k=1}^{K} \pi_k = 0$
4. the lagrangian function is
$$\mathcal{L}(\pi, \mu, \Sigma, \lambda) = -\ell(\pi, \mu, \Sigma) + \lambda(1 - \sum_{k=1}^{K} \pi_k)$$

5. using the KKT condition:

$$\frac{\partial \mathcal{L}(\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Sigma},\lambda)}{\partial \boldsymbol{\mu}_k} = \frac{-\partial \sum_{n=1}^N 1_{[z^{(n)}=k]} \ln p\left(\mathbf{x}^{(n)};\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\right)}{\partial \boldsymbol{\mu}_k} = \mathbf{0},$$
$$\frac{\partial \mathcal{L}(\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Sigma},\lambda)}{\partial \boldsymbol{\Sigma}_k} = \frac{-\partial \sum_{n=1}^N 1_{[z^{(n)}=k]} \ln p\left(\mathbf{x}^{(n)};\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\right)}{\partial \boldsymbol{\Sigma}_k} = \mathbf{0},$$
$$\frac{\partial \mathcal{L}(\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Sigma},\lambda)}{\partial \pi_k} = \frac{\partial \sum_{n=1}^N 1_{[z^{(n)}=k]} \ln \pi_k}{\partial \pi_k} - \lambda = 0,$$
$$1 - \sum_{k=1}^K \pi_k = 0.$$

6. solve the system, we get the solution:

$$\boldsymbol{\mu}_k = \frac{\sum_{n=1}^N 1_{[z^{(n)}=k]}\mathbf{x}^{(n)}}{\sum_{n=1}^N 1_{[z^{(n)}=k]}}$$
$$\boldsymbol{\Sigma}_k = \frac{\sum_{n=1}^N 1_{[z^{(n)}=k]}\left(\mathbf{x}^{(n)}-\boldsymbol{\mu}_k\right)\left(\mathbf{x}^{(n)}-\boldsymbol{\mu}_k\right)^T}{\sum_{n=1}^N 1_{[z^{(n)}=k]}}$$
$$\pi_k = \frac{1}{N}\sum_{n=1}^N 1_{[z^{(n)}=k]}$$

EM algorithm

1. E-step: compute the posterior probability over z given the current model: $p(z|x,\Theta)$, which tells how much do we think each Gaussian generates each data point
2. M-step: assuming that the data was really generated this way, update the parameters of each Gaussian component to maximize the probability that it would generate the data it is currently responsible for.
3. Summary of EM: Initialize the means $\boldsymbol{\mu}_k$, covariances $\boldsymbol{\Sigma}_k$ and mixing coefficients π_k Iterate until convergence:

   (a) E-step: Evaluate the responsibilities given current parameters

$$\gamma_k^{(n)} = p\left(z^{(n)}=k \mid \mathbf{x}^{(n)};\Theta\right) = \frac{\pi_k \mathcal{N}\left(\mathbf{x}^{(n)} \mid \boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\right)}{\sum_{j=1}^K \pi_j \mathcal{N}\left(\mathbf{x}^{(n)} \mid \boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j\right)}.$$

   (b) M-step: compute the expected log likelihood:

$$\begin{aligned}
&\sum_n \mathbb{E}_{P(z^{(n)}|\mathbf{x}^{(n)})}\left[\ln\left(P\left(\mathbf{x}^{(n)},z^{(n)} \mid \Theta\right)\right)\right]\\
=&\sum_n\sum_k \gamma_k^{(n)}\left(\ln\left(P\left(z^{(n)}=k \mid \Theta\right)\right)\right.\\
&\left.+\ln\left(P\left(\mathbf{x}^{(n)} \mid z^{(n)}=k,\Theta\right)\right)\right)\\
=&\sum_n\sum_k \gamma_k^{(n)}\left(\ln(\pi_k)+\ln\left(\mathcal{N}\left(\mathbf{x}^{(n)} \mid \boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\right)\right)\right)\\
=&\sum_n\sum_k \gamma_k^{(n)}\ln(\pi_k)+\sum_n\sum_k \gamma_k^{(n)}\ln\left(\mathcal{N}\left(\mathbf{x}^{(n)} \mid \boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\right)\right),
\end{aligned}$$

Given the posterior probability $\gamma_k^{(n)} = p\left(z^{(n)}=k \mid \mathbf{x}^{(n)}\right)$, we want to update the model parameters $\Theta = \{\boldsymbol{\mu},\boldsymbol{\Sigma},\boldsymbol{\pi}\}$ by maximizing the expected log likelihood, i.e.,

$$\max_\Theta \sum_n^N\sum_k^K \gamma_k^{(n)}\ln(\pi_k)+\sum_n^N\sum_k^K \gamma_k^{(n)}\ln\left(\mathcal{N}\left(\mathbf{x}^{(n)} \mid \boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\right)\right)$$
$$\text{s.t. } \sum_k^K \pi_k = 1.$$

Re-estimate the parameters given current responsibilities

$$\boldsymbol{\mu}_k = \frac{1}{N_k}\sum_{n=1}^N \gamma_k^{(n)}\mathbf{x}^{(n)}$$
$$\boldsymbol{\Sigma}_k = \frac{1}{N_k}\sum_{n=1}^N \gamma_k^{(n)}\left(\mathbf{x}^{(n)}-\boldsymbol{\mu}_k\right)\left(\mathbf{x}^{(n)}-\boldsymbol{\mu}_k\right)^\top,$$
$$\pi_k = \frac{N_k}{N}, \text{ with } N_k = \sum_{n=1}^N \gamma_k^{(n)}.$$

   (c) Evaluate log likelihood and check for convergence

$$\ln p(\mathbf{X} \mid \boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Sigma}) = \sum_{n=1}^N \ln\left(\sum_{k=1}^K \pi_k \mathcal{N}\left(\mathbf{x}^{(n)} \mid \boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\right)\right)$$

Expectation maximization

recall

1. we use EM algorithm as a way of fitting a Gaussian mixture model for clustering
2. next we derive EM from principled approach and see how EM can be applied to general latent variable models
3. why do we introduce latent variables? we can build a complex model out of simple parts, which can simplify the description of the model. and we can sometimes use the latent variables as a representation of the original data.

preliminaries: Jensen's inequality

Theorem 0.5 (Jensen's inequality)**.** *suppose f is a convex function, and X is a random variable, then we have*

$$f(E(x) \leq E[f(X)]$$

if f is a concave function, we have

$$f(E[X] \geq E[f(X)]$$

Proof. □

EM for Latent variable models

1. using x to denote observed data and z to denote the latent variables, we assume we have an observed dataset $D = \{x^{(n)}\}_{n=1}^{N}$ and would like to fit θ using maximum log likelihood:
$$\log p(D;\theta) = \sum_{n=1}^{N} \log p(x^{(n)};\theta)$$
to compute $p(x,\theta)$, we have to marginalize over z:
$$p(x;\theta) = \sum_{z} p(z,x;\theta)$$
where $p(z,x;\theta)$ denotes the probabilistic model we should define.
2. there is no closed form solution to the maximum likelihood problem
$$\log p(D;\theta) = \sum_{n=1}^{N} \log p(x^{(n)};\theta) = \sum_{n=1}^{N} \log\left(\sum_{z^{(n)}} p(z^{(n)},x^{(n)};\theta)\right)$$
we introduce a new distribution w.r.t. each latent variable $z^{(n)}$, denoted as $q_n(z^{(n)})$, we assume that the distributions w.r.t. different latent variables could be different, and they are independent, i.e.
$$q(z) = \prod_{n=1}^{N} q_n(z^{(n)}$$
note that here we do not specify the parameter value of $q_n(z^{(n)})$, which will be learned later. and note that
$$q_n(z^{(n)}) \neq p(z;\pi)$$
3.
$$\begin{aligned}\ln p(x;\theta) &= E_{q(z)}\left[\ln\left(\frac{p(x;\theta)\cdot q(z)}{q(z)}\right)\right]\\ &= E_{q(z)}\left[\ln\left(\frac{p(x,z;\theta)}{q(z)}\cdot\frac{q(z)}{p(z|x;\theta)}\right)\right]\\ &= E_{q(z)}\left[\ln\left(\frac{p(x,z;\theta)}{q(z)}\right)\right] + E_{q(z)}\left[\ln\left(\frac{q(z)}{p(z|x;\theta)}\right)\right]\end{aligned}$$
we extend the above decomposition to the log likelihood of the whole data set D:
$$\begin{aligned}\ln p(D;\theta) = \sum_{n=1}^{N} E_{q(z)}\left[\ln\left(\frac{p(x^{(n)},z^{(n)};\theta)}{q(z^{(n)})}\right)\right]\\ + \sum_{n=1}^{N} E_{q(z^{(n)})}\left[\ln\left(\frac{q(z^{(n)})}{p(z^{(n)}|x^{(n)};\theta)}\right)\right]\\ = \mathcal{L}(q,\theta) + KL(q(z)||p(z|D;\theta)\end{aligned}$$
4.
Theorem 0.6.
$$\ln p(D;\theta) \geq \mathcal{L}(q;\theta), \quad q,\theta$$
Proof. we provide two proofs for the theorem □
5. we know that learning θ by maximizing $\ln p(D;\theta)$ is difficult, we resort to maximize its lower bound $\mathcal{L}(q;\theta)$ with some auxiliary distribution $q(z)$:
$$\max_{q(z),\theta} \mathcal{L}(q,;\theta) \equiv \max_{q(z),\theta} \sum_{n=1}^{N} E_{q_n(z^{(n)})}\left[\ln\left(\frac{p(x^{(n)},z^{(n)};\theta)}{q_n(z^{(n)})}\right)\right]$$
with constraint $\sum_{z^{(n)}=1}^{K} q_n(z^{(n)}) = 1, \forall n$
6. we adopt the coordinate descent algorithm to solve the above optimization problem, i.e. iteratively update $q(z)$ and θ
7. given θ, update $q(z)$ by solving the following sub-problem:
$$\begin{aligned}\max_{\mathbf{q}(\mathbf{z})} \mathcal{L}(\mathbf{q};\boldsymbol{\theta}) &\equiv \max_{\mathbf{q}(\mathbf{z})} \sum_{n=1}^{N} \mathbb{E}_{q_n(z^{(n)})}\left[\ln\left(\frac{p\left(\mathbf{x}^{(n)},z^{(n)};\boldsymbol{\theta}\right)}{q_n\left(z^{(n)}\right)}\right)\right]\\ &\equiv \max_{\mathbf{q}(\mathbf{z})} \sum_{n=1}^{N} \mathbb{E}_{q_n(z^{(n)})}\left[\ln\left(\frac{p\left(z^{(n)}\mid\mathbf{x}^{(n)};\boldsymbol{\theta}\right)\cdot p\left(\mathbf{x}^{(n)};\boldsymbol{\theta}\right)}{q_n\left(z^{(n)}\right)}\right)\right]\\ &\equiv \max_{\mathbf{q}(\mathbf{z})} \sum_{n=1}^{N} \mathbb{E}_{q_n(z^{(n)})}\left[\ln\left(\frac{p\left(z^{(n)}\mid\mathbf{x}^{(n)};\boldsymbol{\theta}\right)}{q_n\left(z^{(n)}\right)}\right) + \ln p\left(\mathbf{x}^{(n)};\boldsymbol{\theta}\right)\right]\\ &\equiv \max_{\mathbf{q}(\mathbf{z})} \sum_{n=1}^{N} \mathbb{E}_{q_n(z^{(n)})}\left[\ln\left(\frac{p\left(z^{(n)}\mid\mathbf{x}^{(n)};\boldsymbol{\theta}\right)}{q_n\left(z^{(n)}\right)}\right)\right] + \text{ constant}\\ &\equiv \min_{\mathbf{q}(\mathbf{z})} \sum_{n=1}^{N} \mathbb{E}_{q_n(z^{(n)})}\left[\ln\left(\frac{q_n\left(z^{(n)}\right)}{p\left(z^{(n)}\mid\mathbf{x}^{(n)};\boldsymbol{\theta}\right)}\right)\right]\\ &\equiv \min_{\mathbf{q}(\mathbf{z})} \sum_{n=1}^{N} \mathrm{KL}\left(q_n\left(z^{(n)}\right)\|p\left(z^{(n)}\mid\mathbf{x}^{(n)};\boldsymbol{\theta}\right)\right),\end{aligned}$$
with the constraint $\sum_{k=1}^{K} q_n(z^{(n)}=k) = 1, \forall n$. according to the property of KL divergence, it is easy to find the optimal solution, as follows:
$$q_n^*(z^{(n)}) = p(z^{(n)}|x^{(n)};\theta)$$
Remark 0.7. *we note that the optimal auxiliary distribution $q_n^*(z^{(n)})$ is exactly the posterior distribution $p(z^{(n)}|x^{(n)};\theta)$; since $KL(q_n^*(z)||p(z|D;\theta)) = 0$, then*
$$\ln p(D;\theta) = \mathcal{L}(q_n^*;\theta)$$
it means that the gap between $\ln p(D;\theta)$ and its lower bound $\mathcal{L}(q^*;\theta)$ becomes 0, given the current θ
8. Given $\mathbf{q(z)}$, update $\boldsymbol{\theta}$ by solving the following sub-problem:
$$\begin{aligned}\max_{\boldsymbol{\theta}} \mathcal{L}(\mathbf{q};\boldsymbol{\theta}) &\equiv \max_{\boldsymbol{\theta}} \sum_{n=1}^{N} \mathbb{E}_{q_n(z^{(n)})}\left[\ln\left(\frac{p\left(\mathbf{x}^{(n)},z^{(n)};\boldsymbol{\theta}\right)}{q_n\left(z^{(n)}\right)}\right)\right]\\ &\equiv \max_{\boldsymbol{\theta}} \sum_{n=1}^{N} \mathbb{E}_{q_n(z^{(n)})}\left[\log p\left(\mathbf{x}^{(n)},z^{(n)};\boldsymbol{\theta}\right)\right]\\ &\quad \underbrace{-\mathbb{E}_{q_n(z^{(n)})}\left[\log q_n\left(z^{(n)}\right)\right]}_{\text{constant w.r.t. }\boldsymbol{\theta}}\end{aligned}$$
Substitute in $q_n\left(z^{(n)}\right) = p\left(z^{(n)}\mid\mathbf{x}^{(n)};\boldsymbol{\theta}^{\text{old}}\right)$:
$$\boldsymbol{\theta}^{\text{new}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{n=1}^{N} \mathbb{E}_{p\left(z^{(n)}\mid\mathbf{x}^{(n)};\boldsymbol{\theta}^{\text{old}}\right)}\left[\log p\left(z^{(n)},\mathbf{x}^{(n)};\boldsymbol{\theta}\right)\right]$$
This is the expected complete data log-likelihood, which is easy to optimize.
9. we can deduce that an iteration of EM will improve the log-likelihood by using the fact that the bound is tight at θ^{old} after the E-step: let q denote the q_n after the E-step, i.e. $q_n(z^{(n)}) = p(z^{(n)}|x^{(n)};\theta^{old})$, we have
$$\begin{aligned}&\log p\left(\mathcal{D};\boldsymbol{\theta}^{\text{new}}\right) \geq \mathcal{L}\left(q,\boldsymbol{\theta}^{\text{new}}\right) \text{ since } \log p(\mathcal{D};\boldsymbol{\theta}) \geq \mathcal{L}(q,\boldsymbol{\theta})\\ &\geq \mathcal{L}\left(q,\boldsymbol{\theta}^{\text{old}}\right) \text{ since } \boldsymbol{\theta}^{\text{new}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\mathcal{L}(q,\boldsymbol{\theta})\\ &= \log p\left(\mathcal{D};\boldsymbol{\theta}^{\text{old}}\right) \text{ since } \log p\left(\mathcal{D};\boldsymbol{\theta}^{\text{old}}\right) \overset{\theta}{=} \left(q;\boldsymbol{\theta}^{\text{old}}\right)\end{aligned}$$

EM for gaussian mixture models

1. recall our GMM model is

$$p(x;\theta)=\sum_z p(x,z;\theta)=\sum_z p(x|z;\theta)p(z|\theta)$$

$$p(z=k;\theta)=\pi_k,\sum_{k=1}^K \pi_k=1$$
$$p(x|z=k;\theta)=\mathcal{N}(x;\mu_k,\Sigma_k)$$

in this scenario, we have $\theta=\{\pi_k,\mu_k,\sigma_k\}_{k=1}^K$

2. Let the current parameters be $\boldsymbol{\theta}^{\text{old}}=\{\boldsymbol{\pi}_{\boldsymbol{k}}^{\text{old}},\boldsymbol{\mu}_{\boldsymbol{k}}^{\text{old}},\boldsymbol{\Sigma}_{\boldsymbol{k}}^{\text{old}}\}_{k=1}^K$
E-step: For all n, set $q_n\left(z^{(n)}\right)=p\left(z^{(n)}\mid\mathbf{x}^{(n)};\boldsymbol{\theta}^{\text{old}}\right)$, i.e.,

$$\begin{aligned}\gamma_k^{(n)}&:=q_n\left(z^{(n)}=k\right)=p\left(z^{(n)}=k\mid\mathbf{x}^{(n)};\theta^{\text{old}}\right)\\&=\frac{\pi_k^{\text{old}}\mathcal{N}\left(\mathbf{x}^{(n)}\mid\boldsymbol{\mu}_k^{\text{old}},\boldsymbol{\Sigma}_k^{\text{old}}\right)}{\sum_{j=1}^K\pi_j^{\text{old}}\mathcal{N}\left(\mathbf{x}^{(n)}\mid\mu_j^{\text{old}},\Sigma_j^{\text{old}}\right)}\end{aligned}$$

M-step:

$$\boldsymbol{\theta}^{\text{new}}=\underset{\boldsymbol{\theta}}{\operatorname{argmax}}\sum_{n=1}^N\mathbb{E}_{q_n\left(z^{(n)}\right)}\left[\log p\left(z^{(n)},\mathbf{x}^{(n)};\boldsymbol{\theta}\right)\right]$$

s.t. $\sum_{k=1}^K\pi_k=1$.
Substitute in: $\log p\left(z^{(n)},\mathbf{x}^{(n)};\boldsymbol{\theta}\right)=\sum_{k=1}^K 1_{\{z^{(n)}=k\}}\left(\log\pi_k+\log\mathcal{N}\left(\mathbf{x}^{(n)};\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_{\boldsymbol{k}}\right)\right)$
$q_n\left(z^{(n)}\right)=p\left(z^{(n)}\mid\mathbf{x}^{(n)};\boldsymbol{\theta}^{\text{old}}\right)$
We have: $\boldsymbol{\theta}^{\text{new}}$

$$\begin{aligned}&=\underset{\boldsymbol{\theta}}{\operatorname{argmax}}\sum_{n=1}^N\mathbb{E}_{q_n\left(z^{(n)}\right)}\left[\sum_{k=1}^K 1_{\{z^{(n)}=k\}}\left(\log\pi_k+\log\mathcal{N}\left(\mathbf{x}^{(n)};\boldsymbol{\mu}_{\boldsymbol{k}},\boldsymbol{\Sigma}_{\boldsymbol{k}}\right)\right)\right]\\&=\underset{\boldsymbol{\theta}}{\operatorname{argmax}}\sum_{n=1}^N\sum_{k=1}^K\gamma_k^{(n)}\left(\log\pi_k+\log\mathcal{N}\left(\mathbf{x}^{(n)};\boldsymbol{\mu}_{\boldsymbol{k}},\boldsymbol{\Sigma}_{\boldsymbol{k}}\right)\right)\end{aligned}$$

$$\boldsymbol{\theta}^{\text{new}}=\underset{\boldsymbol{\theta}}{\operatorname{argmax}}\sum_{n=1}^N\sum_{k=1}^K\gamma_k^{(n)}\left(\log\pi_k+\log\mathcal{N}\left(\mathbf{x}^{(n)};\boldsymbol{\mu}_{\boldsymbol{k}},\boldsymbol{\Sigma}_{\boldsymbol{k}}\right)\right)$$

Taking derivatives and setting to zero, and utilizing the constraint $\sum_{k=1}^K\pi_k=1$, we get the exactly same updates from last lecture:

$$\boldsymbol{\mu}_{\boldsymbol{k}}=\frac{1}{N_k}\sum_{n=1}^N\gamma_k^{(n)}\mathbf{x}^{(n)}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{k}}=\frac{1}{N_k}\sum_{n=1}^N\gamma_k^{(n)}\left(\mathbf{x}^{(n)}-\boldsymbol{\mu}_{\boldsymbol{k}}\right)\left(\mathbf{x}^{(n)}-\boldsymbol{\mu}_{\boldsymbol{k}}\right)^T$$

$$\pi_k=\frac{N_k}{N}\quad\text{with}\quad N_k=\sum_{n=1}^N\gamma_k^{(n)}$$

3. EM Recap
   (a) A general algorithm for optimizing many latent variable models, such as GMMs, mixture of Bernoulli distribution
   (b) Iteratively computes a lower bound then optimizes it.
   (c) Converges but maybe to a local minima.
   (d) Can use multiple restarts.
   (e) Can initialize from k-means for mixture models.
   (f) Limitation: need to be able to compute $p(z\mid\mathbf{x};\boldsymbol{\theta})$, not possible for more complicated models.

Principal component analysis

preliminary

1. projection onto a subspace:
   (a) given a data set $D=\{x^{(1)},...,x^{(N)}\}$ with $x^{(n)}\in\mathbb{R}^D$ and D being the original dimension. and we define the mean as $\mu=\frac{1}{N}\sum_{n=1}^N x^{(N)}\in\mathbb{R}^D$
   (b) K-dimensional subspace $\mathcal{S}$ is spanned by an orthonormal basis $\{u_k\}_{k=1}^K$ with $u_k\in\mathbb{R}^D$
   (c) $\|u_k\|=1,\forall k;i^\top u_j=0$ if $i\neq j,\forall i,j$
   (d) approximate each data point $x\in\mathbb{R}^D$ as:

$$\tilde{x}=\mu+\text{Proj}_{\mathcal{S}}(x-\mu)=\mu+\sum_{k=1}^K z_k u_k$$

   where $z_k=u_k^\top(x-\mu)$ can be seen as the projection length of $x-\mu$ on the k-th basis u_k
   (e) let $U\in\mathbb{R}^{D\times K}$ be a matrix with columns $\{u_k\}_{k=1}^K$, then we have

$$\tilde{x}=\mu+Uz\in\mathbb{R}^D,\text{ which is called reconstruction of }x$$
$$z=U^\top(x-\mu)\in\mathbb{R}^K,\text{ which is called representation/code of }x$$

2.
Theorem 0.8. *the vector $x-\tilde{x}$ is orthogonal to the subspace $\mathcal{S}$, i.e.*

$$U^\top(X-\tilde{X})=0$$

Proof. utilizing the definition of $\tilde{x}$, we have

$$x-\tilde{x}=x-\mu-Uz$$

then utilizing the definition of z and the orthonormality of U, we have

$$U^\top(x-\tilde{x})=U^\top(X-\mu)-U^\top UZ=z-z=0$$

□

dimensionality reduction

1. dimensionality reduction aims to find a low-dimensional data vector to represent the original high-dimensional data vector.
2. it can be implemented by unsupervised learning method or supervised learning method. Here we only introduce one typical unsupervised dimensionality reduction method, called principal component analysis.
3. there are several usages of dimensionality reduction, such as visualization, alleviate overfitting, reduce the computational cost.

4. inputs of dimensionality reduction: given a dataset $D = \{x^{(1)}, ..., x^{(N)}\} \subset \mathbb{R}^D$, with D being the original dimension.
5. goal: find a K-dimensional $K < D$ subspace $\mathcal{S}$, which consists of K orthogonal basis vectors $\{u_k\}_{k=1}^K$, and $u_i^\top u_j = 0$ for $i \neq j$, while $u_i^\top u_i = 1, \forall i$. when projecting all points in D onto $\mathcal{S}$, it is desired that the structure or property of the original data is well preserved.
6. outputs of dimensionality reduction: the basis vectors $\{u_k\}_{k=1}^K$, and a new representation $\mathcal{D}' = \{z^{(1)}, ..., z^{(N)}\} \subset \mathbb{R}^K$
7. we aim to find a one-dimensional sub-space $\mathcal{S} = \{u_1\} \in \mathbb{R}^2$, such that when projecting each point $x^{(n)}$ onto this subspace, we obtain the corresponding reconstruction $\tilde{x}^{(n)}$ and the representation z

derivations of principal component analysis

1. derivation 1:
   (a) given a data set $D = \{x^{(1)}, ..., x^{(N)}\} \in \mathbb{R}^D$, we want to find a K-dimensional $(K < D)$ subspace $\mathcal{S}$, which consists of K orthonormal basis vectors $\{u_k\}_{k=1}^K$ s.t. the variance of the reconstructions $\tilde{D} = \{\tilde{x}^1, ..., \tilde{x}^N\}$ is maximal, i.e.
   $$\max_{U, U^\top U = I} \frac{1}{N} \sum_{n=1}^N \|\tilde{x}^{(n)} - \tilde{\mu}\|^2$$
   where $\tilde{\mu} = \frac{1}{N}\sum_{n=1}^N \tilde{x}^{(n)}$ denotes the mean of the reconstructions
   (b) utilizing the definitions of $\tilde{x}^{(n)}$ and $z^{(n)}$, it is easy to prove
   $$\tilde{\mu} = \frac{1}{N}\sum_{n=1}^N \tilde{x}^{(n)} = \mu + U(\frac{1}{N}\sum_{n=1}^N z^{(n)})$$
   $$= \mu + \frac{1}{N} U U^\top \sum_{n=1}^N (x^{(n)} - \mu) = \mu$$
   substitute it into the above max program, we have
   $$\max_{U, U^\top U = I} \frac{1}{N} \sum_{n=1}^N \|\tilde{x}^{(n)} - \mu\|^2$$
   $$\equiv \max_{U, U^\top U = I} \frac{1}{N} \sum_{n=1}^N \|U z^{(n)}\|^2 \equiv \max_{U, U^\top U = I} \frac{1}{N} \sum_{n=1}^N \|z^{(n)}\|^2$$
   substitute in the definition $z = U^\top (x - \mu)$, we have
   $$\max_{U, U^\top U = I} \frac{1}{N} \sum_{n=1}^N \|U^\top (x - \mu)\|^2$$
   $$\equiv \max_{U, U^\top U = I} \frac{1}{N} \sum_{n=1}^N Trace(U^\top (X^{(n)}))$$
2. derivation 2: minimal reconstruction error
   (a) given a dataset $D = \{x^{(1), ..., x^{(N)}}\}$, we want to find a K-dimensional subspace $\mathcal{S}$, which consists of K orthonormal basis vectors $\{u_k\}_{k=1}^K$ s.t. the reconstruction loss between x and $\tilde{x}$ is minimized, i.e.
   $$\min_{U, U^\top U = I} \frac{1}{N} \sum_{n=1}^N \|x^{(n)} - \tilde{x}^{(n)}\|^2$$
   (b)

   Theorem 0.9. *two derivations are equivalent:*
   $$\max_{U, U^\top U = I} \frac{1}{N} \sum_{n=1}^N \|x^{(n)}\|^2$$
   Proof. □

principal component analysis algorithm

1. Until now, we have known that PCA aims to solve the following optimization problem,
$$\max_{\mathbf{U}, \mathbf{U}^\top \mathbf{U} = \mathbf{I}} \frac{1}{N} \sum_{n=1}^N \text{Trace}\left(\mathbf{U}^\top \left(\mathbf{x}^{(n)} - \boldsymbol{\mu}\right)\left(\mathbf{x}^{(n)} - \boldsymbol{\mu}\right)^\top \mathbf{U}\right).$$
2. We define the empirical covariance matrix, as follows:
$$\boldsymbol{\Sigma} = \frac{1}{N} \sum_{n=1}^N \left(\mathbf{x}^{(n)} - \boldsymbol{\mu}\right)\left(\mathbf{x}^{(n)} - \boldsymbol{\mu}\right)^\top.$$
3. Then, the above optimization can be reformulated as follows:
$$\max_{\mathbf{U}} \text{Trace}\left(\mathbf{U}^\top \boldsymbol{\Sigma} \mathbf{U}\right) = \sum_{k=1}^K \mathbf{u}_k^\top \boldsymbol{\Sigma} \mathbf{u}_k, \text{ s.t. } \mathbf{U}^\top \mathbf{U} = \mathbf{I}.$$
4. The Lagrangian function is formulated as follows:
$$L\left(\mathbf{U}, \boldsymbol{\Lambda}_K\right) = \text{Trace}\left(\mathbf{U}^\top \boldsymbol{\Sigma} \mathbf{U}\right) + \text{Trace}\left(\boldsymbol{\Lambda}_K^\top \left(\mathbf{I} - \mathbf{U}^\top \mathbf{U}\right)\right),$$
where $\boldsymbol{\Lambda}_K = \text{diag}\left(\left[\hat{\lambda}_1, \ldots, \hat{\lambda}_K\right]\right) \in \mathbb{R}^{K \times K}$.
5. Then, its optimal solution should satisfy
$$\frac{\partial L\left(\mathbf{U}, \boldsymbol{\Lambda}_K\right)}{\partial \mathbf{U}} = 2\boldsymbol{\Sigma}\mathbf{U} - 2\mathbf{U}\boldsymbol{\Lambda}_K = \mathbf{0}$$
$$\Rightarrow \boldsymbol{\Sigma}\mathbf{u}_k = \hat{\lambda}_k \mathbf{u}_k, k = 1, \ldots, K.$$
6. It implies that the optimal primal solution $\mathbf{u}_k$ and the corresponding dual optimal solution $\hat{\lambda}_k$ are one of the eigenvectors and one of the eigenvalues of $\boldsymbol{\Sigma}$, which also satisfy the constraint $\mathbf{U}^\top \mathbf{U} = \mathbf{I}$.
7. Utilizing SVD decomposition, we have
$$\boldsymbol{\Sigma} = \mathbf{Q}\boldsymbol{\Lambda}_D \mathbf{Q}^\top = \sum_{i=1}^D \lambda_i \mathbf{q}_i \mathbf{q}_i^\top$$
where $\mathbf{Q} = [\mathbf{q}_1, \ldots, \mathbf{q}_D] \in \mathbb{R}^{D \times D}$ with $\mathbf{q}_i$ being the eigenvector corresponding to the i-th largest eigenvalue λ_i, and $\boldsymbol{\Lambda}_D = \text{diag}\left([\lambda_1, \ldots, \lambda_D]\right)$ with $\lambda_1 \geq \ldots \geq \lambda_D$.
8. Substitute into the objective function, we have
$$\sum_{k=1}^K \mathbf{u}_k^\top \boldsymbol{\Sigma} \mathbf{u}_k = \sum_{k=1}^K \sum_{i=1}^D \lambda_i \left(\mathbf{u}_k^\top \mathbf{q}_i\right) \cdot \left(\mathbf{q}_i^\top \mathbf{u}_k\right) = \sum_{t \in T} \lambda_t,$$
where we utilize the property of eigenvectors (unit and orthogonal to each other), and $T \subset \{1, \ldots, D\}$ with $|T| = K$ denotes the index subset of K picked eigenvalues.
9. It is obvious that we should pick the top-K eigenvalues. Correspondingly, the first K columns of $\mathbf{Q}$ should be used as the optimal solution to $\mathbf{U}$.
10. One eigenvector corresponds to one of the basis vectors of the subspace obtained by PCA.
11. One eigenvalue correspond to the variance of the projected points on one basis vector (i.e., eigenvector). Larger eigenvalue indicates more information about the original data.

applications