Classification

EE530 Image Processing

Outline

Pattern Recognition
- PCA
- SVM
- Feature Extraction
- Sparse Representation
- Neural Networks

Kahunun-Lowe Transform: PCA

Find the eigenvectors

$$C_{xx} = E\{\mathbf{x}\mathbf{x}^T\} \quad (1)$$

Keep the eigenvectors corresponding to the K largest eigenvalues to form

$$A = \left[\begin{array}{ccc} \mathbf{a}_1 & \mathbf{a}_2 & \cdots \mathbf{a}_K \end{array}\right] \quad (2)$$

The KLT

$x = Ab$ inverse transform
$b = A^H x$ forward transform

Working with Images

1. Collect M images x_i for $i = 1, 2, \cdots, M$, where x_i's are $N \times 1$ vector.
2. Subtract the mean image μ.

$$\Phi_i = x_i - \mu \quad (3)$$

3. Cov matrix

$$C = E\{\Phi\Phi^T\} = \frac{1}{M}\sum_{i=1}^{M} \Phi_i \Phi_i^T \quad (4)$$

Let

$$L = [\Phi_1, \Phi_2, \cdots, \Phi_m] \quad (5)$$

Then

$$C = \frac{1}{M} L L^T \quad (6)$$

4. Eigen analysis of $\frac{1}{M}LL^T$ will give us $a_1, a_2, \cdots, a_N$ and $\lambda_1, \lambda_2, \cdots, \lambda_N$.
5. Pick the largest K to form

$$A = [a_1, a_2, \cdots, a_K] \quad (7)$$

6. KLT

$$b_i = A^T \Phi_i \quad (8)$$
$$\Phi_i = A b_i \quad (9)$$

* Basis can be obtained by reshaping a_i's to the size of the image.

Eigen Analysis

- If the size of LL^T is small enough

```
[V,D] = eig(L*L');
```

- If the size of LL^T is too big

$$LL^T a = \lambda a \quad (10)$$
$$L^T L L^T a = \lambda L^T a \quad (11)$$
$$L^T L e = \lambda e \quad (12)$$

Use

```
[V,D] = eig(L'*L);
```

and find

$$a = Le \quad (13)$$

Eigenface

Computing eigenfaces

1. Obtain face images $I_1, \cdots, I_M$.
2. Represent the image I_i into a vector Γ_i.
3. Compute the mean fector Ψ.
4. Subtract the mean face $\Phi_i = \Gamma_i - \Psi$.
5. Compute the covariance C.
6. Fin K eigenvectors u_i's (corresponding to the K largest eigenvalues.

Representing faces

1. Each face Φ_i is approximated by

$$\hat{\Phi}_i = \sum_{j=1}^{K} w_j^i u_j \tag{14}$$

2. Each face Φ_i is represented by a vector

$$\Omega_i = [w_1^i, \cdots, w_K^i]^\mathsf{T} \tag{15}$$

Face recognition

1. Unknown face image Γ
2. Normalize $\Phi = \Gamma - \Psi$.
3. Unknown face Φ is approximated by

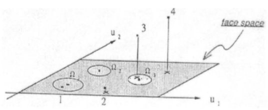
$$\hat{\Phi} = \sum_{j=1}^{K} w_j u_j \tag{16}$$

4. Unknown face Φ is represented by a vector

$$\Omega = [w_1, \cdots, w_K]^\mathsf{T} \tag{17}$$

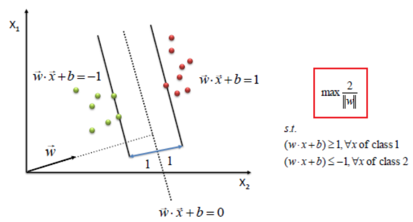
5. Recognition

$$\hat{i} = \arg\min_i \|\Omega - \Omega^i\| \tag{18}$$



Support Vector Machine

Optimal hyperplane for linearly separable patterns



We use this as a decision rule for classification

$$x^T w + b \lesseqgtr 0$$

We give a breathing room such that

$$x^T w + b \geq 1$$
$$x^T w + b \leq -1$$

Define a label y_i such that

$$y_i = \begin{cases} 1 & \text{for } x_+ \text{ samples} \\ -1 & \text{for } x_- \text{ samples} \end{cases}$$

Then we have

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

The width of the breathing room is given by

$$\text{width} = (x_+ - x_-)^T \frac{w}{\|w\|^2}$$

for the corner case x_+ and x_-.

We have $y_i(x_i^T w + b) = 1$. Hence,

$$1(x_+^T w + b) = 1$$
$$-1(x_-^T w + b) = 1$$

The width becomes

$$\text{width} = \frac{2}{\|w\|^2}$$

We want to maximize the with of the breathing room.

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2}\|w\|^2 \\ &\text{subject to} \quad y_i(x_i^T w + b) \geq 1 \quad \text{for all } i \end{aligned}$$

Inequality constrained optimization problem.

We know how to solve the optimization problem.

$$\text{minimize} \quad \frac{1}{2}\|w\|^2 - \sum_i \lambda_i y_i (x_i^\mathsf{T} w + b) + \sum_i \lambda_i \tag{19}$$

for $\lambda_i \geq 0$.

Take partial derivative w.r.t. w, $\quad w = \sum \lambda_i y_i x_i$
Take partial derivative w.r.t. b, $\quad \sum \lambda_i y_i = 0$

Substitute them back to have

$$\text{maximize} \sum_i \lambda_i - \frac{1}{2}\sum_{i,j} \lambda_i \lambda_j y_i y_j x_i^\mathsf{T} x_j \tag{20}$$

subject to $\lambda_i \geq 0$.

Training of SVM

$$\underset{\lambda_i \geq 0}{\text{maximize}} \sum_i \lambda_i - \frac{1}{2}\sum_{i,j} \lambda_i y_i x_i^\mathsf{T} x_j y_j \lambda_j \tag{21}$$

There will be only a few non-zero λ_i's. (Active constraints)
x_i's that have non-zero λ_i are called the support vectors.

Let $l = [\lambda_1, \cdots, \lambda_M]^\mathsf{T}$

$$\text{maximize} \|l\|_1 - \frac{1}{2} l^\mathsf{T} Q l \tag{22}$$

Q involves computation of the inner product $x_i^\mathsf{T} x_j$.

The weight is computed with the support vectors

$$w = \sum_i \lambda_i y_i x_i \tag{23}$$

The decision is

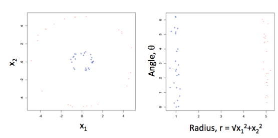
$$x^T w + b > 0$$
$$x^T w + b < 0$$

Or

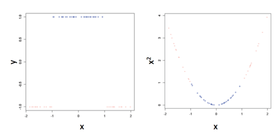
$$\text{sign}(x^T w + b) = \text{sign}(\sum_i \lambda_i y_i x^T x_i + b) \tag{24}$$

Decision involves computation of the inner product $x^T x_i$.

What if samples are not linearly separable?



Use $[r, \theta]^\mathsf{T}$ instead of $[x_1, x_2]$.



Use $[x^2, x]^\mathsf{T}$ instead of x.

We can map features to higher dimension and separate them with a line.

Decision

$$\text{sign}\left(\sum_i \lambda_i y_i K(x, x_i) + b\right) \tag{25}$$

The kernel choices are

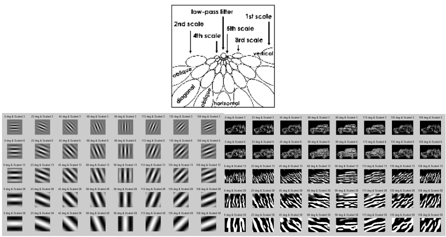
Support Vector Machine	$x^T x_i$
Polynomial Learning Machine	$(x^T x_i + 1)^p$
Radial Base Function	$\exp(-\Vert x - x_i\Vert^2/\sigma^2)$

Recognition

1. PCA to reduce dimension, or extract features
2. SVM to classify into classes

Garbor Transform

Transform in polar coordinate.



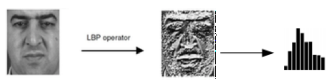
Local Binary Pattern

For sliding blocks,

$$\begin{matrix}5 & 9 & 1\\ 4 & 4 & 6\\ 7 & 2 & 3\end{matrix} \quad \rightarrow \quad \begin{matrix}1 & 1 & 0\\ 1 & & 0\\ 1 & 0 & 0\end{matrix}$$

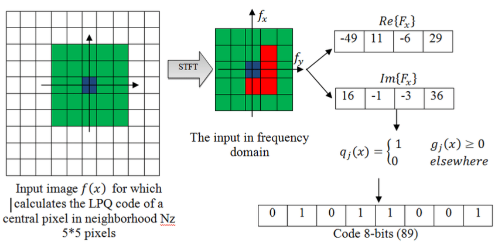
Binary: 11010011
Decimal: 211

Find the histogram of the decimal values $\rightarrow$ feature

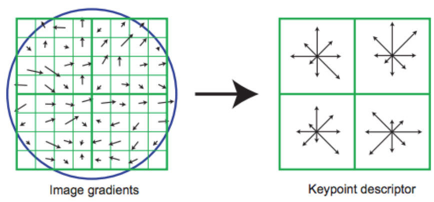


Local Phase Quantization

Phase information is invariant to blur operations.



Keypoint Descriptor



Sparse Representation

Collect all data into a matrix

$$\Phi = [\Phi_1, \Phi_2, \cdots, \Phi_N] \tag{26}$$

We want to represent a query x as

$$x = \Phi u \tag{27}$$

Sove

$$\begin{aligned} \text{minimize} \quad & \|u\|_1 \\ \text{subject to} \quad & x = \Phi u \end{aligned} \tag{28}$$

Check which data in the matrix has larger weight $\rightarrow$ recognition.

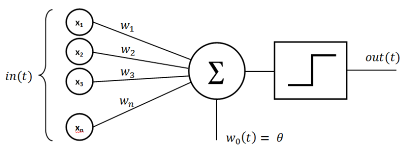
Instead of working with data Φ_i's directly,
use features ϕ extracted by

$$\phi = R\Phi \tag{29}$$

Then

$$\begin{aligned} \text{minimize} \quad & \|u\|_1 \\ \text{subject to} \quad & Rx = R\Phi u \end{aligned} \tag{30}$$

Neural Network



Signle Neuron

Neural Network

$$y = f(\sum_{i=1}^{n} w_i x_i)$$

Training set

$$\{(x_{d1}, y_{d1}), \cdots, (x_{dp}, y_{dp})\}$$

We want to find w such that

$$\text{minimize} \quad \frac{1}{2}\sum_{i=1}^{p}(y_{di} - f(x_{di}^T w))^2$$

Three Layered Neural Network

Network with one hidden layer:

Input: x_i for $i = 1, \cdots, n$
hidden layer: z_j for $j = 1, \cdots, l$
output: y_s for $s = 1, \cdots, m$

$$z_j = f^h\left(\sum_{i=1}^{n} w_{ji}^h x_i\right)$$

$$y_s = f^o\left(\sum_{j=1}^{l} w_{sj}^o z_j\right)$$

Network output

$$y_s = f^o\left(\sum_{j=1}^{l} w_{sj}^o f^h\left(\sum_{i=1}^{n} w_{ji}^h x_{di}\right)\right)$$

Optimization problem

$$\text{minimize} \quad \frac{1}{2}\sum_{s=1}^{m}(y_{ds} - y_s)^2$$

Cost function

$$E(w^h, w^o) = \frac{1}{2}\sum_{s=1}^{m}\left(y_{ds} - f^o\left(\sum_{j=1}^{l} w_{sj}^o f^h\left(\sum_{i=1}^{n} w_{ji}^h x_{di}\right)\right)\right)^2$$

We use gradient descent with a fixed step size η.

$$w_{sj}^{o(k+1)} = w_{sj}^{o(k)} - \eta\frac{\partial}{\partial w_{sj}^o}E(w^h, w^o)$$
$$w_{ji}^{h(k+1)} = w_{ji}^{h(k)} - \eta\frac{\partial}{\partial w_{ji}^h}E(w^h, w^o)$$

Update for w_{sj}^o:
Dummy variables for the sums are replaced by p and q.
The weights are variables only when $p = s$ and $q = j$.

$$E = \frac{1}{2}\sum_{p=1}^{m}\left(y_{dp} - f^o\left(\sum_{q=1}^{l} w_{pq}^o z_q\right)\right)^2$$

$$\begin{aligned}\frac{\partial E}{\partial w_{sj}^o} &= -(y_{ds} - f_s)\dot{f}^o\left(\sum_{q=1}^{l} w_{sq}^o z_q\right) z_j \\ &= -\delta_s z_j\end{aligned}$$

Update for w_{ji}^h:
Dummy variables for the sums are replaced by p, q and r.
The weights are variables only when $q = j$ and $r = i$.

$$E = \frac{1}{2}\sum_{p=1}^{m}\left(y_{dp} - f^o\left(\sum_{q=1}^{l} w_{pq}^o f^h\left(\sum_{r=1}^{n} w_{qr}^h x_r\right)\right)\right)^2$$

$$\begin{aligned}\frac{\partial E}{\partial w_{ji}^h} &= -\sum_{p=1}^{m}(y_{dp} - y_p)\dot{f}^o\left(\sum_{q=1}^{l} w_{sq}^o z_q\right) w_{pj}^o \dot{f}^h\left(\sum_{r=1}^{n} w_{jr}^h x_r\right) x_i \\ &= -\left(\sum_{p=1}^{m}\delta_p w_{pj}^o\right)\dot{f}^h(v_j)x_i\end{aligned}$$

Gradient descent

$$w_{sj}^{o(k+1)} = w_{sj}^{o(k)} + \eta\delta_s z_j$$
$$w_{ji}^{h(k+1)} = w_{ji}^{h(k)} + \eta\left(\sum_{p=1}^{m}\delta_p w_{pj}^o\right)\dot{f}_j^h(v_j)x_i$$

Rewrite

$$w_{sj}^{o(k+1)} = w_{sj}^{o(k)} + \eta\delta_s^o z_j$$
$$w_{ji}^{h(k+1)} = w_{ji}^{h(k)} + \eta\delta_j^h x_i$$

The update term is

$$\text{(error) (input)} \tag{31}$$

When to update?

1. With the average of the entire training set.
2. With the average of a batch of training set.
3. With every training sample.
4. With the average of randomly selected training samples.

Perceptron

Consider a network with one layer with sign() as an activation function

$$y = \mathrm{sign}(w^T x + b) \tag{32}$$

with a training set

$$\{((x_1, y_1), (x_2, y_2), \cdots, (x_P, y_P)\} \tag{33}$$

The cost function

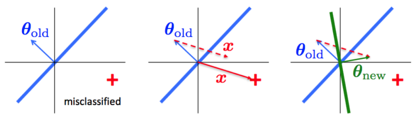
$$E = -\sum_p y_p(w^T x_p + b) \tag{34}$$

With $x = [x_1, \cdots, x_n, 1]$, for a training data (x_i, y_i),

$$\begin{aligned} E &= -y_i(w^T x_i) \\ \frac{\partial E}{\partial x} &= -y_i x_i \end{aligned} \tag{35}$$

Update is

$$w^{(k+1)} = w^{(k)} + \eta y_i x_i \tag{36}$$



Logistic Regression

When decision is

$$y = \text{sign}(w^T x + b) \tag{37}$$

We have $y = 0$, or -1.

Replace sign() with the sigmoid or logistic function

$$y = \sigma(w^T x + b) \tag{38}$$

where

$$\sigma(z) = \frac{1}{1 + \exp(-z)} \tag{39}$$

1) The function is differentiable.
2) The value is between $[0, 1]$.

odds

$$p/q \tag{40}$$

logit

$$\text{logit}(p) = \log(\text{odds}) = \log(p/q) \tag{41}$$

logistic regression

$$\text{logit}(p) = a + bx \tag{42}$$

Inverse of logit function

$$p = \frac{\exp{(a + bx)}}{1 + \exp{(a + bx)}} \tag{43}$$

$\sigma(z)$ can be interpreted as probability.

We consider $\sigma(z)$ to be the conditional probability.

Two classes $C = 0, 1$.

$$\begin{aligned} p(C = 1|x) &= \sigma(w^T x + b) \\ &= \frac{1}{1 + \exp(-(w^T x + b))} \end{aligned} \tag{44}$$

$$\begin{aligned} p(C = 0|x) &= 1 - p(C = 1|x) \\ &= \frac{\exp(-(w^T x + b)}{1 + \exp(-(w^T x + b))} \end{aligned} \tag{45}$$

For a given x_i, the likelihood of that sample being the y_i is

$$p(y_i|x_i; w) = p(y_i = 1|x_i; w)^{y_i} p(y_i = 0|x_i; w)^{(1-y_i)} \tag{46}$$

We want to maximize the likelihood.

Assuming iid,

$$\begin{aligned} L(w) &= \prod_i p(y_i|x_i; w) \\ l(w) &= \log(w) \\ &= \sum_i y_i \log p(y_i = 1|x_i; w) + (1 - y_i) \log p(y_i = 0|x_i; w) \end{aligned} \tag{47}$$

Optimization

$$\underset{w}{\text{minimize}} - \sum_i y_i \log p(y_i = 1|x_i; w) + (1 - y_i) \log(1 - p(y_i = 1|x_i; w)) \tag{48}$$

Update

$$w^{(k+1)} = w^{(k)} - \eta \frac{\partial l(w)}{\partial w} \tag{49}$$

Google for the derivative

$$\begin{aligned} \sigma(z) &= \frac{1}{1 + \exp(-z)} \\ \frac{d\sigma(z)}{dz} &= \sigma(z)(1 - \sigma(z)) \end{aligned} \tag{50}$$

The cost function

$$-(y_i \log p(y_i = 1|x_i; w) + (1 - y_i) \log(1 - p(y_i = 1|x_i; w))) \quad (51)$$

We used

$$p(y_i = 1|w_i; w) = \sigma(w^T x_i + b) \quad (52)$$

1) when $y_i = 1$, the cost is

$$-\log \sigma(w^T x_i + b) \quad (53)$$

hence, we try to make $\sigma(w^T x_i + b)$ as large as possible.

2) when $y_i = 0$, the cost is

$$-\log(1 - \sigma(w^T x_i + b)) \quad (54)$$

hence, we try to make $\sigma(w^T x_i + b)$ as small as possible.

Multiclass Logistic Regression

For class $c = \{1, 2, \cdots, C\}$,

$$\begin{aligned} p(y = c|x; w_1, w_2, \cdots, w_C) &\propto \exp(-(w_c^T x + b_c)) \\ &= \frac{\exp(-(w_c^T x + b_c))}{\sum_c \exp(-(w_c^T x + b_c))} \end{aligned} \quad (55)$$

The function is called softmax.

Two class example,

$$\begin{aligned} p(y = 1|x; w_1, w_2) &\propto \exp(-(w_1^T x + b_1)) \\ &= \frac{\exp(-(w_1^T x + b_1))}{\exp(-(w_0^T x + b_0) + \exp(-(w_1^T x + b_1))} \\ &= \frac{1}{1 + \exp(-((w_0 - w_1)^T x + (b_0 - b_1))} \\ &= \frac{1}{1 + \exp(-(w^T x + b)} \end{aligned} \quad (56)$$

This is the sigmoid function.

1-of-K encoding

$$t = [0, 1, 0, 0]^{\mathsf{T}} \quad (57)$$

$$p(T|x_i; w) = \prod_k p(y = k|x_i; w)^{t_i(k)} \quad (58)$$

The likelyhood

$$\begin{aligned} l(w) &= \prod_k p(y_i = k|x_i; w)^{t_i(k)} \\ L(w) &= \sum_k t_i(k) \log p(y_i = k|x_i; w) \end{aligned} \quad (59)$$

Derivative of softmax

$$D_j S_i = S_i(\delta_{ij} - S_j) \quad (60)$$