CS 559 Machine Learning

Logistic Regression and PCA

Yue Ning
Department of Computer Science
Stevens Institute of Technology

Plan for today

Probabilistic Generative Models

Logistic Regression

Principal Component Analysis

Review of last lecture

- Generative methods vs Discriminative methods
- Linear classifiers: Linear discriminant functions, least square classification, Fisher's linear discriminant (Geometrical properties of Linear Discriminant Analysis), the Perceptron algorithm
- Naive Bayes classifier

Review of gradient descent algorithms

- Three important elements: data($\mathbf{x}, y$), loss function, model parameters $\mathbf{w}$
- One important line, gradient update:

$$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \eta \nabla \mathbf{w}$$

Gradient Descent

procedure BATCH GRADIENT DESCENT
    for i in range(epochs) **do**
        $g^{(i)}(\mathbf{w}) = \text{evaluate\_gradient}(\text{TrainLoss, data, } \mathbf{w})$
        $\mathbf{w} = \mathbf{w} - \text{learning\_rate} * g^{(i)}(\mathbf{w})$
    end for
end procedure

- Can be very slow.
- Intractable for datasets that don't fit in memory.
- Doesn't allow us to update our model online, i.e. with new examples on-the-fly.
- Guaranteed to converge to the global minimum for convex error surfaces and to a local minimum for non-convex surfaces.

Stochastic Gradient Descent

```
procedure STOCHASTIC GRADIENT DESCENT
    for i in range(epochs) do
        np.random.shuffle(data)
        for example ∈ data do
            g^(i)(w) = evaluate_gradient(loss, example, w)
            w = w − learning_rate * g^(i)(w)
        end for
    end for
end procedure
```

- Allow for online update with new examples.
- With a high variance that cause the objective function to fluctuate heavily.

Mini-batch Gradient Descent

```
procedure MINI-BATCH GRADIENT DESCENT
    for i in range(epochs) do
        np.random.shuffle(data)
        for batch ∈ get_batches(data, batch_size=50) do
            g^(i)(w) = evaluate_gradient(loss, batch, w)
            w = w − learning_rate ∗ g^(i)(w)
        end for
    end for
end procedure
```

- Reduces the variance of the parameter updates, which can lead to more stable convergence;
- Can make use of highly optimized matrix optimizations common to state-of-the-art deep learning libraries that make computing the gradient w.r.t. a mini-batch very efficient.

Evaluation Metrics for Classification

- The contingency table or confusion matrix:

		True value	
		Positive	Negative
Predicted	Positive	True positive (TP)	False positive (FP)
	Negative	False negative (FN)	True negative (TN)

- Recall $= \frac{tp}{tp+fn}$
- Precision $= \frac{tp}{tp+fp}$
- Accuracy $= \frac{tp+tn}{tp+tn+fp+fn}$

Features for Text Classification

- Bag of words: word countings, TFIDF, ...
- Embeddings: word2vec, doc2vec,...
- Domain specific keywords
- Latent variables: topic distributions
- ......

Part I: Probabilistic Generative Models

Probabilistic Generative Models

- We now turn to a probabilistic approach to classification.
- How models with linear decision boundaries arise from simple assumptions about the distribution of the data.

Generative Approach

- Solve the inference problem of estimating the class-conditional densities $p(\mathbf{x}|C_k)$ for each class C_k.
- Infer the prior class probabilities $p(C_k)$.
- Use Bayes' theorem to find the class posterior probabilities:

$$p(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)p(C_k)}{p(\mathbf{x})} \tag{1}$$

where

$$p(x) = \sum_k p(\mathbf{x}|C_k)p(C_k) \tag{2}$$

- Use decision theory to determine class membership for each new input $\mathbf{x}$.

Probabilistic Generative Approach-Two Class

Two-class case:

$$
\begin{aligned}
p(C_1|\mathbf{x}) &= \frac{p(\mathbf{x}|C_1)p(C_1)}{p(\mathbf{x}|C_1)p(C_1) + p(\mathbf{x}|C_2)p(C_2)} \\
&= \frac{1}{1+e^{-a}} = \sigma(a)
\end{aligned}
\tag{3}
$$

where

$$
a = \ln \frac{p(\mathbf{x}|C_1)p(C_1)}{p(\mathbf{x}|C_2)p(C_2)} \tag{4}
$$

$$
\sigma(a) = \frac{1}{1+e^{-a}} \quad \text{(logistic sigmoid function)} \tag{5}
$$

Probabilistic Generative Approach-$K > 2$ Class

$K > 2$ classes:

$$p(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)p(C_k)}{\sum_{j=1}^{K} p(\mathbf{x}|C_j)p(C_j)} = \frac{e^{a_k}}{\sum_{j=1}^{K} e^{a_j}} \tag{6}$$

where

$$a_k = \ln\ p(\mathbf{x}|C_k)p(C_k) \tag{7}$$

$$\sigma(a) = \frac{e^{a_k}}{\sum_{j=1}^{K} e^{a_j}} \quad \text{(softmax function)} \tag{8}$$

Probabilistic Generative Models

- Lets assume the class-conditional densities are Gaussian with the same covariance matrix:

$$p(\mathbf{x}|C_k) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} e^{\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T \Sigma^{-1} (\mathbf{x}-\boldsymbol{\mu}_k)\right)} \tag{9}$$

- Two class case first. We can show the following result:

$$P(C_1|\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x} + \omega_0) \tag{10}$$

$$\mathbf{w} = \Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \tag{11}$$

$$\omega_0 = -\frac{1}{2}\boldsymbol{\mu}_1^T \Sigma \boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T \Sigma \boldsymbol{\mu}_2 + \ln \frac{p(C_1)}{p(C_2)} \tag{12}$$

- How?

Probabilistic Generative Models

$$P(C_1|\mathbf{x}) = \sigma(a) = \frac{1}{1+e^{-a}} \tag{13}$$

$$\begin{aligned} a &= \ln p(\mathbf{x}|C_1) - \ln p(\mathbf{x}|C_2) + \ln \frac{p(C_1)}{p(C_2)} \rightarrow \text{(Replace } p(\mathbf{x}|C_k)) \\ &= -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_1) + \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_2)^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_2) \\ &+ \ln \frac{p(C_1)}{p(C_2)} \end{aligned} \tag{14}$$

Probabilistic Generative Models

$$
\begin{aligned}
a &= -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_1) + \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_2)^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_2) + \ln\frac{p(C_1)}{p(C_2)} \\
&= -\frac{1}{2}\mathbf{x}^T\Sigma^{-1}\mathbf{x} + \frac{1}{2}\boldsymbol{\mu}_1^T\Sigma^{-1}\mathbf{x} + \frac{1}{2}\mathbf{x}^T\Sigma^{-1}\boldsymbol{\mu}_1 - \frac{1}{2}\boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{\mu}_1 \\
&+ \frac{1}{2}\mathbf{x}^T\Sigma^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_2^T\Sigma^{-1}\mathbf{x} - \frac{1}{2}\mathbf{x}^T\Sigma^{-1}\boldsymbol{\mu}_2 + \frac{1}{2}\boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{\mu}_2 + \ln\frac{p(C_1)}{p(C_2)} \\
&= \boldsymbol{\mu}_1^T\Sigma^{-1}\mathbf{x} - \boldsymbol{\mu}_2^T\Sigma^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{\mu}_2 + \ln\frac{p(C_1)}{p(C_2)} \\
&= (\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^T\Sigma^{-1}\mathbf{x} + \omega_0 = \mathbf{w}^T\mathbf{x} + \omega_0 \qquad (15)
\end{aligned}
$$

Thus:

$$
\mathbf{w} = \Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \qquad (16)
$$

Probabilistic Generative Models

- We have shown:

$$P(C_1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + \omega_0)$$
$$\mathbf{w} = \Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$$
$$\omega_0 = -\frac{1}{2}\boldsymbol{\mu}_1^T\Sigma\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T\Sigma\boldsymbol{\mu}_2 + \ln\frac{p(C_1)}{p(C_2)}$$

- Decision boundary:

$$p(C_1|\mathbf{x}) = p(C_2|\mathbf{x}) = 0.5 \tag{17}$$

$$\Rightarrow \frac{1}{1 + e^{-(\mathbf{w}^T\mathbf{x}+\omega_0)}} = 0.5 \Rightarrow \mathbf{w}^T\mathbf{x} + \omega_0 = 0 \tag{18}$$

Probabilistic Generative Models

- $\underline{K > 2 \text{ classes}}$:

$$P(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)p(C_k)}{\sum_j p(\mathbf{x}|C_j)p(C_j)} = \frac{e^{a_k}}{\sum_j e^{a_j}} \tag{19}$$

$$a_k = \ln\ p(\mathbf{x}|C_k)p(C_k) \tag{20}$$

- We can show the following result:

$$p(C_k|\mathbf{x}) = \frac{e^{\mathbf{w}_k^T\mathbf{x}+\omega_{k0}}}{\sum_j e^{\mathbf{w}_j^T\mathbf{x}+\omega_{j0}}} \tag{21}$$

$$\mathbf{w}_k = \Sigma^{-1}\boldsymbol{\mu}_k \tag{22}$$

$$\omega_{k0} = -\frac{1}{2}\boldsymbol{\mu}_k^T\Sigma^{-1}\boldsymbol{\mu}_k + \ln\ p(C_k) \tag{23}$$

Maximum Likelihood Solution

- We have a parametric functional form for the class-conditional densities:

$$p(\mathbf{x}|C_k) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} e^{\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T \Sigma^{-1} (\mathbf{x}-\boldsymbol{\mu}_k)\right)} \tag{24}$$

- We can estimate the parameters and the prior class probabilities using maximum likelihood.
  - Two class case with shared covariance matrix.
  - Training data:

$$\{\mathbf{x}_n, y_n\}, \quad n = 1, ..., N$$

$y_n = 1$ denotes class C_1; $y_n = 0$ denotes class C_2;

Priors: $p(C_1) = \gamma, p(C_2) = 1 - \gamma$

Maximum Likelihood Solution

- For a data point $\mathbf{x}_n$ from class C_1, we have $y_n = 1$ and therefore:

$$p(\mathbf{x}_n, C_1) = p(\mathbf{x}|C_1)p(C_1) = \gamma\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \Sigma) \tag{25}$$

- For a data point $\mathbf{x}_n$ from class C_1, we have $y_n = 1$ and therefore:

$$p(\mathbf{x}_n, C_2) = p(\mathbf{x}|C_2)p(C_2) = (1-\gamma)\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \Sigma) \tag{26}$$

- Assuming observations are drawn independently, the likelihood function is as below:

$$\begin{aligned} p(\mathcal{D}|\gamma, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \Sigma) &= \prod_{n=1}^{N} \left[p(\mathbf{x}_n, C_1)\right]^{y_n} \left[p(\mathbf{x}_n, C_2)\right]^{1-y_n} \\ &= \prod_{n=1}^{N} \left[\gamma\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \Sigma)\right]^{y_n} \left[(1-\gamma)\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \Sigma)\right]^{1-y_n} \end{aligned}$$

Maximum Likelihood Solution

- We want to find the values of the parameters that **maximize the likelihood function**, i.e., fit a model that best describes the observed data.

$$p(\mathcal{D}|\gamma, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \Sigma) = \prod_{n=1}^{N} \left[\gamma \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \Sigma)\right]^{y_n} \left[(1-\gamma)\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \Sigma)\right]^{1-y_n} \tag{27}$$

- As usual, we consider the log of the likelihood:

$$\begin{aligned} &\ln\, p(\mathcal{D}|\gamma, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \Sigma) = \\ &\sum_{n=1}^{N} \left[y_n \ln\gamma + y_n \ln\, \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \Sigma)\right. \\ &\left. + (1-y_n)\ln\, (1-\gamma) + (1-y_n)\ln\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \Sigma)\right] \end{aligned} \tag{28}$$

Maximum Likelihood Solution - parameter γ

$$\ln p(\mathcal{D}|\gamma,\boldsymbol{\mu}_1,\boldsymbol{\mu}_2,\Sigma) = \sum_{n=1}^{N} \big[y_n \ln \gamma + y_n \ln \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1,\Sigma) + (1-y_n)\ln(1-\gamma) + (1-y_n)\ln \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2,\Sigma) \big]$$

- We first maximize the log-likelihood with respect to γ (set derivate to 0)

$$\gamma = \frac{1}{N}\sum_{n=1}^{N} y_n = \frac{N_1}{N} = \frac{N_1}{N_1+N_2} \tag{29}$$

- The maximum likelihood estimate of γ is the fraction of points in class C_1.
- For multi-class: ML estimate for $p(C_k)$ is given by the fraction of points in the training set in C_k.

Maximum Likelihood Solution - parameter $\boldsymbol{\mu}$

$$\ln p(\mathcal{D}|\gamma,\boldsymbol{\mu}_1,\boldsymbol{\mu}_2,\Sigma) = \sum_{n=1}^{N}\Big[y_n \ln\gamma + y_n \ln\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1,\Sigma) + (1-y_n)\ln(1-\gamma) + (1-y_n)\ln\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2,\Sigma)\Big]$$

- We them maximize the log-likelihood with respect to $\boldsymbol{\mu}_1$ (set derivate to 0)

$$\boldsymbol{\mu}_1 = \frac{1}{N_1}\sum_{n=1}^{N} y_n\mathbf{x}_n \qquad (30)$$

- The maximum likelihood estimate of $\boldsymbol{\mu}_1$ is the sample mean of all input $\mathbf{x}_n$ in class C_1.
- The maximum likelihood estimate of μ_2 is:

$$\boldsymbol{\mu}_2 = \frac{1}{N_2}\sum_{n=1}^{N}(1-y_n)\mathbf{x}_n \qquad (31)$$

Maximum Likelihood Solution - parameter Σ

- Maximize the log-likelihood with respect to Σ (set derivate to 0), we obtain the estimate Σ_{ML}

$$\Sigma_{ML} = \frac{N_1}{N} S_1 + \frac{N_2}{N} S_2 \tag{32}$$

where

$$S_1 = \frac{1}{N_1} \sum_{n \in C_1} (\mathbf{x}_n - \boldsymbol{\mu}_1)(\mathbf{x}_n - \boldsymbol{\mu}_1)^T \tag{33}$$

$$S_2 = \frac{1}{N_2} \sum_{n \in C_2} (\mathbf{x}_n - \boldsymbol{\mu}_2)(\mathbf{x}_n - \boldsymbol{\mu}_2)^T \tag{34}$$

- The maximum likelihood estimate of the covariance is given by the weighted average of the sample covariance matrices associated with each of the classes.
- The results extend to K classes.

Summary so far

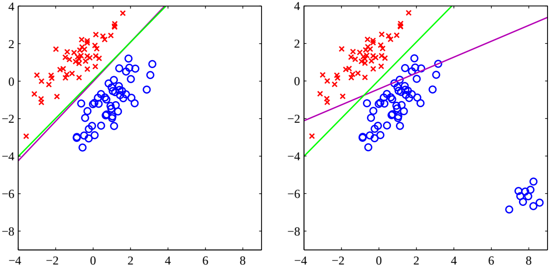
- We assumed $p(\mathbf{x}|(y=1)) \sim \mathcal{N}(\boldsymbol{\mu}_+, \Sigma_+)$ and $p(\mathbf{x}|(y=-1)) \sim \mathcal{N}(\boldsymbol{\mu}_-, \Sigma_-)$, and two class-probabilities $p(y=1)$ and $p(y=-1)$.

Summary so far

- We assumed $p(\mathbf{x}|(y=1)) \sim \mathcal{N}(\boldsymbol{\mu}_+, \Sigma_+)$ and $p(\mathbf{x}|(y=-1)) \sim \mathcal{N}(\boldsymbol{\mu}_-, \Sigma_-)$, and two class-probabilities $p(y=1)$ and $p(y=-1)$.
- This is called an generative model, as we have written down a full joint model over the data.

Summary so far

- We assumed $p(\mathbf{x}|(y=1)) \sim \mathcal{N}(\boldsymbol{\mu}_+, \Sigma_+)$ and $p(\mathbf{x}|(y=-1)) \sim \mathcal{N}(\boldsymbol{\mu}_-, \Sigma_-)$, and two class-probabilities $p(y=1)$ and $p(y=-1)$.
- This is called an generative model, as we have written down a full joint model over the data.
- We saw that violations of the model assumption can lead to "bad" decision boundaries.



Figures from Bishop PRML, 44a and b

Probabilistic Generative Models - parameters

- How many parameters did we estimate to fit Gaussian class-conditional densities (the generative approach)?

$$
\begin{aligned}
p(C_1) &\Rightarrow 1 \\
\text{2 mean vectors } &\Rightarrow 2d \\
\Sigma &\Rightarrow d + \frac{d^2 - d}{2} = \frac{d^2 + d}{2} \\
\text{total } &= 1 + 2d + \frac{d^2 + d}{2} = O(d^2)
\end{aligned}
$$

Part II: Logistic Regression

Logistic Regression

$$P(C_1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + \omega_0) = f(\mathbf{x}) \tag{35}$$

- We use maximum likelihood to determine the parameters of the logistic regression model.

$$\{\mathbf{x}_n, y_n\}, \quad n = 1, ..., N$$

$$y_n = 1 \text{ denotes class } C_1; \; y_n = 0 \text{ denotes class } C_2;$$

Logistic Regression

$$P(C_1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + \omega_0) = f(\mathbf{x}) \tag{35}$$

- We use maximum likelihood to determine the parameters of the logistic regression model.

$$\{\mathbf{x}_n, y_n\}, \quad n = 1, ..., N$$

$$y_n = 1 \text{ denotes class } C_1;\ y_n = 0 \text{ denotes class } C_2;$$

- We want to find the values of **w** that maximize the posterior probabilities associated to the observed data
- Likelihood function:

$$\begin{aligned} L(\mathbf{w}) &= \prod_{n=1}^{N} p(C_1|\mathbf{x}_n)^{y_n}(1 - p(C_1|\mathbf{x}_n))^{1-y_n} \\ &= \prod_{n=1}^{N} f(\mathbf{x}_n)^{y_n}(1 - f(\mathbf{x}_n))^{1-y_n} \end{aligned} \tag{36}$$

Logistic Regression

$$P(C_1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + \omega_0) = f(\mathbf{x})$$

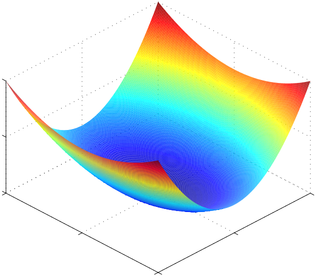
- We consider the negative logarithm of the likelihood (Cross Entropy):

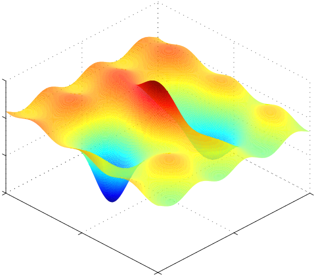
$$\begin{aligned} E(\mathbf{w}) = -\ln\ L(\mathbf{w}) &= -\ln\ \prod_{n=1}^{N} p(C_1|\mathbf{x}_n)^{y_n}(1 - p(C_1|\mathbf{x}_n))^{1-y_n} \\ &= -\sum_{n=1}^{N}(y_n \ln f(\mathbf{x}_n) + (1 - y_n)\ln\ (1 - f(\mathbf{x}_n))) \end{aligned} \tag{37}$$

- Thus:

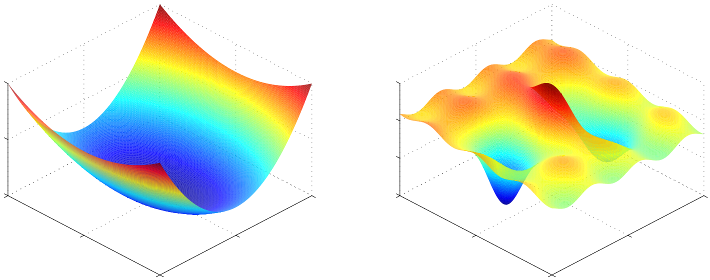
$$\max L(\mathbf{w}) = \min E(\mathbf{w}) \tag{38}$$

The cost-function for logistic regression is convex.



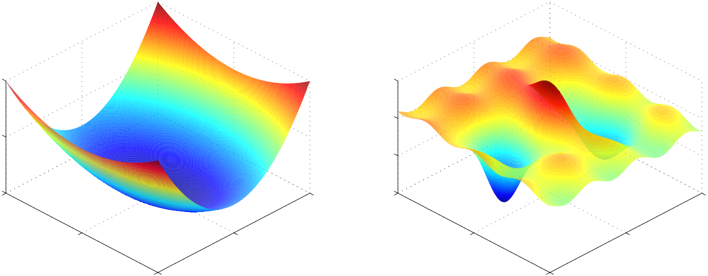


The cost-function for logistic regression is convex.



- Fact: The negative log-likelihood is *convex* – this makes life much more easier.

The cost-function for logistic regression is convex.



- Fact: The negative log-likelihood is *convex* – this makes life much more easier.
- There are no local minima to get stuck in, and there is good optimization techniques for convex problems.

Logistic Regression - Gradient

$$P(C_1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + \omega_0) = f(\mathbf{x})$$

- We compute the derivative of the error function with respect to **w**

$$\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}}\left[-\ln \prod_{n=1}^{N} p(C_1|\mathbf{x}_n)^{y_n}(1 - p(C_1|\mathbf{x}_n))^{1-y_n}\right] \quad (39)$$

- The derivative of the logistic sigmoid function:

$$\begin{aligned} \frac{\partial}{\partial a}\sigma(a) &= \frac{\partial}{\partial a}\frac{1}{1+e^{-a}} = \frac{e^{-a}}{(1+e^{-a})^2} \\ &= \frac{1}{1+e^{-a}}\frac{e^{-a}}{(1+e^{-a})} = \frac{1}{1+e^{-a}}(1 - \frac{1}{1+e^{-a}}) \\ &= \sigma(a)(1-\sigma(a)) \end{aligned} \quad (40)$$

Logistic Regression

Homework: The derivative of the error function with respect to $\mathbf{w}$ is:

$$\nabla_{\mathbf{w}} E(\mathbf{w}) = \sum_{n=1}^{N} (f(\mathbf{x}) - y_n)\mathbf{x}_n \tag{41}$$

Probabilistic Discriminative Models - parameters

- Two-class case:

$$P(C_1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + \omega_0) = f(\mathbf{x})$$

$$P(C_2|\mathbf{x}) = 1 - P(C_1|\mathbf{x})$$

- This model is know as Logistic Regression.
- Assuming $\mathbf{x} \in \mathbb{R}^d$, how many parameters do we need to estimate?

$$d + 1$$

Summary so far

- From the previous introduction, we know that

$$P(y = 1|\mathbf{x}) = \sigma(a(\mathbf{x}))$$

where $\sigma(a) = 1/(1 + \exp(-a))$ and $a(x) = \mathbf{w}^\top \mathbf{x} + \omega_o$.
- Notation is simpler if we use 0 and 1 as class labels, so we define $y_n = 1$ as the label for the positive class, and $y_n = 0$ as label for the negative class.
- In other words, $y|x \sim \text{Bernoulli}(\sigma(f(x)))$.

Summary so far

- From the previous introduction, we know that

$$P(y=1|\mathbf{x}) = \sigma(a(\mathbf{x}))$$

  where $\sigma(a) = 1/(1+\exp(-a))$ and $a(x) = \mathbf{w}^\top \mathbf{x} + \omega_o$.
- Notation is simpler if we use 0 and 1 as class labels, so we define $y_n = 1$ as the label for the positive class, and $y_n = 0$ as label for the negative class.
- In other words, $y|x \sim \text{Bernoulli}(\sigma(f(x)))$.
- The parameters of $a(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + \omega_o$ can be learned by maximizing the conditional log-likelihood $L(\mathbf{w}) = \sum_n \log p(y_n|x_n, \mathbf{w})$

Summary so far

- From the previous introduction, we know that

$$P(y = 1|\mathbf{x}) = \sigma(a(\mathbf{x}))$$

  where $\sigma(a) = 1/(1 + \exp(-a))$ and $a(x) = \mathbf{w}^\top \mathbf{x} + \omega_o$.
- Notation is simpler if we use 0 and 1 as class labels, so we define $y_n = 1$ as the label for the positive class, and $y_n = 0$ as label for the negative class.
- In other words, $y|x \sim \text{Bernoulli}(\sigma(f(x)))$.
- The parameters of $a(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + \omega_o$ can be learned by maximizing the conditional log-likelihood $L(\mathbf{w}) = \sum_n \log p(y_n|x_n, \mathbf{w})$
- This is a discriminative approach to classification, as we only model the labels, and not the inputs.

Summary so far

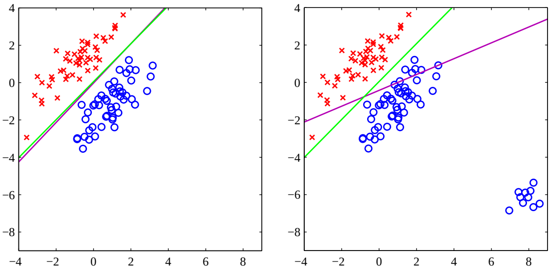
- From the previous introduction, we know that

$$P(y = 1|\mathbf{x}) = \sigma(a(\mathbf{x}))$$

where $\sigma(a) = 1/(1 + \exp(-a))$ and $a(x) = \mathbf{w}^\top \mathbf{x} + \omega_o$.

- Notation is simpler if we use 0 and 1 as class labels, so we define $y_n = 1$ as the label for the positive class, and $y_n = 0$ as label for the negative class.
- In other words, $y|x \sim \text{Bernoulli}(\sigma(f(x)))$.
- The parameters of $a(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + \omega_o$ can be learned by maximizing the conditional log-likelihood $L(\mathbf{w}) = \sum_n \log p(y_n|x_n, \mathbf{w})$
- This is a discriminative approach to classification, as we only model the labels, and not the inputs.
- Decision rule and function shape of $p(y|\mathbf{x})$ will be the same for the generative and the discriminative model, but the parameters were obtained differently.

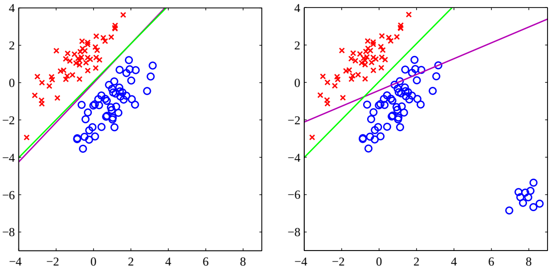
Maximum likelihood estimation of Logistic Regression



Bishop PRML Figure 44 a and b

- Logistic regression is a *much* better algorithm than the algorithms we discussed last week.
- Need to optimize log-likelihood numerically.

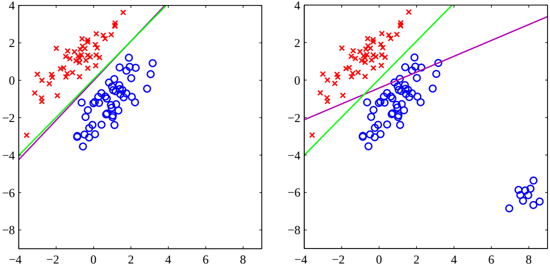
Maximum likelihood estimation of Logistic Regression



Bishop PRML Figure 44 a and b

- Logistic regression is a *much* better algorithm than the algorithms we discussed last week.
- Need to optimize log-likelihood numerically.
- People typically minimize the negative log-likelihood $\mathcal{L}$ rather than maximize the log-likelihood...

Maximum likelihood estimation of Logistic Regression



Bishop PRML Figure 44 a and b

- Logistic regression is a *much* better algorithm than the algorithms we discussed last week.
- Need to optimize log-likelihood numerically.
- People typically minimize the negative log-likelihood $\mathcal{L}$ rather than maximize the log-likelihood...
- To numerically minimize the negative log-likelihood, we need its gradient (and maybe its hessian)

Multiclass Logistic Regression

- Multiclass case:

$$p(C_k|\mathbf{x}) = \frac{e^{\mathbf{w}_k^T\mathbf{x}+\omega_{k0}}}{\sum_j e^{\mathbf{w}_j^T\mathbf{x}+\omega_{j0}}} = f_k(\mathbf{x}) \quad (42)$$

- We use maximum likelihood to determine the parameters:

$$\{\mathbf{x}_n, y_n\}, \quad n = 1, ..., N$$

$$y_n = (0, ..., 1, ..., 0) \text{ denotes class } C_k$$

- We want to find the values of $\mathbf{w}_1, ..., \mathbf{w}_k$ that maximize the posterior probabilities associated to the observed data likelihood function:

$$\mathcal{L}(\mathbf{w}_1, ..., \mathbf{w}_k) = \prod_{n=1}^{N}\prod_{k=1}^{K} p(C_k|\mathbf{x}_n)^{y_{nk}} = \prod_{n=1}^{N}\prod_{k=1}^{K} f_k(\mathbf{x}_n)^{y_{nk}} \quad (43)$$

Multiclass Logistic Regression

$$\mathcal{L} = \prod_{n=1}^{N} \prod_{k=1}^{K} p(C_k|\mathbf{x}_n)^{y_{nk}} = \prod_{n=1}^{N} \prod_{k=1}^{K} f_k(\mathbf{x}_n)^{y_{nk}}$$

- Consider the negative logarithm of the likelihood:

$$\mathcal{E}(\mathbf{w}_1, ..., \mathbf{w}_k) = -\ln\mathcal{L}(\mathbf{w}_1, ..., \mathbf{w}_k) = -\sum_{n=1}^{N} \sum_{k=1}^{K} y_{nk} \ln f_k(\mathbf{x}_n) \quad (44)$$

$$\min_{\mathbf{w}_j} \mathcal{E}((\mathbf{w}_j)) \quad (45)$$

- The gradient of the error function w.r.t one of the parameter vectors:

$$\frac{\partial}{\partial \mathbf{w}_j}\mathcal{E}(\mathbf{w}_1, ..., \mathbf{w}_k) = \frac{\partial}{\partial \mathbf{w}_j}\Big[-\sum_{n=1}^{N} \sum_{k=1}^{K} y_{nk} \ln\ f_k(\mathbf{x}_n)\Big] \quad (46)$$

Multiclass Logistic Regression

$$\frac{\partial}{\partial \mathbf{w}_j}\mathcal{E}(\mathbf{w}_1, ..., \mathbf{w}_k) = \frac{\partial}{\partial \mathbf{w}_j}\Big[- \sum_{n=1}^{N}\sum_{k=1}^{K} y_{nk} \ln\ f_k(\mathbf{x}_n)\Big]$$

- The derivatives of the softmax function:

$$\frac{\partial}{\partial a_k} f_k = \frac{\partial}{\partial a_k} \frac{e^{a_k}}{\sum_j e^{a_j}} = \frac{e^{a_k}\sum_j e^{a_j} - e^{a_k}e^{a_k}}{(\sum_j e^{a_j})^2} = f_k - f_k^2 = f_k(1 - f_k)$$

- Thus:

$$\text{for } j \neq k, \frac{\partial}{\partial a_j} f_k = \frac{\partial}{\partial a_j} f_k = \frac{\partial}{\partial a_j} \frac{e^{a_k}}{\sum_j e^{a_j}} = \frac{-e^{a_k}e^{a_j}}{(\sum_j e^{a_j})^2} = -f_k f_j$$

- Compact expression (I_{kj} are the elements of the identity matrix)

$$\frac{\partial}{\partial a_j} f_k = f_k(I_{kj} - f_j)$$

Multiclass Logistic Regression

$$\nabla_{\mathbf{w}_j}\mathcal{E}(\mathbf{w}_1, \ldots, \mathbf{w}_k) = \sum_{n=1}^{N}(f_{nj} - y_{nj})\mathbf{x}_n$$

Multiclass Logistic Regression

$$\nabla_{\mathbf{w}_j}\mathcal{E}(\mathbf{w}_1, ..., \mathbf{w}_k) = \sum_{n=1}^{N}(f_{nj} - y_{nj})\mathbf{x}_n$$

- It can be shown that $\mathcal{E}$ is a convex function of $\mathbf{w}$. Thus, it has a unique minimum.
- For a batch solution, we can use the Newton-Raphson optimization technique.
- Online solution (SGD):

$$\mathbf{w}_j^{t+1} = \mathbf{w}_j^t - \eta\nabla_{\mathbf{w}_j}\mathcal{E}_n(\mathbf{w}) = \mathbf{w}_j^t - \eta(f_{nj} - y_{nj})\mathbf{x}_n$$

Part III: PCA

Eigenvalues and Eigenvectors

- For an $n \times n$ square matrix A, $\mathbf{e}$ is an eigenvector with eigenvalue λ if:

$$A\mathbf{e} = \lambda\mathbf{e}$$

or

$$(A - \lambda I)\mathbf{e} = 0$$

- If $(A - \lambda I)$ is invertible, the only solution is $\mathbf{e} = \mathbf{0}$ (trivial).

Eigenvalues and Eigenvectors

- For non-trivial solutions

$$det(A - \lambda I) = 0$$

- This is called the "characteristic polynomial"
- Solutions are not unique because if $\mathbf{e}$ is an eigenvector $a\mathbf{e}$ is also an eigenvector.

Eigenvalues and Eigenvectors - simple example

- For a 2×2 matrix A:
$$\begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}$$
- Given:
$$det(A - \lambda I) = \begin{vmatrix} a_{11} - \lambda & a_{12} \\ a_{21} & a_{22} - \lambda \end{vmatrix} = (a_{11} - \lambda)(a_{22} - \lambda) - a_{12}a_{21} = 0$$
- We get: $a_{11}a_{22} - a_{12}a_{21} - (a_{11} + a_{22})\lambda + \lambda^2 = 0 \rightarrow$
$1 \cdot 4 - 2 \cdot 2 - (1 + 4)\lambda + \lambda^2 = 0$
- Solutions are $\lambda = 0$ and $\lambda = 5$.

Eigenvalues and Eigenvectors - simple example

- The eigenvector for the first eigenvalue, $\lambda = 0$ is:

$$\begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1x + 2y \\ 2x + 4y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

- One solution for both equations is $x = 2, y = -1$
- The eigenvector for the second eigenvalue, $\lambda = 5$ is:

$$\begin{bmatrix} -4 & 2 \\ 2 & -1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -4x + 2y \\ 2x - 1y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

- One solution for both equations is $x = 1, y = 2$

Eigenvalues and Eigenvectors - properties

- The product of the eigenvalues is the determinant of A: $det(A)$
- The sum of the eigenvalues $=$ trace(A)
- The eigenvectors are pairwise orthogonal

Dimensionality Reduction

- Many dimensions are often interdependent (correlated);
- Solution 1: reduce the dimensionality of problems;
- Solution 2: transform interdependent coordinates into significant and independent ones;

Principal Component Analysis

It is also called Karhunen-Loeve transformation

- PCA transforms the original input space into a lower dimensional space, by constructing dimensions that are linear combinations of the given features;
- The objective is to consider independent dimensions along which data have largest variance (i.e., greatest variability)

Principal Component Analysis

- PCA involves a linear algebra procedure that transforms a number of possibly correlated variables into a smaller number of uncorrelated variables called principal components;
- The first principal component accounts for as much of the variability in the data as possible;
- Each succeeding component (orthogonal to the previous ones) accounts for as much of the remaining variability as possible.

Principal Component Analysis

- So: PCA finds n linearly transformed components, $s_1, s_2, ..., s_n$, so that they explain the maximum amount of variance;
- We can define PCA in an intuitive way using a recursive formulation:

Principal Component Analysis

- Suppose data are first centered at the origin (i.e., their mean is $\mathbf{0}$)
- We define the direction of the first principal component, say, $\mathbf{w}_1$ as follows:

$$\mathbf{w}_1 = \arg\max_{||\mathbf{w}||=1} E[(\mathbf{w}^T\mathbf{x})^2]$$

  where $\mathbf{w}_1$ is of the same dimensionality d as the data vector $\mathbf{x}$.
- Thus: the first principal component is the projection on the direction along which the variance of the projection is maximized.

Principal Component Analysis

- Having determined the first $k-1$ principal components, the k-th principal component is determined as the principal component of the data residual:

$$\mathbf{w}_k = \arg\max_{||\mathbf{w}||=1} E\{[\mathbf{w}^T(\mathbf{x} - \sum_{i=1}^{k-1} \mathbf{w}_i \mathbf{w}_i^T \mathbf{x})]^2\}$$

- The principal components are then given by:

$$s_i = \mathbf{w}_i^T \mathbf{x}$$

Simple illustration of Principal Component Analysis

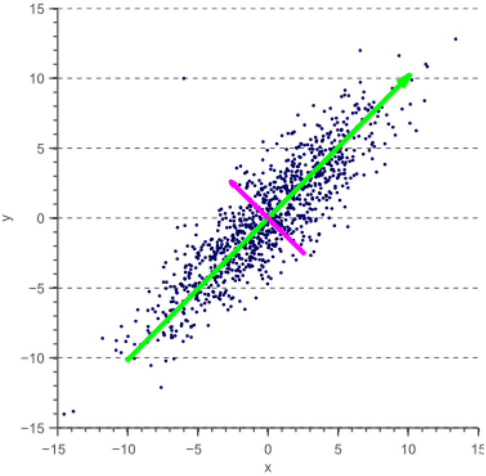


Figure: Green: first principal component of a two-dimensional dataset; Magenta: second principal component

PCA - Geometric interpretation

PCA rotates the data (centered at the origin) in such a way that the maximum variability is visible (i.e., aligned with the axes.)

PCA - How to compute the principal components

- Let $\mathbf{w}$ be the direction of the first principal component, with $||\mathbf{w}|| = 1$
- $s_i = \mathbf{w}^T \mathbf{x}_i$ is the projection of $\mathbf{x}_i$ along $\mathbf{w}$.
- $\bar{s} = \frac{1}{N} \sum_{i=1}^{N} s_i = \frac{1}{N} \sum_{i=1}^{N} \mathbf{w}^T \mathbf{x}_i$
- Variance of data along $\mathbf{w}$:

$$\frac{1}{N} \sum_{i=1}^{N} (s_i - \bar{s})^2 = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{w}^T \mathbf{x}_i - \frac{1}{N} \sum_{j=1}^{N} \mathbf{w}^T \mathbf{x}_j)^2$$

PCA - How to compute the principal components

$$
\begin{aligned}
\frac{1}{N}\sum_{i=1}^{N}(s_i - \bar{s})^2 &= \frac{1}{N}\sum_{i=1}^{N}(\mathbf{w}^T\mathbf{x}_i - \frac{1}{N}\sum_{j=1}^{N}\mathbf{w}^T\mathbf{x}_j)^2 \\
&= \frac{1}{N}\sum_{i=1}^{N}\left[\mathbf{w}^T\;(\mathbf{x}_i - \frac{1}{N}\sum_{j=1}^{N}\mathbf{x}_j)\right]^2 \\
&= \frac{1}{N}\sum_{i=1}^{N}\left[\mathbf{w}^T(\mathbf{x}_i - \bar{\mathbf{x}})\right]^2 \\
&= \frac{1}{N}\sum_{i=1}^{N}\left[\mathbf{w}^T(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T\mathbf{w}\right] \\
&= \mathbf{w}^T\underbrace{\left\{\frac{1}{N}\sum_{i=1}^{N}\left[(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T\right]\right\}}_{\text{sample covariance matrix}}\mathbf{w} \\
&= \mathbf{w}^T\Sigma\mathbf{w}
\end{aligned}
$$

PCA - How to compute the principal components

- Thus, the variance of data along direction $\mathbf{w}$ can be written as:

$$\mathbf{w}^T \Sigma \mathbf{w}$$

- Our objective is to find $\mathbf{w}$ such that:

$$\mathbf{w} = \arg\max_{\mathbf{w}} \mathbf{w}^T \Sigma \mathbf{w}$$

  with the constraint $\mathbf{w}^T \mathbf{w} = 1$

- By introducing one Lagrange multiplier λ we obtain the following unconstrained optimization problem:

$$\mathbf{w} = \arg\max_{\mathbf{w}} [\mathbf{w}^T \Sigma \mathbf{w} - \lambda(\mathbf{w}^T \mathbf{w} - 1)]$$

- Setting $\frac{\partial}{\partial \mathbf{w}} = 0$, gives $2\Sigma\mathbf{w} - 2\lambda\mathbf{w} = 0$
- That is $\Sigma\mathbf{w} = \lambda\mathbf{w}$ (reduced to an eigenvalue problem)

PCA - How to compute the principal components

The solution $\mathbf{w}$ is the eigenvector of Σ corresponding to the largest eigenvalue λ:

$$\Sigma \mathbf{w} = \lambda \mathbf{w}$$

PCA - Summary

- The computation of the $\mathbf{w}_i$ is accomplished by solving an eigenvalue problem for the sample covariance matrix (assuming data have 0 mean):

$$\Sigma = E[\mathbf{x}\mathbf{x}^T]$$

- The eigenvector associated with the largest eigenvalue corresponds to the first principal component; the eigenvector associated with the second largest eigenvalue corresponds to the second principal component; and so on...
- Thus: The $\mathbf{w}_i$ are the eigenvectors of Σ that correspond to the i largest eigenvalues of Σ.

PCA - In practice

- The basic goal of PCA is to reduce the dimensionality of the data. Thus, one usually chooses:

$$n \ll d$$

- But how do we select the number of components n?

Determining the number of component

- Plot the eigenvalues–each eigenvalue is related to the amount of variation explained by the corresponding axis (eigenvector);
- If the points on the graph tend to level out (show an "elbow" shape), these eigenvalues are usually close enough to zero that they can be ignored;
- In general: Limit the variance accounted for.

Determining the number of component

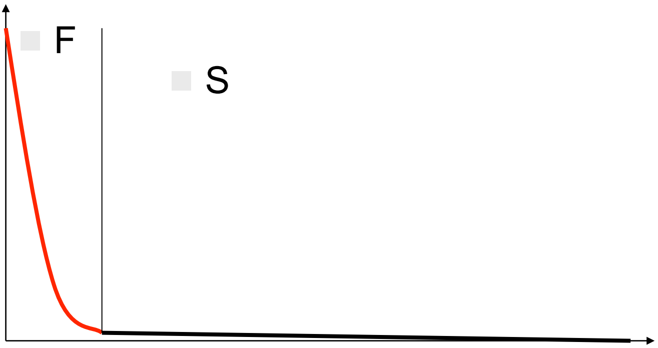
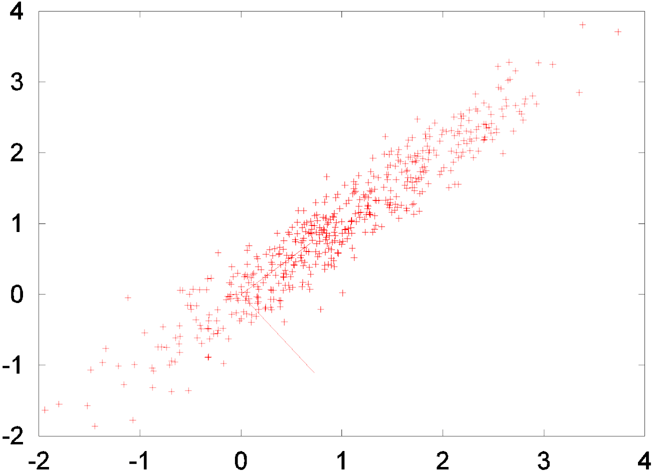


Figure: A typical eigenvalue spectrum and its division into two orthogonal subspaces

Determining the number of component



$$\lambda_1 = 1.98, \quad \lambda_2 = 0.05$$

Determining the number of component

- $\mathbf{x}_i \in \mathbb{R}^d, i = 1, ..., N$
- $\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_d$: d eigenvectors (principal component directions)
- $||\mathbf{w}_i|| = 1$ (the $\mathbf{w}_i$s are orthonormal vectors)
- Representation of $\mathbf{x}_i$ in eigenvector space:

$$\mathbf{z}_i = (\mathbf{w}_1^T \mathbf{x}_i)\mathbf{w}_1 + (\mathbf{w}_2^T \mathbf{x}_i)\mathbf{w}_2 + ... + (\mathbf{w}_d^T \mathbf{x}_i)\mathbf{w}_d$$

- Suppose we retain the first k principal component:

$$\mathbf{z}_i^k = (\mathbf{w}_1^T \mathbf{x}_i)\mathbf{w}_1 + (\mathbf{w}_2^T \mathbf{x}_i)\mathbf{w}_2 + ... + (\mathbf{w}_k^T \mathbf{x}_i)\mathbf{w}_k$$

- Then:

$$\mathbf{z}_i - \mathbf{z}_i^k = (\mathbf{w}_{k+1}^T \mathbf{x}_i)\mathbf{w}_{k+1} + ... + (\mathbf{w}_d^T \mathbf{x}_i)\mathbf{w}_d$$

Determining the number of component

$$
\begin{aligned}
&(\mathbf{z}_i - \mathbf{z}_i^k)^T(\mathbf{z}_i - \mathbf{z}_i^k) = \\
&\left[(\mathbf{w}_{k+1}^T\mathbf{x}_i)\mathbf{w}_{k+1} + \ldots + (\mathbf{w}_d^T\mathbf{x}_i)\mathbf{w}_d\right]^T\left[(\mathbf{w}_{k+1}^T\mathbf{x}_i)\mathbf{w}_{k+1} + \ldots + (\mathbf{w}_d^T\mathbf{x}_i)\mathbf{w}_d\right] \\
&= \mathbf{w}_{k+1}^T(\mathbf{w}_{k+1}^T\mathbf{x}_i)^2\mathbf{w}_{k+1} + \ldots + \mathbf{w}_d^T(\mathbf{w}_d^T\mathbf{x}_i)^2\mathbf{w}_d \\
&(\text{note } \mathbf{w}_i^T\mathbf{w}_j = 0 \forall i \neq j \text{ since } \mathbf{w}_i \text{ and } \mathbf{w}_j \text{ are orthogonal vectors}) \\
&= (\mathbf{w}_{k+1}^T\mathbf{x}_i)^2\mathbf{w}_{k+1}^T\mathbf{w}_{k+1} + \ldots + (\mathbf{w}_d^T\mathbf{x}_i)^2\mathbf{w}_d^T\mathbf{w}_d \\
&= (\mathbf{w}_{k+1}^T\mathbf{x}_i)^2 + \ldots + (\mathbf{w}_d^T\mathbf{x}_i)^2 \\
&= (\mathbf{w}_{k+1}^T\mathbf{x}_i)(\mathbf{x}_i^T\mathbf{w}_{k+1}) + \ldots + (\mathbf{w}_d^T\mathbf{x}_i)(\mathbf{x}_i)^T\mathbf{w}_d \\
&= \mathbf{w}_{k+1}^T(\mathbf{x}_i\mathbf{x}_i^T)\mathbf{w}_{k+1} + \ldots + \mathbf{w}_d^T(\mathbf{x}_i\mathbf{x}_i^T)\mathbf{w}_d
\end{aligned}
$$

Determining the number of component

$$\frac{1}{N}\sum_{i=1}^{N}(\mathbf{z}_i - \mathbf{z}_i^k)^T(\mathbf{z}_i - \mathbf{z}_i^k) =$$

$$\frac{1}{N}\sum_{i=1}^{N}\left[\mathbf{w}_{k+1}^T(\mathbf{x}_i\mathbf{x}_i^T)\mathbf{w}_{k+1} + ... + \mathbf{w}_d^T(\mathbf{x}_i\mathbf{x}_i^T)\mathbf{w}_d\right]$$

$$= \mathbf{w}_{k+1}^T\left[\frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i\mathbf{x}_i^T)\right]\mathbf{w}_{k+1} + ... + \mathbf{w}_d^T\left[\frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i\mathbf{x}_i^T)\right]\mathbf{w}_d$$

$$= \mathbf{w}_{k+1}^T\Sigma\mathbf{w}_{k+1} + ... + \mathbf{w}_d^T\Sigma\mathbf{w}_d$$

(Note: $\Sigma\mathbf{w}_{k+1} = \lambda_{k+1}\mathbf{w}_{k+1}, ..., \Sigma\mathbf{w}_d = \lambda_d\mathbf{w}_d$)

$$= \mathbf{w}_{k+1}^T\lambda_{k+1}\mathbf{w}_{k+1} + \mathbf{w}_d^T\lambda_d\mathbf{w}_d$$

$$= \lambda_{k+1} + ... + \lambda_d$$

Determining the number of component

$$\frac{1}{N}\sum_{i=1}^{N}(\mathbf{z}_i - \mathbf{z}_i^k)^T(\mathbf{z}_i - \mathbf{z}_i^k) = \lambda_{k+1} + ... + \lambda_d$$

The mean square error of the truncated representation is equal to the sum of the remaining eigenvalues.

In general: choose k so that $90-95\%$ of the variance of the data is captured.

PCA - Advantages

- Optimal linear dimensionality reduction technique in the mean-square sense;
- Reduce the curse-of-dimensionality;
- Computational overhead of subsequent processing stages is reduced;
- Noise may be reduced;
- A projection into a subspace of a very low dimensionality, e.g. two dimensions, is useful for visualizing the data.

Acknowledgements

Slides adapted from Dr. Carlotta Domeniconi's *Pattern Recognition* at George Mason University.