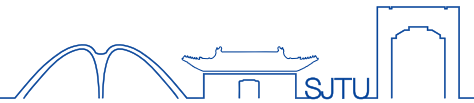
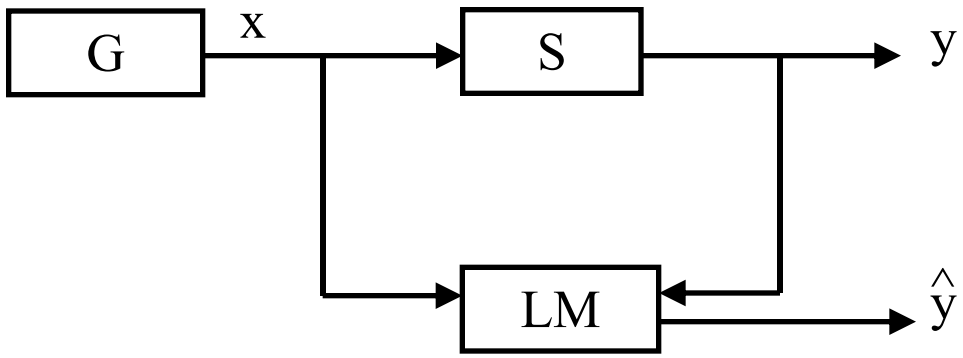
Review:
Statistical Learning Theory & Methods



Lecturer: Liqing Zhang

Dept. Computer Science & Engineering
Shanghai Jiao Tong University

Function Estimation Model



- **Key concepts**: $F(x,y)$, an i.i.d. k-sample on F, functions $f(x,\alpha)$ and the equivalent representation of each f using its index α

The Problem of Risk Minimization

- The **loss functional** (*L, Q)*
  - the error of a given function on a given example

$$\begin{aligned} L:(x,y,f_\alpha) &\mapsto L(y,f(x,\alpha)) \\ Q:(z,\alpha) &\mapsto L(z_y,f(z_x,\alpha)) \end{aligned}$$

- The **risk functional** (*R*)
  - the expected loss of a given function on an example drawn from $F(x,y)$
  - the (usual concept of) generalisation error of a given function

$$R(\alpha)=\int Q(z,\alpha)dF(z)$$

The Problem of Risk Minimization

- Three Main Learning Problems
  - Pattern Recognition:

$$y \in \{0,1\} \text{ and } L(y, f(x,\alpha)) = \mathbf{1}[y = f(x,\alpha)]$$

  - Regression Estimation:

$$y \in \Re \text{ and } L(y, f(x,\alpha)) = (y - f(x,\alpha))^2$$

  - Density Estimation:

$$y \in [0,1] \text{ and } L(p(x,\alpha)) = -\log p(x,\alpha)$$

General Formulation

- The Goal of Learning
  - Given an i.i.d. k-sample $z_1,\ldots, z_k$ drawn from a fixed distribution $F(z)$
  - For a function class' loss functionals $Q(z, \alpha)$, with α in Λ
  - To **minimise the risk**, finding a function α^*

$$\alpha^* = \arg\min_{\alpha \in \Lambda} R(\alpha)$$

General Formulation

- The Empirical Risk Minimization (ERM) Inductive Principle
  - Define the **empirical risk** (sample/training error):

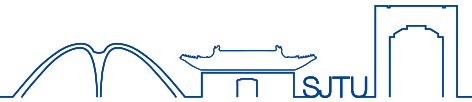
$$R_{\text{emp}}(\alpha) = \frac{1}{k}\sum_{i=1}^{k} Q(z_i, \alpha)$$

  - Define the empirical risk minimiser:

$$\alpha_k = \arg\min_{\alpha \in \Lambda} R_{\text{emp}}(\alpha)$$

  - ERM approximates $Q(z, \alpha^*)$ with $Q(z, \alpha_k)$ the R_{emp} minimiser…that is ERM approximates α^* with α_k
  - Least-squares and Maximum-likelihood are realisations of ERM

Overview of the Course



- Introduction
- Overview of Supervised Learning
- Linear Method for Regression and Classification
- Basis Expansions and Regularization
- Kernel Methods
- Model Selections and Inference
- Support Vector Machine
- Unsupervised Learning
- Latent Dirichlet Allocation
- Deep Networks and Regularization

Key Points in SL

- Curse of dimensionality
  - How many samples are necessary for estimating one dimensional pdf in general
  - How many samples are good for 10-d pdf estimation
  - Pseudo orthogonality for any two high-dimensional vectors
- Regression function

$$EPE(f) = E[Y - f(X)]^2$$

极小解为：

$$f(x) = E\ (\mathrm{Y} \mid X = x)$$

Error Decomposition

The generalization error is decompozed into model bias and model variance

$$
\begin{aligned}
EPE(x_0) &= E_{T,y_0|x_0}[y_0 - \hat{y}_0]^2 \\
&= E_T[f(x_0) - E[\hat{y}_0] + E[\hat{y}_0] - \hat{y}_0]^2 + \sigma_\varepsilon^2 \\
&= E_T[\hat{y}_0 - E_T(\hat{y}_0)]^2 + E_T[E(\hat{y}_0) - f(x_0)]^2 \\
&\quad + 2E\{[\hat{y}_0 - E_T(\hat{y}_0)][E(\hat{y}_0) - f(x_0)]\} + \sigma_\varepsilon^2 \\
&= Var_T(\hat{y}_0) + Bias^2(\hat{y}_0) + \sigma_\varepsilon^2
\end{aligned}
$$

Key Points in Linear Regression

- Model is $f(\mathbf{x}_i) = \beta_0 + \sum_{j=1}^{P-1} x_{ij}\beta_j = \bar{\mathbf{x}}_i^T \beta$
- MSE:

$$\hat{\beta} = \arg\min(\mathrm{Y} - \mathbf{X}\beta)^T (Y - \mathbf{X}\beta)$$

Solution is $\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$

$$\mathrm{Cov}(\hat{\beta}) = (\mathbf{X}^T\mathbf{X})^{-1}\sigma^2$$

- The variance of the predicted model

$$Var(\hat{y}) = \mathbf{x}^T(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{x}\sigma^2, \qquad \hat{\sigma}^2 = \frac{1}{N-p-1}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Key Points in Linear Regr.

- How to formulate the regularization for Linear Regression?
  - Data -- subset selection
  - Model
  - Bootstrap

Shrinkage methods

Ridge regression

- The ridge estimator is defined by

$$\hat{\beta}^{ridge} = \arg\min(\mathrm{Y} - \mathbf{X}\beta)^T (Y - \mathbf{X}\beta) + \lambda\beta^T\beta$$

- Equivalently,

$$\hat{\beta}^{ridge} = \arg\min(\mathrm{Y} - \mathbf{X}\beta)^T (\mathrm{Y} - \mathbf{X}\beta)$$

$$\text{subject to } \sum \beta_j^2 \le s$$

Ridge regression

- Ridge solution:

$$\hat{\beta}_\lambda = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathrm{Y}$$

- Singular value Decomposition:

$$\mathbf{X}=\mathbf{U}\mathbf{D}\mathbf{V}^T; \quad \mathbf{D} \text{ is a diagonal matrix with}$$

$$d_1 \geq d_2 \geq d_3 \geq \ldots \geq d_p \geq 0$$

- For Ordinary Regression

$$\mathbf{X}\hat{\beta}^{\mathrm{ls}}=\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathrm{Y}=\mathbf{U}\mathbf{U}^T\mathrm{Y}$$

The Lasso

- The lasso (least absolute shrinkage and selection operator) is a shrinkage method like ridge, but acts in a nonlinear manner on the outcome y.
- The lasso is defined by

$$\hat{\beta}^{lasso} = \arg\min(\mathrm{Y} - \mathbf{X}\beta)^T(\mathrm{Y} - \mathbf{X}\beta)$$

$$\text{subject to } \sum_{j=1}^{p}\left|\beta_j\right| \le t$$

- No constraint on β_0

Q&A

- To prove that the norm of regression solution $\hat{\beta}=\left(\mathbf{X}^T\mathbf{X}+\lambda\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y}$ will not increase as λ increases.

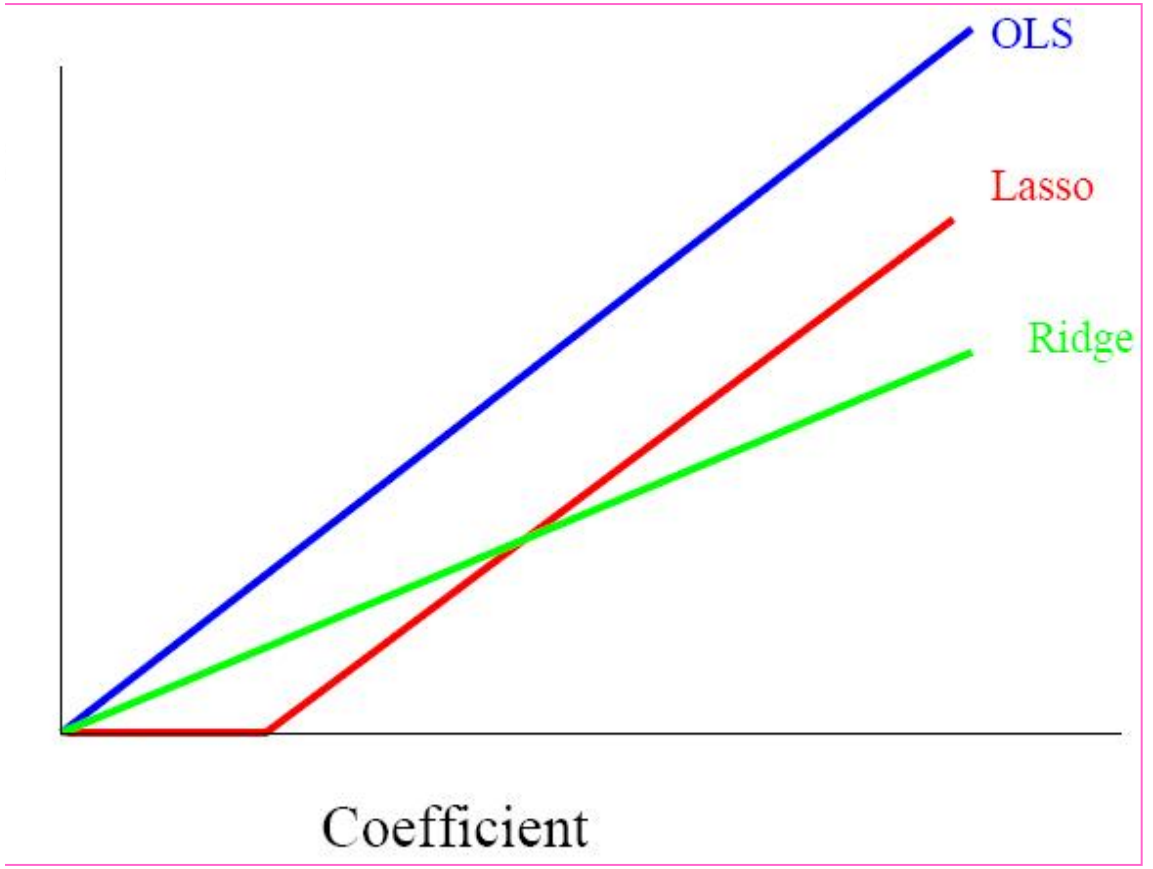
$$\hat{\beta}_\lambda = \arg\min(\mathrm{Y}-\mathbf{X}\beta)^T(Y-\mathbf{X}\beta)+\lambda\|\beta\|_2^2$$

- How about the LASSO regression?

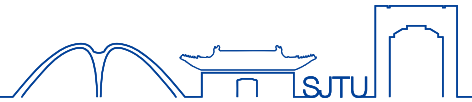
$$\hat{\beta}_\lambda = \arg\min(\mathrm{Y}-\mathbf{X}\beta)^T(Y-\mathbf{X}\beta)+\lambda\|\beta\|_1$$

The Lasso

- The parameter t should be adaptively chosen to minimize an estimate of expected, using say cross-validation
- *Ridge vs Lasso:* if inputs are orthogonal,
  - ridge *multiplies* least squares coefficients by a constant < 1,
  - lasso *translates* them towards zero by a constant, truncating at zero.



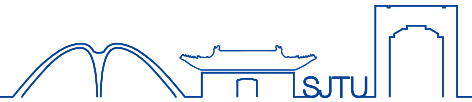
Linear Regression Summary



- How to use LR methods appropriately.
- How to evaluate the performance of LR methods
  - Confidence Interval
  - MSE / Generalization
- How to improve the generalization performance
  - Data: Feature selection (based on p-value)； Cross-validation
  - Shrinkage: Ridge; Lasso; PC regression; Partial least squares

- What is the purpose for imposing constraints on models?

Linear Classification



- Linear and Quadratic Discriminant Functions
- Reduced Rank Linear Discriminant Analysis
- Logistic Regression
- Separating Hyperplanes

Linear Discriminant Analysis

- According to the Bayes optimal classification mentioned in chapter 2, the posteriors is needed.

  post probability : $\Pr(G \mid X)$

  Assume:

  $f_k(x)$ —— condition-density of $\mathbf{X}$ in class $G=k$.

  π_k —— prior probability of class k, with $\sum_{k=1}^{K} \pi_k = 1$

  Bayes theorem give us the discriminant:

$$\Pr(G = k \mid X = x) = \frac{f_k(x)\pi_k}{\sum_{l=1}^{K} f_l(x)\pi_l}$$

Linear Discriminant Analysis

- Multivariate Gaussian density:

$$f_k(x) = \frac{1}{(2\pi)^{p/2}|\Sigma_k|^{1/2}} e^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)}$$

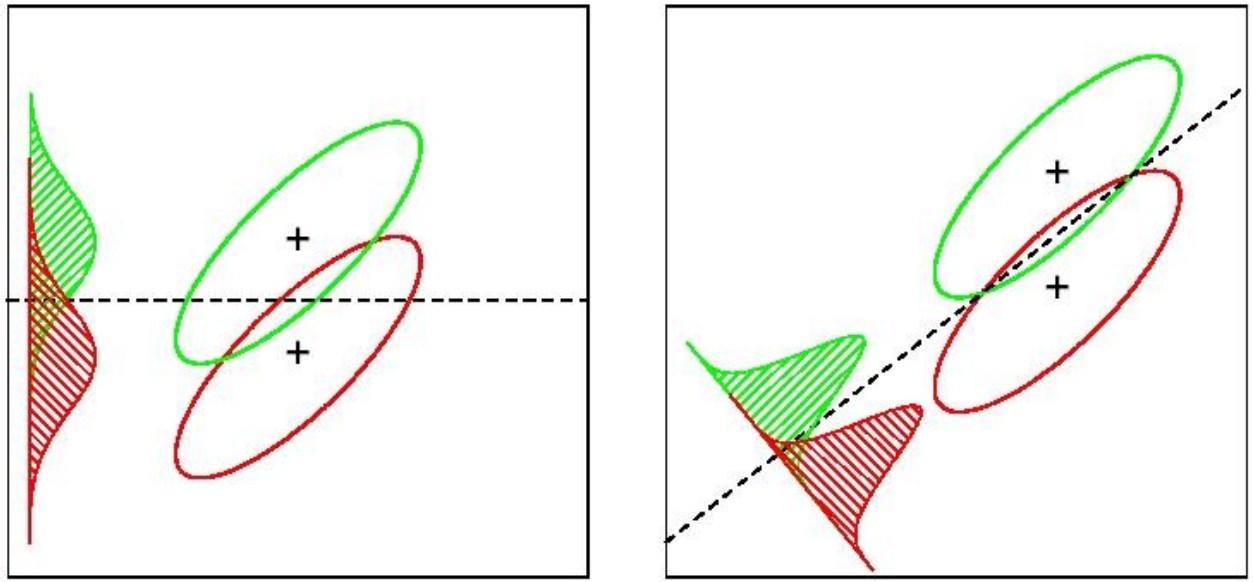
- Comparing *pdfs* of two classes k and l, assume $\Sigma_k = \Sigma, \forall k$

$$\log \frac{\Pr(G=k \mid X=x)}{\Pr(G=l \mid X=x)} = \log \frac{f_k(x)}{f_l(x)} + \log \frac{\pi_k}{\pi_l}$$

$$= \log \frac{\pi_k}{\pi_l} - \frac{1}{2}(\mu_k + \mu_l)\Sigma^{-1}(\mu_k - \mu_l)$$

$$+ x^T \Sigma^{-1}(\mu_k - \mu_l)$$

Reduced Rank LDA



- Although the line joining the centroids defines the direction of greatest centroid spread, the projected data overlap because of the covariance (left panel).
- The discriminant direction minimizes this overlap for Gaussian data (right panel).

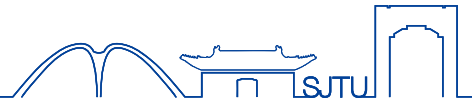
Reduced Rank LDA

降秩线性判别分析的计算过程描述如下：

1. 计算 $K \times p$ 类形心矩阵 M 和公共协方差矩阵 W;
2. 使用 W 的特征值分解计算 $W = \hat{\Sigma}$, and $M^* = MW^{-1/2}$;
3. 计算 M^* 的协方差矩阵 B^* 和它的特征值分解 $B^* = V^* D_B V^{*T}$;
4. 利用投影 $Z_l = v_l^T X$ with $v_l = W^{-1/2} v_l^*$ 最佳子空间坐标。

- Can project data onto K-1 dim subspace spanned by $\hat{U}_1^*, \cdots, \hat{U}_{K-1}^*$, and lose nothing!
- Can project even lower dim using principal components of $\hat{U}_k^*$, $k=1, \ldots, M(<K)$.

Overview of the Course



- Introduction
- Overview of Supervised Learning
- Linear Method for Regression and Classification
- Basis Expansions and Regularization
- Kernel Methods
- Model Selections and Inference
- Support Vector Machine
- Unsupervised Learning
- Latent Dirichlet Allocation
- Deep Networks and Regularization

Smoothing Splines

- Base on the spline basis method: $f(x) = \sum_{k=1}^{N} \beta_k h_k(x)$
- $y = \sum_{k=1}^{m} \beta_k h_k(x) + \varepsilon$, ε is the noise.
- Minimize the penalized residual sum of squares

$$RSS(f, \lambda) = \sum_{i=1}^{N} \{y_i - f(x_i)\}^2 + \lambda \int \{f''(t)\}^2 dt$$

λ is a fixed smoothing parameter

$\lambda = 0$: f can be any function that interpolates the data

$\lambda = \infty$: the simple least squares line fit

Smoothing Splines

- The solution is a natural spline: $f(x) = \sum_{j=1}^{N} N_j(x)\theta_j$
- Then the criterion reduces to:

$$RSS(\theta, \lambda) = (\mathbf{y} - \mathbf{N}\theta)^T (\mathbf{y} - \mathbf{N}\theta) + \lambda \theta^T \Omega_N \theta$$

  - where $N = \{N_j(x_i)\}; \quad \Omega_{Nij} = \int N_i''(t) N_j''(t) dt$

- So the solution:

$$\hat{\theta} = (\mathbf{N}^T \mathbf{N} + \lambda \Omega_N)^{-1} \mathbf{N}^T y$$

- The fitted smoothing spline:

$$\hat{f}(x) = \sum_{j=1}^{N} N_j(x) \hat{\theta}_j$$

Spaces of Functions Generated by Kernel

- Important subclass are generated by the positive kernel $K(x,y)$.
- The corresponding space of functions H_k is called reproducing kernel Hilbert space.
- Suppose that K has an eigen-expansion

$$K(x,y)=\sum_{i=1}^{\infty}\gamma_i\phi_i(x)\phi_i(y),\quad \gamma_i\geq 0,\sum_{i=1}^{\infty}\gamma_i^2<\infty$$

- Elements of H have an expansion

$$f(x)=\sum_{i=1}^{\infty}c_i\phi_i(x),\quad \|f\|_{H_k}^2\doteq\sum_{i=1}^{\infty}c_i^2/\gamma_i<\infty$$

*

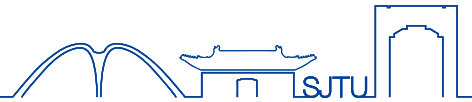
Spaces of Functions Generated by Kernel

- Reproducing properties of kernel function

$$< K(\bullet, x_i), f >_{H_K} = f(x_i), \quad < K(\bullet, x_i), K(\bullet, x_j) >_{H_K} = K(x_i, x_j)$$

$$f(x) = \sum_{i=1}^{\infty} c_i \phi_i(x), \quad \|f\|_{H_k}^2 \triangleq \sum_{i=1}^{\infty} c_i^2 / \gamma_i < \infty$$

Overview of the Course



- Introduction
- Overview of Supervised Learning
- Linear Method for Regression and Classification
- Basis Expansions and Regularization
- Kernel Methods
- Model Selections and Inference
- Support Vector Machine
- Unsupervised Learning
- Latent Dirichlet Allocation
- Deep Networks and Regularization

Local Linear Regression

- Locally weighted linear regression make a first-order correction
- Separate weighted least squares at each target point x_0:

$$\min_{\alpha(x_0),\beta(x_0)} \sum_{i=1}^{N} K_\lambda(x_0, x_i)[y_i - \alpha(x_0) - \beta(x_0)x_i]^2$$

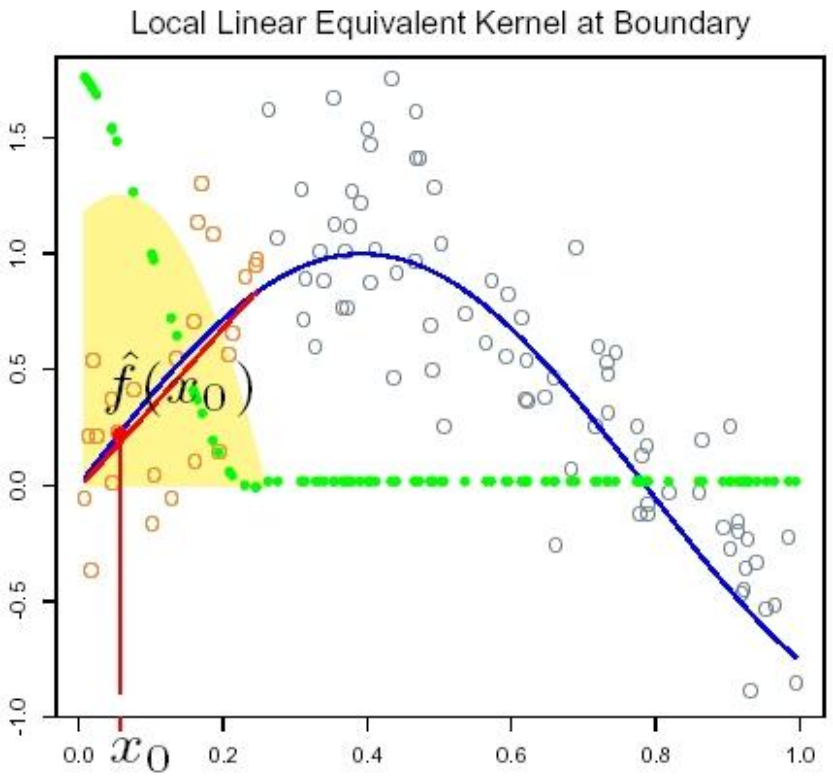
- The estimate: $\hat{f}(x_0) = \hat{\alpha}(x_0) + \hat{\beta}(x_0)x_0$
- $b(x)^T=(1,x)$; B: Nx2 regression matrix with i-th row $b(x)^T$;

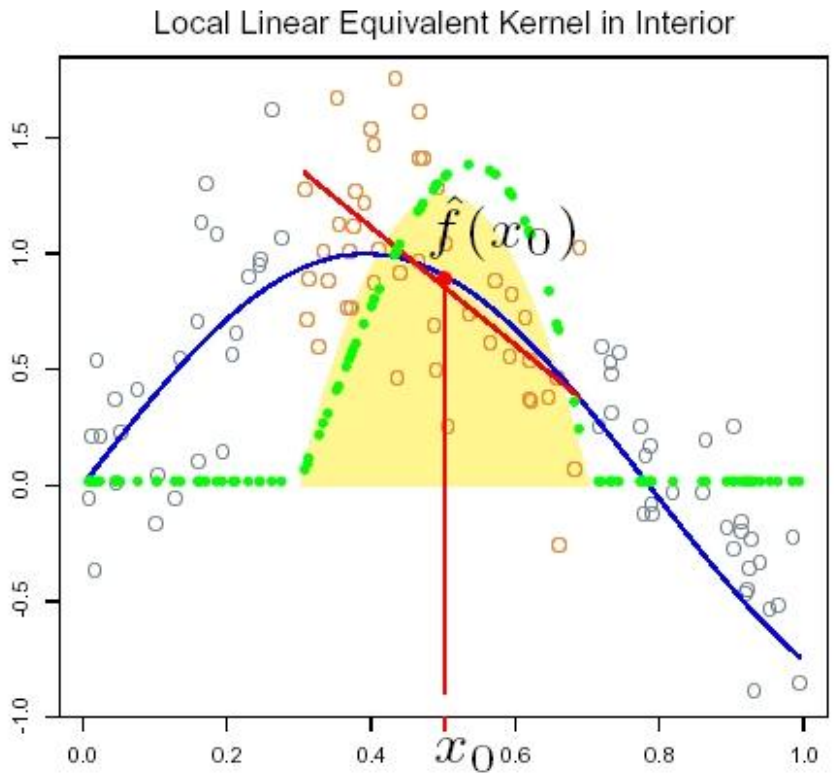
$$W_{N\times N}(x_0) = diag\big(K_\lambda(x_0, x_i)\big), i = 1, \ldots, N$$

$$\hat{f}(x_0) = b(x_0)^T (B^T W(x_0) B)^{-1} B^T W(x_0) y = \sum_{i=1}^{N} l_i(x_0) y_i$$

Local Linear Regression

- The weights $l_i(x_0)$ combine the weighting kernel $K_\lambda(x_0,\cdot)$ and the least squares operations——Equivalent Kernel





Local Linear Regression

- The expansion for $E\hat{f}(x_0)$, using the linearity of local regression and a series expansion of the true function f around x_0

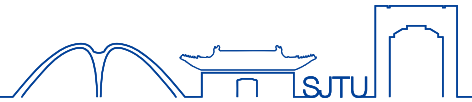
$$E\hat{f}(x_0) = \sum_{i=1}^{N} l_i(x_0) f(x_i) = f(x_0)\sum_{i=1}^{N} l_i(x_0) + f'(x_0)\sum_{i=1}^{N}(x_i - x_0) l_i(x_0)$$
$$+ \frac{f''(x_0)}{2}\sum_{i=1}^{N}(x_i - x_0)^2 l_i(x_0) + R$$

$$\sum_{i=1}^{N} l_i(x_0) = 1, \quad \sum_{i=1}^{N}(x_i - x_0) l_i(x_0) = 0$$

For local regression

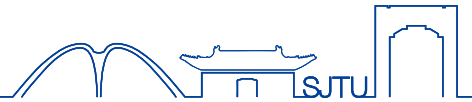
- The bias $E\hat{f}(x_0) - f(x_0)$ depends only on quadratic and higher-order terms in the expansion of f.

Overview of the Course



- Introduction
- Overview of Supervised Learning
- Linear Method for Regression and Classification
- Basis Expansions and Regularization
- Kernel Methods
- Model Selections and Inference
- Support Vector Machine
- Unsupervised Learning
- Latent Dirichlet Allocation
- Deep Networks and Regularization

Outline



- Bias, Variance and Model Complexity
- The Bias-Variance Decomposition
- Optimism of the Training Error Rate
- Estimates of In-Sample Prediction Error
- The Effective Number of Parameters
- The Bayesian Approach and BIC
- Minimum Description Length
- Vapnik-Chernovenkis Dimension
- Cross-Validation
- Bootstrap Methods

Bias-Variance Decomposition

- Basic Model: $Y = f(X) + \varepsilon, \quad \varepsilon \sim N(0, \sigma_\varepsilon^2)$
- The expected prediction error of a regression fit $\hat{f}(X)$

$$
\begin{aligned}
Err(x_0) &= E[(Y - \hat{f}(x_0))^2 \mid X = x_0] \\
&= \sigma_\varepsilon^2 + [E\hat{f}(x_0) - f(x_0)]^2 + E[E\hat{f}(x_0) - \hat{f}(x_0)]^2 \\
&= \sigma_\varepsilon^2 + Bias(\hat{f}(x_0))^2 + Var(\hat{f}(x_0)) \\
&= \text{Irreducible Error} + \text{Bias}^2 + \text{Variance}
\end{aligned}
$$

- The more complex the model, the lower the (squared) bias but the higher the variance.

Bias-Variance Decomposition

- For the k-NN regression fit the prediction error:

$$Err(x_0) = E[(Y - \hat{f}(x_0))^2 | X = x_0]$$
$$= \sigma_\varepsilon^2 + [f(x_0) - \frac{1}{k}\sum_{j=1}^{k} f(x_j)]^2 + \sigma_\varepsilon^2 / k$$

$$\hat{f}(x_0) = \frac{1}{k} \sum_{x_j \in N(x_0)} y_j,$$

- The in-sample error of the Linear Model

$$\frac{1}{N}\sum_{i=1}^{N} Err(x_i) = \sigma_\varepsilon^2 + \frac{1}{N}\sum_{i=1}^{N}[f(x_i) - E\hat{f}(x_i)]^2 + \frac{p}{N}\sigma_\varepsilon^2$$

  - The model complexity is directly related to the number of parameters p.

Optimism of the Training Error Rate

- Training Error < True Error

$$\text{Training Error} \quad \overline{err} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}(x_i))$$

$$\text{True Error} \quad Err = E[L(Y, \hat{f}(X)]$$

- Err is extra-sample error
- The in-sample error

$$Err_{in} = \frac{1}{N}\sum_{i=1}^{N} E_{Y^{new}}\left[L(Y_i^{new}, \hat{f}(x_i)) | \mathbb{T} \right]$$

- Optimism:

$$op \equiv Err_{in} - \overline{err}$$

Optimism of the Training Error Rate

- The average optimism is the expectation of the optimism over training sets

$$\omega \equiv \mathrm{E}_y(op) \equiv \mathrm{E}_y(Err_{in} - \overline{err})$$

$$\text{Training Error} \quad \overline{err} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}(x_i))$$

$$\text{In sample error:} \quad Err_{in} = \frac{1}{N}\sum_{i=1}^{N} E_{Y^{new}}\left[L(Y_i^{new}, \hat{f}(x_i)) | \mathbb{T} \right]$$

Akaike Information Criterion

- For the logistic regression model, using the binomial log-likelihood.

$$\mathrm{AIC} = -\frac{2}{N} E[\log lik] + 2\frac{d}{N}$$

- For Gaussian model the **AIC** statistic equals to the $\mathbf{C_p}$ statistic.

$$\mathrm{AIC} = C_p = \overline{err} + 2\frac{d}{N}\hat{\sigma}_\varepsilon^2$$

Bayesian Approach & BIC

- The Bayesian Information Criterion (BIC)

$$\mathrm{BIC} = -2loglik + (\log N)d$$

- Gaussian model:

$$\mathrm{BIC} = \overline{err} + (\log N)\frac{d}{N}\sigma_\varepsilon^2$$

$$\mathrm{AIC} = C_p = \overline{err} + 2\frac{d}{N}\hat{\sigma}_\varepsilon^2$$

*

Cross Validation

- Denote the fitted function by $\hat{f}^{-k}(x)$ with removing k-th fold data. Then the cross-validation estimate of prediction error is

$$\mathrm{CV}(\hat{f}) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}^{-k}(x_i))$$

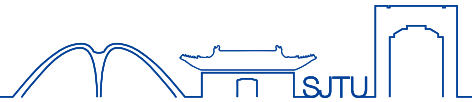
- The case K = N, *leave-one-out* cross-validation.
- Given model family $f(x,\alpha)$ indexed by a tuning parameter α.
- $\hat{f}^{-k}(x,\alpha)$: fited with the kth part of the data removed.

$$\mathrm{CV}(\hat{f},\alpha) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}^{-k}(x_i,\alpha))$$

The Correct Way to Do CV

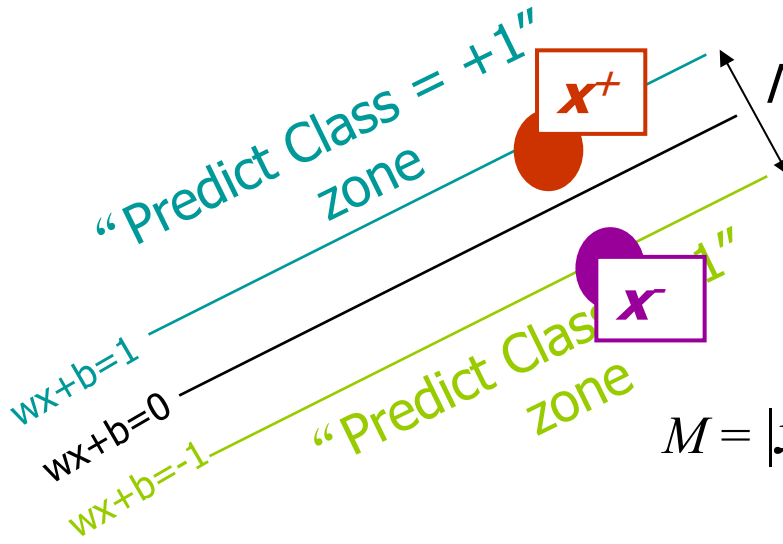
- A typical strategy for CV might be as follows:
  - Divide the samples into K cross-validation folds (groups) at random.
  - For each fold k = 1, 2, . . . ,K
    - Find a subset of “good” predictors that show fairly strong (univariate) correlation with the class labels, using all of the samples except those in fold k.
    - Using just this subset of predictors, build a multivariate classifier, using all of the samples except those in fold k.
    - Use the trained classifier to predict the class labels for the samples in fold k.

Overview of the Course



- Introduction
- Overview of Supervised Learning
- Linear Method for Regression and Classification
- Basis Expansions and Regularization
- Kernel Methods
- Model Selections and Inference
- Support Vector Machine
- Unsupervised Learning
- Latent Dirichlet Allocation
- Deep Networks and Regularization

Computing the margin width



What we know:

- $\boldsymbol{w} \cdot \boldsymbol{x}^+ + b = +1$
- $\boldsymbol{w} \cdot \boldsymbol{x}^- + b = -1$
- $\boldsymbol{x}^+ = \boldsymbol{x}^- + \lambda \boldsymbol{w}$
- $|\boldsymbol{x}^+ - \boldsymbol{x}^-| = M$
- $\lambda = \dfrac{2}{\mathbf{w}.\mathbf{w}}$

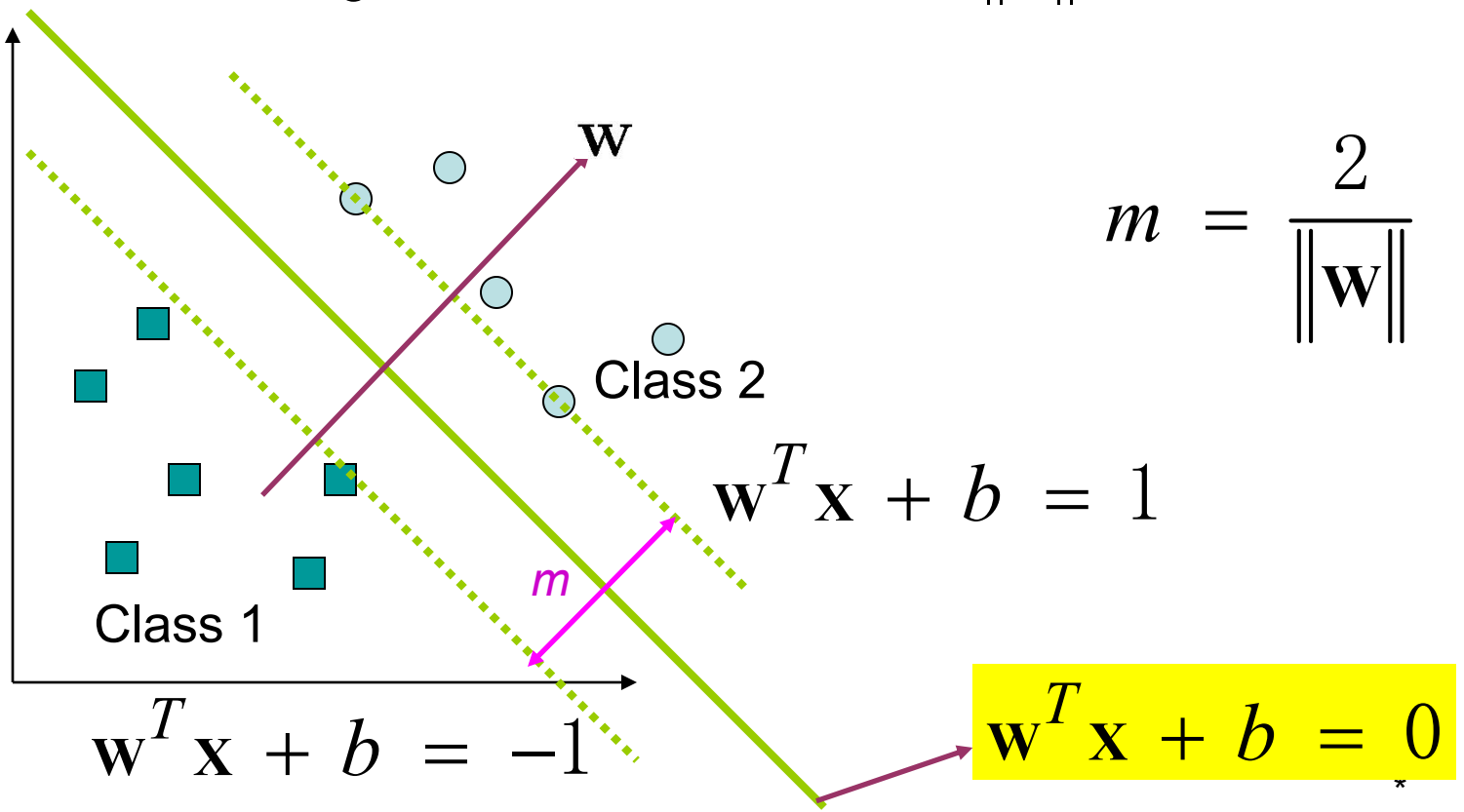
$$M = |\boldsymbol{x}^+ - \boldsymbol{x}^-| = |\lambda \boldsymbol{w}| =$$

$$= \lambda |\mathbf{w}| = \lambda \sqrt{\mathbf{w}.\mathbf{w}}$$

$$= \frac{2\sqrt{\mathbf{w}.\mathbf{w}}}{\mathbf{w}.\mathbf{w}} = \frac{2}{\sqrt{\mathbf{w}.\mathbf{w}}}$$

Large-margin Decision Boundary

- The decision boundary should be as far away from the data of both classes as possible
  - We should maximize the margin, *m*
  - Distance between the origin and the line $\boldsymbol{w}^t\boldsymbol{x}=k$ is $k/\|\mathbf{w}\|$



$$m = \frac{2}{\|\mathbf{w}\|}$$

Finding the Decision Boundary

- Let $\{x_1, ..., x_n\}$ be our data set and let $y_i \in \{1,-1\}$ be the class label of x_i
- The decision boundary should classify all points correctly $\Rightarrow$

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \qquad \forall i$$

- The decision boundary can be found by solving the following constrained optimization problem

$$\text{Minimize } \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \qquad \forall i$$

- This is a constrained optimization problem. Solving it requires some new tools
  - Feel free to ignore the following several slides; what is important is the constrained optimization problem above

*

The Dual Problem

- It is known as the dual problem: if we know **w**, we know all α_i; if we know all α_i, we know **w**
- The objective function of the dual problem needs to be maximized!
- The dual problem is therefore:

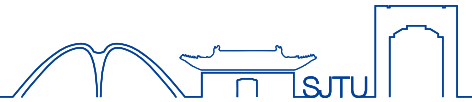
$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } \alpha_i \geq 0, \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

Properties of α_i when we introduce the Lagrange multipliers

The result when we differentiate the original Lagrangian w.r.t. b

Question?



- How to formulate margin maximum problem into a constrained optimal problems
- How to estimate $\boldsymbol{w}$ and $\boldsymbol{b}$?
- How to deal with nonlinear separatable problem?

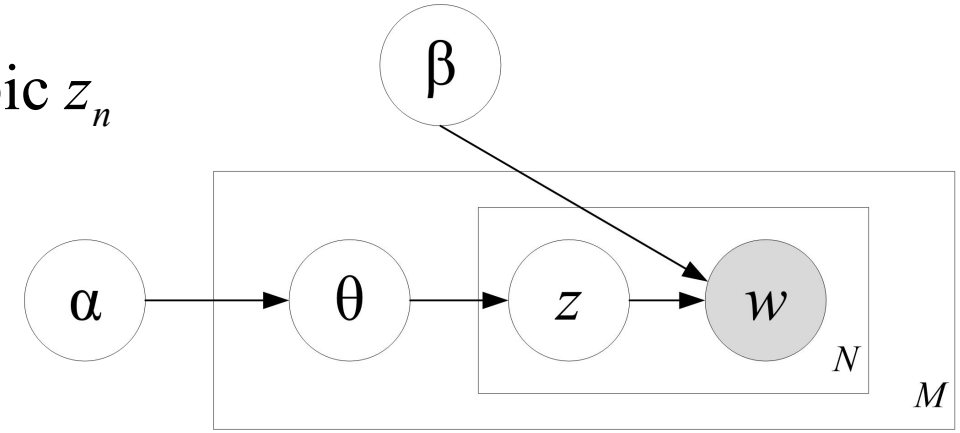
Latent Dirichlet Allocation

- The basic idea: Documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words.
- For each document *d*, we generate as follows:

1. Choose $N \prec \text{Poisson}(\xi)$
2. Choose $\boldsymbol{\theta} \prec \text{Dir}(\boldsymbol{\alpha})$
3. For each of the N words w_n :
   (a) Choose a topic $z_n \prec \text{Multinomial}(\boldsymbol{\theta})$
   (b) Choose a word w_n from $p(w_n \mid z_n, \boldsymbol{\beta})$;
   a multinomial probability conditioned on the topic z_n

k topics **z**

$\boldsymbol{\beta}$ is a $k \times V$ matrix with $\beta_{ij} = p(w_i \mid z_j)$

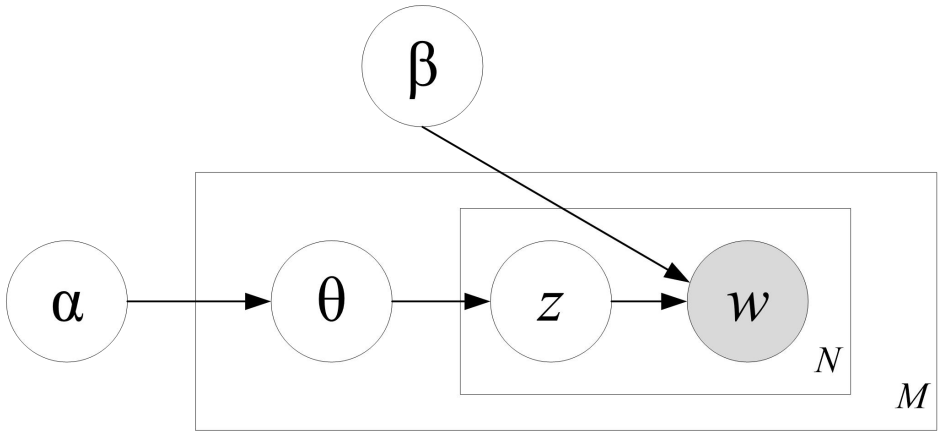


Joint Distribution

- Given the parameters **α** and **β**, the joint distribution of a topic mixture **θ**, a set of N topics **z**, and a set of N words **w** is given by:

$$p(\boldsymbol{\theta}, \mathbf{z}, \mathbf{w} \mid \boldsymbol{\alpha}, \boldsymbol{\beta}) = p(\boldsymbol{\theta} \mid \boldsymbol{\alpha}) \prod_{n=1}^{N} p(z_n \mid \boldsymbol{\theta}) p(w_n \mid z_n, \boldsymbol{\beta}) \quad (1)$$

where $p(z_n|\boldsymbol{\theta})$ is simply θ_i for the unique i such that $z_n^i=1$.



General Latent Variable Model

- $\mathbf{X}$ input data, with log-likelihood $\ell(\theta, \mathbf{X}) = \log p(\mathbf{X} \mid \theta)$
- $\mathbf{Z}$ latent data (in our example Δ_i)
- $\mathbf{T} = (\mathbf{X}, \mathbf{Z})$ complete data with log-likelihood $\ell_0(\theta, \mathbf{T})$

$$\Pr(\mathbf{Z} \mid \mathbf{X}, \theta') = \frac{\Pr(\mathbf{Z}, \mathbf{X} \mid \theta')}{\Pr(\mathbf{X} \mid \theta')}; \qquad \Pr(\mathbf{X} \mid \theta') = \frac{\Pr(\mathbf{T} \mid \theta')}{\Pr(\mathbf{Z} \mid \mathbf{X}, \theta')}$$

General Latent Variable Model

- Evidence Lower Bound (ELBO)

$$\ell(\theta; \mathbf{X}) = \log p(\mathbf{X} \mid \theta) = \log \int p(\mathbf{X}, \mathbf{Z} \mid \theta)\, d\mathbf{Z}$$

$$= \log \int \frac{p(\mathbf{X}, \mathbf{Z} \mid \theta)}{q(\mathbf{Z} \mid \varphi)} q(\mathbf{Z} \mid \varphi)\, d\mathbf{Z}$$

$$= \log \mathrm{E}_{q(Z)} \left[\frac{p(\mathbf{X}, \mathbf{Z} \mid \theta)}{q(\mathbf{Z} \mid \varphi)} \right] \geq \mathrm{E}_{q(Z)} \left[\log \frac{p(\mathbf{X}, \mathbf{Z} \mid \theta)}{q(\mathbf{Z} \mid \varphi)} \right]$$

$$= \mathrm{E}_{q(Z)} \left[\log p(\mathbf{X}, \mathbf{Z} \mid \theta) - \log q(\mathbf{Z} \mid \varphi) \right] = ELBO(\theta, \varphi)$$

If $\phi(x)$ is convex, then $\mathrm{E}[\phi(x)] \geq \phi(E[x])$

General Latent Variable Model (EM algorithm)

$$\ell(\theta; \mathbf{X}) = \boxed{\mathrm{E}_{q(Z)}\left[\log p(\mathbf{X}, \mathbf{Z}\,|\theta) - \log q(\mathbf{Z}\,|\varphi)\right]} + \boxed{KL\left[q(\mathbf{Z}\,|\varphi)\,;\, p(\mathbf{Z}\,|\mathbf{X}, \theta)\right]}$$

1. **Start with** initial params $\theta^{(0)}, \varphi^{(0)}$
2. **Expectation Step**: at the k-th step compute

$$\mathrm{ELBO}(\theta, \varphi) = \mathrm{E}_{q(Z|\varphi^{(k)})}\left[\log p(\mathbf{X}, \mathbf{Z}\,|\theta) - \log q(\mathbf{Z}\,|\varphi)\right]$$

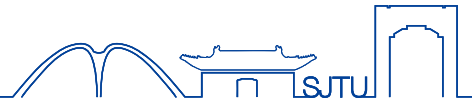
   as a function of the dummy argument θ, φ

3. **Maximization Step**: Determine the new params by maximizing

$$\hat{\theta}^{(k+1)} = \arg\max_{\theta} ELBO(\theta, \varphi^{(k)})\,; \quad \varphi^{(k+1)} = \arg\max_{\varphi} ELBO(\hat{\theta}^{(k+1)}, \varphi)\,;$$

- **Iterate** 2 and 3 until convergence

Unsupervised Learning



- Introduction
- Association Rules & Cluster Analysis
- Self-Organizing Maps
- Principal Components, Curves and Surfaces
- Non-negative Matrix Factorization
- Independent Component Analysis
- Multidimensional Scaling
- Nonlinear Dimension Reduction
- The Google PageRank Algorithm

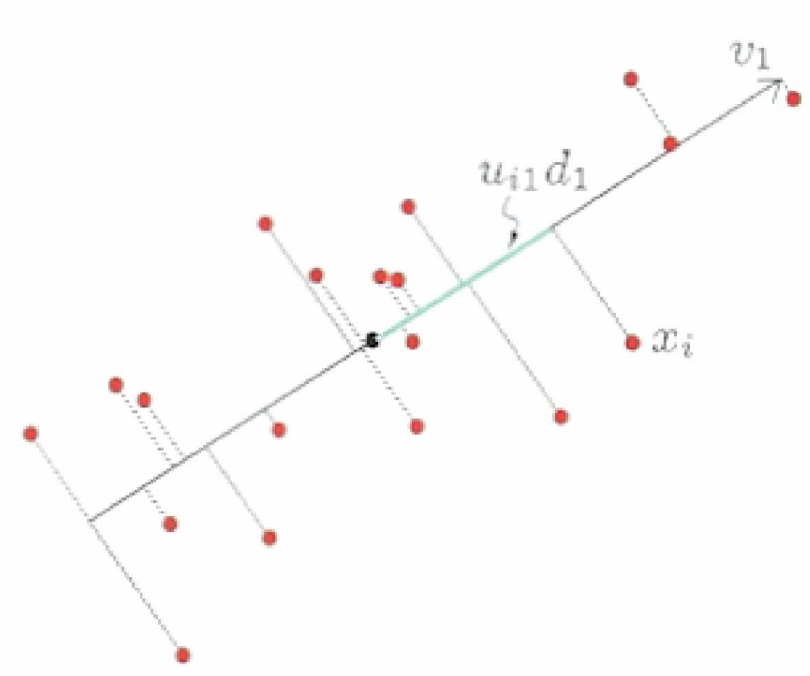
Principal Components (PCA)

Set of q directions towards which p-dimensional data are linearly projected

-> dimension reduction since $q \leq p$

Data : $x_1, x_2, \cdots, x_n$

Model : $f(\lambda) = (\mu + \lambda_i V_q)$



Model fitted by least squares

w.r.t. reconstruction error

$$\{\mu, \lambda_i, V_q\} = \text{var}\min_{\mu,\lambda,V_q} \sum_{i=1}^{n} \left\| x_i - (\mu + \lambda_i V_q) \right\|^2$$

Principal Components (PCA)

Principal Components are computed from matrix operations

1. Given V_q, solve the optimal problem

$$\hat{\mu} = \bar{x}, \quad \hat{\lambda}_i = V_q^T (x_i - \bar{x}).$$

2. Substitute them into original cost function

$$V_q = \text{var min}_{V_q} \sum_{i=1}^{n} \left\| (x_i - \bar{x}) - V_q V_q^T (x_i - \bar{x}) \right\|^2$$

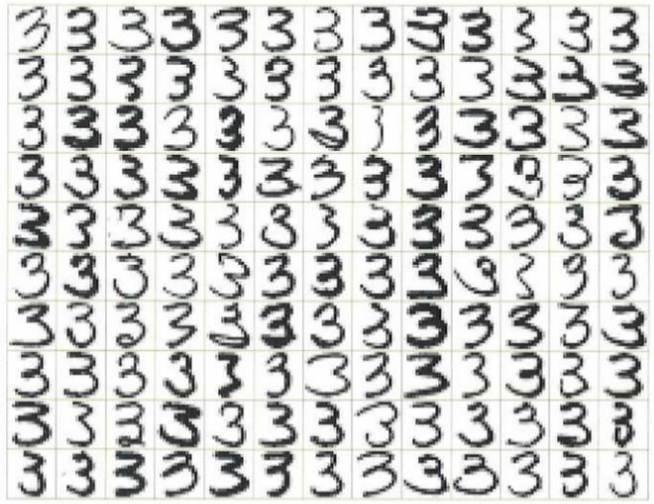
Data are usually centered, and $V_q V_q^T = H_q$ is the projection matrix. Solution V_q are the q first columns of V obtained from

$$\mathbf{X} = \mathbf{UDV}^T = \mathbf{SA}^T; \ \mathbf{U} \ \textit{is} \ \text{N} \times \text{p} \ \text{orthognal matrix} \ \text{U}^T\text{U}=\text{I}$$

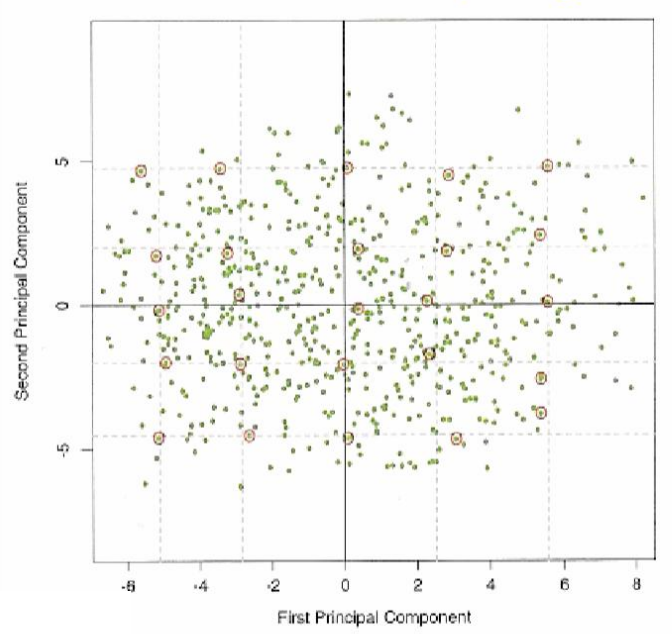
14.5. Principal Components (PCA)

- PCA can be used for handwritten digits

Data in $\mathbb{R}^{256}$



2-D projections



$$\hat{f}(\lambda) = \bar{x} + \lambda_1 v_1 + \lambda_2 v_2$$

$$= \bar{x} + \lambda_1 \cdot v_1 + \lambda_2 \cdot v_2 .$$

$$\hat{f}(\lambda) = \bar{x} + \lambda_1 v_1 + \lambda_2 v_2$$

SVD (Singular Value Decomposition)

- Definition:
  - **Singular value decomposition** is a method of decomposing a matrix into the product of three matrices. For any $m \times n$ matrix $\boldsymbol{A}$, there exist an $m \times m$ orthogonal matrix $\boldsymbol{U}$, an $n \times n$ orthogonal matrix $\boldsymbol{V}$, and an $m \times n$ diagonal matrix $\boldsymbol{D}$ such that

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T$$

- Importance: It has wide applications in many fields such as data compression, image recognition, recommendation systems, and principal component analysis. It is an effective means of handling large-scale data and complex matrix operations.

SVD (Singular Value Decomposition)

- Proof outline:

$$\mathbf{A}^T\mathbf{A} = \mathbf{V}\mathbf{D}\mathbf{V}^T, \qquad (\mathbf{AV})^T\mathbf{AV} = \mathbf{D} = \left[diag(\lambda_1^2, \lambda_2^2, \cdots, \lambda_n^2)\right],$$

$$\mathbf{P} = \mathbf{AV}; \qquad \mathbf{P}^T\mathbf{P} = \mathbf{D}; \qquad \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r; \ \lambda_k = 0, \ k > r$$

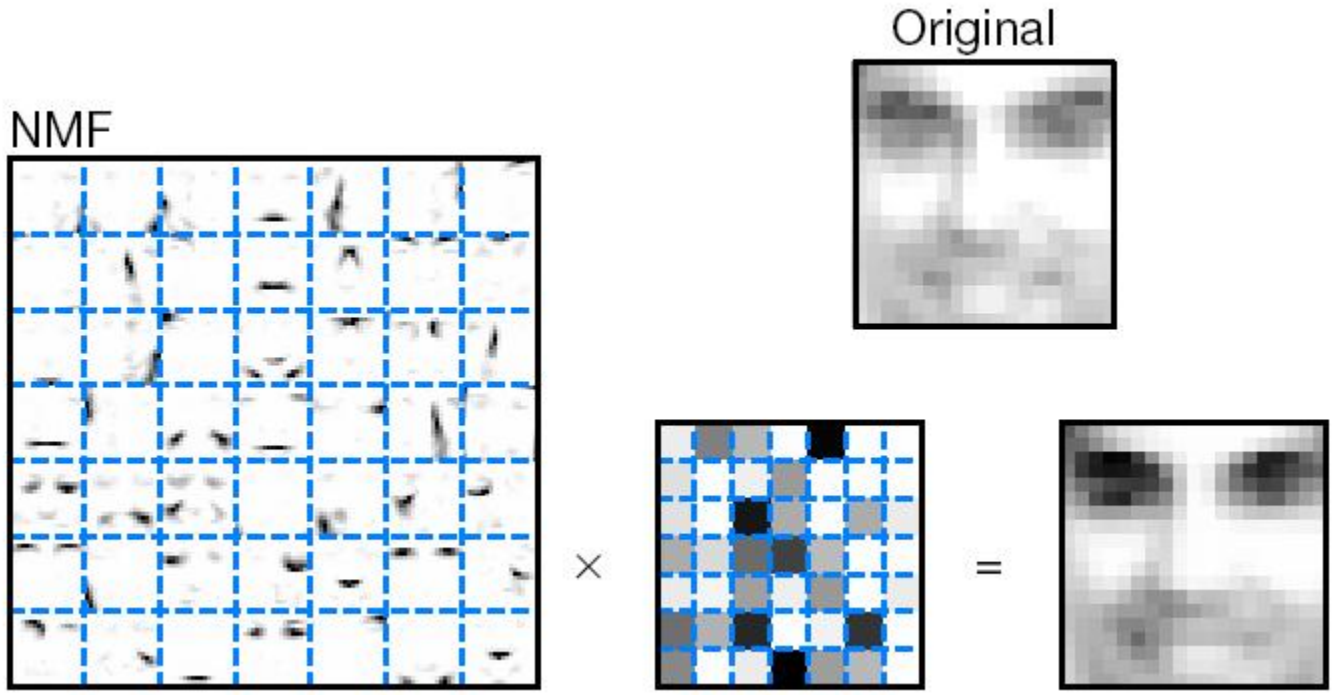
$$\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_n], \qquad \mathbf{u}_j = \mathbf{p}_j / \lambda_j, \ j = 1,2,\cdots,r$$

$$\mathbf{u}_i^T\mathbf{u}_j = \delta_{ij}, \ i, j = 1,2,\cdots,r$$

- Construct the rest vectors $\mathbf{u}_j$, $j > r$ from the null space of $\mathbf{A}^T$ such that $\{\mathbf{u}_j\}_{j=1}^m$ are orthogonal.

Non-negative Matrix Factorization

- Like PCA, except the coefficients in the linear combination cannot be negative



- By constraining the weights, we can control how images are represented into the weighted sum of basis functions

Objective function

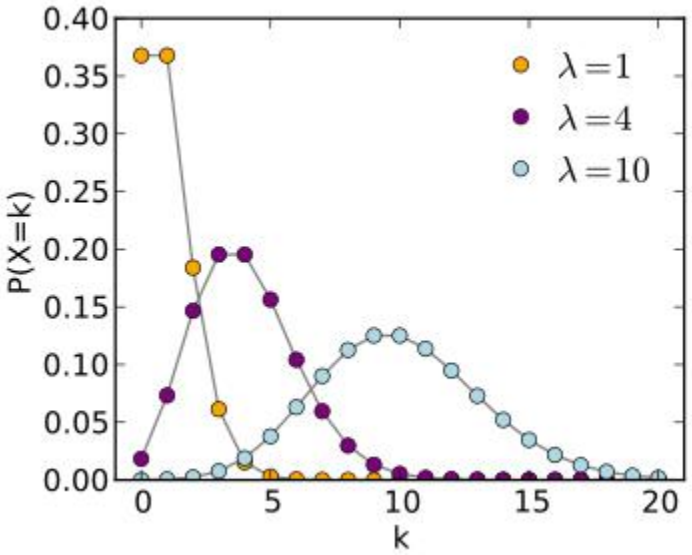
- Let the value of a pixel in an original input image be X.

$$X = WH + \varepsilon, \qquad W \in R^{N \times r}, H \in R^{r \times p}$$

- Let $(WH)_{ij}$ be the reconstructed pixel.
- If we consider X to be a noisy version of $(WH)_{ij}$, then the Poisson PDF of V is

$$P(x \mid (WH)_{ij}) = \frac{(WH)_{ij}^{x} e^{-(WH)_{ij}}}{x!}$$

$$P(x \mid \lambda) = \frac{\lambda^{x} e^{-\lambda}}{x!}$$



Objective function

$$P(x \mid (WH)_{ij}) = \frac{(WH)_{ij}^{x} e^{-(WH)_{ij}}}{x!}$$

- Now we will maximize the log probability of this PDF over W and H, leaving the relevant objective function to be:

$$L(W,H) = \sum_{i=1}^{N}\sum_{j}^{p}\left[x_{ij} \log(WH)_{ij} - (WH)_{ij} \right]$$

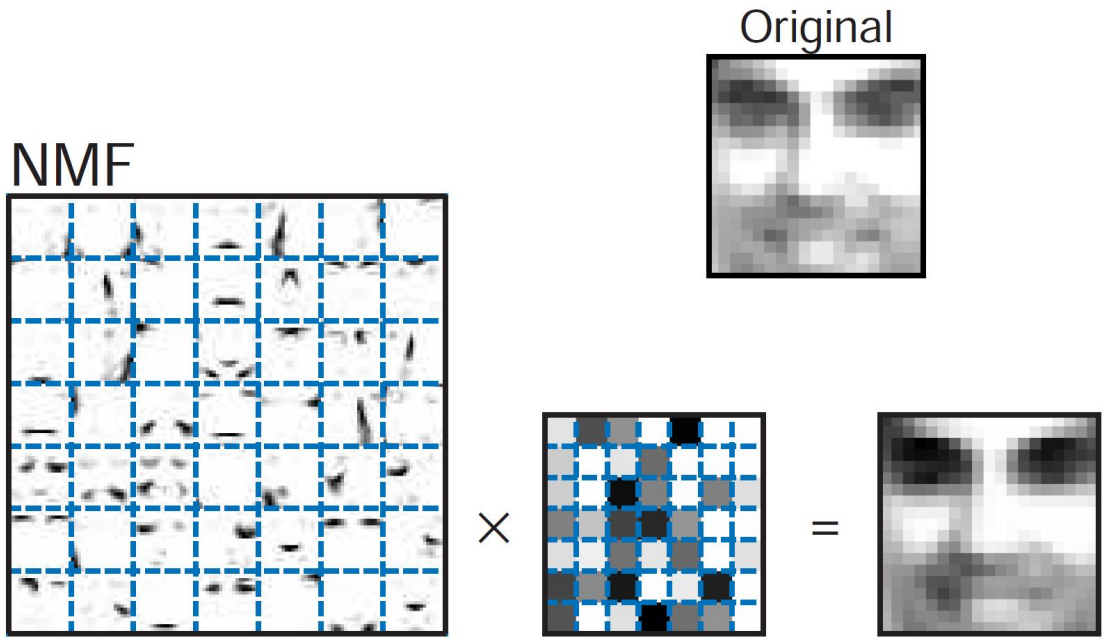
$$W = W + \eta \frac{\partial L(W,H)}{\partial W} = \left(w_{ik} + \eta \sum_{j=1}^{p} x_{ij} h_{kj} \,/\, (WH)_{ij} - \sum_{j=1}^{p} h_{kj} \right)$$

$$H = H + \eta \frac{\partial L(W,H)}{\partial H} = \left(h_{kj} + \eta \sum_{i=1}^{N} x_{ij} w_{ik} \,/\, (WH)_{ij} - \sum_{i=1}^{N} w_{ik} \right)$$

Deriving Update Rules

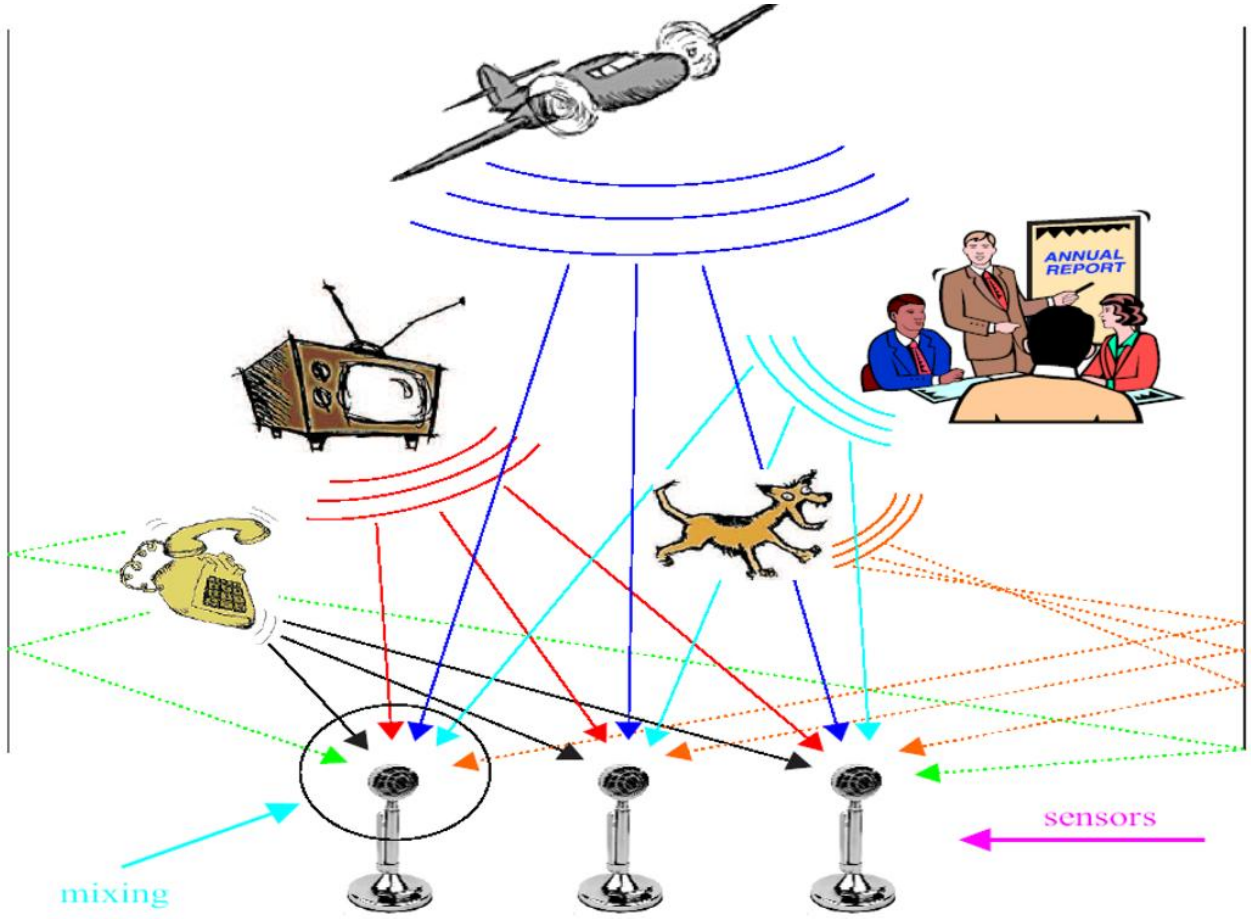
- Gradient Descent Rule： (Alternating algorithm)

$$w_{ik} \leftarrow w_{ik} \frac{\sum_{j=1}^{p} h_{kj}\, x_{ij} / (WH)_{ij}}{\sum_{j=1}^{p} h_{kj}}; \qquad h_{kj} \leftarrow h_{kj} \frac{\sum_{i=1}^{N} w_{ik}\, x_{ij} / (WH)_{ij}}{\sum_{i=1}^{N} w_{ik}}$$



6. Independent Component Analysis (ICA)

- Speech Separation (Cocktail Party Problem)

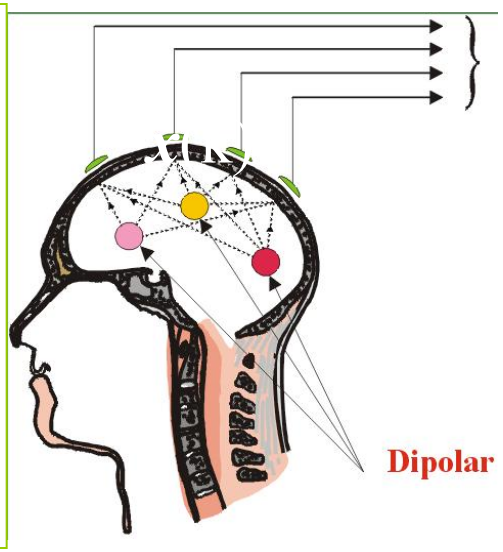


Mathematical Formulation

Mixing Model

$$x_i(k) = a_{i1}s_1(k) + a_{i2}s_2(k) + \cdots + a_{in}s_n(k) + \varepsilon(k)$$

$$\mathbf{x}(k) = \mathbf{A}\mathbf{s}(k) + \boldsymbol{\varepsilon}(k)$$



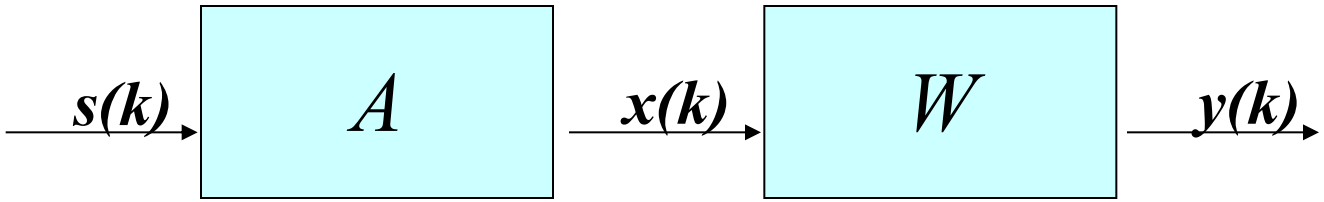
- $\boldsymbol{s}(k) = (s_1(k), \ldots, s_n(k))^T$: the vector of n-source signals;
- $\boldsymbol{x}(k) = (x_1(k), \ldots, x_m(k))^T$: the vector of m-sensor signals;
- $\boldsymbol{\varepsilon}(k)$: the vector of sensor noises.
- $\boldsymbol{A}$ is the mixing matrix.

Demixing Model

Problem: to estimate the source signals (or event-related potentials) by using the sensor signals

$$\mathbf{y}(k) = \mathbf{W}\,\mathbf{x}(k)$$

- $\boldsymbol{y}(k) = (y_1(k), \ldots, y_m(k))^T$: the vector of recovered signals
- **W** is the demixing matrix.



Cost Function

Kullback-Leibler (KL) divergence between

$$p(y_1,\cdots,y_K) \text{ and } \prod_{k=1}^{K} q_k(y_k)$$

$$KL(W) = \int p(\mathbf{y}) \log \frac{p(\mathbf{y})}{\prod_{k=1}^{K} q_k(y_k)} d\mathbf{y}$$

$$= -H(\mathbf{Y};W) + \sum_{k=1}^{K} H(Y_k;W)$$

1. Joint Entropy of $\boldsymbol{y}$

2. Sum of marginal entropy of y_k

- Minimized when y_k are mutually independent

Cost Function

Assume that $\mathbf{y} = \mathbf{W}\mathbf{x}$, the differential entropy can be expressed by

$$H(\mathbf{y}) = H(\mathbf{x}) + \log\left|\det(\mathbf{W})\right|$$

$$KL(W) = \int p(\mathbf{y}) \log \frac{p(\mathbf{y})}{\prod_{k=1}^{K} p(y_k)} d\mathbf{y}$$

$$= -H(\mathbf{Y}; W) + \sum_{k=1}^{K} H(Y_k; W)$$

$$= -H(x) - \log\left|\det(W)\right| - \sum_{k=1}^{K} E\left[\log(p_k(y_k))\right]$$

Gradient Descent

To update $\boldsymbol{W}$ along the negative gradient of $KL(W)$

$$\Delta \boldsymbol{W} \propto -\frac{\partial KL(\boldsymbol{W})}{\partial \boldsymbol{W}} = \left((\boldsymbol{W}^{\mathrm{T}})^{-1} - \int p(x)\boldsymbol{\phi}(\boldsymbol{y})\boldsymbol{x}^{\mathrm{T}} d\boldsymbol{x}\right)$$

$$= \left((\boldsymbol{W}^{\mathrm{T}})^{-1} - \mathrm{E}_x\left[\boldsymbol{\phi}(\boldsymbol{y})\boldsymbol{x}^{\mathrm{T}}\right]\right)$$

$$= \left(\boldsymbol{I} - \mathrm{E}_y\left[\boldsymbol{\phi}(\boldsymbol{y})\boldsymbol{y}^{\mathrm{T}}\right]\right)\left(\boldsymbol{W}^{\mathrm{T}}\right)^{-1}$$

Nonlinear Function 2 ⇒ To be diagonalized

where

$$\boldsymbol{\phi}(\boldsymbol{y}) \equiv \left[\frac{\partial \log p(y_1)}{\partial y_1}, \ldots, \frac{\partial \log p(y_K)}{\partial y_K}\right]^{\mathrm{T}}$$

This can be approximated by Sigmoid Function in speech signal.

Gradient Descent

To update W along the negative gradient of $KL(W)$

$$\Delta \mathbf{W} = \alpha \left(\mathbf{I} - \mathrm{E}_y \left[\varphi(\mathbf{y}) \mathbf{y}^{\mathrm{T}} \right] \right) \left(\mathbf{W}^{\mathrm{T}} \right)^{-1}$$

Modified Learning Algorithm

$$\phi(\boldsymbol{y}) \equiv \left[\frac{\partial \log p(y_1)}{\partial y_1}, \ldots, \frac{\partial \log p(y_K)}{\partial y_K} \right]^{\mathrm{T}}$$

$$\Delta \mathbf{W} = \alpha \left(\mathbf{I} - \mathrm{E}_y \left[\varphi(\mathbf{y}) \mathbf{y}^{\mathrm{T}} \right] \right) \mathbf{W}$$

The natural gradient of Nonsingular Matrix Manifold

$$\nabla l(\mathbf{W}) = \frac{\partial l(\mathbf{W})}{\partial \mathbf{W}} \mathbf{W}^T \mathbf{W}$$

ICA theoretical problems

- Convergence of gradient descent $\Delta \mathbf{W} = \alpha \left(\mathbf{I} - \mathrm{E}_y \left[\varphi(\mathbf{y}) \mathbf{y}^{\mathrm{T}} \right] \right) \mathbf{W}$
  - Depends on the choice of activation function
  - For super Gaussian, $\varphi(y) = \tanh(y)$
  - For sub Gaussian, $\varphi(y) = y^3$
- Adaptation of activation function
  - Using generalized Gaussian family

$$f(x) = \frac{c_p}{\alpha} \exp\left(-\frac{|x-\mu|^p}{2\alpha^p} \right), \quad c_p = \frac{p}{2^{(p+1)/p}\,\Gamma(1/p)}, \quad \text{p>0}$$

- Determination of the number of ICs

Unsupervised Learning

1. Introduction
2. Association Rules & Cluster Analysis
4. Self-Organizing Maps
5. Principal Components, Curves and Surfaces
6. Non-negative Matrix Factorization
7. Independent Component Analysis
8. **Multidimensional Scaling**
9. **Nonlinear Dimension Reduction**
10. **The Google PageRank Algorithm**

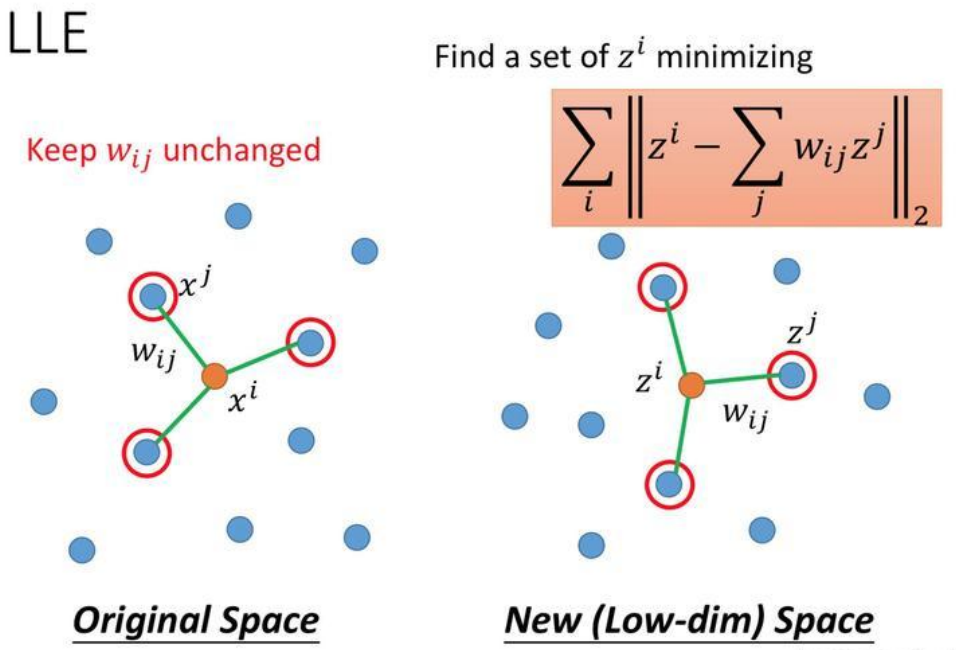
Local linear embedding

- For each data point x_i in p dimensions, we find its K-nearest neighbors $N(i)$ in Euclidean distance.
- We approximate each point by an affine mixture of the points in its neighborhood:

$$\min_{W_{ik}} \left\| x_i - \sum_{k \in N(i)} w_{ik} x_k \right\|^2$$

$$w_{ik} = 0, \quad k \notin N(i); \quad \sum_{k=1}^{N} w_{ik} = 1$$

- We must have K < p.



Local linear embedding

- Given w_{ik} , we find points $y_i \in R^d$ in a space of dimension d < p to minimize

$$\min_{y_i} \sum_{i=1}^{N} \left\| y_i - \sum_{k=1} w_{ik} y_k \right\|^2$$

- In compact form:

$$\min_{Y} tr\left[(Y - WY)^T (Y - WY) \right] = tr\left[Y^T (I - W)^T (I - W) Y \right]$$

where $W \in \mathrm{R}^{N \times N}, Y \in \mathrm{R}^{N \times d}$. The solution is the trailing eigenvectors of

$$\mathrm{M} = (I - W)^T (I - W)$$

- Since 1 is a trivial eigenvector with eigenvalue 0, we discard it and keep the next d. This has the side effect that $1^T Y = 0$

t-SNE: Student-t distribution SNE

- For each data point $x_i \in R^p$, we find points $y_i \in R^d$ in a space of dimension d < p to minimize *KL* divergence between two distributions.
- Student-t distribution SNE:
  - A symmetrized version of the SNE cost function with simpler gradients.
  - A Student-t distribution to compute the similarity in the low-dimensional
- Symetry : $p_{ij} = \frac{p_{i|j} + p_{j|i}}{2}$

$$p_{j|i} = \frac{\exp(-\left\|x_i - x_j\right\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\left\|x_i - x_j\right\|^2 / 2\sigma_i^2)}$$

- A Student-t distribution for $y_i \in R^d$

$$q_{ij} = \frac{\left(1 + \left\|y_i - y_j\right\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \left\|y_i - y_j\right\|^2\right)^{-1}}$$

$$p_{st}(z \mid p) = \frac{\Gamma\left(\frac{p+1}{2}\right)\left(1 + \left\|z\right\|^2 / p\right)^{-\frac{p+1}{2}}}{\sqrt{p\pi}\,\Gamma(p/2)}$$

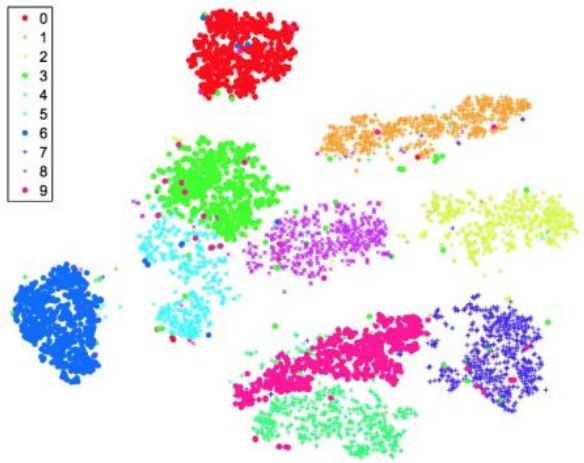
t-SNE: Student-t distribution SNE

- Cost function: $$\mathrm{L} = \sum_i KL(P_i \mid Q_i) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

- The gradient $$\frac{\partial L}{\partial y_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij}) \left(1 + \left\| y_i - y_j \right\|^2\right)^{-1} (y_i - y_j)$$

- The heavy tails of the normalized Student-t kernel allow dissimilar input objects $x_i, x_j \in R^p$ to be modeled by low-dimensional counterparts $y_i, y_j \in R^d$ that are too far apart because q_{ij} is not very small for two embedded points that are far apart.

Comparisons



(a) Visualization by t-SNE.

(b) Visualization by Sammon mapping.

(a) Visualization by Isomap.

(b) Visualization by LLE.

Deep Learning

- Background
- Deep Neural Networks
  - Regularization - Dropout
- Disentangled Representation in DNN
- CNN for Computer Vision
  - Convolutional Networks
  - Filter; Pooling
- Perspectives and Future Directions

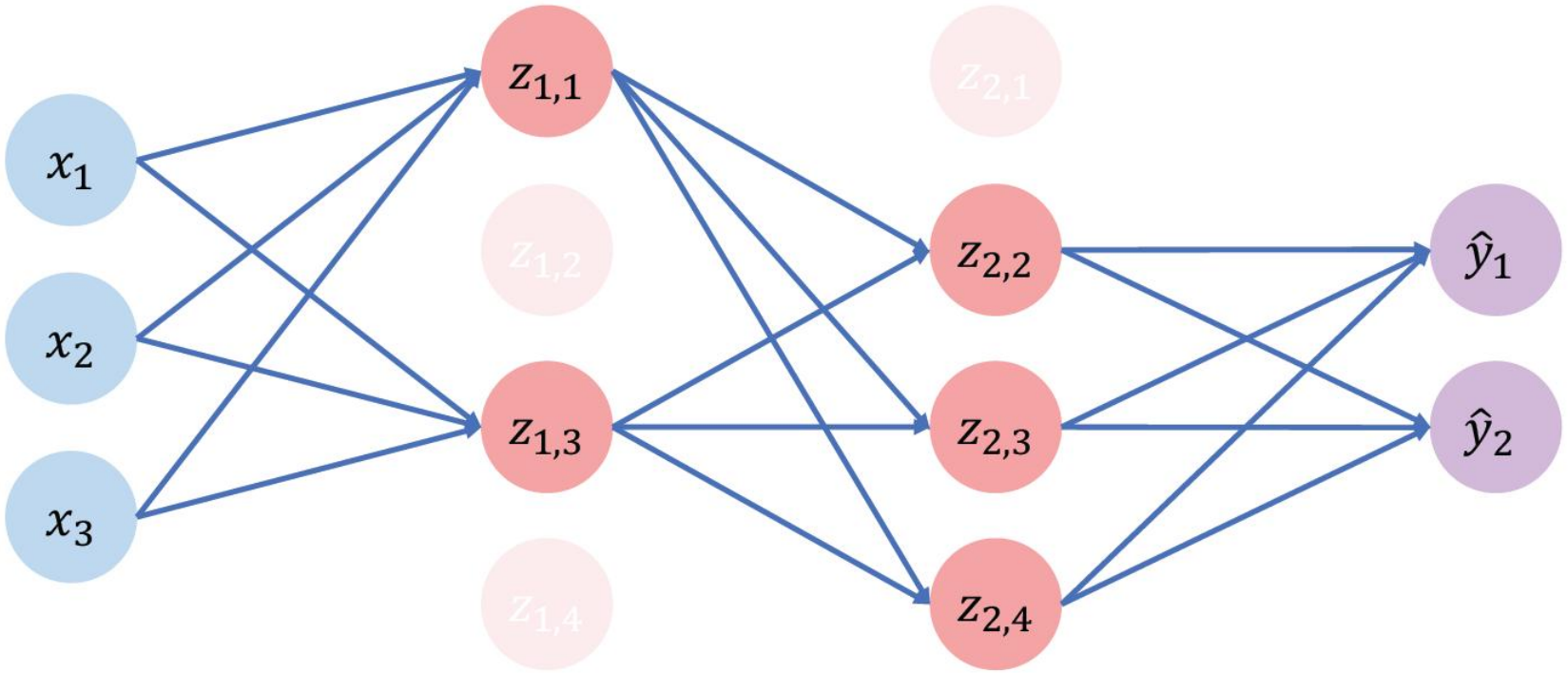
Dropout Regularization

- **Dropout regularization** is used for reducing overfitting and improving the generalization of deep neural networks.
- Large weights in a neural network are a sign of a more complex network that has overfit the training data.
- **Probabilistically dropping out** nodes in the network is a simple and effective regularization method.
- **Large neural nets trained on relatively small datasets can overfit the training data.**
  The statistical noise in the training data results in poor performance when the model is evaluated on new data, e.g. a test dataset

Regularization: Dropout

- During training, randomly set some activations to 0
  - Typically 'drop' 50% of activations in layer
  - Forces network to not rely on any 1 node

`tf.keras.layers.Dropout(p=0.5)`

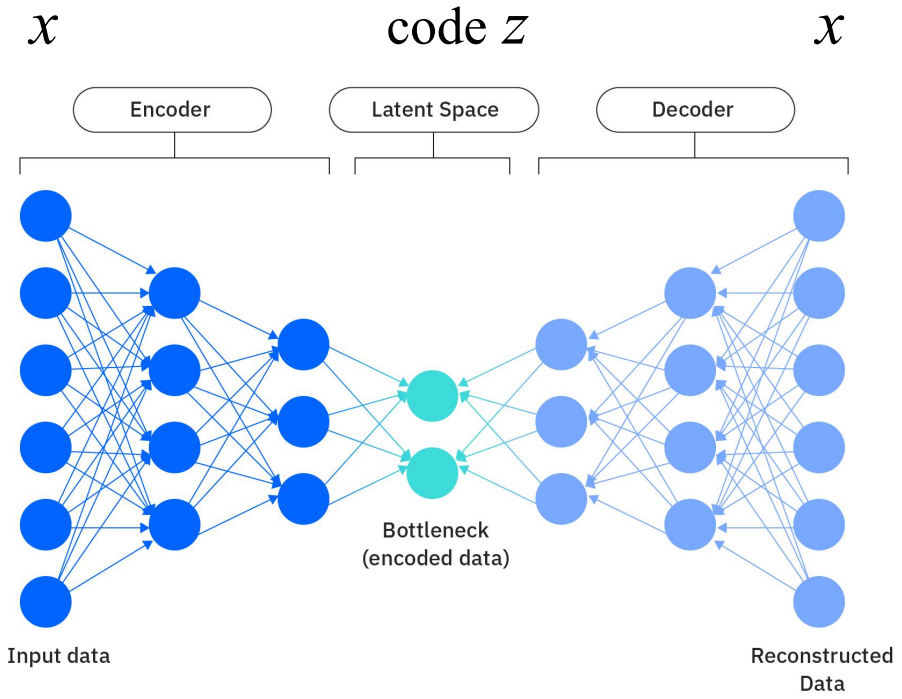


AutoEncoder

- Autoencoder (AE)
- Input x; Encoder: z

Encoder $f : X \to Z; \quad z = f(x, \theta);$

Decoder $g : Z \to X; \quad \hat{x} = g(z, \varphi);$



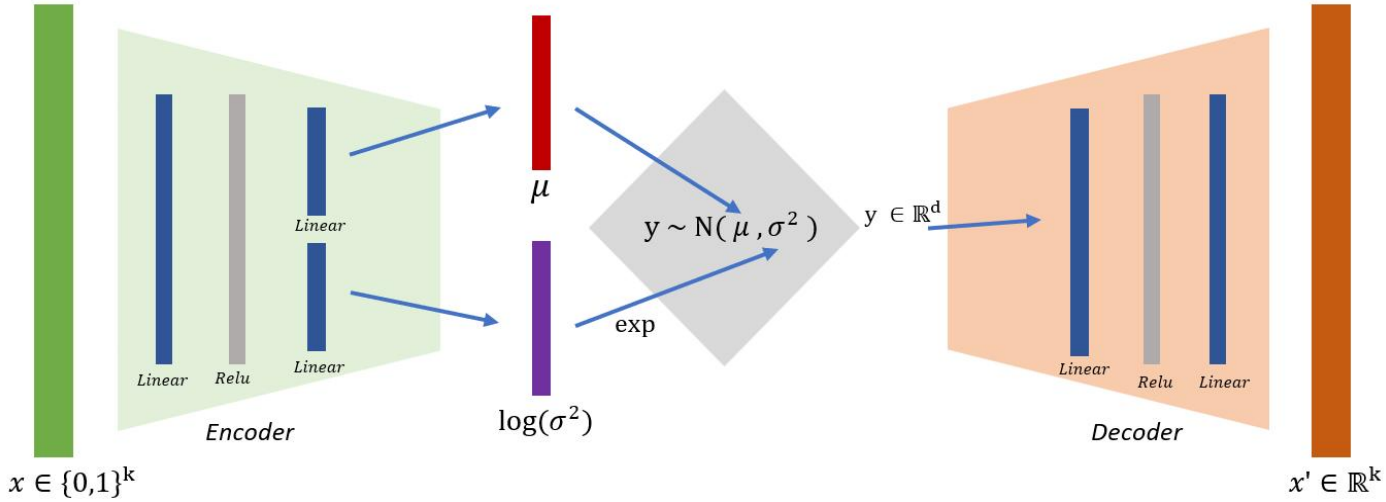
Risk function(Cost Function)

$$f, g = \arg\min_{f,g} \left\| x - g[f(x)] \right\|^2$$

AutoEncoder is a natrual extension of PCA to Nonlinear Cases

Variational Autoencoder (VAE)

- VAE =AutoEncoders + Variational inference
  - to learn a probability distribution over the input data, which can then be used to generate new data similar to the training data.



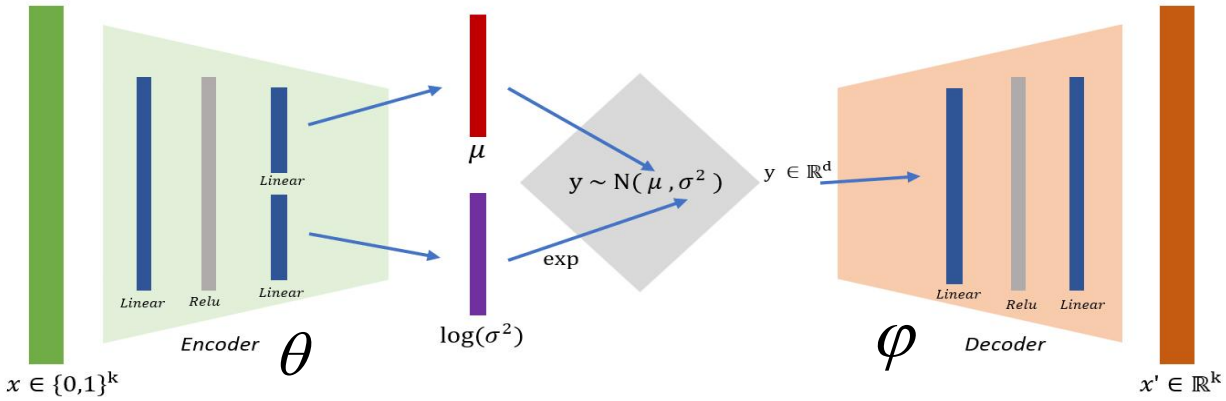
VAES are *probabilistic* models. VAEs encode latent variables of training data not as a fixed discrete value z, but as a continuous range of possibilities expressed as a probability distribution $p(z)$.

Mathematical Formulation of the Encoder

- $\mathbf{X}$: the input data point (e.g., an image, a sequence of text, etc.).
- The encoder network $p(z|X,\theta)$ maps the input X to a latent variable z.
  - Here, θ represents the parameters of the encoder network.
- The encoder network outputs the mean $\mu_\theta(x)$ and the covariance matrix $\sigma_\theta(x)$ or more commonly, the log - variance $\log \sigma_\theta^2(x)$ of a Gaussian distribution.
- The latent variable z is then sampled from this distribution:

$$z = \mu_\theta(x) + \varepsilon \, \sigma_\theta(x)$$

$$\varepsilon \sim N(0, I)$$



Mathematical Formulation of the Decoder

- The decoder network $q(x|z,\varphi)$ maps the latent variable z back to the original data space.
  - Here, φ represents the parameters of the decoder network.
- The decoder is usually modeled as a conditional probability distribution. For example, if the data is continuous (like images), it might be a Gaussian distribution

$$q(x|z,\varphi) = N(z \mid \mu_\varphi, \Sigma_\varphi)$$

where $\mu_\varphi, \Sigma_\varphi$ are the mean and covariance of the distribution predicted by the decoder. If the data is discrete (like text), it might be a categorical distribution.

Variational Inference

- Maximizing the variational lower bound

$$\log p(x) \geq F(\theta, \varphi) = E_{q(z)}\left[\log p(x \mid z)\right] - D_{KL}(q \mid p)$$

- The relation with cost function for VAE

$$\log p(x \mid z, \varphi) = -\frac{1}{2\sigma^2}\left\|x - G_{\varphi}(z)\right\|^2 + const.$$

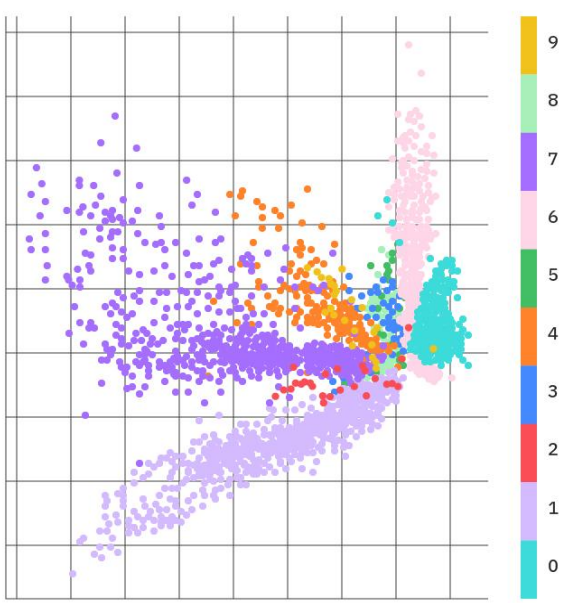
- Miximizing the variational lower bound is equivalent to minimizing the total Loss function

$$\max_{\theta,\varphi} F(\theta, \varphi) \sim \min_{\theta,\varphi} L_{total}$$

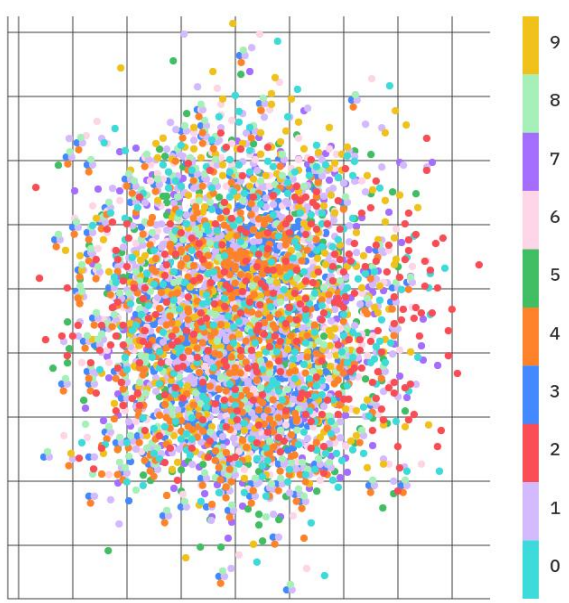
Experimental Results

- Examples
  latent spac

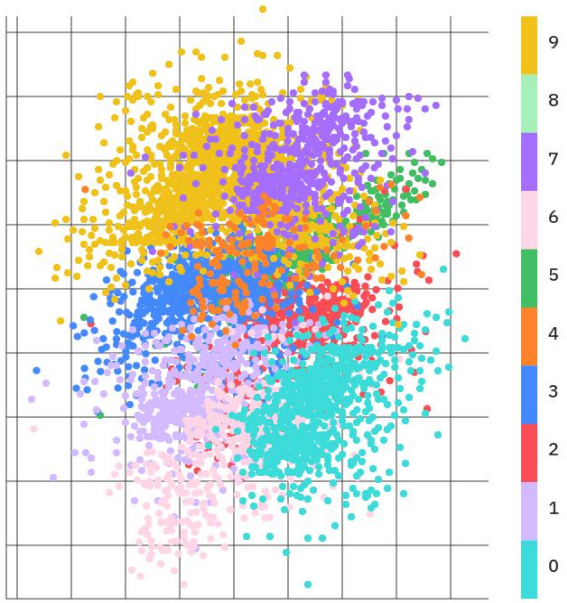
Only reconstruction loss



Only KL divergence



Reconstruction loss
and KL divergence



Final Report

- 时间：2025-01-07 13:10-15:10
- 地点：东上院412 东上院415

- 答疑时间：2025-01-03，地点：电信学院3-435室