Hongyi Zhang

$y=\begin{pmatrix}y_A\\y_B\end{pmatrix}\sim N(\mu,\Sigma)$ $\mu=\begin{bmatrix}\mu_A\\\mu_B\end{bmatrix}$ $\Sigma=\begin{bmatrix}\Sigma_{AA}&\Sigma_{AB}\\\Sigma_{BA}&\Sigma_{BB}\end{bmatrix}$ $y\sim N(\mu,\Sigma), y'\sim N(\mu',\Sigma') \Rightarrow y+y'\sim N(\mu+\mu',\Sigma+\Sigma')$ $x\sim N(\mu,\sigma^2)$

$y_A\sim N(\mu_A,\Sigma_{AA})$ $y_B\sim N(\mu_B,\Sigma_{BB})$

$p(x;\mu,\Sigma)=\frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}}\exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$ $p(x)=\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$

$y_A|y_B\sim N(\mu_A+\Sigma_{AB}\Sigma_{BB}^{-1}(y_B-\mu_B),\ \Sigma_{AA}-\Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA})$

Performance Evaluation

2.1 Performance Measures for Regression

2.1.1 Error: Mean absolute error (MAE) $e=\frac{1}{n}\sum_{i=1}^{n}|f(x_i)-y_i|$

Mean square error (MSE) $e=\frac{1}{n}\sum_{i=1}^{n}(f(x_i)-y_i)^2$

2.1.2 $R^2=1-\frac{SS_{RES}}{SS_{TOT}}=\frac{SS_{REG}}{SS_{TOT}}=\frac{\frac{1}{n}SS_{REG}}{\frac{1}{n}SS_{TOT}}$

$SS_{RES}=\sum_{i=1}^{n}(f(x_i)-y_i)^2$ $SS_{REG}=\sum_{i=1}^{n}(f(x_i)-\bar{y})^2$ $SS_{TOT}=\sum_{i=1}^{n}(y_i-\bar{y})^2$

2.1.3 Negative log predictive density

$NLPD=-\log p(y)=-\frac{1}{n}\sum_{i=1}^{n}\log p(y_i=f(x_i)|x_i)$

Gaussian Process: $NLPD=\frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{2}\log(2\pi\sigma_i^2)+\frac{(y_i-\mu_i)^2}{2\sigma_i^2}\right)$

2.2 Performance Measures for Classification

2.2.1 0/1 loss: $\frac{1}{n}\sum_{i=1}^{n}\delta_{h(x_i)\neq y_i}\times 100\%$ (error)

1-error: $\frac{1}{n}\sum_{i=1}^{n}\delta_{h(x_i)=y_i}\times 100\%$ (accuracy)

2.2.2 Confusion Matrix

prediction \ true class	+1	-1
+1	TP	FP
-1	FN	TN

$P=TP+FN$; $N=TN+FP$

$FPR=\frac{FP}{N}$ $FNR=1-TPR=\frac{FN}{P}$ $TNR=\frac{TN}{N}$ $TPR=\frac{TP}{P}$ (recall)

$precision=\frac{TP}{TP+FP}$ $accuracy=\frac{TP+TN}{P+N}$ $F\text{-score}=2\cdot\frac{precision\cdot recall}{precision+recall}$

2.2.3 Area under the ROC curve (AUC) — (only for binary) — threshold parameter t (change)

x轴: FPR, y轴: TPR AUC = probability that a randomly selected positive example is ranked higher than a randomly selected negative example

2.4 Efficiency Runtime (training time / test time) Memory

3. Null hypothesis H_0: A and B perform equivalently Aim: reject H_0

3.1 Parametric Tests t-Test split $1\cdots n$

H_0: the observed means μ_A and μ_B are the same (same performance)

$d_i=acc_i^A-acc_i^B$, $\bar{d}=\mu_A-\mu_B$, $\hat{\sigma}_d=\sqrt{\frac{\Sigma(d_i-\bar{d})^2}{n-1}}$, $t=\frac{\bar{d}}{\hat{\sigma}_d/\sqrt{n}}$

degree of freedom $df=n-1$; two-tailed test $|t|\geq t_\alpha$ $(\alpha=0.05)$

we can reject H_0

3.2 Non-parametric Tests 3.2.1 McNemar's Test (classification)

C_{01} the number of instances misclassified by A correctly classified by B

C_{10} misclassified by B correctly classified by A

$H_0: p(C_{01})=p(C_{10})$ (same performance)

$\chi^2_{MC}=\frac{(|C_{01}-C_{10}|-1)^2}{C_{01}+C_{10}}$ Chi-squared distribution with 1 degree of freedom

3.2.2 Sign Test (used on multiple domains)

A_{win} = number of experiments A outperform B, B_{win} = B outperformed A

$H_0: p(A_{win}>B_{win})=0.5$ (same performance) $A_{win}\sim B(n,\theta=0.5)$

$A_{win}>w_\alpha$, reject H_0

ARD Kernel $K(x,x')=\sigma_f^2\exp(-\frac{1}{2}(x-x')^TM(x-x'))+\sigma_n^2\delta_{x,x'}$ $M=diag(l)^{-2}$

Kernel Methods

不需计算 $\phi(x)$, 只需计算 $\langle\phi(x),\phi(x')\rangle$ 内积 2. Kernel Trick

$\phi(x)^T\phi(x')$

2.1 linear model $h(x)=w^Tx$, $L(w)=\sum_{i=1}^{n}(w^Tx-y_i)^2$, $g(w)=\frac{\partial L}{\partial w}=\sum_{i=1}^{n}2(w^Tx_i-y_i)x_i$

$w_{t+1}=w_t-\eta g(w_t)\rightarrow w_t=\sum_{i=1}^{n}\alpha_i^{(t)}x_i\rightarrow w_t^Tx_i=\sum_{j=1}^{n}\alpha_j^{(t)}x_j^Tx_i\Rightarrow$ new test input x^*

$y^*=\sum_{i=1}^{n}\alpha_i^{(conv)}x_i^Tx^*\xrightarrow{Kernel}y^*=\sum_{i=1}^{n}\alpha_i^{(conv)}\langle\phi(x_i),\phi(x^*)\rangle$ 预测时 $h(x)=\sum_i\alpha_iK(x_i,x_i)$

Definition: Kernel $K(x,x')=\langle\phi(x),\phi(x')\rangle$ Definition: Valid Kernel $\forall R^{m\times m}$, $\{x_1,\cdots,x_m\}$

① symmetric $k(x,x')=k(x',x)$ ② positive semi-definite K is a PSD matrix $K_{ij}=k(x_i,x_j)$

(① $\forall q\in R^m, q^TAq\geq 0$ ② $Av=\lambda v$, all $\lambda\geq 0$ ③ 左上子阵行列式 ≥ 0)

3.2 Common Kernel Linear kernel $K(x,x')=x^Tx'$ Polynomial kernel d: degree of poly

$K(x,x')=(1+x^Tx')^d$ RBF/Gaussian kernel $K(x,x')=\exp(-\frac{1}{2l^2}\|x-x'\|^2)$ 认1, 过拟合

3.4 Kernel Construction ① $k(x,x')=x^Tx'$ ② $K(x,x)=ck_1(x,x')$, $c\geq 0$ ③ $k(x,x')=k_1+k_2$

④ $K=g(k_1)$, g: polynomial 正系数 ⑤ $K=k_1\times k_2$ ⑥ $k=f(x)k_1f(x')$ $f:R^d\rightarrow R$ ⑦ $k=e^{k_1}$ ← RBF valid proof

⑧ $k(x,x')=x^TAx'$, A is PSD 4. Kernel Machines

4.1 Kernel Ridge Regression $\min_w\|Xw-y\|^2+\lambda\|w\|^2\rightarrow w=(XX^T+\lambda I)^{-1}Xy$

$w=X\alpha$, $\alpha=(X^TX+\lambda I)^{-1}y=(K+\lambda I)^{-1}y$, where $K=X^TX$, $\alpha\in R^n$

$w=(XX^T+\lambda I)^{-1}Xy=X\alpha\Rightarrow(XX^T+\lambda I)X\alpha=Xy\Rightarrow X^TX(X^TX+\lambda I)\alpha=X^TXy\Rightarrow(K+\lambda I)\alpha=y$

$y=w^Tx+\varepsilon\Rightarrow y=w^T\phi(x)+\varepsilon$ or $y=K_x\alpha$, $\alpha=(K+\lambda I)^{-1}y$, $K_x=K(x,X)$ prediction $h(x)=K_x\alpha=\sum_{i=1}^{n}\alpha_ik(x_i,x)$

$y=w^Tx+\varepsilon$ (Primal) $y=x^TX\alpha$ (Dual) Gaussian Process (非参贝叶斯 model)

$y=f(x)+\varepsilon=w^Tx+\varepsilon$, $\varepsilon\sim N(0,\sigma_n^2)$, $p(y|X,w)=N(X^Tw,\sigma_n^2I)$ $p(y|x,w)=N(w^Tx,\sigma_n^2)$

$p(w|X,y)=\frac{P(y|X,w)p(w)}{p(y|X)}=\frac{likelihood\times prior}{marginal\ likelihood}$ $p(w)=N(0,\sigma_p^2I)$

$y=f(x)+\varepsilon=w^T\phi(x)+\varepsilon=\sum_{i=1}^{n}\alpha_ik(x_i,x)+\varepsilon$

$p(y^*|x^*,X,y)=N(\mu_{y^*|D},\Sigma_{y^*|D})$ $\mu_{y^*|D}=K_*^T(K+\sigma_n^2I)^{-1}y$

$\Sigma_{y^*|D}=K_{**}-K_*^T(K+\sigma_n^2I)^{-1}K$ $K=K(X,X)\in R^{n\times n}$ $K_*=K(X,x^*)\in R^n$, $K_{**}=k(x^*,x^*)\in R$

2. Gaussian Process joint distribution $f\sim GP(\mu,k)$ $x_1,\cdots,x_n$ μ: mean $k(x,x')$: covariance

3. Gaussian Process Regression (GPR) $f(x)=f|x\sim GP(\mu(x),k(x,x))$

3.1 Noise free observation $D=\{X,f\}$, $f\in R^n$ $p(f|X)=N(\mu(X),K(X,X))$

test point (x_*,f_*), $p(f_*|x_*)=N(\mu(x_*),k(x_*,x_*))$

$$p\left(\begin{bmatrix}f|X\\f_*|x_*\end{bmatrix}\right)=N\left(\begin{bmatrix}\mu(X)\\\mu(x_*)\end{bmatrix},\begin{bmatrix}K(X,X)&K(X,x_*)\\K(x_*,X)&K(x_*,x_*)\end{bmatrix}\right)$$

$p(f_*|x_*,X,f)=N(\mu(x_*)+K(x_*,X)K(X,X)^{-1}(f-\mu(X)),\ K(x_*,x_*)-K(x_*,X)K(X,X)^{-1}K(X,x_*))$

$E(f_*)=\bar{f}_*=\mu(x_*)+K(x_*,X)K(X,X)^{-1}(f-\mu(X))$ $Cov(f_*)=V(f_*)=K_{**}-K_*^TK^{-1}K_*$

3.2 Noise Observation $D=\{X,y\}$, $y=f(X)+\varepsilon=f+\varepsilon$, $\varepsilon\sim N(0,\sigma_n^2)$

$$p\left(\begin{bmatrix}y|X\\f_*|x_*\end{bmatrix}\right)=N\left(\begin{bmatrix}\mu(X)\\\mu(x_*)\end{bmatrix},\begin{bmatrix}K(X,X)+\sigma_n^2I&K(X,x_*)\\K(x_*,X)&K(x_*,x_*)\end{bmatrix}\right)$$

$k(X,X)\leftarrow k(X,X)+\sigma_n^2I$

$\bar{f}_*=\mu(x_*)+K(x_*,X)(K(X,X)+\sigma_n^2I)^{-1}(y-\mu(X))$ $cov(f_*)=k(x_*,x_*)-K(x_*,X)(K(X,X)+\sigma_n^2I)^{-1}K(X,x_*)$

3.3 Noisy prediction y_* $cov(y_*)=cov(f_*)+\sigma_n^2$

4. Algorithm

Train (X,y), k (covariance) σ_n^2 (noise) x_* (test) Rescale y_i, zero-mean GP $\mu(x)=0$

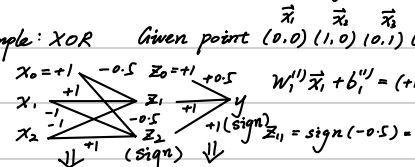
$L=cholesky(K+\sigma_n^2I)$ 代替 inverse $LV=y$, $L^T\alpha=V\Rightarrow\alpha=L^T\backslash(L\backslash y)$ $\bar{f}_*=K_*^T\alpha$, $V=L\backslash K_*$

$V(f_*)=k(x_*,x_*)-V^TV$ 4.2 Training GP CV(hyperparameter) $\theta^*=\arg\max_\theta p(y|X,\theta)$ max marginal likelihood

Perceptron: $h(\vec{x}) = \text{sign}(\vec{w}^T\vec{x})$ L_2-norm $\|\vec{x}\|_2^2 = \vec{x}^T\vec{x} = \sum x_i^2$ $L_1 + \lambda\|w\|_1 = \lambda\sum|w_i|$ $L_2 + \lambda\|w\|_2^2$ Eigenvector Eigenvalue $Av=\lambda v \Rightarrow (A-\lambda I)v=0$ $\det(A-\lambda I)=0$ $\Rightarrow \lambda_1,\cdots,\lambda_n$

Neural Network 2. Multi-layer Perceptron

Example: XOR Given point $\vec{x}_1$ $(0,0)$ $\vec{x}_2$ $(1,0)$ $\vec{x}_3$ $(0,1)$ $\vec{x}_4$ $(1,1)$



$W_1^{(1)}\vec{x}_1 + b_1^{(1)} = (+1\ -1)\begin{pmatrix}0\\0\end{pmatrix} - 0.5 = -0.5$

$z_{11} = \text{sign}(-0.5) = -1$ similarly $z_{12}=1$, $z_{13}=-1$, $z_{14}=-1$ z_{21} z_{22} z_{23}, z_{24}

$W^{(1)} = \begin{pmatrix}+1 & -1\\ -1 & +1\end{pmatrix}$ $b^{(1)} = \begin{pmatrix}-0.5\\ -0.5\end{pmatrix}$ $W^{(2)} = (+1\ +1)$ $b^{(2)} = (0.5)$

$W_j^{(L)}$ jth row for unit j

3. Parameter Learning Notation: $W^{(L)}$ weight parameter matrix for layer L $L \in \{1,\cdots,L\}$

$b^{(L)}$ bias terms for layer L; $a^{(L)} = W^{(L)}z^{(L-1)} + b^{(L)}$, $z^{(L)} = g_L(a^{(L)})$, m_L number of unit

3.1 Forward Propagation $\vec{x}^{(0)} = x \in R^d$, $W^{(L)} \in R^{m_L \times m_{L-1}}$, $z^{(L)} = g_L(W^{(L)}z^{(L-1)} + b^{(L)})$

3.2 Compute Loss (Regression) $l(\vec{x},y) = \frac{1}{2}(z^{(L)}-y)^2$ squared loss. GD update rules. layer L

$l(\vec{x},y) = \frac{1}{2}(g_L(\underbrace{W^{(L)}z^{(L-1)} + b^{(L)}}_{a_1^{(L)}}) - y)^2$; $\frac{\partial l}{\partial w_{ij}^{(L)}} = \frac{\partial l}{\partial g_L}\cdot\frac{\partial g_L}{\partial a_1^{(L)}}\cdot\frac{\partial a_1^{(L)}}{\partial w_{ij}^{(L)}}$

α – GD learning rate $= (z^{(L)}-y)\cdot g_L'(a_1^{(L)})\cdot z_j^{(L-1)} = \delta_1^{(L)}\cdot z_j^{(L-1)}$

$W^{(L)} \leftarrow W^{(L)} - \alpha\Delta W^{(L)} \Rightarrow W^{(L)} \leftarrow W^{(L)} - \alpha\delta^{(L)}z^{(L-1)T} \in R^{m_L \times m_{L-1}}$

3.3 Back Propagation $\frac{\partial l}{\partial w_{jk}^{(L-1)}} = \frac{\partial l}{\partial g_L}\frac{\partial g_L}{\partial a^{(L)}}\frac{\partial a^{(L)}}{\partial g_{L-1}}\frac{\partial g_{L-1}}{\partial a^{(L-1)}}\frac{\partial a^{(L-1)}}{\partial w_{jk}^{(L-1)}}$

$= \underbrace{(z^{(L)}-y)\cdot g_L'(a^{(L)})}_{\delta^{(L)}}\cdot W^{(L)}\begin{bmatrix} g'(a_1^{(L-1)}) & & 0\\ & \ddots & \\ 0 & & g'(a_{m_{L-1}}^{(L-1)})\end{bmatrix}\begin{bmatrix}0\\ \vdots\\ z_k\\ \vdots\\ 0\end{bmatrix} \leftarrow j\text{th row}$

$= \delta^{(L)}\begin{bmatrix}W_{11}^{(L)} & \cdots & W_{1j}^{(L)} & \cdots & W_{1m_{L-1}}^{(L)}\end{bmatrix}\begin{bmatrix}0\\ \vdots\\ g'(a_j^{(L-1)})z_k^{(L-2)}\\ \vdots\\ 0\end{bmatrix} = \underbrace{\delta^{(L)}W_{1j}^{(L)}g'(a_j^{(L-1)})}_{\delta_j^{(L-1)}}z_k^{(L-2)}$

$\frac{\partial l}{\partial b_i^{(L)}} = \frac{\partial l}{\partial z_i^{(L)}}\frac{\partial z_i^{(L)}}{\partial a_i^{(L)}}\frac{\partial a_i^{(L)}}{\partial b_i^{(L)}} = (z_i^{(L)}-y_i)\cdot g'(a_i^{(L)}) = \delta_i^{(L)}$ $b_i^{(L)} \leftarrow b_i^{(L)} - \alpha\delta_i^{(L)}$

$b^{(L)} \leftarrow b^{(L)} - \alpha\delta^{(L)}$

3.4 Training Algorithm $\forall L = 1,\cdots,L$

Input: $D = \{(\vec{x}_i, y_i)\}_{i=1,\cdots,n}$ $NN = \begin{cases}L\\ g_L\\ m_L\end{cases}$ Initialization: $\vec{b}^{(L)}$, $W^{(L)}$

repeat for each training example $(\vec{x},y) \in D$ do $z^{(0)} = \vec{x}$

Forward Prop for $L = 1,\cdots,L$ do $a^{(L)} = W^{(L)}z^{(L-1)} + b^{(L)}$, $z^{(L)} = g_L(a^{(L)})$

$\delta^{(L)} = (z^{(L)}-y)\cdot g_L'(a^{(L)})$ (compute loss in output layer)

Back Prop $\delta^{(L-1)} = g_{L-1}'(a^{(L-1)})\circ W^{(L)T}\delta^{(L)}$ $W^{(L)} \leftarrow W^{(L)} - \alpha\delta^{(L)}z^{(L-1)T}$ $b^{(L)} \leftarrow b^{(L)} - \alpha\delta^{(L)}$

for $L = L-1,\cdots,1$ do

$\Delta W^{(L)} = \delta^{(L)}z^{(L-1)T}$ $\delta^{(L-1)} = g_{L-1}'(a^{(L-1)})\circ W^{(L)T}\delta^{(L)}$

$\Delta b^{(L)} = \delta^{(L)}$ $W^{(L)} \leftarrow W^{(L)} - \alpha\Delta W^{(L)}$ $b^{(L)} \leftarrow b^{(L)} - \alpha\Delta b^{(L)}$

until error $= \frac{1}{n}\sum_{i=1}^{n}(z_i^{(L)}-y_i) < \varepsilon$

5. Model Choice 5.1 Output Units (activation function used in output unit)

regression – linear unit $h(a) = a$, binary-sigmoid classification $h(a) = \frac{1}{1+e^{-a}}$, multi-class-softmax $a \in R^K$ $h_c(a) = \frac{e^{ac}}{\sum_j e^{aj}}$

5.2 loss function regression: squared loss $l(\vec{x},y) = \frac{1}{2}(z^{(L)}-y)^2$ binary classification $y \in \{0,1\}$

cross-entropy loss $l(\vec{x},y) = -[(1-y)\ln z_0^{(L)} + y\ln z_1^{(L)}]$ z_0, z_1 sigmoid 概率

multi class cross-entropy loss $l(\vec{x},y) = -\sum_{c=1}^{K}\delta_{y=c}\ln z_c^{(L)}$

5.3 Activation Function sigmoid $(a) = \frac{1}{1+e^{-a}}$ $\text{sigm}'(a) = \text{sigm}(a)(1-\text{sigm}(a))$

sign; $\tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$ $\tanh'(a) = 1 - \tanh^2(a)$;

$\text{relu}(a) = \max(a,0)$, $\text{relu}'(a) = \begin{cases}0 & a<0\\ 1 & a>0\\ \text{NAN} & a=0\end{cases}$

6. Autoencoder (unsupervised) $\vec{x} \in R^m \xrightarrow{\text{encoder } g} \vec{z} \in R^d \xrightarrow{\text{decoder } f} \hat{\vec{x}}$ over $m>d$ deep autoencoder $L>2$

undercomplete autoencoder, where $m<d$. loss function $l(\vec{x}, \underbrace{f(g(\vec{x}))}_{\hat{\vec{x}}})$

identity activation $g(\vec{x}) = W\vec{x}+b$ and f get non-linear generalization of PCA

Deep Learning challenges: overfitting, difficult optimization

2. Regularization 2.1 Weight Decay $W_{i:}^{(L)}$ i-th row in $W^{(L)}$

$a_i = \vec{W}^T\vec{x} + b_i^{(L)} = W_{i:}^{(L)}\vec{x} + b_i^{(L)}$ L_2 regularization: $\frac{\lambda}{2}\|\vec{w}\|_2^2 = \frac{\lambda}{2}\vec{w}^T\vec{w}$

$\frac{\partial\frac{\lambda}{2}\vec{w}^T\vec{w}}{\partial\vec{w}} = \lambda\vec{w}$ gradient update $W_{i:}^{(L)} = w = w - \alpha(\Delta w + \lambda w) = (1-\alpha\lambda)w - \alpha\Delta w$

2.2 Constrained Optimization constraint $\Omega(\vec{w}) \le k$ new objective $l(w,x,y) + \lambda(\Omega(w)-k)$

2.3. Early Stopping (use validation set) 2.4 Transfer learning as regularizer)

2.5 Ensemble Learning and Dropout 2.6 Batch Normalization (↑lr, speeds up training

2.8 Data Augmentation 2.9 Adversarial Training 3. Optimization

(1) SGD → mini-batch (2) adaptive learning rate α (lr) linear search (3) momentum method Adam + (2) uniform Normal

(4) conjugate GD (avoid compute Hessian) 4 Parameter Initialization 4.1 Random Init

4.2 Greedy Pretraining Supervised Pretraining (omit $W^{(L)}$ for later layer)

Unsupervised Pretraining (X connect output layer, like auto encoder, reconstruction) layer

Dimensionality Reduction 3. PCA (project $x_i \in R^d$ to first r ($r<d$) principal components)

find $U \in R^{d\times r}$ s.t. $\{v_i\}_{i=1}^n$, $v_i = U^Tx_i$ have maximum spread v_i: $r\times 1$, x_i: $d\times 1$ $\bar{x} = \frac{1}{n}\sum_{i=1}^n x_i$ $u \in R^d$

$C = \frac{1}{n-1}\sum_{i=1}^n(x_i-\bar{x})(x_i-\bar{x})^T$ Step 1: center the data $x_i \leftarrow x_i - \bar{x}$ Step 2: $u = \underset{u^Tu=1}{\text{argmax}}\sum_{i=1}^n(u^Tx_i)^2 \Rightarrow$

$\underset{u^Tu=1}{\text{argmax}}\ u^TCu \Rightarrow \max_u u^TCu$ s.t. $u^Tu=1 \Rightarrow \alpha(u,\lambda) = u^TCu - \lambda(u^Tu-1) \Rightarrow$

u is eigenvector of C

$\frac{\partial\alpha}{\partial u} = 2Cu - 2\lambda u = 0 \Rightarrow Cu = \lambda u \Rightarrow \max u^TCu = \max\lambda \Rightarrow$ pick the r eigenvectors

with the largest eigenvalues (principal components) $U_r = [u_1,\cdots,u_r] \in R^{d\times r}$

Data Projection: $z_i = U_r^Tx_i$ (C symmetric u_j orthogonal $u_j^Tu_k=0$, $u_j^Tu_j=1$)

3.3 Reconstruction Error reconstruction $\hat{x}_i = U_r\vec{z}_i = \sum_{\alpha=1}^r z_\alpha\vec{u}_\alpha$ original x U (all eigenvectors) $x = Uz = \sum_{\alpha=1}^d z_\alpha u_\alpha \Rightarrow$ total reconstruction error

$\sum_{i=1}^n\|x_i-\hat{x}_i\|^2 = \sum_{i=1}^n\sum_{\alpha=r+1}^d[z_i]_\alpha^2 = \sum_{\alpha=r+1}^d\sum_{i=1}^n u_\alpha^Tx_ix_i^Tu_\alpha = (n-1)\sum_{\alpha=r+1}^d u_\alpha^TCu_\alpha = (n-1)\sum_{\alpha=r+1}^d\lambda_\alpha u_\alpha^Tu_\alpha = (n-1)\sum_{\alpha=r+1}^d\lambda_\alpha$

(compute U)

4.1. SVD Input $x_1,\cdots,x_n$, r; $\bar{x} = \frac{1}{n}\sum_{i=1}^n x_i$. $X = [x_1,\cdots,x_n] \in R^{d\times n}$, $X = X - \bar{x}$ (center)

$X = UDV^T$

$[U,D,V] = \text{svd}(X)$ $u_1,\cdots,u_d$ sorted left SVs of X, where $d_1 \ge \cdots \ge d_d$

$U_r = [u_1,\cdots,u_r] \in R^{d\times r}$. $Z = U_r^TX$ 4. MDS (PCA equivalent) $G = X^TX \in R^{n\times n}$,

$V = [v_1,\cdots,v_n]$ eigenvectors of G. $\tilde{\lambda}_1 \ge \cdots \ge \tilde{\lambda}_n$, $\lambda_k = d_k^2$, $z_i = d_i[V_r]_{i:}$ or $Z = D_rV_r^T$

4.3. $X = X_{raw} - \bar{x}$ (data centering) $\tilde{X} = \underbrace{\frac{1}{\Sigma}}_{\text{diagonal}}X$, $\sigma_\alpha^2 = \frac{1}{n-1}\sum_{i=1}^n[x_i]_\alpha^2$ (data standardization)

$\hat{\tilde{X}} = D_\alpha^{-1}U^T\tilde{X}$, D_α 奇异值对角阵 (特征值平方根), U (from PCA) (data whitening)

5 Non-linear Dimensionality Reduction.

5.1 MDS. kernelize MDS by replacing the $G = X^TX$ by kernel Matrix $K = K(X,X)$

5.2 Kernel PCA $C = \frac{1}{n}\sum_{i=1}^n\phi(x_i)\phi(x_i)^T \Rightarrow Cu_\alpha = \lambda u_\alpha$ ← eigenvector

$\Rightarrow \frac{1}{n}\sum_{i=1}^n\phi(x_i)\underbrace{\phi(x_i)^Tu_\alpha}_{a_{\alpha i}} = \lambda_\alpha u_\alpha \Rightarrow \frac{1}{n}\sum_{i=1}^n K(x_i,x_k)\sum_{j=1}^n a_{\alpha j}K(x_i,x_j) = \lambda_\alpha\sum_{i=1}^n a_{\alpha i}K(x_i,x_k)$

$(K(x_i,x_j) = \phi(x_i)^T\phi(x_j))$ $\Rightarrow K^2a_\alpha = \lambda_\alpha nKa_\alpha \Rightarrow Ka_\alpha = \lambda_\alpha n\,a_\alpha$ ← eigenvector of K

$[z_i]_\alpha = u_\alpha^T\phi(x_i) = \sum_{j=1}a_{\alpha j}\underbrace{\phi(x_i)^T\phi(x_j)}_{K(x_i,x_j)} = a_\alpha^TK_{:i}$ $\forall\alpha = 1,\cdots,r$ 取前 r 个

6. Autoencoder. identity activation – PCA.

non-linear f, g, non linear surface. deep ↑ highly non-linear