

10-701 Cheat Sheet

Non-Parametric

MaxLikelihood learning window will give you delta functions, which is a kind of over fitting. Use Leave-one-out cross validation for model selection. Idea: Use some of the data to estimate density; Use other part to evaluate how well it works. Pick the parameter that works best.

$\log p(x_i|X \setminus \{x_i\}) = \log \frac{1}{n-1} \sum_{j \neq i} k(x_i, x_j)$, the sum over all points is $\frac{1}{n} \sum_{i=1}^n \log \left[\frac{n}{n-1} p(x_i) - \frac{1}{n-1} k(x_i, x_i) \right]$ where $p(x) = \frac{1}{n} \sum_{i=1}^n k(x_i, x)$.

why must we not check too many parameters? that you can overfit more; for a given dataset, a few particular parameter values might happen to do well in k-fold CV by sheer chance, where if you had a new dataset they might not do so well. Checking a reasonable number of parameter values makes you less likely to hit those “lucky” spots helps mitigate this risk.

Silverman’s Rule for kernel size Use average distance from k nearest neighbors $r_i = \frac{r}{k} \sum_{x \in \text{NN}(x_i, k)} \|x_i - x\|$.

Watson Nadaraya 1. estimate $p(x|y=1)$ and $p(x|y=-1)$; 2. compute by Bayes rule

$p(y|x) = \frac{p(x|y)p(y)}{p(x)} = \frac{\frac{1}{m_y} \sum_{y_i=y} k(x_i, x) \cdot \frac{m_y}{m}}{\frac{1}{m} \sum_i k(x_i, x)}$. 3. Decision boundary

$p(y=1|x) - p(y=-1|x) = \frac{\sum_j y_j k(x_j, x)}{\sum_i k(x_i, x)} = \sum_j y_j \frac{k(x_j, x)}{\sum_i k(x_i, x)}$

Actually, we assume that $p(x=y)$ is equal to

$1/m_y * \sum_y k(x_i, x)$. Using this definition, we can see

$p(x, -1) + p(x, 1) = p(x|-1)p(-1) + p(x|1)p(1) = p(x)$.

This can be incorporated into the regression framework in chap 6 of PRML. Where we define $f(x - x_n, t \neq t_n) = 0$, and $f(x - x_n, t = t_n) = f(x - x_n)$. Using this definition, we can derive all the probabilities on this slide. (see my handwritten notes on chap 6 of PRML).

Regression case is the same equation.

kNN Let optimal error rate be p . Given unlimited iid data, 1NN’s error rate is $\leq 2p(1-p)$.

Matrix Cookbook

$$\frac{\partial \mathbf{x}^T \mathbf{a}}{\partial \mathbf{x}} = \frac{\partial \mathbf{a}^T \mathbf{x}}{\partial \mathbf{x}} = \mathbf{a}$$

$$\frac{\partial \mathbf{a}^T \mathbf{X} \mathbf{b}}{\partial \mathbf{X}} = \mathbf{a} \mathbf{b}^T, \frac{\partial \mathbf{a}^T \mathbf{X}^T \mathbf{b}}{\partial \mathbf{X}} = \mathbf{b} \mathbf{a}^T, \frac{\partial \mathbf{a}^T (\mathbf{X}^T | \mathbf{X}) \mathbf{a}}{\partial \mathbf{X}} = \mathbf{a} \mathbf{a}^T$$

$$\mathbf{W} \in \mathbf{S}, \frac{\partial}{\partial \mathbf{s}} (\mathbf{x} - \mathbf{A} \mathbf{s})^T \mathbf{W} (\mathbf{x} - \mathbf{A} \mathbf{s}) = -2 \mathbf{A}^T \mathbf{W} (\mathbf{x} - \mathbf{A} \mathbf{s}),$$

$$\frac{\partial}{\partial \mathbf{x}} (\mathbf{x} - \mathbf{s})^T \mathbf{W} (\mathbf{x} - \mathbf{s}) = 2 \mathbf{W} (\mathbf{x} - \mathbf{s}),$$

$$\frac{\partial}{\partial \mathbf{s}} (\mathbf{x} - \mathbf{s})^T \mathbf{W} (\mathbf{x} - \mathbf{s}) = -2 \mathbf{W} (\mathbf{x} - \mathbf{s}),$$

$$\frac{\partial}{\partial \mathbf{x}} (\mathbf{x} - \mathbf{A} \mathbf{s})^T \mathbf{W} (\mathbf{x} - \mathbf{A} \mathbf{s}) = 2 \mathbf{W} (\mathbf{x} - \mathbf{A} \mathbf{s}),$$

$$\frac{\partial}{\partial \mathbf{A}} (\mathbf{x} - \mathbf{A} \mathbf{s})^T \mathbf{W} (\mathbf{x} - \mathbf{A} \mathbf{s}) = -2 \mathbf{W} (\mathbf{x} - \mathbf{A} \mathbf{s}) \mathbf{s}^T.$$

$$\text{Tr}(\mathbf{A}) = \sum_i \mathbf{A}_{ii}. \text{ For two equal sized matrices, } \text{Tr}(\mathbf{A}^T \mathbf{B}) =$$

$$\text{Tr}(\mathbf{B}^T \mathbf{A}) = \text{Tr}(\mathbf{A} \mathbf{B}^T) = \text{Tr}(\mathbf{B} \mathbf{A}^T) = \sum_{i,j} \mathbf{A}_{ij} \mathbf{B}_{ij}.$$

$\text{Tr}(\mathbf{A}) = \text{Tr}(\mathbf{A}^T)$, $\text{Tr}(\mathbf{A} + \mathbf{B}) = \text{Tr}(\mathbf{A}) + \text{Tr}(\mathbf{B})$. For square matrices,

$$\text{Tr}(\mathbf{A} \mathbf{B}) = \text{Tr}(\mathbf{B} \mathbf{A}), \text{Tr}(\mathbf{A} \mathbf{B} \mathbf{C}) = \text{Tr}(\mathbf{C} \mathbf{A} \mathbf{B}) = \text{Tr}(\mathbf{B} \mathbf{C} \mathbf{A})$$

(trace rotation).

Classifiers and Regressors

Naive Bayes Conditionally independent:

$P(x_1, x_2, \dots | C) = \prod_i P(x_i | C)$. One way to avoid divide by zero: add $(1, 1, \dots, 1)$ and $(0, 0, \dots, 0)$ to both classes.

Learns $P(x_i|y)$ for *Discrete* $x_i - P(x_i|y) = \frac{\#D(X_i=x_i, Y=y)}{\#D(Y=y)}$

For smoothing, use $P(x_i|y) = \frac{\#D(X_i=x_i, Y=y)+k}{\#D(Y=y)+n_i k}$, where n_i is the number of different possible values for X_i (In practice problem set, Jing Xiang used $k=1$?) *Continuous* x_i - Can use any PDF, but usually use Gaussian $P(x_i|y) = \mathcal{N}(\mu_{X_i|y}, \sigma_{X_i|y}^2)$, where $\mu_{X_i|y}$ and $\sigma_{X_i|y}$ are, respectively, the average and variance of X_i for all data points where $Y=y$. The Gaussian distribution already provides smoothing.

Perceptron Produces linear decision boundaries. *Classifies* using $\hat{y} = X_{test} w + b$ *Learns* w and b by updating w whenever $y_i(w^T x_i + b) \leq 0$ (i.e. incorrectly classified). Updates as $w \leftarrow w + x_i y_i, b \leftarrow b + y_i$ Repeat until all examples are correctly classified. w is some linear combination $\sum_i \alpha_i x_i (y_i * x_i)$ of data points, and decision boundary is the linear hyperplane $f(x) = w^T x + b$. **Note** that the perceptron is the same as stochastic gradient descent with a hinge loss function of $\max(0, 1 - y_i[< w, x_i > + b])$ (**we can’t remove 1 in the loss function; otherwise we can set $w, b = 0$**).

Convergence of perceptron proof 1 Here we use a perceptron without b . Assume we have w^* that has margin γ ($\min(w^*)^T y_i x_i = \gamma$), and $\|w^*\| = 1, \|x_i\| = 1$. We start from $w_0 = 0$. Assume that we have made M mistakes. We have 1) $w_M \cdot w^* = (w_{M-1} + y_i x_i) \cdot w^* \geq w_{M-1} \cdot w^* + \gamma$. So we have $w_M \cdot w^* \geq M\gamma$.

2) $w_M \cdot w_M = (w_{M-1} + y_i x_i) \cdot (w_{M-1} + y_i x_i) = w_{M-1} \cdot w_{M-1} + 2y_i x_i \cdot w_{M-1} + (y_i x_i) \cdot (y_i x_i) \leq w_{M-1} \cdot w_{M-1} + 1$. So we have $w_M \cdot w_M < M$.

Combining them, using Cauchy-Schwarz, we have $M\gamma \leq w_M \cdot w^* \leq \|w_M\| \|w^*\| \leq \sqrt{M}$. So $M \leq 1/\gamma^2$. **proof 2** Let potential function $Q_i = \|w_i\| - w_i \cdot w^*$, where i is the number of iterations. Assuming up to iteration i , we have M mistakes, so we have $Q_i \leq \sqrt{M} - M\gamma$. Clearly $Q_i \geq 0$ by Cauchy-Schwarz. So we have $\sqrt{M} - M\gamma \geq 0$.

Kernel

Kernel function $k(x, x') = \phi(x)^T \phi(x')$ for some $\phi(\cdot)$. For a set of data points $\{x_i\}$, we have Gram matrix (kernel matrix) $K_{ij} = k(x_i, x_j)$. A **necessary and sufficient** condition for being a valid kernel function: K always positive semidefinite. Proof: $\alpha^T K \alpha = \sum_{ij} \alpha_i \alpha_j K_{ij} = \sum_{ij} \alpha_i \alpha_j \langle \phi(x_i), \phi(x_j) \rangle = \langle \sum_i \alpha_i \phi(x_i), \sum_j \alpha_j \phi(x_j) \rangle \geq 0$.

Mercer’s Theorem for any symmetric function

$k: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ which is square integrable and satisfying $\int_{\mathcal{X} \times \mathcal{X}} k(x, x') f(x) f(x') dx dx' \geq 0$ for $f \in L_2(\mathcal{X})$, we have a feature space $\Phi(x)$ and $\lambda \geq 0$ that $k(x, x') = \sum_i \lambda_i \phi_i(x) \phi_i(x')$.

new kernel from old ones Given kernels $k_1(x, x'), k_2(x, x')$, $ck_1(x, x'), f(x)k_1(x, x')f(x'), k_1(x, x') + k_2(x, x'), k_1(x, x')k_2(x, x')$ are new valid kernels. Proof: 1) write the kernel as the dot product of two vectors; 2) use

Mercer’s Theorem; 3) any Gram matrix derived from it is positive semidefinite. $k_1(x, x') - k_2(x, x')$ is **invalid**: let $k_1(x, x') = 1$ for $x = x'$, and 0 otherwise, and $k_2(x, x') = 2k_1(x, x')$. The Gram matrix of new kernel is not PSD.

PSD matrices Products of two PSD matrices are not always PSD. Let A be 2×2 PSD, and B be $\text{diag}(1, 2)$. AB is not PSD (columns scaled differently). **PSD’s eigen decomposition** $A = UDU^T$. $A^{m+1} = AA^m = UDU^T(UD^mU^T) = UD(U^T U)D^mU^T = UD^{m+1}U^T$. **Any PSD is a covariance matrix** Let x be a random vector with covariance I , PSD Q is the covariance matrix for $Q^{1/2}x$: $\text{cov}(Q^{1/2}x) = Q^{1/2} \text{cov}(x) Q^{1/2} = Q^{1/2} Q^{1/2} = Q$. **examples** polynomial: $\langle x, x' \rangle + c^d, c \geq 0$. For $c = 0$, it’s a polynomial having all terms of order d ; for $c > 0$, it contains all terms of order up to d .

Convexity

Convex Sets

Definition: A set C is *convex* if the line segment between any two points in C lies in C , i.e. if for any $x_1, x_2 \in C$ and any θ with $0 \leq \theta \leq 1$, we have

$$\theta x_1 + (1 - \theta)x_2 \in C$$

Examples:

- Empty set \emptyset , single point x_0 , the whole space \mathbb{R}^n
- Hyperplane $\{x|a^T x = b\}$, halfspaces $\{x|a^T x \leq b\}$
- Euclidean balls $\{x|\|x - x_c\|_2 \leq r\}$
- Positive semidefinite matrices $S_+^n = \{A \in S^n | A \succeq 0\}$ (S^n is the set of symmetric $n \times n$ matrices)

Convexity preserving set operations:

- Translation $\{x + b|x \in C\}$
- Scaling $\{\lambda x|x \in C\}$
- Affine function $\{Ax + b|x \in C\}$
- Intersection $C \cap D$
- Set sum $C + D = \{x + y|x \in C, y \in D\}$

Convex Functions

Definition: A function $f: \mathbb{R}^n \rightarrow \mathbb{R}$ is *convex* if **dom** f is a convex set and if for all $x, y \in \text{dom}f$, and θ with $0 \leq \theta \leq 1$, we have

$$f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 - \theta)f(y)$$

First-order conditions: Suppose f is differentiable. Then f is convex if and only if **dom** f is convex and

$$f(y) \geq f(x) + \nabla f(x)^T (y - x)$$

Second-order conditions: Assume that f is twice differentiable