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 $r_i = \frac{r}{k} \sum_{x \in \mathrm{NN}(x_i, k)} \|x_i - x\|$.
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

$\frac{\partial \boldsymbol{x}^T \boldsymbol{a}}{\partial \boldsymbol{x}} = \frac{\partial \boldsymbol{a}^T \boldsymbol{x}}{\partial \boldsymbol{x}} = \boldsymbol{a}$
$\frac{\partial \boldsymbol{a}^T \boldsymbol{X} \boldsymbol{b}}{\partial \boldsymbol{X}} = \boldsymbol{a}\boldsymbol{b}^T$, $\frac{\partial \boldsymbol{a}^T \boldsymbol{X}^T \boldsymbol{b}}{\partial \boldsymbol{X}} = \boldsymbol{b}\boldsymbol{a}^T$, $\frac{\partial \boldsymbol{a}^T (\boldsymbol{X}^T |\boldsymbol{X}) \boldsymbol{a}}{\partial \boldsymbol{X}} = \boldsymbol{a}\boldsymbol{a}^T$
$W \in \boldsymbol{S}$, $\frac{\partial}{\partial \boldsymbol{s}} (\boldsymbol{x} - \boldsymbol{A}\boldsymbol{s})^T \boldsymbol{W} (\boldsymbol{x} - \boldsymbol{A}\boldsymbol{s}) = -2\boldsymbol{A}^T \boldsymbol{W} (\boldsymbol{x} - \boldsymbol{A}\boldsymbol{s})$,
$\frac{\partial}{\partial \boldsymbol{x}} (\boldsymbol{x} - \boldsymbol{s})^T \boldsymbol{W} (\boldsymbol{x} - \boldsymbol{s}) = 2\boldsymbol{W}(\boldsymbol{x} - \boldsymbol{s})$,
$\frac{\partial}{\partial \boldsymbol{s}} (\boldsymbol{x} - \boldsymbol{s})^T \boldsymbol{W} (\boldsymbol{x} - \boldsymbol{s}) = -2\boldsymbol{W}(\boldsymbol{x} - \boldsymbol{s})$,
$\frac{\partial}{\partial \boldsymbol{x}} (\boldsymbol{x} - \boldsymbol{A}\boldsymbol{s})^T \boldsymbol{W} (\boldsymbol{x} - \boldsymbol{A}\boldsymbol{s}) = 2\boldsymbol{W}(\boldsymbol{x} - \boldsymbol{A}\boldsymbol{s})$,
$\frac{\partial}{\partial \boldsymbol{A}} (\boldsymbol{x} - \boldsymbol{A}\boldsymbol{s})^T \boldsymbol{W} (\boldsymbol{x} - \boldsymbol{A}\boldsymbol{s}) = -2\boldsymbol{W}(\boldsymbol{x} - \boldsymbol{A}\boldsymbol{s})\boldsymbol{s}^T$.
$\mathrm{Tr}(\boldsymbol{A}) = \sum_i \boldsymbol{A}_{ii}$. For two equal sized matrices, $\mathrm{Tr}(\boldsymbol{A}^T \boldsymbol{B}) = \mathrm{Tr}(\boldsymbol{B}^T \boldsymbol{A}) = \mathrm{Tr}(\boldsymbol{A}\boldsymbol{B}^T) = \mathrm{Tr}(\boldsymbol{B}\boldsymbol{A}^T) = \sum_{i,j} \boldsymbol{A}_{ij} \boldsymbol{B}_{ij}$.
$\mathrm{Tr}(\boldsymbol{A}) = \mathrm{Tr}(\boldsymbol{A}^T), \mathrm{Tr}(\boldsymbol{A} + \boldsymbol{B}) = \mathrm{Tr}(\boldsymbol{A}) + \mathrm{Tr}(\boldsymbol{B})$. For square matricis,
$\mathrm{Tr}(\boldsymbol{A}\boldsymbol{B}) = \mathrm{Tr}(\boldsymbol{B}\boldsymbol{A}), \mathrm{Tr}(\boldsymbol{A}\boldsymbol{B}\boldsymbol{C}) = \mathrm{Tr}(\boldsymbol{C}\boldsymbol{A}\boldsymbol{B}) = \mathrm{Tr}(\boldsymbol{B}\boldsymbol{C}\boldsymbol{A})$ (trace rotation).

Classifers and Regressors

Naive Bayes Conditionally independent:
$P(x_1, x_2, \ldots |C) = \prod_i P(x_i|C)$. One way to avoid divide by zero: add $(1, 1, \ldots, 1)$ and $(0, 0, \ldots, 0)$ 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
$P(x_i|y) = \mathcal{N}(\mu_{X_i|y}, \sigma^2_{X_i|y})$, 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 $\boldsymbol{w}^*$ that has margin γ ($\min(\boldsymbol{w}^*)^T y_i x_i = \gamma$), and $\|\boldsymbol{w}^*\| = 1, \|x_i\| = 1$. We start from $\boldsymbol{w}_0 = 0$. Assume that we have made M mistakes. We have 1) $\boldsymbol{w}_M \cdot \boldsymbol{w}^* = (\boldsymbol{w}_{M-1} + y_i x_i) \cdot \boldsymbol{w}^* \geq \boldsymbol{w}_{M-1} \cdot \boldsymbol{w}^* + \gamma$. So we have $\boldsymbol{w}_M \cdot \boldsymbol{w}^* \geq M\gamma$.
2) $\boldsymbol{w}_M \cdot \boldsymbol{w}_M = (\boldsymbol{w}_{M-1} + y_i x_i) \cdot (\boldsymbol{w}_{M-1} + y_i x_i) = \boldsymbol{w}_{M-1} \cdot \boldsymbol{w}_{M-1} + 2 y_i x_i \cdot \boldsymbol{w}_{M-1} + (y_i x_i) \cdot (y_i x_i) \leq \boldsymbol{w}_{M-1} \cdot \boldsymbol{w}_{M-1} + 1$. So we have $\boldsymbol{w}_M \cdot \boldsymbol{w}_M < M$.
Combining them, using Cauchy-Schwarz, we have $M\gamma \leq \boldsymbol{w}_M \cdot \boldsymbol{w}^* \leq \|\boldsymbol{w}_M\| \|\boldsymbol{w}^*\| \leq \sqrt{M}$. So $M \leq 1/\gamma^2$. **proof 2** Let potential function $Q_i = \|\boldsymbol{w}_i\| - \boldsymbol{w}_i \cdot \boldsymbol{w}^*$, 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 $k(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})^T \phi(\boldsymbol{x}')$ for some $\phi(\cdot)$. For a set of data points $\{\boldsymbol{x}_i\}$, we have Gram matrix (kernel matrix) $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. A **necessary and sufficient** condition for being a valid kernel function: K always positive semidefinite. Proof: $\boldsymbol{\alpha}^T K \boldsymbol{\alpha} = \sum_{ij} \alpha_i \alpha_j K_{ij} = \sum_{ij} \alpha_i \alpha_j \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle = \langle \sum_i \alpha_i \phi(\boldsymbol{x}_i), \sum_j \alpha_j \phi(\boldsymbol{x}_j) \rangle \geq 0$.
Mercer's Theorem for any symmtric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ which is square integrable and satisfying $\int_{\mathcal{X} \times \mathcal{X}} k(x, x') f(x) f(x') \mathrm{d}x \mathrm{d}x' \geq 0$ for $f \in L_2(\mathcal{X})$, we have a feature space $\Phi(x)$ and $\boldsymbol{\lambda} \geq 0$ that $k(x, x') = \sum_i \lambda_i \phi_i(x) \phi_i(x')$.

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 marices $S^n_+ = \{A \in S^n | A \succeq 0\}$ (S^n is the set of symmetric $n \times n$ matrices)

Convexit 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 $f : \mathbb{R}^n \to \mathbb{R}$ is *convex* if $\mathbf{dom} f$ is a convex set and if for all $x, y \in \mathbf{dom} f$, 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 $\mathbf{dom} f$ is convex and

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

Second-order conditions: Assume that f is twice differentiable