CS189 Midterm 1 Study Guide

Equations

Probability

Bayes' rule	$P(A\|B) = P(A)P(B\|A)/P(B)$
Expected value	$\mathbb{E}[cX] = c\mathbb{E}[X]$
Variance	$Var(X) = E[(X - E[X])^2] = \sigma^2$
	$Var(cX) = c^2Var(X)$
Covariance	$\mathbb{E}[(x_i - \mu_i)(x_j - \mu_j)]$
Covariance matrix	$X^TX = US^2U^T$ for centered dataset
	$\sigma_{ij} = \frac{1}{N}\sum_{k=1:N}(x_i^k - \mu_i)(x_j^k - \mu_{xj})$
Hessian	$[\partial^2 R/\partial w_i \partial w_j]$
Hessian matrix	$\partial^2 R/\partial w^2$

Distributions

Isotropic normal	$\frac{1}{\sigma\sqrt{2\pi}}\exp(-(x-\mu)^2/(2\sigma^2))$
Multivariate normal	$\frac{1}{\sqrt{(2\pi)^k\|\Sigma\|}}\exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu))$
Poisson distribution	$p(x,\lambda) = (e^{-\lambda}\lambda^x)/x!$

Matrices:

$(AB)^T$	B^TA^T
$(A^T)^{-1}$	$(A^{-1})^T$
$\partial(x^Ta)/\partial x$	a
$\partial(x^TAx)/\partial x$	$(A + A^T)x$
$\partial Trace(XA)/\partial X$	A^T
$\partial(a^TXb)/\partial X$	ab^T

Maximum Likelihood Estimation Find likelihood function, take negative log likelihood, take derivative with respect to optimized parameter. Set derivative to 0. $argmax[P(data|model)]$

Maximum A Posteriori (MAP) $argmin[-\log P(D|f)(-\log P(f))]$. $-\log P(f)$ serves the role of regularizer. $argmax[P(data|model)P(model)]$

Training Methods

Hebb's rule $w = \sum_k y^kx^k$, $w \leftarrow w + y^k\Phi(x^k)$, $\alpha^k \leftarrow \alpha^k + y^k$

Weight decay $w_i \leftarrow (1-\gamma)w_i + y^kx_i^k$

Perceptron algorithm Always finds separating hyperplane if samples linearly separable. Equations: $\Delta w_i = \eta y\Phi_i(x_i)$, $\Delta\alpha_k = \eta y^k$. Normal perceptron: $z < 0$, update if misclassified. Large-margin: $z < 1$, update if misclassified or within margin. Optimum margin: $min(z)$, only update if closest point to margin.

Linear SVM Fits a hyperplane to separate and classify data. Soft-margin hyperparameter C represents tradeoff between robustness and fit. Minimizes $1/\text{margin} + C(\text{training error})$. Approaches hard margin as $C \to \infty$, and vice versa

Newton's Method $w \leftarrow w - H^{-1}\nabla_w R$

Stochastic Gradient Descent $\Delta w_i = -\eta\partial L/\partial w_i$. Obtain dual form $\Delta\alpha$ by using kernel trick, NOT by differentiating loss. Does not exploit convexity of risk function; best approach for big data but NOT when N or d is small.

Update rules:

least mean squares:	$\Delta w_i = \eta(y - f(x))\Phi_i(x)$
	$\Delta\alpha_k = \eta(y^k - f(x^k))$

Risk Minimization

Loss functions

Square	$(1-z)^2$
Perceptron	$max(0, -z)$
Hinge	$max(0, 1-z)$
Logistic	$\ln(1 + e^{-z})$
Adaboost	e^{-z}

Risk/Loss Functionals

functional margin	$z = yf(x), y = \pm 1$
risk	$(1/N)\sum_{k=1:N} L(f(x^k, w), y^k)$
batch gradient	$w \leftarrow w - \eta\nabla_w R$
stochastic gradient	$w \leftarrow w - \eta\nabla_w L$
SRM/regularization	$w_i \leftarrow (1-\gamma)w_i - \eta\partial R_{train}/\partial w_j$
linear discriminant	$f(x) = \sum_i w_ix_i = wx$

Regularization penalizes model complexity at expense of more training error, often explicitly part of loss function. Takes the form of shrinkage (weight decay, ridge regression, SVM), feature selection, kernel parameters, etc.

Risk Types

guaranteed risk: $R_{gua}[f] = R_{train}[f] + \epsilon(\delta, C/N)$ with high probability $(1-\delta)$

regularized risk: $R_{reg}[f] = R_{train}[f] + \lambda\|w\|^2$

Norms $\|x\|_p = (|x_1|^p + |x_2|^p + \cdots)^{1/p}$

L_0 norm	penalizes number of features considered
L_1 norm (lasso)	makes weight vector more sparse
L_2 norm (ridge)	shrinks weight vector to reduce variance

Logistic Regression

Like Hebb's rule but weighted; misclassifications are more heavily weighted (multiplied by $S(-z)$). A link function links the functional margin z and the likelihood $P(D|f)$.

Linear logistic regression log odds ratio (logit) is a linear function of x, $\ln[P_f(Y=1|X=x)/P_f(Y=-1|X=x)] = w\cdot x + b$

Logistic function $S(t) = g^{-1}(t) = 1/(1+e^{-t})$

$$\begin{aligned}\Delta w_i &= -\eta\partial L/\partial w_i = -\eta\partial L/\partial z \cdot \partial z/\partial w_i \\ &= \eta S(-z)yx_i\end{aligned}$$

Update equations with shrinkage:

$$\Delta w_i = (-\gamma w_i) + \eta S(-z)y\Phi_i(x)$$

$$\Delta\alpha_k = (-\gamma\alpha_k) + \eta S(-z)y^k \text{ for example k}$$

$$\Delta\alpha_h = (-\gamma\alpha_h) \text{ for other examples}$$

Ridge Regression

$\sum_i w_ix_i^k = y^k$ for all $k = 1..m$

$$\begin{aligned}Xw^T &= y \\ X^TXw^T &= X^Ty \\ w^T &= (X^TX)^{-1}X^Ty\end{aligned}$$

Optimal solution: $w^T = X^+y$

Residual $y - \hat{y} = (I - XX^+)y$

Pseudo-inverse

Case 1) $N > d$ overdetermined, no exact solution. Optimal RSS solution is
$X^+ = \lim_{\lambda\to 0}(X^TX + \lambda I)^{-1}X^T$
Case 2) $N < d$ underdetermined, optimize for $\min(\|w\|)$
$X^+ = \lim_{\lambda\to 0} X^T(XX^T + \lambda I)^{-1}$
Not limit when $\lambda \to 0$, but find optimal value through cross-validation.

Kernel trick In case 2 where $N << d$, dimensionality of features can approach ∞. Instead, replace XX^T by a (N, N) kernel matrix $K = k(x^k, x^h)$. $\alpha = (K + \lambda I)^{-1}y$ yields the nonlinear regression function $f(x) = \sum_k \alpha_k k(x, x_k)$.

Principal Component Analysis (PCA) decrease dimensionality of features by constructing linear combinations of the features such that the reconstructed patterns are as close as possible to the original features (minimize RSS). Accomplishes this by removing the dimensions with the smallest eigenvalues (smallest variance, affects the data the least).

PCA Solution: $X' = XU$, $X'' = X'U^T = XUU^T$, $min_U\|X - XUU^T\|^2$

Kernel Machines

Kernels similarity measure between two datapoints, dot product in some (potentially infinite) feature space: $k(s,t) = \Phi(s)\cdot\Phi(t)$ (don't need to know Φ representation), good kernels are symmetric: $k(x,x') = k(x',x)$, kernel matrix should be invertible, possibly after regularization $(K + \lambda I)$. Satisfied if matrix is PSD, all eigenvalues $\lambda_i > 0$.

Kernel Trick

parametric	$f(x) = w\cdot\Phi(x)$
	$w = \sum_k \alpha_k\Phi(x^k)$
non-parametric	$f(x) = \sum_k \alpha_k k(x^k, x)$
	$k(x^k, x) = \Phi(x^k)\cdot\Phi(x)$

Parzen window $f(x) = \sum_{k=1:N} y_k k(x, x_k)$

Radial kernels

Top-hat	$k(x, x_k) = 1(\|x - x_k\|^2 < \sigma^2)$
Gaussian	$\exp -(\|x - x_k\|^2/2\sigma^2)$

Non-radial kernels

Linear	$k(x, x_k) = x \cdot x_k$
Polynomial	$k(x, x_k) = (1 + x\cdot x_k)^q$

Bayesian Decision Theory

Assumptions Datasets should be IID for naive Bayes.

Bonferroni Correction: $p' = mp$, where we use m classifiers.

Performance Evaluation

Metrics error rate, bit-error rate (BER), cross-entropy, area under the ROC curve (AUC), R^2

Error bar Σ of the error rate is $\sqrt{E(1-E)/n}$

Sample size Number of test examples needed is inversely proportional to error rate. Rule of thumb: $n = 100/E$

Model Selection

Filters reduce number of hyperparameters by measuring feature relevance. Does not make use of the learning machine, uses statistics instead. Techniques used are prior knowledge, unsupervised learning to reduce dimensionality of input space, transfer learning, meta-learning, surrogate learning machines

Wrappers use the learning machine to evaluate the performance of alternative feature subsets. Use a search method to explore the space of possible subsets. Considered a black box, no knowledge of the learning algorithm is needed to apply this method.

Embedded methods reduce the number of hyperparameters as much as possible before passing into wrapper method (too many hyperparameters leads to overfitting). Pushes hyperparameters to a lower level and have them estimated in the process of learning parameters. This cannot be done by optimizing the training error; need two levels of inference in order to enjoy finite capacity of the learning problem. Optimizing the kernel parameters (kernel with in the Gaussian kernel or degree in the polynomial kernel) leads to infinite VC dimension (can learn perfectly any training set); optimizing the ridge or regularization parameter Λ leads to zero capacity ($w = 0$ is the optimum).

Search strategies grid search, simulated annealing, pattern search
Voting methods multiple models cast a vote for a class, classifier weighs each vote and makes a decision.

Gaussian Classification

Assumptions: Generating model (draw y first, draw x given y), variance in dataset explained by Gaussian noise, independence of features in given class (no covariance), same variance for all classes. NOT optimum Bayes classifier because assumptions almost always violated. We can post-fit bias term by adjusting the threshold. If two classes have same variance, Gaussian classifier is a linear discriminant (equivalent to centroid method, Hebb's rule with target values $y^k = 1/N_1$ if in class 1 and $-1/N_0$ otherwise). Posterior of Gaussian model is logistic with Poisson or Gaussian class conditionals.
Bayes' rule $P(Y = y|X = x)\ P(Y = y)P(X = x|Y = y)$
Prior $P(Y = y)$, relative class abundance (occurrences over total count)
Likelihood $P(X = x|Y = y)$, probability x belongs to class y, proportional to $\exp(-\|x - \mu^{|y|}\|^2/2\sigma^2)$, Gaussian noise model

Linear Discriminant Analysis

Centering: subtracting mean of the features
Standardizing sphering; subtracting by mean, dividing by standard deviation (component-wise), $\Phi(x^k) = (x^k - \mu)/\sigma$
Principal axes Diagonalize $\Sigma = X^T X = US^2U^T$, where U is an orthogonal matrix of column eigenvectors such that $U^T U = I$ and S^2 is a diagonal matrix of positive eigenvalues s_i^2. Matrix $R = USU^T = \Sigma^{1/2}$. To obtain a diagonal covariance matrix S, $\Phi = XU$. To axis-align the Gaussian, apply $U^{-1}(X - \mu)$
Whitening multiply by square root of inverse covariance matrix, $\Phi = X(X^T X)^{-1/2} = XR^{-1} = X\Sigma^{-1/2}$
Linear Discriminant Analysis Generalization of Gaussian classifier for cases where the input variables aren't statistically independent, but all classes have same covariance matrix Σ. Once the input space is rotated into the principal axes of the covariance matrix and rescaled by the eigenvalues, LDA is like the isotropic Gaussian classifier (centroid method). Useful for multi-class classification and data visualization.

Rotate input space, then find the direction that maximizes the ratio of the between class variance over the within class variance.

$$max\left(\frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2}\right)$$

PCA, ridge regression use covariance matrix of all data combined. LCA uses pooled, within-class covariance. In both cases, whitening is applied with $\Phi = X\Sigma^{-1/2}$.