TERMINOLOGY

CONSISTENCY

An estimator is consistent when its **bias is zero as the number of examples grows to infinity for any distribution $P(y|x)$.**

A model selection criterion is consistent if **the probability of selecting the right model approaches 1 as n goes to infinity**.

UNBIASED ESTIMATOR

Based on a dataset, if:

$$E[\hat{\theta}] = \theta$$

Then the estimation is unbiased; else it is biased.

Note that 'bias' $E[\hat{\theta} - \theta]$here represent the difference between estimated value of a parameter and the true value of the parameter. It is a metric of the estimator (e.g. MLE or MAP)

HAT-MATRIX

Maps the labels y to their estimation $\hat{y}$

$$\hat{y} = Hy$$

Describes the influence each y has on each prediction

Diagonal elements of H are leverages

For linear regression: $H = X(X^TX)^{-1}X^T$

CONVEX

We can use gradient descent to solve convex optimization.

PROBABILITY

COVARIANCE

Let $\boldsymbol{X} = (X, Y)^T$

$$Cov(\boldsymbol{X}) = E[(\boldsymbol{X} - E[\boldsymbol{X}])^T(\boldsymbol{X} - E[\boldsymbol{X}])]$$

BAYES

$p(\theta)$ – parameter prior

$p(\theta|D)$ - posterior

$p(D|\theta)$ – likelihood function

KL-DIVERGENCE

$$D_{KL}(P\|Q) = \sum_i P(i)\log\frac{P(i)}{Q(i)}$$

CONJUGATE PRIOR

If the posterior distributions $p(\theta|D)$ are in the same family as the prior probability distribution $p(\theta)$, the prior and posterior are then called conjugate distributions, and the **prior** is called a **conjugate prior** for the **likelihood function** $p(D|\theta)$.

likelihood	conjugate prior
Bernoulli	Beta
Poisson	Gamma
Gaussian with known variance (mean as p)	Gaussian
Gaussian with known mean (variance as p)	Inverse Gamma

MLE V.S. MAP

Both are parameter estimation method.

MLE	MAP/Bayesian
$\hat{\theta}_{MLE} = argmaxP(D\|\theta)$	$\hat{\theta}_{MAP} = argmaxP(D\|\theta)$
unbiased	Biased
Minimize $bias$	Minimize $bias^2 + variance$
Large variance	Lower variance
Better use MAP when do not have enough data (avoid overfitting)	
When (1) $n \to \infty$ or (2) $\sigma_{prior} \to \infty$, $\hat{\theta}_{MAP}$ reduce to $\hat{\theta}_{MLE}$	

ESTIMATE GAUSSIAN PARAMETERS

MLE method: skip

MAP method: using conjugate prior!

INFORMATION THEORY

CONDITIONAL ENTROPY

$$H(Y|X)$$

INFORMATION GAIN

$$IG(Y|X) = H(Y) - H(Y|X) = H(Y) - \sum_{x_i} P(X = x_i)H(Y|X = x_i) = H(Y) - \sum_{x_i} P(X = x_i)(-\sum_{y_i} P(Y = y_i)\log P(Y = y_i)) = \sum_{x_i,y_i} P(x_i, y_i)\log\left(P(x_i)/P(x_i, y_i)\right)$$

Notice that information gain is "myopic". When building a decision tree, we should continue splitting even if all IG are 0.

MACHINE LEARNING IN GENERAL

TYPE OF ML

Supervised; Unsupervised; Semi-supervised; Reinforcement

GENERATIVE V.S. DISCRIMINATIVE

Generative models: $P(X, Y)$

Discriminative models: $P(Y|X)$

	SUP	UNSUP
GEN	Naïve Bayes	GMM
DIS	Logistic Reg.	*

* most discriminative models are inherently supervised

ESTIMATION V.S. OPTIMIZATION

Two different form of the same thing. Difference is that Bayesian way gives us a probability distribution of the parameter(s) we try to estimate.

OVERFITTING AND REGULARIZATION

Training error, true error and test (set) error.

How to adjust complexity penalty:

Cross Validation	LOOCV
	K-fold cross validation
MDL	AIC, BIC, RIC …

NEAREST NEIGHBOR

K-NN

1-NN is non-parametric;

Under some reasonable regularity conditions on $P(y|x)$, 1-Nearest Neighbor is consistent;

KERNEL REGRESSION

$\hat{y}(\mathbf{x}) = \frac{\sum_i k(\mathbf{x},\mathbf{x_i})y_i}{\sum_i k(\mathbf{x},\mathbf{x_i})}$ No parameter estimated!

σ increase, the decision boundary became smoother; $\sigma \to 0$, KR becomes like 1-NN

DECISION BOUNDARY

K-NN (K>1) has smoother decision boundary than 1-NN (for the same training data, of course), the decision boundary of kernel regression is smoother than K-NN;

DECISION TREE

ID3-ALGORITHM

ADVANTAGE OVER KNN

- Faster
- Smaller to save
- relative robustness to noisy or irrelevant features

REGRESSION

LINEAR REGRESSION

$$p(y|w, x, \sigma)$$

$$y \sim N(wx, \sigma^2)$$

σ is the **noise** of data, which we cannot control.

MLE

$$w_{MLE} = (X^TX)^{-1}X^Ty$$

(When features are not linearly independent, X^TX is not invertible, problem!)

MAP/Ridge Regression

Assume $w \sim N(0, \lambda^2)$

$$w_{MAP} = \left(X^TX + {\sigma^2}/{\lambda^2}\, I\right)^{-1} X^Ty$$

- $\sigma \to 0$, w_{MAP} reduce to w_{MLE}
- $\lambda \to \infty$, w_{MAP} reduce to w_{MLE}
- $\sigma \to \infty$, $w_{MAP} \to 0$
- N increase, the effect of prior vanishes.

NON-LINEAR REGRESSION

Non close-form solutions: Line search, simulated annealing, gradient descent, conjugate gradient, Levenberg Marquant, Newton's method, EM algorithm …

Polynomial Regression: add high order terms, then conduct linear regression

For q[th] degree polynomial regression, number of terms: $\boldsymbol{C^m_{Q+m}}$ (m is the dimension of original X)

BIAS VARIANCE TRADE OFF / TRUE ERROR (OF MODEL) DECOMPOSTION

Both bias and variance quantifies the error.

Bias - how much cannot a model represents the complexity of the true label distribution;

Variance – how much a model represents the too much so that it leads to overfitting.

Bias-variance decomposition for regression*:*

$$E_{x,y,D}[(h(x;D) - y)^2] = E_{x,D}[\left(h(x;D) - \bar{h}(x)\right)^2\{variance\} + E_x\left[\left(\bar{h}(x) - \bar{y}(x)\right)^2\right]\{bias^2\} + E_{x,y}[(\bar{y}(x) - y)^2]\{noise, cannot\ control\}$$

$bias^2$~training error, the more complex the model is, the smaller $bias^2$ is;

$variance$~test error-training error, the simpler a model is/the more data we have, the smaller $variance$ is

REGULARIZATION

Objective: Control the extent to which we

"trust" the training data.

L_2	Convex	Gaussian Prior	Ridge Regression
L_1	Convex	Laplace Prior	LASSO
L_0	Non-convex	Spike and slab	Stepwise Regression

Elastic net regularization: L_1+L_2
Convex, global optimum

STREAM/STEP/STAGE WISE (FOR L_0)

1. Stream-wise: For each feature, try adding it to the model and train a new set of *'w'*s each time. If error decreases, accept new model. [Complexity: $p(nq^2+q^3)$]
2. Step-wise: Pick a feature (not in current model) which gives the lowest error (while calculating the error, train a new set of *'w'*s each time); if error decreases, accept new model, otherwise stop calculating. [Complexity: : $qp(nq^2+q^3)$]
3. Stage-wise: Like step-wise. Except keep all the coefficients for the former model. [Complexity: $qp(2n)$]
4. All subsets regression: Expensive! [Complexity: $2^p(nq^2+q^3)$]

NAÏVE BAYES

$P(y|x) \sim P(x,y) = P(y)P(x|y)$

$P(y)$ is class prior; $P(x|y)$ is class model.

They are separately estimated.

When lack of samples, MLE estimation may lead to undesired results. MAP with conjugate prior, does better. So use MLE for prior estimation, use MAP for $P(x|y)$

LOGISTIC REGRESSION

For binary label data:

$$P(Y=\pm 1|\mathbf{x},\mathbf{w}) = \frac{1}{1+\exp(-y\mathbf{w}_i^{\mathrm{T}}\mathrm{x})}$$

For multinomial MAP, use

$$P(Y=i|\mathbf{x},\mathbf{w}) = \frac{\exp(\mathbf{w}_i^{\mathrm{T}}\mathrm{x})}{\sum_{j=1}^{K}\exp(\mathbf{w}_j^{\mathrm{T}}\mathrm{x})}$$

to avoid bias.

A problem of using MLE in LG, it tends to increase w so it will never converge. MAP can solve this problem by adding a prior: (same as linear regression)

NB VS LR

Both are classification methods.

NB	LR
Generative	Discriminative
(MLE) close form solution	(MLE) iterative solution (concave)
Linear/quadratic boundary	Linear boundary
Small dataset vs parameter	Enough data vs parameter
Number of training samples increase, naive Bayes initially performs better but is then overtaken by logistic regression	

Note that the decision rule of GNB with shared variance parameter (for each feature over classes) is of the **same parametric form** as logistic regression; However, the decision boundaries are **NOT** the same!

RADIAL BASIS FUNCTION

Basis function: turn non-linear separable data into separable data.

Radial basis function:

Can either increase/decrease/keep the dimension of input.

$\phi_i(x) = KernelFunction(|x-c_i|/KW)$

Use gradient descent to decide parameters.

ROBUST REGRESSION

LOESS-based robust regression:

Give large weight to well estimated data points and small weight to badly estimated data points.

Thought, assume with probability p, the linear regression model is right, elsewise $y \sim N(\mu, \sigma^2_{huge})$

MDL

Minimize: $\frac{Err_q}{2\sigma^2} + \lambda|w|_0$

THE NATURE OF AIC, BIC AND RIC

They are regularization strategy using the thought of MDL. We can train λ using these models.

	AIC	BIC	RIC
λ	1	$\log\sqrt{n}$	$\log p$
when	$p \gg \sqrt{n}$	$p \ll \sqrt{n}$	$p \gg \sqrt{n}$
	$q \approx p/2$	-	$q \ll p$
u/o	-	under	-
consis	N	Y	N

P-VALUE

The probability of getting false positive

BOOSTING

ADABOOST

Training algorithm

Input: n examples (x_i, y_i)

Initialize: $D_1(i) = 1/n$

For t = 1 to T

Train weak classifier on $D_i(i), h_t(x)$

$\epsilon_t = \sum_i D_t(i)\mathbf{1}(y_i \neq h_t(x_i))$

$\alpha_t = \frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}$

$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$, $Z_t = \sum_i D_t(i)\exp(-\alpha_t y_i h_t(x_i))$

Output classifier: $h(x) = \sum_t \alpha_t h_t(x)$

Exponential loss minimization:

$\frac{1}{n}\sum_i 1(y_i \neq h(x_i)) \leq$

$\frac{1}{n}\sum_i \exp(-y_i f_\alpha(x_i)) = \prod_t Z_t \leq$

$\exp(-2T\gamma^2), \gamma = \min_t(0.5-\epsilon_t)$

Test error keeps decreasing long after the training error has gone down to zero – confidence of classification, margin: $\frac{y_i f_\alpha(x_i)}{\|\alpha\|_1}$

measure the distance between the example and the boundary. The more confident the boundary is, the more robust to noise the model is.

BOOSTING V.S. LR

Advantage of boosting: No need to enumerate all possible classifiers; Saves time

Disadvantage of boosting: Exponential loss is more sensitive to label noise

APPENDIX

NORM $L^p = (\sum_i |x_i|^p)^{1/p}$

Frobenius norm: $\|A\|_F = \sqrt{\sum_i \sum_j a_{ij}^2} = \sqrt{}$

$L_0\ pseudo-norm: number\ of\ x_i \neq 0$

PRECISION AND MATRIX

$precision = \frac{true\ pos}{true\ pos + false\ pos}$

$recall = \frac{true\ pos}{true\ pos + false\ neg}$

$accuracy = \frac{true\ pos + true\ neg}{total}$

$specifity = \frac{true\ neg}{true\ neg + false\ pos}$

RUC and AUC:

RUC: x axis p(FP), y axis p(TP)

AUC [0,1] AUC=0.5 is random guessing, AUC=1.0 is perfect prediction.

OTHERS

$x \to 0, x\log x \to 0$

ML SPEED

Expedite: faster language, GPU, vectorize; streaming algorithm; data sampling; dimensionality reduction; faster algorithm

slow	Fast
Logistic	Linear
Kernel SVM	Linear SVM
Step-wise	Stage-wise
K-NN	K-means

Speed: ridge < lasso < step-wise

Usually sparse code is faster

LOSS FUNCTION

Capture the performance of a model /

Log	Hinge
$\log(1+\exp\{-yf(x)\})$	$\max\{0, 1-yf(x)\}$
0–1 (binary)	**Squared**	**Exponential**
$\mathbf{1}(y \neq \mathrm{sign}(f(x)))$	$(y-f(x))^2$	$\exp\{-yf(x)\}$

optimization over some parameters

All convex except for 0-1!

DEEP LEARNING

BASICS IN NEURAL NETWORK

Weight decay: regularization / Bayesian prior; Batch learning; Data has noise, not all steps reduce error, on average the error goes down; Non-linearity; Can be supervised or unsupervised

DEEP NETWORKS

Semi-parametric; flexible model form; no prior, no feature selection

Used on vast amounts of data

CNN

Convolution, pooling and full connection

REGULARIZATION

Early stopping; L1/L2 penalty; set max norm; dropout (randomly remain a fraction of the weights unchanged on each iteration; bounce out of local minimum)

KERNEL METHOD

DUAL REPRESENTATION

express the prediction/estimated parameters as the linear combination of examples

Linear regression in dual representation

$\widehat{\mathbf{w}}_{MAP} = \boldsymbol{X}^T\boldsymbol{\alpha},\ \alpha_i = \frac{\lambda^2}{\sigma^2}(y_i - \widehat{\mathbf{w}}_{MAP}^T\mathbf{x}_i)$

Solving linear regression using dual representation (KRR)

$\widehat{\boldsymbol{\alpha}}_{MAP} = \left(\mathbf{K} + \frac{\sigma^2}{\lambda^2}\boldsymbol{I}\right)^{-1} y\ ,\ \mathbf{K} = \boldsymbol{X}\boldsymbol{X}^T$

$\hat{y}_{MAP}(\mathbf{x}) = \mathbf{k}(\mathbf{x})^{\mathrm{T}}\left(\mathbf{K} + \frac{\sigma^2}{\lambda^2}\boldsymbol{I}\right)^{-1}\mathbf{y}$

$\mathbf{k}(\mathbf{x}) = \mathbf{X}\mathbf{x}$ is the kernel

Logistic regression in dual

$\widehat{\mathbf{w}}_{MAP} = \boldsymbol{X}^T\boldsymbol{\alpha},\ \alpha_i = \lambda^2 y_i(1 - P(y_i|\widehat{\mathbf{w}}_{MAP}^T, \mathbf{x}_i))$

KERNEL

Objective: expand (or decrease) number of features

Intuitional meaning: "similarity" among samples, can be the opposite of a distance metric (i.e. $k(x,y) = \exp(-d(x,y))$) (but for kernels not generated this way, it's probable that $k(x,y) > k(x,x)$)

Trick: by defining **ANY** feature mapping $\phi(\mathbf{x})$, we can redefine the kernel function as the dot product in the new expanded feature space $K(\mathbf{x},\mathbf{x}') = \phi(\mathbf{x})^T\phi(\mathbf{x}')$, but calculating kernel is usually more efficient; Using quadratic kernel in linear regression is different from introducing quadratic term in linear regression (no extra parameters involved)

Construction: Kernel matrix $\mathbf{K}$ should be positive semi-definite

[About PSD: non-negative singular values; $det(K) \geq 0$; can have negative entries; the covariance matrix is PSD]

$\mathbf{K} = \sum_{i=1}^n \lambda_i \boldsymbol{z_i}\boldsymbol{z_i^T}\ ,\ \boldsymbol{z^T}\mathbf{K}\boldsymbol{z} = \sum_{i=1}^n \lambda_i\left(\boldsymbol{z_i}\boldsymbol{z_i^T}\right)^2$

1. $k(\mathbf{x},\mathbf{x}') = \mathbf{x}^{\mathrm{T}}\mathbf{x}'$
2. $k(\mathbf{x},\mathbf{x}') = ck_1(\mathbf{x},\mathbf{x}')$
3. $k(\mathbf{x},\mathbf{x}') = k_1(\mathbf{x},\mathbf{x}') + k_2(\mathbf{x},\mathbf{x}')$
4. $k(\mathbf{x},\mathbf{x}') = k_1(\mathbf{x},\mathbf{x}')k_2(\mathbf{x},\mathbf{x}')$
5. $k(\mathbf{x},\mathbf{x}') = q(k_1(\mathbf{x},\mathbf{x}'))$
6. $k(\mathbf{x},\mathbf{x}') = f(\mathbf{x})k_1(\mathbf{x},\mathbf{x}')f(\mathbf{x}')$
7. $k(\mathbf{x},\mathbf{x}') = \exp(k_1(\mathbf{x},\mathbf{x}'))$
8. $k(\mathbf{x},\mathbf{x}') = \mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x}'$

* q is polynomial with positive coefficients, c>0, A>=0

Common kernels: RBF(Gaussian), linear, polynomial, intersection …

Use: nearest neighbors, regression, SVMs and perceptron, PCA, CCA

KR & KRR & KNN

KRR - intelligent KR/KNN, optimize α_i to discover *which examples are useful for prediction*, and *how those examples should influence the output of another example*

LOCALLY WEIGHTED REGRESSION

$\widehat{\mathbf{w}}(\mathbf{x}) = argmin_w \frac{\sum_i k(\mathbf{x},\mathbf{x_i})(\mathbf{w^T}\mathbf{x_i} - y_i)^2}{\sum_i k(\mathbf{x},\mathbf{x_i})}$

Nearness is measured by $k(x,x_i)$

LINEAR SMOOTHERS* (from HW)

$\hat{y} = \sum_i l_i(x)y_i$

OPTIMIZATION & LAGRANGIAN

Optimization: non-probabilistic, discriminative; convexity for optimization;

Lagrangian: for optimization with constrains

The problem (primal)

$$\begin{cases} minimize\ f_0(x) \\ constraints: f_i(x) \leq 0, i = 1,\dots,m \\ h_j(x) = 0, j = 1,\dots,n \end{cases}$$

Introduce a set of λs (KKT multipliers) and vs (Lagrange multiplier), let:

$L(x,\lambda,v) = f_0(x) + \sum_i \lambda_i f_i(x) + \sum_j v_j h_j(x)$

The Lagrangian (dual)

$$\begin{cases} maximize\ g(\lambda,v) = \inf_x L(x,\lambda,v) \\ constraints: \lambda_i \geq 0 \end{cases}$$

the maximum of the dual problem = the minimum of the primal function

(Dual problem is always convex even if the primal is not!)

KKT Conditions:

For optimal solution (x^*,λ^*,v^*):

$\partial f_0(x^*) + \sum_i \lambda_i \partial f_i(x^*) + \sum_j v_j \partial h_j(x^*) = 0$

(complementary slackness)$\forall i, \lambda_i f_i(x^*) = 0$

(primal feasibility) $f_i(x^*) \leq 0, f_j(x^*) = 0$

(dual feasibility) $\lambda_i \geq 0$

> ***How to solve Lagrangian***
> Get $L(x,\lambda,v)$, take derivative of x, express x with λ, v; Plug in get$g(\lambda,v)$, take derivative of λ, v, plug back to get x

SVM

HINGE LOSS

Convex; grow slow below zero; nice geometric interpretation of a maximum margin hyperplane separating pos from neg; fits well with kernels (many examples are not part of solution), efficient

SEPARABLE SVM

Primal (maximize the margin $1/\|\boldsymbol{w}\|_2$):

$$\begin{cases} \min_{\mathbf{w},b} \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w} \\ s.t.\ y_i(\mathbf{w}^{\mathbf{T}}\mathbf{x_i} + b) \geq 1, i = 1,\dots,n \end{cases}$$

only support vectors participate in defining the SVM hyperplane

$L(\mathbf{w},b,\alpha) = \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w} + \sum_i \alpha_i\left(1 - y_i(\mathbf{w}^{\mathbf{T}}\mathbf{x_i} + b)\right)$

Take derivatives of $\mathbf{w}, b$, we have:

$\sum_i \alpha_i y_i = 0\ ,\ \mathbf{w} = \alpha_i y_i \mathbf{x}_i^T, b = y_i - \mathbf{w}^{\mathbf{T}}\mathbf{x_i}$

Plug in $L(\mathbf{w},b,\alpha)$ we got:

Separable SVM Dual

$$\begin{cases} \max_{\alpha\geq 0} \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i\alpha_j y_i y_j \mathbf{x}_i\mathbf{x}_j^T \\ s.t. \sum_i \alpha_i y_i = 0 \end{cases}$$

plug the result back in to get $\mathbf{w}, b$

Kernelized SVM Dual

$$\begin{cases} \max_{\alpha\geq 0} \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i\alpha_j y_i y_j k(\mathbf{x}_i,\mathbf{x}_j) \\ s.t. \sum_i \alpha_i y_i = 0 \end{cases}$$

prediction: $\sum_i \alpha_i y_i k(\mathbf{x_i},\mathbf{x}) + b$

where $b = y_i - \sum_j \alpha_j y_j k(\mathbf{x_j},\mathbf{x_i}), \forall i\ \alpha_i > 0$

NON-SEPARABLE SVM

Hinge primal

$$\begin{cases} \min_{\mathbf{w},b} \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w} + C\sum_i \xi_i \\ s.t. y_i(\mathbf{w}\mathbf{x_i} + b) \geq 1 - \xi_i, i = 1,\dots,n \end{cases}$$

ξ_i - $\max(0, 1 - y_i(\mathbf{x_i}\mathbf{w} + b))$

general case: L_p regularized, L_q loss

Hinge dual

$$\begin{cases} \max_{\alpha\geq 0} \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i\alpha_j y_i y_j \mathbf{x}_i\mathbf{x}_j^T \\ s.t. \sum_i \alpha_i y_i = 0\ ,\ \alpha_i \leq C \end{cases}$$

Hinge Kernelized Dual

$$\begin{cases} \max_{\alpha\geq 0} \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i\alpha_j y_i y_j k(\mathbf{x}_i,\mathbf{x}_j) \\ s.t. \sum_i \alpha_i y_i = 0\ ,\ \alpha_i \leq C \end{cases}$$

prediction: $\sum_i \alpha_i y_i k(\mathbf{x_i},\mathbf{x}) + b$

where $b = y_i - \sum_j \alpha_j y_j k(\mathbf{x_j},\mathbf{x_i}), \forall i, C > \alpha_i > 0\ (points\ on\ the\ margin)$

• In real problem, you can run the slack SVM directly, but you have to choose C.

SUPPORT VECTORS

In separable case, support vectors are always on the margin; in non-separable case, support vectors can be inside the margin, besides points on the margin ($0 < \alpha_i < C$) any point on the wrong side of ITS margin is a support vector ($\alpha_i = C$)!

For the same data, higher dimension means more points become support vectors

PROPERTIES

Convex, sparse, dual, kernel trick -> faster

Assume less about data -> accurate

Parametric, not probabilistic

PERCEPTRON

ONLINE LINEAR REGRESSION

Objective: $\min \sum_i (y_i - w^T\mathrm{x})^2$

LMS (least mean square) algorithm

$w_{i+1} = w_i - \left(\frac{\eta}{2}\right)\frac{dErr_i}{dw_i} = w_i + \eta r_i \mathrm{x_i}$

- converges for $0 < \eta < \lambda_{max}$, λ_{max} is the largest eigenvalue for X^TX

- converge rate is inversely proportional to $\lambda_{max}/\lambda_{min}$

Perceptron have higher bias compared to SVM; Perceptron may not converge to the same solution as SVM

Perceptron learning algorithm

Online SVM; hinge loss

> **Input:** a list of training examples T $< \boldsymbol{x_0}, \boldsymbol{y_0} >, \dots, < \boldsymbol{x_n}, \boldsymbol{y_n} >, \boldsymbol{y_i} = \pm\mathbf{1}$
> **Output:** classifying hyperplane $\boldsymbol{w}$
>
> Randomly initialize $\boldsymbol{w}$
> While $\boldsymbol{w}$ makes error on dataset:
> For $< \boldsymbol{x_i}, \boldsymbol{y_i} > \in \boldsymbol{T}$
> $\boldsymbol{w_{i+1}} = \boldsymbol{w_i} + \boldsymbol{\eta r_i x_i}$, where $\boldsymbol{r_i} = \boldsymbol{y_i} - \boldsymbol{sign}(\boldsymbol{w}\cdot\boldsymbol{x_i}), \boldsymbol{\eta} = \mathbf{1/2}$

Will converge if linear separable, number of mistakes $M < R^2/\gamma$ where $R = \max|x_i|_2, \gamma > y_i w^{*T} x_i$

Voted perceptron

Learn: keep track of all intermediate models; Classify: let the models vote and take the majority
More accurate;

Average perceptron

Final model = average of all intermediate models

Passive aggressive perceptron

Objective: minimize the hinge loss at each observation, $L(w_i; x_i, y_i) = 0$ if $y_i w_i^T x_i > 1$, else $L(w_i; x_i, y_i) = 1 - y_i w_i^T x_i$, $w_{i+1} = w_i + \eta y_i x_i$, where $\eta = L(w_i; x_i, y_i)/\|x_i\|^2$ (make the current data sample fall on the new border)

Margin-infused relaxed algorithm

(MIRA): for multiclass, each class has a prototype vector
Learn: passive aggressive
Classify: choosing the class whose prototype has the greatest dot product the instance

PCA

n m-dimensional example $\mathrm{x}^1, \dots, x^n$
k orthonormal basis $\mathrm{u}_1, \dots, \mathrm{u}_\mathrm{k}$ - loading
$z^i = ((\mathrm{x}^\mathrm{i} - \bar{\mathrm{x}})^\mathrm{T}\mathrm{u}_1, \dots, (\mathrm{x}^\mathrm{i} - \bar{\mathrm{x}})^\mathrm{T}\mathrm{u}_\mathrm{k})$ - principle components

$\hat{\mathrm{x}}_i = \bar{\mathrm{x}} + \sum_{j=1}^{k} z_j^i \mathrm{u}_\mathrm{j}$

DISTORTION AND VARIANCE

Distortion: $\sum_i \left\|\mathrm{x}^i - \hat{\mathrm{x}}_i\right\|_2^2 = \sum_i \sum_j (x_j^i - \hat{x}_j^i) = \sum_i \sum_{j=k+1}^{m} \left(z_j^i\right)^2 = n \sum_{j=k+1}^{m} \mathrm{u}_\mathrm{j}^\mathrm{T} \Sigma \mathrm{u}_\mathrm{j}$

Σ is the covariance matrix of examples.
Minimize distortion – u_j is the eigenvector of Σ, then distortion = $n \sum_{j=k+1}^{m} \lambda_j$, λ_j is the corresponding eigenvalue of u_j

Variance: $\sum_\mathrm{i} \sum_j \left(u_j^T \mathrm{x}^\mathrm{i} - u_j^T \bar{\mathrm{x}}\right)^2 = n \sum_{j=1}^{k} \mathrm{u}_\mathrm{j}^\mathrm{T} \Sigma \mathrm{u}_\mathrm{j} = n \sum_{j=1}^{k} \lambda_j$

Variance + Distortion=$n \cdot trace(\Sigma)$

PCA ALGORITHM

PCA

- Given $\boldsymbol{D} = \mathbf{x^1}, \dots, \mathbf{x^n}$
- Compute $\bar{\mathbf{x}}, \boldsymbol{\Sigma} = \sum_\mathbf{i} (\mathbf{x^i} - \bar{\mathbf{x}})(\mathbf{x^i} - \bar{\mathbf{x}})^\mathbf{T}/\boldsymbol{n}$
- Find k eigenvectors of $\boldsymbol{\Sigma}$ with largest eigenvalues $\boldsymbol{u_1}, \dots, \boldsymbol{u_k}$ (loadings)
- Get principle components:

$\boldsymbol{z^i} = ((\boldsymbol{x^i} - \bar{\boldsymbol{x}})^\boldsymbol{T}\boldsymbol{u_1}, \dots, (\boldsymbol{x^i} - \bar{\boldsymbol{x}})^\boldsymbol{T}\boldsymbol{u_k})$

(PCA via SVD: $X_{centered} = USV^T$, k rows of V^T with largest eigenvalues = loadings)
PCA can also use kernel, if the high covariance manifold is not linear, kernels may let you still find it

PCR (PCA+regression) vs Ridge

Ridge shrink in the sing. value space, effectively shrinking the small singular values more than the big ones; PCR does a "feature selection" in the sing. value space, zeroing out all the small sing. values.
PCR is not scale invariant

PLS or Canonical Covariance Analysis

Maximize the covariance of two sets of data X and Y by the singular value of X^TY

CCA (Canonical Correlation Analysis)

Maximize the correlation of two sets of data X and Y; CCA is scale invariant because of whitening finding the left and right singular value of $(X^TX)^{\frac{1}{2}}X^TY(Y^TY)^{\frac{1}{2}}$

UNSUPERVISED LEARNING

K-MEANS

Data intensive; quantize or compress data
* k-means++ initialization

K-means objectives (minimize distortion)

$J(\mu, r) = \sum_{i=1}^{n} \sum_{k=1}^{K} r_{ik} \|\mu_k - \mathrm{x_i}\|^2$

K-means algorithm continue to minimize this distortion in every step and will converge to a local minimum

Kernelized K-means

GMM

$p(x) = \sum_k \pi_k N(\mu_k, \Sigma_k) = \sum_k p_\theta(z = k) p_\theta(x|z = k)$

Variance parameters

1. Full covariance: Σ_k is arbitrary for each class $O(m^2)$ | 2. Shared Full covariance: Σ_k is arbitrary but same for each class $O(m^2)$ | 3. Diagonal (Naive Bayes): Σ_k is a diagonal matrix $O(Km)$ | 4. Shared Diagonal: Σ_k is a diagonal matrix same for each class O(m) | 5. Spherical: Σ_k is $\sigma_k I$ | 6. Shared Spherical: Σ_k is σI

EM

Input: model $\boldsymbol{p(z, \mathbf{x}|\theta)}$
starting parameter $\boldsymbol{\theta_0}$
unlabeled data $\mathbf{x_1}, \dots, \mathbf{x_n}$

For *t = 1 to T*
• E-step:
$\boldsymbol{q^t(Z_i|\mathbf{x_i}) = \arg\max_q F(q, \theta^{t-1}) = p(Z_i|\mathbf{x_i}, \theta^{t-1})}$
• M-step:
$\boldsymbol{\theta^t = \arg\max_\theta F(q^t, \theta) = \arg\max_\theta \sum_i \sum_z q^t(Z_i = z|\mathbf{x_i}) \log p(z, \mathbf{x_i})}$
(Use Lagrange to solve M-step)

GENERATIVE MODELS

BAYES NET

Compact, sparse representation

D-separation*: Rule 1: x and y are d-connected if there is an unblocked path between them (no collider) |Rule 2: x and y are d-connected, conditioned on a set Z of nodes, if there is a collider-free path between x and y that traverses no member of Z. If no such path, x and y are d-separated by Z, (every path between x and y is "blocked" by Z)|Rule 3: If a collider is a member of the conditioning set Z, or has a descendant in Z, then it no longer blocks any path that traces this collider.*

Belief Network Construction Algorithm

Choose an ordering of variables $X_1, \dots, X_n$
For $i = 1 to n$
  Add X_i to network
  Select parents from $X_1, \dots, X_{i-1}$ such that $P(X_i|Parents(X_i)) = P(X_i|X_1, \dots, X_{i-1})$

(Parameter) learning

loss function $P(X|\theta) + const * n\_param$
Do stochastic gradient descent
When hidden variables exist: EM
Given a known structure $\Longrightarrow$ Local maximum likelihood model

Criteria for good Bayes network

MDL – fewer bits to build model, fewer bits to code data

HIDDEN MARKOV MODEL

Transfer matrix, emission matrix, start case

Joint probability

$P(X_{1,\dots,T} = x_{1,\dots,T}, O_{1,\dots,T} = o_{1,\dots,T}) =$

$\prod_{i=1}^{T} P(X_i = x_i|X_{i-1} = x_{i-1}) P(O_i = o_i|X_i = x_i)$

Conditional independency

$X_{i-1} \perp X_{i+1}|X_i$
$O_i \perp X_{i+1}|X_i$
$O_i \perp O_{i+1}|X_i \quad and \quad O_i \perp O_{i+1}|X_{i+1}$

Inference

Posterior decoding (marginal probability for each X) and Viterbi decoding (joint probability for all Xs) [HW8]

UNSUPERVISED DEEP NETWORK

Auto-encoder

RECURRENT NEURAL NETWORKS

Generalize HMMs: Hidden state dynamical models, but nonlinear
Needed if you have inputs of varying length e.g. speech,text, robot observation
Gated neural nets (like LSTM) are replacing HMM in speech recognition

APPENDIX

VECTOR, MATRICES, EIGENVECTORS

Matrix: orthogonal ($AA^T = A^TA = I$);

Tensor

SVD

$rank(A)$=number of non-zero eigens

SVD (for any matrix), singular value and singular vector

$$A^TAv = \lambda v, AA^Tu = \lambda u$$

$$A_{n\times p} = U_{n\times n}\Lambda_{\mathrm{n\times p}}V^T_{p\times p}$$

Thin SVD: keep the top k singular values:

$$A_{n\times p} = U_{n\times k}\Lambda_{\mathrm{k\times k}}V^T_{k\times p}$$

The diagonal elements of Λ are the singular values A
The eigenvalues of A^TA and AA^T are the same, and they are the square of A's singular values
A's eigenvalue (if exist) = A's sing. value

Generalized Inverse

$$X^+ = (X^TX)^{-1}X^T$$
$$X^+ = (U\Lambda^{-1}V^T)^T = V\Lambda^{-1}U^T$$

in case that Λ is not invertible, we use thin SVD: $X^+ = V_k\Lambda_k^{-1}U_k^T$