@ operator needs py3.5 [what's new in py3.5?]

140

HW Assignment 1 (Due by 10:30am on Sep 26)

Bhishan Poudel

1 Theory (40 points)

1. [**Polynomial Curve Fitting**, 20 points]
Consider the problem of fitting a dataset of N points with a polynomial of degree M, by minimizing the sum-of-squares error:

$$J(\mathbf{w}) = \frac{1}{2N}\sum_{n=1}^{N}(h_{\mathbf{w}}(\mathbf{x}_n) - t_n)^2 \tag{1}$$

where $h_{\mathbf{w}}(\mathbf{x}) = \sum_{j=0}^{M} w_j x^j$. We have shown in class that the solution to minimizing $J(\mathbf{w})$ satisfies the following set of linear equations:

$$\sum_{j=0}^{M} A_{ij} w_j = T_i \tag{2}$$

$$\text{where } A_{ij} = \sum_{n=1}^{N} x_n^{i+j} \text{ and } T_i = \sum_{n=1}^{N} x_n^i t_n \tag{3}$$

Derive the solution for the regularized version of polynomial curve fitting, which minimizes the objective function below:

$$J(\mathbf{w}) = \frac{1}{2N}\sum_{n=1}^{N}(h_{\mathbf{w}}(\mathbf{x}_n) - t_n)^2 + \frac{\lambda}{2}||\mathbf{w}||^2 \tag{4}$$

2. [**Probability Theory**, 20 points] *Exercise 1.3, page 58 in PRML.*
Suppose that we have three coloured boxes r (red), b (blue), and g (green). Box r contains 3 apples, 4 oranges, and 3 limes, box b contains 1 apple, 1 orange, and 0 limes, and box g contains 3 apples, 3 oranges, and 4 limes. If a box is chosen at random with probabilities $p(r) = 0.2$, $p(b) = 0.2$, $p(g) = 0.6$, and a piece of fruit is removed from the box (with equal probability of selecting any of the items in the box), then what is the probability of selecting an apple? If we observe that the selected fruit is in fact an orange, what is the probability that it came from the green box?

3. [**Ridge Regression (*)**, 20 points]
Consider the regularized linear regression objective shown below:

$$J(\mathbf{w}) = \frac{1}{2N}\sum_{n=1}^{N}(h(x_n, \mathbf{w}) - t_n)^2 + \frac{\lambda}{2}||\mathbf{w}||^2$$

   (a) Minimizing the L2 norm of $\mathbf{w}$ drives all parameters, including w_0, towards 0. Are there situations in which we do not want to constrain w_0 to be small? If yes, give an example, if not show why it is useful to constrain all the weights to be small, including w_0 .

   (b) We have seen in class how to compute the weights $\mathbf{w}$ that minimize $J(\mathbf{w})$. Assume now that we replace $||\mathbf{w}||$ in $J(\mathbf{w})$ with $||\mathbf{w}_{[1:]}||$, where $\mathbf{w}_{[1:]} = [w_1, w_2, ..., w_M]$. Derive the solution for $\mathbf{w}$ that minimizes this new objective function.

2 Implementation (80 points)

In this exercise, you are asked to run an experimental evaluation of linear regression on the Athens houses dataset, and on an artificial dataset with and without L2 regularization. The input data is available at `http://ace.cs.ohio.edu/~razvan/courses/ml4900/hw01.zip`. Make sure that you organize your code in folders as shown in the table below. Write code only in the Python files indicated in bold.

$w = (X^T X)^{-1} X^T t$ = two w (row 2d array)

$RMSE = \sqrt{\frac{\sum_{n=1}^{N} (h-t)^2}{N}}$

$= np.sqrt(np.sum(h-t)^2) / N$

$= np.sqrt((h-t)^2.mean())$

```
ml4900/
  hw01/
    code/
      univariate.py
      multivariate.py
      polyfit.py
      train_test_line.png
    data/
      univariate/
        train.txt, test.txt
      multivariate/
        train.txt, test.txt
      polyfit/
        train.txt, test.txt, devel.txt
```

1. [**Univariate Regression**, 20 points]
   Train a univariate linear regression model to predict house prices as a function of their floor size, based on the solution to the system with 2 linear equations discussed in class. Use the dataset from the folder `hw01/data/univariate`. Python3 skeleton code is provided in `univariate.py`. After training print the parameters and report the RMSE and the objective function values on the training and test data. Plot the training using the default blue circles and test examples using lime green triangles. On the same graph also plot the linear approximation.

2. [**Multivariate Regression**, 20 points]
   Train a univariate linear regression model to predict house prices as a function of their floor size, number of bedrooms, and year. Use the normal equations discussed in class, and evaluate on the dataset from the folder `hw01/data/multivariate`. Python3 skeleton code is provided in `multivariate.py`. After training print the parameters and report the RMSE and the objective function values on the training and test data. Compare the test RMSE with the one from the univariate case above.

3. [**Polynomial Curve Fitting**, 40 points]
   In this exercise, you are asked to run an experimental evaluation of a linear regression model, with and without regularization. Use the normal equations discussed in class, and evaluate on the dataset from the folder `hw01/data/polyfit`.

   (a) Select 30 values for $x \in [0, 1]$ uniformly spaced, and generate corresponding t values according to $t(x) = sin(2\pi x) + x(x+1)/4 + \epsilon$, where $\epsilon = N(0, 0.005)$ is a

zero mean Gaussian with variance 0.005. Save and plot all the values. Done in `dataset.txt`.

(b) Split the 30 samples (x_n, t_n) in three sets: 10 samples for training, 10 samples for validation, and 10 samples for testing. Save and plot the 3 datasets separately. Done in `train.txt, test.txt, devel.txt`.

(c) Consider a linear regression model with polynomial basis functions, trained with the objective shown below:

$$J(\mathbf{w}) = \underbrace{\frac{1}{2N}\sum_{n=1}^{N}(h(x_n, \mathbf{w}) - t_n)^2}_{\text{half MSE}} + \underbrace{\frac{\lambda}{2}||\mathbf{w}||^2}_{\text{L2 Regularizer}}$$

$\sum_{j=1}^{M} w_j^2$ generally we don't regularize w_0.

Show the closed form solution (vectorized) for the weights $\mathbf{w}$ that minimize $J(\mathbf{w})$.

(d) Train and evaluate the linear regression model in the following scenarios:

1. Without regularization: Use the training data to infer the parameters $\mathbf{w}$ for all values of $M \in [0, 9]$. For each order M, compute the RMSE separately for the training and test data, and plot all the values on the same graph, as shown in class.
2. With regularization: Fixing $M = 9$, use the training data to infer the parameters $\mathbf{w}$, one parameter vector for each value of $\ln \lambda \in [-50, 0]$ in steps of 5. For each parameter vector (lambda value), compute the RMSE separately for the training and validation data, and plot all the values on the same graph, as shown in class. Select the regularization parameter that leads to the parameter vector that obtains the lowest RMSE on the validation data, and use it to evaluate the model on the test data. Report and compare the test RMSE with the one obtained without regularization.

3 Submission

Turn in a hard copy of your homework report at the beginning of class on the due date. Electronically submit on Blackboard a hw01.zip file that contains the hw01 folder in which your code is in the 3 required files.

On a Linux system, creating the archive can be done using the command:

```
> zip -r hw01.zip hw01.
```

Please observe the following when handing in homework:

1. Structure, indent, and format your code well.
2. Use adequate comments, both block and in-line to document your code.
3. On the theory assignment, **clear and complete explanations and proofs of your results are as important as getting the right answer.**
4. Make sure your code runs correctly when used in the directory structure shown above.

HW 1

Due: Sep 19 TU

Bhishan Poudel

ON 1.1 polynomial curve Fitting

cost function without the regularization is

$$J(w) = \frac{1}{2N} \sum_{n=1}^{N} \left(h_w(x_n) - t_n\right)^2 \quad \text{---(1)}$$

where, predicted values

$$h_w(x) = \underbrace{w_0 x_0}_{=w_0} + w_1 x_1 + w_2 x_2 + \cdots + w_M x_M$$

$w_0, w_1, \ldots, w_M$ = model parameters

w_0 = bias term or intercept term

Given

$$h_w(x) = \sum_{j=0}^{M} w_j x^j$$

minimizing the cost function w/o regularization,

$$\frac{\partial J}{\partial w} = 0 \quad \text{gives}$$

$$\sum_{j=0}^{M} A_{ij} w_j = T_i \quad \text{---(2)}$$

where

$$A_{ij} = \sum_{n=1}^{N} x_n^{i+j} \quad \text{---(3)}$$

$$T_{i8} = \sum_{n=1}^{N} x_n^{i} t_n$$

problem: minimize the function, λ is shrinkage

penalized residual sum of squares (PRSS)

$$J(w) = \underbrace{\frac{1}{2N} \sum_{n=1}^{N} \underbrace{\left(h_w(x_n) - t_n\right)^2}_{(h_n - t_n)^2}}_{\frac{1}{2} \cdot SSE} + \underbrace{\frac{\lambda}{2} \|w\|^2}_{\text{L2 regularizer or shrinkage penalty}}$$

solution: the hypothesis is

$$h_w(x_n) = \sum_{j=0}^{M} w_j x_n^j$$

$n = 0, \dots, N \rightarrow N$ samples $x_1, x_2, \dots, x_N$

$j = 0, \dots, M \rightarrow$ M features $w_0, w_1, \dots w_M$

w_0 bias, w_1 slope, w_2 age

t = target (price) (t is y)

we can write cost function as,

$$J(w) = \frac{1}{2N} \sum_{n=1}^{N} \left(\sum_{j=0}^{M} w_j x_n^j - t_n \right)^2 + \frac{\lambda}{2} \sum_{j=0}^{M} w_j^2$$

we want to minimize the cost function $J(w)$ w.r.t. weights w_i,

$$\nabla_{w_i} J(w) = 0$$

$$\frac{\partial}{\partial w_i} J(w_i) = 0$$

$$0 = \frac{1}{2N} \cdot 2 \sum_{n=1}^{N} \left(\sum_{j=0}^{M} w_j x_n^j - t_n \right) \cdot x_n^i + \frac{\lambda}{2} \cdot 2 w_i \cdot N$$

$$0 = \sum_{n=1}^{N} \sum_{j=0}^{M} w_j x_n^{i+j} - \underbrace{\sum_{n=1}^{N} t_n x_n^i}_{} + \lambda N w_i$$

$$\text{or.} \quad \sum_{j=0}^{M} \left(\sum_{n=1}^{N} x_n^{i+j} \right) w_i + \lambda N w_i = \sum_{n=1}^{N} t_n x_n^i$$

$$\Rightarrow \boxed{\sum_{j=0}^{M} A_{ij} w_j + \lambda N w_i = T_i} \quad \# \text{ Answer}$$

where,

$$A_{ij} = \sum_{n=1}^{N} x_n^{i+j}$$

$$T_i = \sum_{n=1}^{N} t_n x_n^i$$

Q.N1.2 Probability Theory

Ex 1.3, page 58 in PRML > By Bishop

Red R: 3a 4o 3l

blue B: 1a 1o 0l

Green G: 3a 3o 4l

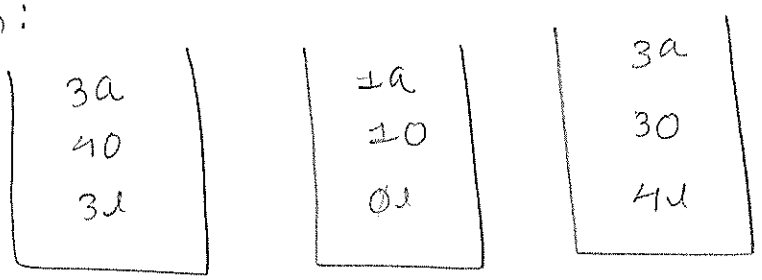
$p(R) = 0.2 \quad p(B) = 0.2 \quad p(G) = 0.6$

(a) If a piece of fruit is removed from the box, then what is the prob of selecting an apple?

(b) If we observe that the selected fruit is in fact an orange, what is the prob that it came from the green box?

Solution:

(a)



Red $N_R = 10$ | Blue $N_B = 2$ | Green $N_G = 10$

$N = N_R + N_B + N_G = 10 + 2 + 10 = 22$

$p(a|R) = \frac{3}{10} = 0.3 \quad p(a|B) = \frac{1}{2} = 0.5 \quad p(a|G) = \frac{3}{10} = 0.3$ } from figure

$p(o|R) = \frac{4}{10} = 0.4 \quad p(o|B) = \frac{1}{2} = 0.5 \quad p(o|G) = \frac{3}{10} = 0.3$

$p(R) = 0.2 \quad p(B) = 0.2 \quad p(G) = 0.6$ } given

Implementation

Qn 3.C

Given objective function for linear regression with polynomial basis functions is,

cost function

$$J(\vec{w}) = \frac{1}{2N} \sum_{n=1}^{N} [h(x_n, \vec{w}) - t_n]^2 + \frac{\lambda}{2} \|\vec{w}\|^2 \quad \text{---} \circledast$$

Now, we have to find the vectorized solution of w which minimizes cost function J.

For this,

we first write hypothesis in terms of design matrix X and weight vector w.

$$h = Xw \quad \text{---} ①$$

then, eqn ⊛ can be written as,

$$J = \frac{1}{2N} (Xw - t)^T (Xw - t) + \frac{\lambda}{2} w^T w$$

we minimize this equation w.r.t. w.

$$0 = \frac{\partial J}{\partial w}$$

$$= \frac{1}{2N} \cdot 2X^T (Xw - t) + \frac{\lambda}{2} \cdot 2w$$

Here we have used matrix derivative formula

$$\frac{d}{dx}(x^T x) = 2x$$

$$\frac{d}{dx}(Ax+b) = 2A^T(Ax+b)$$

A, x, b are matrices

Formula

then,

$$0 = x^T x w - x^T t + \lambda N w$$

$$x^T t = (x^T x + \lambda N) w$$

$$\boxed{w = (\lambda N I + x^T x)^{-1} (x^T t)}$$

Answer

Note that $(x^T x)^{-1} x^T$ is called Moore-Penrose pseudoinverse of any shape matrix X.

eg $x^T x = (4,50)(50,4) = (4,4) = I_{4,4}$

$x^T t = (4,50)(50,1) = (4,1)$

$w = (4,4)(4,1) = (4,1) = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_M \end{bmatrix}$

$w^T = [w_0 \; w_1 \; w_2 \; w_M]_{1 \times (M+1)}$

$$I_{4,4} = \begin{bmatrix} 1 & & & 0 \\ & 1 & & \\ & & 1 & \\ 0 & & & 1 \end{bmatrix}$$

Set $I[0][0] = 0$

we denote regularize bias term.

$$I = \begin{bmatrix} 0 & & & \\ & 1 & & 0 \\ & & 1 & \\ & 0 & & 1 \end{bmatrix}$$

if we regularize bias term, it will shift the mean of target vector. then adding const value to all the target DOES NOT hypothesize similarly shifted hypothesis.

I-1 (20)

I-2 (20)

I-3 (40)

Bhishan poudel

106/120

HW Assignment 3 (Due by 10:30am on Oct 12)

1 Theory (100 points)

1. [**Maximum Likelihood**, 20 points]
The Poisson distribution specifies the probability of observing k events in an interval, as follows:

$$P(k \text{ events in interval}) = e^{-\lambda}\frac{\lambda^k}{k!} \tag{1}$$

For example, k can be the number of meteors greater than 1 meter diameter that strike Earth in a year, or the number of patients arriving in an emergency room between 10 and 11 pm[1].

Suppose we observe N samples k_1, k_2, ..., k_N from this distribution (i.e. numbers of meteors that strike Earth over a period of N years). Derive the maximum likelihood estimate of the event rate λ.

2. [**Logistic Regression**, 20 points]
Consider a dataset that contains the 4 examples below i.e., the truth table of the logical XOR function. Prove that no logistic regression model can perfectly classify this dataset. Do not forget the bias feature $x_0 = 1$.

x_1	x_2	t
0	0	0
0	1	1
1	0	1
1	1	0

Hint: Prove that there cannot be a vector of parameters $\mathbf{w}$ such that $P(t = 1|\mathbf{x}, \mathbf{w}) \geq 0.5$ for all examples $\mathbf{x}$ that are positive, and $P(t = 1|\mathbf{x}, \mathbf{w}) < 0.5$ for all examples $\mathbf{x}$ that are negative.

3. [**Logistic Regression**, 20 points]
Prove that the gradient (with respect to $\mathbf{w}$) of the negative log-likelihood error function for logistic regression corresponds to the formula shown in lecture 4:

$$\nabla_{\mathbf{w}} E(\mathbf{w}) = \sum_{n=1}^{N}(h_n - t_n)\mathbf{x}_n \tag{2}$$

4. [**Logistic Regression**, 20 points]
In `scikit`, the objective function for logistic regression expresses the trade-off between training error and model complexity through a parameter C that is multiplied with the error term, as shown below. See the `scikit` documentation at `http://scikit-learn.org/stable/modules/linear_model.html#logistic-regression`.

$$E(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C * \sum_{n=1}^{N}\ln(e^{-t_n(\mathbf{w}^T\mathbf{x}_n)} + 1) \tag{3}$$

[1] https://en.wikipedia.org/wiki/Poisson_distribution

- Show that the sum in the second term is equal with the negative log-likelihood, where $t_n = +1$ stands for positive labels and $t_n = -1$ stands for negative labels.
- Compute the C parameter such that the objective is equivalent with the standard formulation shown on the slides in which the regularization parameter λ is multiplied with the L2 norm term.

5. [**Softmax Regression**, 20 points]
   Show that Logistic Regression is a special case of Softmax Regression. That is to say, if $\mathbf{w_1}$ and $\mathbf{w_2}$ are the parameter vectors of a Softmax Regression model for the case of two classes, then there exists a parameter vector $\mathbf{w}$ for Logistic Regression that results in the same classification as the Softmax Regression model.

6. [**Softmax Regression (*)**, 20 points]
   Prove that the gradient (with respect to $\mathbf{w}_k$) of the negative log-likelihood error function for regularized softmax regression corresponds to the formula shown in lecture 4, for any class $k \in [1..K]$:

$$\nabla_{\mathbf{w}_k} E(\mathbf{w}) = -\frac{1}{N} \sum_{n=1}^{N} (\delta_k(t_n) - p(C_k|\mathbf{x}_n))\mathbf{x}_n + \alpha \mathbf{w}_k \tag{4}$$

2 Submission

Turn in a hard copy of your homework report at the beginning of class on the due date. On this theory assignment, **clear and complete explanations and proofs of your results are as important as getting the right answer**.

HW03

Bhishan Poudel

Q#1

maximum Likelihood

Derive MLE of event rate parameter λ)

Let $x_1, x_2, \ldots x_n$ be iid poisson random variables with prob mass function,

pmf of poisson dist^n

$$p(x_i, \lambda) = p(x_i) = \frac{e^{-\lambda} \lambda^{x_i}}{x_i!}$$

Now, the joint prob of all variables x_i, called likelihood function, is given by

Likelihood

$$L(\lambda) = \prod_{i=1}^{n} p(x_i, \lambda)$$

$$= \prod_{i=1}^{n} \frac{e^{-\lambda} \lambda^{x_i}}{x_i!}$$

The log likelihood of pmf is,

Log Likelihood

$$\ln L(\lambda) = \ln \prod_{i=1}^{n} \frac{e^{-\lambda} \lambda^{x_i}}{x_i!}$$

$$= \sum_{i}^{} \ln \frac{e^{-\lambda} \lambda^{x_i}}{x_i!}$$

$$= \sum_{i}^{} [-\lambda + x_i \ln \lambda - \ln x_i!]$$

$$\ln L(\lambda) = -n\lambda + \ln \lambda \sum_{i}^{} x_i - \sum_{i}^{} \ln x_i!$$

To get the maximum likelihood estimate of the parameter λ, we maximize the Log likelihood ($\ln L(\lambda)$) w.r.t. λ. [maximizing $(+\ln L)$ is same as minizing $(-\ln L)$]

$$0 = \frac{\partial}{\partial \lambda} \ln L(\lambda)$$

$$= \frac{\partial}{\partial \lambda} \left[-n\lambda + \ln\lambda \sum_i x_i - \sum_i \ln x_i! \right]$$

$$0 = -n + \frac{\sum_i x_i}{\lambda} \quad -(1)$$

$$n = \frac{\sum_i x_i}{\lambda}$$

(2)

MLE of λ

$$\hat{\lambda} = \frac{\sum_{i=1}^{n} x_i}{n}$$

$n \rightarrow$ number of samples

Ans

Here, the maximum likelihood estimate of the poission distribution parameter λ is just the mean (or expectation) of the distribution.

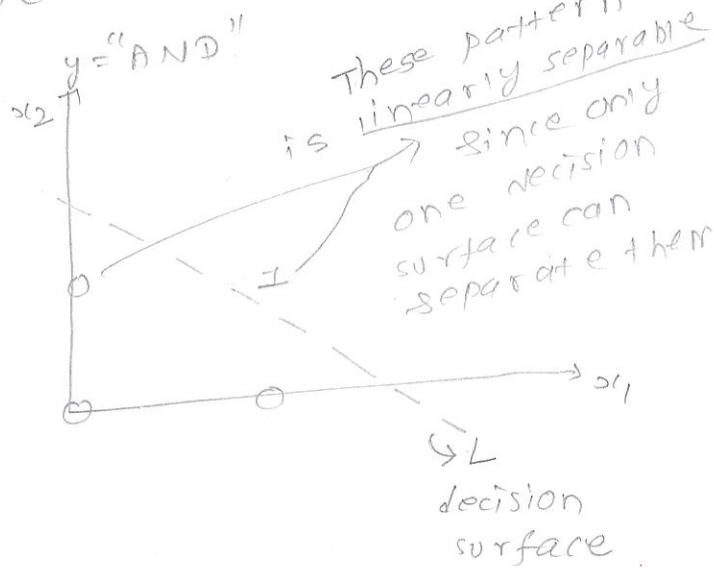
ON 2

ON 2

XOR problem in Logistic Regression

To describe XOR problem in LR, I shall start with "AND" logic.

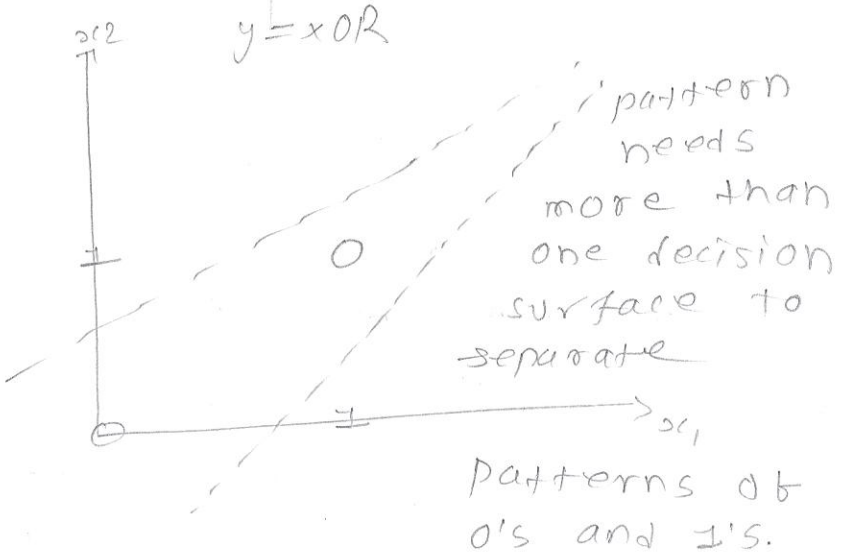
x_1	x_2	$y = x_1 \& x_2$
0	0	0
0	1	0
1	0	0
1	1	1



Now, look at XOR,

Truth table

x_1	x_2	$y = XOR$
0	0	0
0	1	1
1	0	1
1	1	0

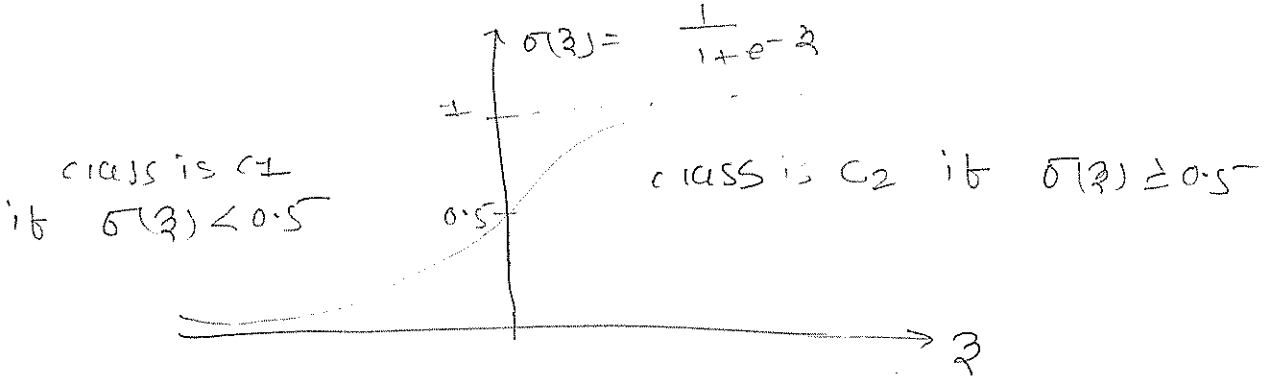


∴ XOR logic is not linearly separable.

P T O.

Now, we shall show that Logistic Regression is linear classifier :

The classifier used in LR is sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}}$



In Logistic Regression,

pattern is 1 if $\frac{1}{1+e^{-w^T x}} \geq \frac{1}{2}$

0 if $\frac{1}{1+e^{-w^T x}} < \frac{1}{2}$

let's set the ~~classifier~~ equal to separator value:

$$\frac{1}{1+e^{-w^T x}} = \frac{1}{2}$$

$$2 = 1+e^{-w^T x}$$

$$\frac{1}{1} = e^{-w^T x} = \frac{1}{e^{w^T x}}$$

$$e^{w^T x} = 1$$

$$w^T x = \ln 1 = 0$$

$$\Rightarrow \boxed{\sum_i w_i x_i = 0}$$

this means LR is linear classifier.

12

conclusion: 1) LR is linear classifier

2) XOR problem is non-linear and linearly inseparable

Use:

$W^T x_n \geq 0$ if $x_n \in +1$

$W^T x_n < 0$ if $x_n \in -1$

then put 4 samples into the inequations.

∴ LR can NOT perfectly classify dataset that follows XOR logic.

ans

Better method

x_0	x_1	x_2	t
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

for linear separability, we want

$w_0 + \sum_i w_i x_i < 0$ if $t_i == 0$

$w_0 + \sum_i w_i x_i \geq 0$ if $t_i == 1$

Now,

$w_0 < 0$ —①

$w_0 + w_2 \geq 0$ —②

$w_0 + w_1 \geq 0$ —③

$w_0 + w_1 + w_2 < 0$ —④

② + ③ $\Rightarrow w_0 + (w_0 + w_1 + w_2) \geq 0$

but $w_0 < 0$

and $w_0 + w_1 + w_2 < 0$

∴ it contradicts our assumptions of linear separability.

Q N3
Q N3

Gradient of Logistic Regression cost fn

(find $\nabla_w E$)

for linear regression, hypothesis $h = w^T x$

for logistic regression, hypothesis $h = \sigma = \frac{1}{1+e^{-w^T x}}$

$$h = \sigma = \frac{1}{1+e^{-wx}} \quad (1)$$

(I write $w^T x$ as wx since they are just matrix dot product of two matrices)

Aside:
Binomial distn

$p(c; p, n) = {}^nC_c \, p^x (1-p)^{n-x}$

for $x \in (0 \cdots n)$

${}^nC_c = \frac{n!}{x!(n-x)!}$

Binomial (Bernoulli) Likelihood function

pmf for logistic regression: $p(t|w) = h_n^{t_n} \cdot (1-h_n)^{1-t_n}$

likelihood function $L(w) = \prod_{n=1}^{N} h_n^{t_n} (1-h_n)^{1-t_n}$

(Bishop eq 4.89 p. 206/223)

-ve log likelihood, E or $J = -\ln L(w)$

$$E = -\ln \prod_{n=1}^{N} h_n^{t_n} (1-h_n)^{1-t_n}$$

$$E = \sum_n E_n = \sum_{n=1}^{N} (-t_n \ln h_n - (1-h_n) \ln (1-h_n))$$

Loss, $E(w) = -\frac{1}{N} \sum_n [t_n \ln h_n + (1-t_n) \ln (1-h_n)]$

for any nth sample the cost can be written as,

LOSS

$$E = -t \ln h - (1-h) \ln (1-h) \quad (2)$$

-ve log likelihood error fn for LR

here $h = \sigma = \sigma(w^T x) = \sigma(wx)$

Before proceeding further, I would derive derivative of sigmoid function.

$$\sigma(z) = \frac{1}{1+e^{-z}} \Rightarrow \frac{d\sigma}{dz} = \frac{-1 \cdot e^{-z} \cdot (-1)}{(1+e^{-z})^2}$$

$$\frac{d\sigma}{dz} = \frac{e^{-z}}{(1+e^{-z})^2} = \frac{1}{1+e^{-z}} \cdot \frac{e^{-z}+1-1}{1+e^{-z}}$$

$$= \frac{1}{1+e^{-z}} \cdot \left(\frac{e^{-z}+1}{1+e^{-z}} - \frac{1}{1+e^{-z}} \right)$$

$$= \frac{1}{1+e^{-z}} \cdot \left(1 - \frac{1}{1+e^{-z}} \right)$$

$$\boxed{\frac{d\sigma(z)}{dz} = \sigma(1-\sigma)}$$

similarly $\boxed{\frac{d\sigma(az)}{dz} = a\sigma(1-\sigma)}$ (here $a \neq a(z)$)

∴ derivative of sigmoid function

$$\boxed{\frac{d\sigma(az)}{dz} = a\sigma(1-\sigma)}$$

Now, going back to the problem, let's calculate the gradient of loss function.

$$\frac{\partial E}{\partial w} = \frac{\partial}{\partial w}\left[-t \ln h - (1-t)\ln(1-h)\right]$$

$$= -\frac{t}{h}\frac{\partial h}{\partial w} - \frac{(1-t)}{1-h}\cdot\frac{\partial h}{\partial w}$$

Note: $h = \sigma(wx)$, σ and h are same thing here

$$= -\frac{t}{h}\frac{\partial \sigma(wx)}{\partial w} + \frac{1-t}{1-h}\frac{\partial \sigma(wx)}{\partial w}$$

Property of sigmoid function

Note: $\frac{\partial h}{\partial w} = \frac{\partial h(wx)}{\partial w} = xh(1-h)$

$$= -\frac{t}{h} \times h(1-h) + \frac{1-t}{(1-h)} \times h(1-h)$$

$$\because \frac{d\sigma(az)}{dz} = a\sigma(1-\sigma)$$

(W) ✓

$$= -tx - txh + xh - txh$$

$$= xh - tx$$

$$\boxed{\frac{\partial E}{\partial w} = (h-t)x}$$

(here this is for a single sample, but for total loss we add up all the losses)

$$\Rightarrow \boxed{\nabla_w E = \sum_{n=1}^{N}(h_n - t_n)x_n}$$

Q.E.D.

Q N 4

Logistic Regression in sklearn

part a The cost function for L2 penalized logistic regression is given by,

L2 Regularized cost fn for LR (in sklearn)

$$E = \underbrace{\frac{1}{2} w^T w}_{\text{L2 regularizer}} + C \underbrace{\sum_{n=1}^{N} \ln\left(1 + e^{-t_n w^T x_n}\right)}_{\text{cost for LR}} \quad \text{--- (1)}$$

where $t_n \in \{-1, 1\}$

we have to show that the summation term is equal to the -ve log likelihood,

-ve log likelihood

$$E_D = -\ln p = \sum_{n=1}^{N} \ln\left(1 + e^{-t_n w^T x_n}\right) \quad \text{--- (2)}$$

$t_n \in \{0, 1\}$

Now, the posterior probabilities for class 0 and 1 are,

posterior prob for LR

$$p(C_1|x) = \sigma(w^T x) = \frac{1}{1 + e^{w^T x}} \quad \text{--- (3)}$$

$$p(C_2|x) = 1 - \sigma(w^T x) = \frac{e^{-w^T x}}{1 + e^{-w^T x}}$$

The likelihood function is,

Likelihood for LR

$$p(t|w) = \prod_{n=1}^{N} h_n^{t_n} (1 - h_n)^{(1 - t_n)} \quad \text{--- (4)}$$

The -ve log likelihood is,

$$-\ln p(t|w) = -\ln \prod_{n=1}^{N} h_n^{t_n} \cdot (1-h_n)^{1-t_n} = -\sum_{n=1}^{N} \ln \left(h_n^{t_n} \cdot (1-h_n)^{1-t_n}\right)$$

$$= -\sum_{n=1}^{N} \left[\ln (h_n^{t_n}) + \ln (1-h_n)^{1-t_n}\right]$$

cost for LR

$$E_D = -\ln p = -\sum_{n=1}^{N} \left[\underbrace{t_n \ln h_n}_{\text{for class 0}} + \underbrace{(1-t_n) \ln (1-h_n)}_{\text{for class 1}}\right] \quad \text{--- (5)}$$

where $t_n \in \{0, 1\}$

$$h_n = \frac{1}{1+e^{-w^T x_n}}$$

$$1-h_n = 1 - \frac{1}{1+e^{-w^T x_n}}$$

$$= \frac{1+e^{-w^T x_n} - 1}{1+e^{-w^T x_n}}$$

$$= \frac{e^{-w^T x_n}}{1+e^{-w^T x_n}}$$

$$= \frac{1}{e^{w^T x_n} + 1}$$

$$1-h_n = \frac{1}{1+e^{w^T x_n}}$$

$$E_D = -\ln p$$

$$= -\sum_{n=1}^{N} \left[t_n \ln \left(\frac{1}{1+e^{-w^T x_n}} \right) + (1-t_n) \ln \left(\frac{1}{1+e^{w^T x_n}} \right) \right.$$

$t_n \in \{0, 1\}$

$$= -\sum_{n=1}^{N} \begin{cases} \ln \dfrac{1}{1+e^{-w^T x_n}} & \text{if } t_n = 1 \\ \ln \dfrac{1}{1+e^{w^T x_n}} & \text{if } t_n = 0 \end{cases}$$

$$= -\sum_{n=1}^{N} \ln \frac{1}{1+e^{-t_n w^T x_n}} \quad \text{if } t_n \in \{-1, 1\}$$

74

$$E_D = +\sum_{n=1}^{N} \ln \left(1+e^{-t_n w^T x_n}\right) \quad \forall (x_n, t_n) \in D.$$

$t_n \in \{-1, 1\}$

QN 4b Find 'C' parameter of LR cost fn in sklearn.

Now, the L2 regularized cost function for LR

$$E = E_W + E_D$$

$$= \frac{\lambda}{2} w^T w + (-\ln p)$$

when $t_n \in \{0,1\}$

$$E = \frac{\lambda}{2} w^T w - \sum_{n=1}^{N} \left[t_n \ln h_n + (1-t_n) \ln (1-h_n) \right] \quad \text{(0)}$$

$t_n \in \{0,1\}$

when $t_n \in \{-1,1\}$

Regularized cost

$$E = \frac{\lambda}{2} w^T w + \sum_{n=1}^{N} \ln \left(1 + e^{-t_n w^T x_n} \right) \quad \text{(1)}$$

$t_n \in \{-1,1\}$

when $t_n \in \{-1,1\}$ in sklearn

$$E(w) = \frac{1}{2} w^T w + C \sum_{n=1}^{N} \ln \left(1 + e^{-t_n (w^T x_n)} \right) \quad \text{(2)}$$

To get maximum Likelihood Estimate of parameter w,

$$\vec{w} = \underset{w}{\operatorname{argmin}} \left[\frac{\lambda}{2} w^T w + \sum_{n=1}^{N} \ln \left(1 + e^{-t_n w^T x_n} \right) \right]$$

$$\Rightarrow \vec{w} = \underset{w}{\operatorname{argmin}} \left[\frac{1}{2} w^T w + \frac{1}{\lambda} \sum_{n=1}^{N} \ln \left(1 + e^{-t_n w^T x_n} \right) \right] \quad \text{(general form)}$$

but, $\vec{w} = \underset{w}{\operatorname{argmin}} \left[\frac{1}{2} w^T w + C \sum_{n=1}^{N} \ln \left(1 + e^{-t_n w^T x_n} \right) \right]$ (scikit form)

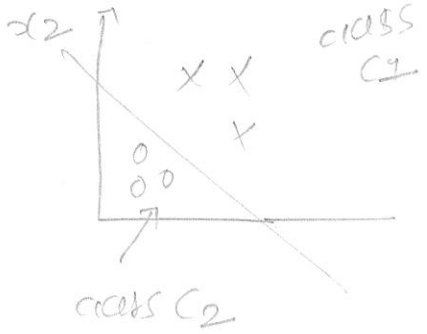
Extra note: to add bias term sklearn uses $w^T \to w^T - C$ in the data term E_D

(b) $\therefore$ $C = \frac{1}{\lambda}$ ✓

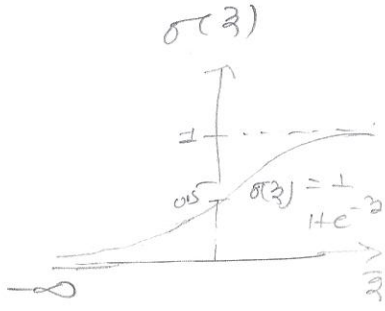
Ans

Softmax Regression as special case of Logistic Regression

FOR LR



(binary classification)



(sigmoid fn)

the posterior prob of class C1 is

posterior prob

$$p(C_1|x;w) = p(C_1|x) = \sigma(w^T x) = \frac{1}{1+e^{-w^T x}}$$

the posterior prob of class C2 is,

$$p(C_2|x) = 1 - \sigma(w^T x) = \frac{e^{-w^T x}}{1+e^{-w^T x}}$$

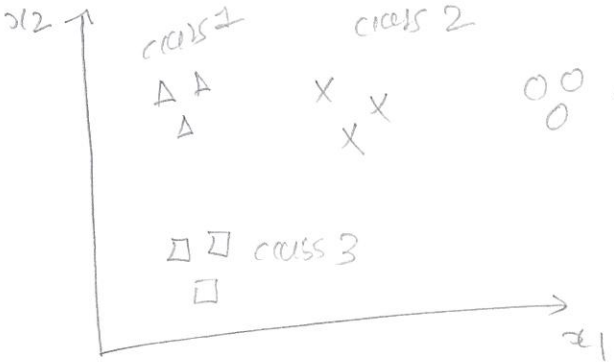
hypothesis the hypothesis for LR,

$$h(w) = \begin{bmatrix} \frac{1}{1+e^{-w^T x}} \\ \frac{e^{-w^T x}}{1+e^{-w^T x}} \end{bmatrix}$$

→ for class 1

→ for class 2

Again, FOR SR (Softmax Regression or multiclass Logistic Regression)



total number of samples = N

input data X

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$ some values

target

$$\begin{bmatrix} \text{triangle } t_1 \\ \text{cross } t_2 \\ \text{rectangle} \\ \cdots \\ \text{circle } t_N \end{bmatrix}$$

In softmax Regression, the posterior probability of any class C_K is given by,

$$p(C_K|x, w) = \frac{e^{w_K^T x}}{\sum_{K=1}^{K} e^{w_K^T x}}$$

posterior prob.

[Bishop, 4.3.4 eq 4.104 p. 209/Pdf 227]

note $a_K = w_K^T x$ is called activation fn for the parameter vector w_K

hyp:

$$h = \frac{1}{\sum_K e^{w_K^T x}} \begin{bmatrix} e^{w_1^T x} \\ e^{w_2^T x} \\ \vdots \\ e^{w_K^T x} \end{bmatrix}$$

plus sign $+w_K^T x$

sigmoid is $-$
softmax is $+$

Show $h(w) = h(w-\psi)$

[SR has overparametrization property]

ψ is any fixed vector

$$h(w-\psi) = \frac{e^{(w_K-\psi)^T x}}{\sum_{K=1}^{K} e^{(w_K-\psi)^T x}}$$

$$= \frac{e^{w_K^T x} \cdot e^{-\psi^T x}}{\sum_K e^{w_K^T x} \cdot e^{-\psi^T x}}$$

$$= \frac{e^{w_K^T x} \cdot \cancel{e^{-\psi^T x}}}{\cancel{e^{-\psi^T x}} \sum_K e^{w_K^T x}}$$

does not depend on K

$$= \frac{e^{w_K^T x}}{\sum_K e^{w_K^T x}}$$

$$\boxed{h(w-\psi) = h(w)}$$

∴ If we change any parameter vector $w_K \to w_K - \psi$ we get <u>same</u> hypothesis for softmax Regression.

for two-class SR,

$$h = \frac{1}{\sum_K e^{w_K^T x}} \begin{bmatrix} e^{w_1^T x} \\ e^{w_2^T x} \end{bmatrix}$$

$$= \frac{1}{e^{w_1^T x} + e^{w_2^T x}} \begin{bmatrix} e^{w_1^T x} \\ e^{w_2^T x} \end{bmatrix}$$

use overparametrization property of SR

$h(w-\psi) = h(w)$ where $\psi = w_2$

$$= \frac{1}{e^{(w_1 - w_2)^T x} + 1} \begin{bmatrix} e^{(w_1 - w_2)^T x} \\ 1 \end{bmatrix}$$

write $\vec{w_2} - \vec{w_1} = \vec{w}$

$$= \frac{1}{1 + e^{-w^T x}} \begin{bmatrix} e^{-w^T x} \\ 1 \end{bmatrix}$$

$$(h)_{SR} = \begin{bmatrix} \frac{e^{-w^T x}}{1 + e^{-w^T x}} \\ \frac{1}{1 + e^{-w^T x}} \end{bmatrix} = (h)_{LR} \quad \#$$

proved

↑ this is same as h for LR

Direct method to prove softmax (K=2) = LR

the hypothesis for softmax Regression is

$$h(w,x) = \frac{1}{\sum_{k=1}^{K} e^{w_k^T x}} \begin{bmatrix} e^{w_1^T x} \\ e^{w_2^T x} \\ \vdots \\ e^{w_K^T x} \end{bmatrix}$$

when $K=2$,

$$h = \frac{1}{e^{w_1^T x} + e^{w_2^T x}} \begin{bmatrix} e^{w_1^T x} \\ e^{w_2^T x} \end{bmatrix}$$

$$(h)_{\text{softmax}} = \begin{bmatrix} \frac{e^{w_1^T x}}{e^{w_1^T x} + e^{w_2^T x}} \times \frac{e^{-w_2^T x}}{e^{-w_2^T x}} \\ \frac{e^{w_2^T x}}{e^{w^T x} + e^{w_2^T x}} \times \frac{e^{-w_2^T x}}{e^{-w_2^T x}} \end{bmatrix}$$

(2a) ✓

$$= \begin{bmatrix} \frac{e^{-(w_2^T - w_1^T)x}}{1 + e^{-(w_2^T - w_1^T)x}} \\ \frac{1}{1 + e^{-(w_2^T - w_1^T)x}} \end{bmatrix} = \begin{bmatrix} \frac{e^{-w^T x}}{1 + e^{-w^T x}} \\ \frac{1}{1 + e^{-w^T x}} \end{bmatrix}$$

where $w^T = w_2^T - w_1^T$

or,

$$(h)_{softmax} = \begin{bmatrix} \frac{e^{-w^T x}}{1+e^{-w^T x}} \\ \\ \frac{1}{1+e^{-w^T x}} \end{bmatrix}$$

this is $(h)_{Logistic\ Regression}$

$$\therefore \boxed{(h)_{SR} = (h)_{LR}}$$ proved!

ANG
ANG

derivative of cost fⁿ for softmax Regr

for softmax Regression,

prob of data x belonging to class C_k i.e.
posterior prob of class C_k given data x is,

posterior prob of class C_k

$$p(C_k|x,w) = \frac{e^{w_k^T x}}{\sum_{k=1}^{K} e^{w_k^T x}} \quad \text{---(1)}$$

Bishop 4.3.4
p. 209 / 227

rename class C_k by target t_n, then the probability that the nth example x_n belong to target t_n is given by,

$$p(t_n|x,w) = \frac{e^{w_{t_n}^T x_n}}{\sum_{n=1}^{N} e^{w_{t_n}^T x_n}} \quad \text{---(1)}$$

then, likelihood

$$l(w) = \prod_{n=1}^{N} p(t_n|x_n,w) \quad (2)$$

the -ve log likelihood is,

$$-\ln l(w) = -\ln \prod_{n=1}^{N} \left(\frac{e^{w_{t_n}^T x_n}}{\sum_{n=1}^{N} e^{w_{t_n}^T x_n}} \right)$$

cost for data

$$E_D = \frac{1}{N} \cdot (-\ln l(w)) = \frac{1}{N} \sum_{n=1}^{N} -\ln \left(\frac{e^{w_{t_n}^T x_n}}{\sum_{n=1}^{N} e^{w_{t_n}^T x_n}} \right) \quad \text{using } w_{t_n} \quad (3)$$

(weight vector for each N examples)

here, t_n is $t_1, t_2, t_3, \ldots, t_N$ (single value)
x_n is $x_1, x_2, x_3, \ldots, x_N$ (x_1 is row vector)
w is $w_1, w_2, \ldots \ldots, w_N$ ($w x_n$ should be single value)
(i.e. x_1 is row w_1 is column)

here, instead of $\vec{w} = \begin{bmatrix} w_1 \\ w_2 \\ w_N \end{bmatrix}$ we can group

the data so that there are only K weight vectors $w = [w_1 w_2, \cdots w_K]$, then,

using w_K

$$p(C_k|x) = \frac{e^{w_k^T x}}{\sum_{k=1}^{K} e^{w_k^T x}} \quad \text{--- (1)}$$

(Regrouping weight vectors for each K classes, we can rewrite rest of the eqs)

and, $$E_D(w) = -\frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} \{\delta_k(t_n)\} \ln \left(\frac{e^{w_k x_n}}{\sum_{k=1}^{K} e^{w_k^T x_n}} \right)$$

insert this!

using w_K (weight vector for each K classes)

$$= -\frac{1}{N} \sum_{n=1}^{N} \sum_{k} \delta_k(t_n) \left[w_k^T x_n - \ln \sum_{k=1}^{K} e^{w_k^T x_n} \right]$$

$$E_D = -\frac{1}{N} \sum_{n=1}^{N} \left[\sum_{k=1}^{K} \delta_k(t_n) w_k^T x_n - \sum_{k=1}^{K} \delta_k(t_n) \ln \left(\sum_{k=1}^{K} e^{w_k^T x_n} \right) \right]$$

$$\frac{\partial}{\partial w_j} E_D = -\frac{1}{N} \sum_{n=1}^{N} \left[\frac{\partial}{\partial w_j} \sum_{k=1}^{K} \delta_k(t_n) w_k^T x_n - \frac{1}{\partial w_j} \sum_{k=1}^{K} \delta_k(t_n) \ln \left(\sum_{k=1}^{K} e^{w_k^T x_n} \right) \right]$$

here w_j operates only when $w_j = w_k$ and summation vanishes

$$\frac{\partial E_D}{\partial w_j} = -\frac{1}{N}\sum_{n=1}^{N}\left[\delta_j(t_n)\,x_n - \delta_j(t_n)\cdot\frac{1\cdot e^{w_j^T x_n}\cdot\delta_j(t_n)\,x_n}{\sum_{k=1}^{K} e^{w_j^T x_n}}\right]$$

$$\frac{\partial E_D}{\partial w_K} = -\frac{1}{N}\sum_{n=1}^{N}\left[\delta_{nk}(t_n)\cdot x_n - x_n\cdot\delta_{nk}(t_n)\frac{e^{w_{nk}^T x_n}}{\sum_{k=1}^{K} e^{w_k^T x_n}}\right]$$

we change dummy index from $j \to K$

$\delta_K(t_n)\cdot\delta_{nk}(t_n) = \delta_K(t_n)$

$p(C_K)$ is prob only when label is K, so basically $\delta_{nk}(t_n)$ is 1 for $p(C_K)$

$= p(C_K|x_n)$ (eq 1)

Look

(2)

$$= -\frac{1}{N}\sum_{n=1}^{N}\left[\delta_K(t_n)\,x_n - x_n\,p(C_K|x_n)\right]$$

gradient of ED

$$\nabla_w E_D = -\frac{1}{N}\sum_{n=1}^{N}\left(\delta_K(t_n) - p(C_K|x_n)\right)x_n \quad (A)$$

Again $\nabla_w E_w(w) = \frac{\partial}{\partial w}\frac{\alpha}{2}w_K^T w_K = \frac{\alpha}{2}\cdot 2w_K^T = \alpha w_K^T$

$$\nabla_w E_w = \alpha w_K^T \quad (B)$$

$\therefore$ Ans

$$\nabla_w E = \nabla_w E_D + \nabla_w E_w = -\frac{1}{N}\sum_{n=1}^{N}\left(\delta_{nk}(t_n) - p(C_K|x_n)\right)x_n + \alpha w_K^T$$

method 2 $E_D(w) = -\frac{1}{N} \sum_n y_n \left(\frac{e^{w_n^T x_n}}{\sum_n e^{w_n^T x_n}} \right)$ (grouping n-weights)

$$E_D(w) = -\frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} \delta_k(t_n) y_n \left(\frac{e^{w_k^T x_n}}{\sum_{k=1}^{K} e^{w_k^T x_n}} \right) \quad \text{(grouping k-weights)}$$

let, $$E_n = \sum_{k=1}^{K} \delta_k(t_n) y_n \left(\frac{e^{w_k^T x_n}}{\sum_k e^{w_k^T x_n}} \right)$$

$$E_n = \sum_k \delta_k(t_n) w_k^T x_n - \sum_k \delta_k(t_n) y_n \left(\sum_k e^{w_k^T x_n} \right)$$

$$\frac{\partial E_n}{\partial w_j} = \delta_j(t_n) x_n - \delta_j(t_n) \cdot \frac{1}{\sum_j e^{w_j^T x_n}} \cdot e^{w_j^T x_n} \cdot \delta_j(t_n) x_n$$

$$= \delta_j(t_n) x_n - \frac{e^{w_j^T x_n}}{\sum_j e^{w_j^T x_n}} x_n$$

$$\frac{\partial E_n}{\partial w_k} = \delta_k(t_n) x_n - \frac{e^{w_k^T x_n}}{\sum_k e^{w_k^T x_n}} x_n$$

$$\boxed{\frac{\partial E_n}{\partial w_k} = \left(\delta_k(t_n) - p(c_k|x_n) \right) x_n}$$

$$\therefore \nabla_{w_k} E_D(w) = -\frac{1}{N} \sum_n \left[\delta_k(t_n) - p(c_k|x_n) \right] x_n \quad \text{——— (a)}$$

also, $$\boxed{E_w(w) = \frac{\alpha}{2} w_k^T w_k}$$

$$\nabla_{w_k} E_w(w) = \frac{\partial}{\partial w_k} \frac{\alpha}{2} w_k^T w_k = \alpha w_k$$

$$\therefore \nabla E = -\frac{1}{N} \sum_{n=1}^{N} \left(\delta_k(t_n) - p(c_k|x_n) \right) x_n + \alpha w_k$$

$$\boxed{\nabla_w E = \alpha \vec{w}_k - \frac{1}{N} \sum_n \left(\delta_k(t_n) - p(c_k|x_n) \right) \vec{x_n}}$$ Ans

HW5

Bhishan poudel

142/150

HW Assignment 5 (Due by 10:30am on Nov 2)

1 Theory (110 points)

1. [**Properties of Linear Discriminants**, 20 points]
We have proven in class that the distance between origin and the decision hyperplane $h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 = 0$ is equal with $-w_0/\|\mathbf{w}\|$. Prove that the margin between a point $\mathbf{x}$ and the same decision hyperplane is equal with $h(\mathbf{x})/\|\mathbf{w}\|$.

2. [**Bonus**, 20 points]
Prove the two properties above for the general n-dimensional case.

3. [**Fisher Criterion and Least Squares**, 30 points]
Show that the Fisher criterion can be written in the vectorized form shown below:
$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} = \frac{\mathbf{w}^T\mathbf{S}_B\mathbf{w}}{\mathbf{w}^T\mathbf{S}_W\mathbf{w}}$$

4. [**Fisher Criterion (*)**, 20 points]
Reference the PRML Chapter 4 material available on Blackboard under Content.
Using the definitions of the between-class and within-class covariance matrices given by (4.27) and (4.28), respectively, together with (4.34) and (4.36) and the choice of target values described in Section 4.1.5, show that the expression (4.33) that minimizes the sum-of-squares error function can be written in the form (4.37).

5. [**Perceptrons**, 40 points]
Consider a training set that contains the following 8 examples:

$\mathbf{x}$	x_1	x_2	x_3	$t(x)$
$\mathbf{x}^{(1)}$	0	0	0	+1
$\mathbf{x}^{(2)}$	0	1	0	+1
$\mathbf{x}^{(3)}$	1.5	0	-1.5	+1
$\mathbf{x}^{(4)}$	1.5	1	-1.5	+1
$\mathbf{x}^{(5)}$	1.5	0	0	−1
$\mathbf{x}^{(6)}$	1.5	1	0	−1
$\mathbf{x}^{(7)}$	0	0	-1.5	−1
$\mathbf{x}^{(8)}$	0	1	-1.5	−1

   (a) Prove that the perceptron algorithm does not converge on this dataset. Do not forget to include the bias.

   (b) Consider a kernel perceptron that uses a polynomial kernel $k(\mathbf{x}, \mathbf{y}) = (1 + \mathbf{x}^T\mathbf{y})^d$. What is the smallest degree d for which the kernel perceptron would converge on this dataset?

6. [**Perceptrons**, 10 points]
A kernel perceptron for binary classification is run for a number of epochs E on a training dataset containing N examples, resulting in the dual parameters α_1, α_2, ..., α_N. What is the total number of mistakes that are made during training?

7. [**Matrix Computations**, 10 points]
Let $U \in R_{k \times m}$ and $X \in R_{n \times m}$. Let u_i and x_i be the i-th columns of U and X, respectively, for $1 \leq i \leq m$. Prove that $UX^T = \sum_{i=1}^{m} u_i x_i^T$.

2 Submission

Turn in a hard copy of your homework report at the beginning of class on the due date. On this theory assignment, **clear and complete explanations and proofs of your results are as important as getting the right answer**.

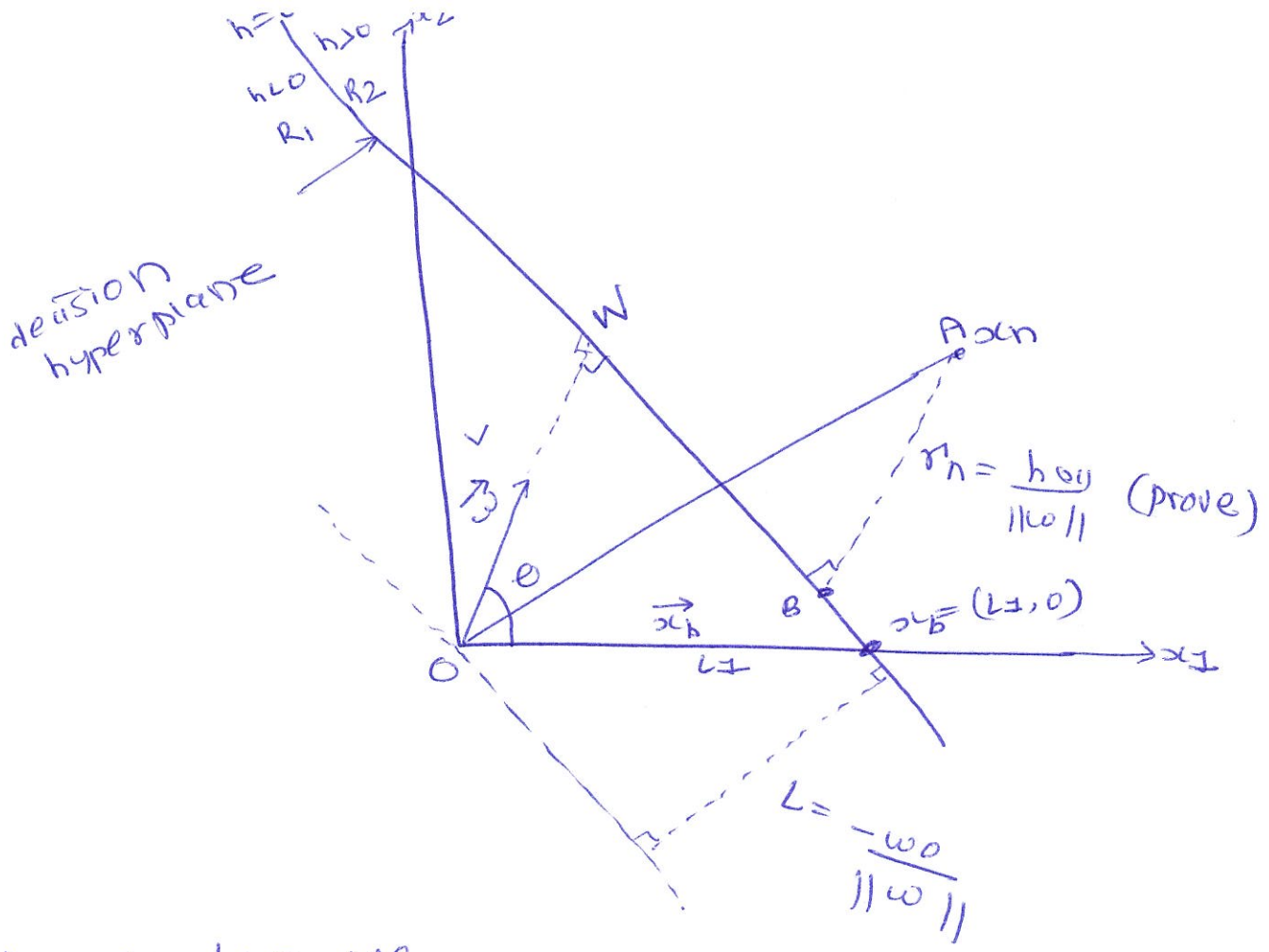
HW 05

Bhishan Poudel

Qn 1

prove:

dist betn data point & decision hyp. $= \frac{h(x)}{||w||}$



a) Prove $L_1 = \frac{-w_0}{\|w\|}$

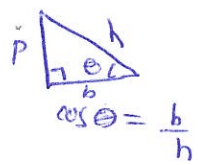
Here, $\vec{w} = [w_1, w_2]$

$\vec{x_b} = [x_1, x_2] = [L_1, 0]$

$\vec{w} \cdot \vec{x_b} + w_0 = 0 = w_1 x_1 + w_2 x_2 + w_0$

$0 = w_1 L_1 + w_2 \cdot 0 + w_0$

$$L_1 = \frac{-w_0}{w_1} \quad \text{---} (1)$$



Again, $\cos\theta = \frac{\text{base}}{\text{hypotenuse}} = \frac{L}{L_1}$

$\cos\theta = \frac{L}{-w_0 / w_1}$

$$L = \frac{-w_0 \cos\theta}{w_1} \quad \text{---} (2)$$

$$L = \frac{-w_0}{w_1} \cdot \frac{\vec{w} \cdot \vec{x_b}}{\|w\| \|x\|} \quad \because \vec{a} \cdot \vec{b} = |\vec{a}| \, |\vec{b}| \cos\theta$$

$$= \frac{-w_0}{w_1} \cdot \frac{(w_1, w_2) \cdot (L_1, 0)}{\|w\| L_1}$$

$$= \frac{-w_0}{w_1} \frac{w_1 L_1}{\|w\| L_1}$$

$$\boxed{L = \frac{-w_0}{\|w\|}} \quad \text{Ans}$$

Again,

To find the coordinate of point B,

$$\vec{OB} = \vec{OA} - \vec{AB}$$

$$\vec{B} = \vec{x_n} - r_n \hat{w} \longrightarrow \hat{w} \text{ is unit vector perpendicular to decision hyperplane}$$

$$\boxed{B = x_n - r_n \frac{w}{\|w\|}}$$

But, point B lies in decision boundary, so,

$$\vec{w} \cdot \vec{B} + w_0 = 0$$

$$w^T \left(x_n - r_n \frac{w}{\|w\|} \right) + w_0 = 0$$

$$\text{or, } w^T x_n - r_n \frac{w^T w}{\|w\|} + w_0 = 0 \qquad \because w^T w = \|w\|^2$$

$$\text{or, } w^T x_n - r_n \|w\| + w_0 = 0$$

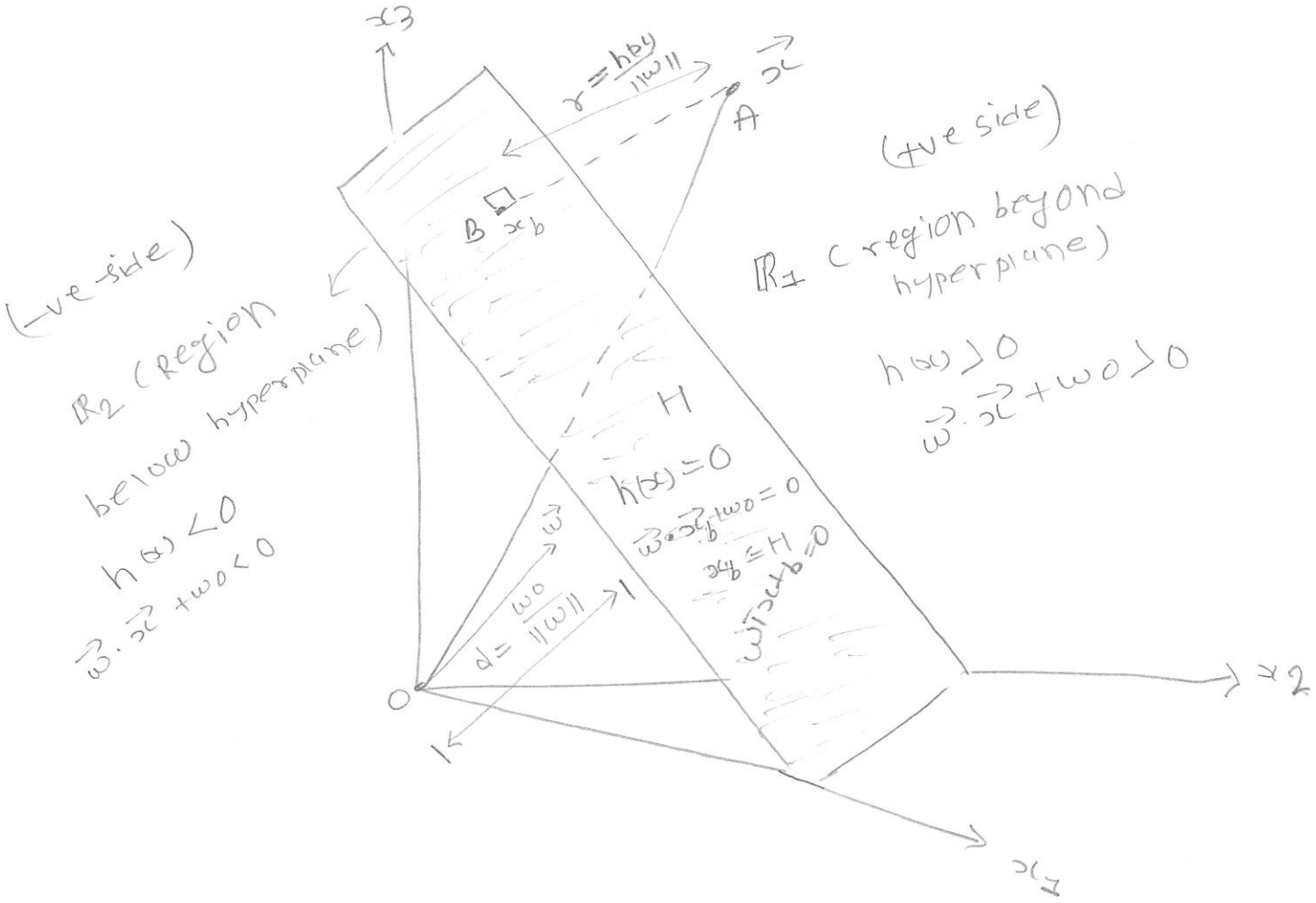
or, $w^T x_n + w_0 = r_n \|w\|$

$\therefore \quad r_n = AB = \dfrac{w^T x_n + w_0}{\|w\|}$ where $h(x) = w^T x + w_0$

$$\boxed{r_n = \frac{h(x)}{\|w\|}}$$ proved!

✓ 20

QN 2



Qn 2 For n-dimensional case, prove:

dist b/w origin and decision hyperplane $= \frac{-w_0}{\|w\|}$

& dist b/w datapoint and decision hyperplane $= \frac{h(x)}{\|w\|}$

Soln Let $\vec{x}$ be any point on the positive side of decision boundary (hyperplane), then,

$$\vec{x} = \vec{x_b} + r\frac{\vec{w}}{\|w\|}$$ where $\frac{w}{\|w\|}$ is unit vector perpendicular to decision hyperplane

$$\boxed{x_b = x - r\frac{w}{\|w\|}} \quad \text{---} (1)$$

but, x_b lies in boundary, we need to find r,

$$\boxed{\vec{w}\cdot\vec{x_b} + w_0 = 0} \quad \text{---} (2)$$

w

or, $w^T\left(x - r\frac{w}{\|w\|}\right) + w_0 = 0 = w^T x - r\frac{w^T w}{\|w\|} + w_0 = 0$

or, $w^T x + w_0 = r\|w\| = h(x)$

distance from datapoint to hyperplane $\Rightarrow$ $r = \frac{w^T x + w_0}{\|w\|}$

$$\boxed{r = \frac{h(x)}{\|w\|}}$$ Ans

Also, we have to show the distance of hyperplane from the origin,

$$\overrightarrow{OB} = -\frac{w_0}{\|w\|} \hat{OB}$$

Here, the point x_b lies in decision hyperplane, so,

$$\vec{w} \cdot \vec{x_b} + w_0 = 0$$

$$w x_b + w_0 = 0$$

$$w x_b = -w_0$$

distance from origin to decision hyperplane

$$\boxed{x_b = \frac{-w_0}{\|w\|}}$$

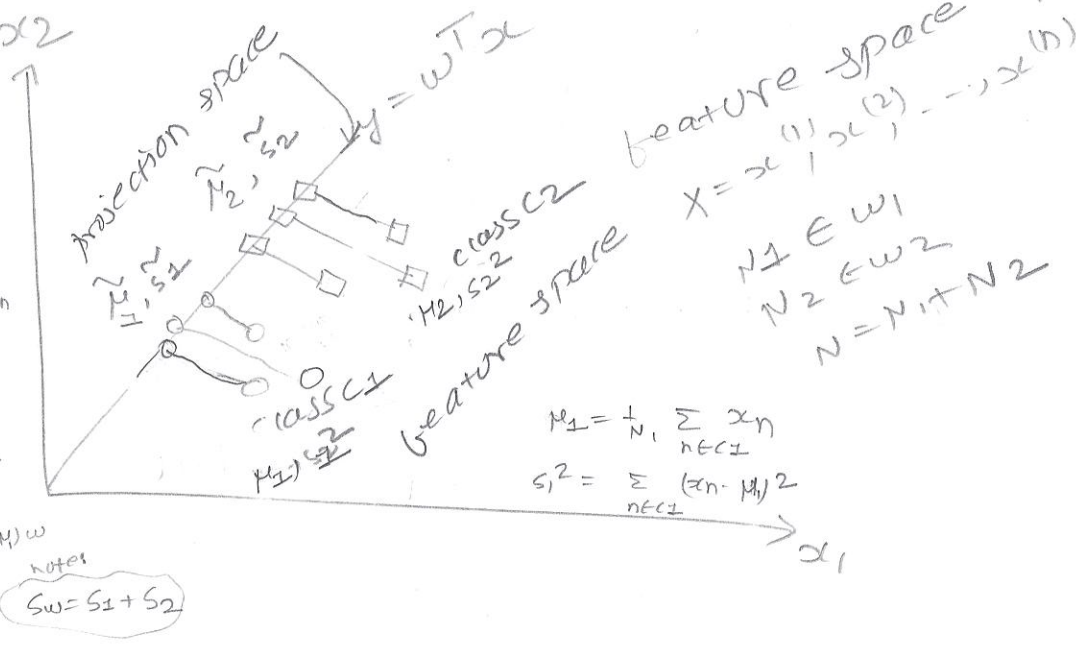
Ans

Q#3 show that Fisher criterion

Bishop (ex 4.5) can be written as,

$$J(w) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} = \frac{w^T S_B w}{w^T S_w w}$$

solution



feature space X

$X = x^{(1)}, x^{(2)}, \ldots, x^{(n)}$

$N_1 \in w_1$

$N_2 \in w_2$

$N = N_1 + N_2$

$\tilde{\mu}_1 = \frac{1}{N_1} \sum_{n \in c_1} w^T x_n = w^T m_1$

$s_1^2 = \sum_{n \in c_1} (y_n - \tilde{\mu}_1)^2$

$= \sum_{n \in c_1} (w^T x_n - w^T m_1)^2$

$= \sum_n w^T (x_n - m_1)^T (x_n - m_1) w$

$s_1^2 = w^T S_1 w$ (big S)

note: $S_w = S_1 + S_2$

$\mu_1 = \frac{1}{N_1} \sum_{n \in c_1} x_n$

$s_1^2 = \sum_{n \in c_1} (x_n - \mu_1)^2$

In feature space X, we have input data,

$$x = (x^{(1)}, x^{(2)}, \ldots, x^{(n)})$$

feature space X, mean $\mu_i = \frac{1}{N_i} \sum_{x \in c_i} x$

projection space y, mean $\tilde{\mu}_i = \frac{1}{N_i} \sum_{y \in c_i} y$

$= \frac{1}{N_i} \sum_{x \in c_i} w^T x$

$\tilde{\mu}_i = w^T \mu_i$

variance

ALSO,

variance, (feature space)

$$\sigma_i^2 = \frac{1}{N_i} \sum_{x \in c_i} (x - M_i)^2$$

variance (projection space)

$$\tilde{\sigma}_i^2 = \frac{1}{N_i} \sum_{y \in c_i} (y - \tilde{M}_i)^2$$

ALSO, scatter in feature space,

$$S_i^2 = \sum_{x \in c_i} (x - M_i)^2 = \sum_{x \in c_i} (x - M_i)(x - M_i)^T$$

scatter in projection space,

scatter

$$\tilde{S}_i^2 = \sum_{y \in c_i} (y - \tilde{\mu}_i)^2$$

ALSO, within-class scatter $S_w = S_1 + S_2$

$$\tilde{S}_w = \tilde{S}_1 + \tilde{S}_2$$

ALSO,

Now, we define Fisher's criterion,

$$J(w) = \frac{|\tilde{\mu}_1 - \tilde{\mu}_2|^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$$

Fisher's linear discriminant

Now,

$$(\tilde{\mu}_1 - \tilde{\mu}_2)^2 = (w^T \mu_1 - w^T \mu_2)^2$$

$$= w^T (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T w$$

$$\boxed{(\tilde{\mu}_1 - \tilde{\mu}_2)^2 = w^T S_B w} \quad (A)$$

where, $S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$ is between-class scatter.

Also,

$$\tilde{s}_i^2 = \sum_{y \in \omega_i} (y - \tilde{\mu}_i)^2$$

$$= \sum_{x \in \omega_i} (w^T x - w^T \mu_i)^2$$

$$= \sum_{x \in \omega_i} w^T (x - \mu_i)(x - \mu_i)^T w$$

$$\boxed{\tilde{s}_i^2 = w^T S_i w}$$ where, feature-space scatter

$$S_i = \sum_{x \in \omega_i} (x - \mu_i)(x - \mu_i)^T$$

$$\Rightarrow \tilde{s}_1^2 = w^T S_1 w$$

$$\tilde{s}_2^2 = w^T S_2 w$$

$$\tilde{s}_1^2 + \tilde{s}_2^2 = w^T (S_1 + S_2) w = w^T S_w w$$

where $S_w = S_1 + S_2$ is within-class scatter.

$$\Rightarrow \boxed{\tilde{s}_1^2 + \tilde{s}_2^2 = w^T S_w w} \quad \longleftarrow (B)$$

using equations (A) & (B), we have,

$$J(w) = \frac{|\tilde{\mu}_1 - \tilde{\mu}_2|^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$$

$$\boxed{J(w) = \frac{w^T S_B w}{w^T S_w w}}$$

Q.E.D.!

QN4 In Bishop BOOK, (p. 190 & 208)

Bishop ex 4.6 p.

use equations: 4.27, 4.28, 4.34, 4.36, 4.33 to prove eq 4.37.

Derive: $\left(S_W + \frac{N_1 N_2}{N} S_B\right) w = N (m_1 - m_2)$ ——— (4.37)

Here, given, SSE is given by

$$E(w) = \frac{1}{2} \sum_{n=1}^{N} (w^T x_n + w_0 - t_n)^2$$ ——— (4.31) p. 190/208

$$\frac{\partial E}{\partial w} = 0 = \sum_{n=1}^{N} (w^T x_n + w_0 - t_n) x_n$$ ——— (4.33)

(4.34)

$$w_0 = -w^T m$$

$$t_n = \begin{cases} \frac{N}{N_1} & \text{if } x_n \in C_1 \\ -\frac{N}{N_2} & \text{if } x_n \in C_2 \end{cases}$$

so that $\sum_{n=1}^{N} t_n = N_1 \frac{N}{N_1} - N_2 \frac{N}{N_2}$

$$\sum_{n=1}^{N} t_n = 0$$

mean of the total dataset (population mean)

(4.36)

$$m = \frac{1}{N} \sum_{n=1}^{N} x_n = \frac{1}{N} N_1 m_1 + N_2 m_2$$

where, (4.21) p.205

(sample mean of class C_1)

$$m_1 = \frac{1}{N_1} \sum_{n \in C_1} x_n$$ (mean of x_n belonging to class C_1)

$$\Rightarrow \sum_{n \in C_1} x_n = N_1 m_1$$

$$m_2 = \frac{1}{N_2} \sum_{n \in C_2} x_n \Rightarrow \sum_{n \in C_2} x_n = N_2 m_2$$

we define within-class scatter as,

eq 4.28 p.207

$$S_W = \sum_{n \in C_1} (x_n - m_1)(x_n - m_1)^T + \sum_{n \in C_2} (x_n - m_2)(x_n - m_2)^T$$

and, between-class scatter as,

eq 4.27 (p.207)

$$S_B = (m_2 - m_1)(m_2 - m_1)^T$$

we have to prove,

TO prove

$$(S_W + \frac{N_1 N_2}{N} S_B) w = N(m_1 - m_2) \quad \text{(eq 4.37)}$$

$$\Rightarrow \frac{N(m_1 - m_2)}{w} = S_W + \frac{N_1 N_2 S_B}{N} \quad \text{——— (1)}$$

we start from gradient zero equation (4.33),

$$0 = \sum_{n=1}^{N} (w^T x_n - w_0 - t_n) x_n$$

$$= \sum_{n=1}^{N} (w^T x_n - w^T m - t_n) x_n$$

$$= \sum_{n=1}^{N} w^T (x_n x_n^T - m x_n) - \sum_{n=1}^{N} t_n x_n$$

$$= \sum_{n \in C_1} (x_n x_n^T - x_n m^T) w - \sum_{n \in C_1} t_n x_n + \sum_{n \in C_2} (x_n x_n^T - x_n m^T) w - \sum_{n \in C_2} t_n x_n$$

use $t_n = \begin{cases} \frac{N}{N_1} & \text{for } C_1 \\ -\frac{N}{N_2} & \text{for } C_2 \end{cases}$

$$m_2 = \frac{1}{N_2} \sum_{n \in C_2} x_n \Rightarrow \sum_{n \in C_2} x_n = N_2 m_2$$

$$0 = \left(\sum_{n\in C_1} x_n x_n^T - N_1 m_1 m^T\right) w - \frac{N}{N_1} N_1 m_1$$

$$\left(\sum_{n\in C_2} x_n x_n^T - N_2 m_2 m^T\right) w + \frac{N}{N_2} N_2 m_2$$

$$0 = \left(\sum_{n\in C_1} x_n x_n^T + \sum_{n\in C_2} x_n x_n^T - (N_1 m_1 + N_2 m_2) m^T\right) w - N(m_1 - m_2)$$

$$\frac{N(m_1 - m_2)}{w} = \underbrace{\sum_{n\in C_1} x_n x_n^T + \sum_{n\in C_2} x_n x_n^T}_{\text{write in terms of } S_w} - (N_1 m_1 + N_2 m_2) m^T \quad \text{(2)}$$

$$m = \frac{N_1 m_1 + N_2 m_2}{N}$$

Now, we expand the first term in S_w,

$$\sum_{n\in C_1} (x_n - m_1)(x_n - m_1)^T = \sum_{n\in C_1} (x_n x_n^T - x_n m_1^T - m_1 x_n^T + m_1 m_1^T)$$

$$= \sum_{n\in C_1} x_n x_n^T - m_1^T \sum_{n\in C_1} x_n - m_1 \sum_{n\in C_1} x_n^T + \sum_{n\in C_1} m_1 m_1^T$$

$$= \sum_{n\in C_1} x_n x_n^T - N_1 m_1 m_1^T - N_1 m_1 m_1^T + N_1 m_1 m_1^T$$

$$\boxed{\sum_{n\in C_1} (x_n - m_1)(x_n - m_1)^T = \sum_{n\in C_1} x_n x_n^T - N_1 m_1 m_1^T}$$

So, $S_w = \sum_{n\in C_1} (x_n - m_1)(x_n - m_1)^T + \sum_{n\in C_2} (x_n - m_2)(x_n - m_2)^T$

$$\boxed{S_w = \sum_{n\in C_1} x_n x_n^T + \sum_{n\in C_2} x_n x_n^T - N_1 m_1 m_1^T - N_2 m_2 m_2^T}$$

$$\Rightarrow \sum_{n\in C_1} x_n x_n^T + \sum_{n\in C_2} x_n x_n^T = S_w + N_1 m_1 m_1^T + N_2 m_2 m_2^T$$

Then eqn ② becomes,

$$\frac{N(m_1-m_2)}{\omega} = S_W + N_1 m_1 m_1^T + N_2 m_2 m_2^T - (N_1 m_1 + N_2 m_2)\cdot\frac{(N_1 m_1 + N_2 m_2)^T}{N}$$

$$= S_W + m_1 m_1^T\left(N_1 - \frac{N_1^2}{N}\right) + m_2 m_2^T\left(N_2 - \frac{N_2^2}{N}\right) - \frac{N_1 N_2}{N}(m_1 m_2^T + m_2 m_1^T)$$

$N = N_1 + N_2$

$N_1(N_1+N_2-N_1) = N_1 N_2$; $N_2(N_1+N_2-N_2) = N_1 N_2$

$$= S_W + \frac{(N_1+N_2)N_1 - N_1^2}{N} m_1 m_1^T + \frac{(N_1+N_2)N_2 - N_2^2}{N} m_2 m_2^T - \frac{N_1 N_2}{N}(m_1 m_2^T + m_2 m_1^T)$$

$$= S_W + \frac{N_1 N_2}{N}(m_1 m_1^T + m_2 m_2^T) - \frac{N_1 N_2}{N}(m_1 m_2^T + m_2 m_1^T)$$

$$= S_W + \frac{N_1 N_2}{N}(m_1 m_1^T + m_2 m_2^T - m_1 m_2^T - m_2 m_1^T)$$

$S_B = (m_2-m_1)(m_2-m_1)^T$

$= m_2 m_2^T - m_2 m_1^T - m_1 m_2^T + m_1 m_1^T$

$= m_1 m_1^T + m_2 m_2^T - m_1 m_2^T - m_2 m_1^T$

$$\frac{N(m_1-m_2)}{\omega} = S_W + \frac{N_1 N_2}{N} S_B$$

$$\therefore \left(S_W + \frac{N_1 N_2}{N} S_B\right)\omega = N(m_1-m_2)$$

Q.E.D.

20

✓

Q#5

Given training set,

x	x_1	x_2	x_3	$t(x)$
$x^{(1)}$	0	0	0	+1
$x^{(2)}$	0	1	0	+1
$x^{(3)}$	1.5	0	-1.5	+1
$x^{(4)}$	1.5	1	-1.5	+1
$x^{(5)}$	1.5	0	0	-1
$x^{(6)}$	1.5	1	0	-1
$x^{(7)}$	0	0	-1.5	-1
$x^{(8)}$	0	1	-1.5	-1

a) prove that the perceptron algorithm does not converge on this dataset. (Note: Include bias term)

b) consider a kernel perceptron that uses a polynomial kernel $K(x,y) = (1+x^Ty)^d$.

What is the smallest degree d for which the kernel perceptron would converge on this dataset?

(5a) for a given example, example belongs to class +1 if

$$w_0 + \vec{w} \cdot \vec{x_n} \geq 0$$

and, example belongs to class −1 if

$$w_0 + \vec{w} \cdot \vec{x_n} < 0$$

now, $x^{(1)} = \overset{w_1\ w_2\ w_3}{[0, 0, 0]}$ $t^{(1)} = +1$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 + w_3 \cdot 0 \geq 0$$

$$w_0 \geq 0$$

similarly,

2a

+ve
① $w_0 + 0 + 0 + 0 \geq 0$ ——①
② $w_0 + 0 + w_2 + 0 \geq 0$ ——②
③ $w_0 + 1.5w_1 + 0 - 1.5w_3 \geq 0$ ——③
④ $w_0 + 1.5w_1 + w_2 - 1.5w_3 \geq 0$ ——④

④ − ③ ⇒ $w_2 \geq 0$

−ve
⑤ $w_0 + 1.5w_1 + 0 + 0 < 0$ ——⑤
⑥ $w_0 + 1.5w_1 + w_2 + 0 < 0$ ——⑥
⑦ $w_0 + 0 + 0 - 1.5w_3 < 0$ ——⑦
⑧ $w_0 + 0 + w_2 - 1.5w_3 < 0$ ——⑧

⑥ − ⑤ ⇒ $w_2 < 0$

here, ④ − ③ gives $w_2 \geq 0$
but ⑥ − ⑤ gives $w_2 < 0$

this is contradictory so perceptron fails.

(5b) Here, the given dataset is not linearly separable in X-space.

we shall propose a perceptron that uses polynomial kernel,

the polynomial kernel is,

$$K(x, x') = (a x^T x' + b)^d$$

for $a = b = 1$

$$\boxed{K(x, x') = (1 + x^T x')^d}$$

a = scaling factor
b = bias term
d = degree of polynomi
x = augmented dataset of shape N rows m+1 cols
x' = one example of datase with m+1 features

when d = 1 : CANNOT — the kernel is linear classifier, but our dataset is NON linear, so it cannot classify correctly.

12

when d = 2: CAN — when degree is 2, quadratic kernel can classify linearly non-separable dataset.

need find a vector to seperate the data set.

when d > 2 : may overfit — higher degree polynomial kernels can fit the linearly non-separable dataset good. but, they may overfit the test dataset.

∴ Smallest degree d = 2 Ans

q1v6

For a kernel perceptron, from lecture note, (Lecture 06) the algorithm is

1. $f(x) = w^T x = \sum_n \alpha_n t_n \vec{x_n}^T x = \sum_n \alpha_n t_n \underbrace{K(x_n, x)}_{\text{kernel}}$

2. initialize α : $\alpha_n = 0$

3. for e in E epochs
   for $n = 1 \cdots N$
   $h_n = \text{sgn}(f(x_n))$
   if $h_n \neq t_n$ then
   $\alpha_n = \alpha_n + 1$

Now, from step 3, if we run perceptron E number of epochs, then total number of mistakes made so far is,

$$M = \sum_{n=1}^{N} \alpha_n$$

ANS

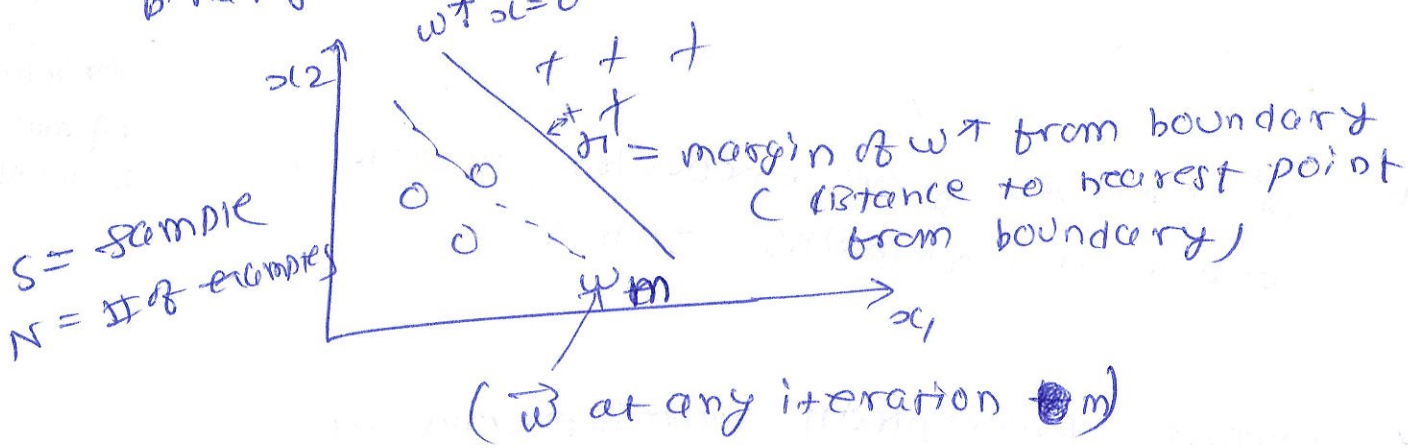
10

Q.N 6B Method 2 (Extra practice)

A Kernel perceptron for binary classification is run for a number of epochs E on a training dataset containing N examples, resulting in the dual parameters $\alpha_1, \alpha_2, \ldots, \alpha_N$. What is the total number of mistakes that are made during training?

- maximum error bound

Soln Consider a kernel perceptron for binary classification.



Let there is a unit vector $\vec{w}^T$ that can separate N training examples into two classes with margin γ.

Let R = norm of maximum value of $\vec{x}$ in dataset

$$\|\vec{R}\| = \max_{x \in S} \|\vec{x}\|$$

perceptron Algorithm:

1. initialize w to zero. $w_1 = [0\ 0\ 0\ 0]$
2. for e in epochs

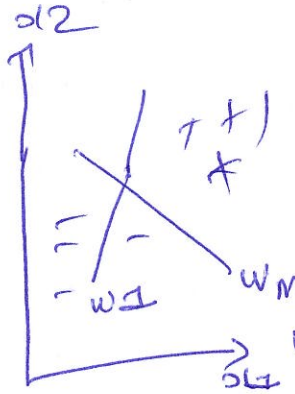
   for n = 1 to N

   if $(w \cdot x_n) < 0$

   $w = w + t x_n$

   if converged w:

   break



$w^* = \frac{w_M}{|w_M|}$

for total M mistakes $1 \le m \le M$

* $\vec{w}$ changes if we encounter mistake, otherwise remains same.

* suppose perceptron makes a mistake at iteration m, then,

$$\vec{w}_m \cdot w^* = (\vec{w}_{m-1} + t\vec{x}) \cdot w^*$$

$$= \vec{w}_{m-1} \cdot w^* + t(\vec{x} \cdot \vec{w}^*)$$

$$w_m \cdot w_* \geq \vec{w}_{m-1} \cdot \vec{w}^* + \gamma$$

from definition of γ. each data point $\vec{x}$ is $\geq \gamma$ from the boundary w^*

$\therefore$ $\boxed{w_m \cdot w^* \geq w_{m-1} \cdot w^* + \gamma}$ ——— (A)

Also,

$$\|w_m\|^2 = \|w_{m-1} + tx\|^2$$

$$= \|w_{m-1}\|^2 + 2t(w_{m-1} \cdot x) + \|tx\|^2$$

(b) maximum $\|x\|$ is R

(a) Here we have mistake at iteration m, so the previous iteration has hypothesis $t(w_{m-1} \cdot x) < 0$

$$\boxed{\|w_m\|^2 \leq \|w_{m-1}\|^2 + R^2} \quad \text{(B)}$$

(from (a) & (b))

Here, inequalities (A) & (B) hold true for all the iterations m,

$$w_m \cdot w^* \geq w_{m-1} \cdot w^* + \gamma$$

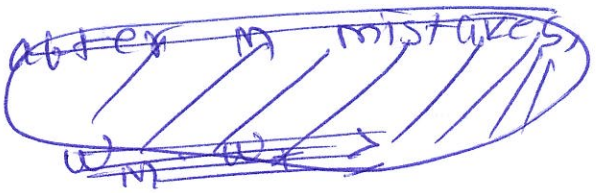
$$\|w_m\|^2 \leq \|w_{m-1}\|^2 + R^2$$

$$\|w_m\|^2 - \|w_{m-1}\|^2 \leq R^2$$

difference of

(if there is a mistake then square of weight vectors is less than R^2).

(for each mistake $w^2 - w_{m-1}^2$ cannot be larger than R^2)



For all iterations,

$$w_m \cdot w_* \geq w_{m-1} \cdot w_* + \gamma$$

after M mistakes,

$$\boxed{\vec{w}_M \cdot \vec{w}_* \geq \gamma M} \quad \text{——} \quad (C)$$

also, for all iterations,

$$\|w_m\|^2 \leq \|w_{m-1}\|^2 + R^2$$

after M mistakes

$$\|w_M\|^2 \leq R^2 M$$

$$\boxed{\|w_M\| \leq R\sqrt{M}} \quad \text{——} \quad (D)$$

so, from (C) & (D),

$$\gamma M \leq \vec{w}_M \cdot \vec{w}_*$$

$$\leq \|\vec{w}_m\| \, \|\vec{w}_*\|$$

$$\leq \|w_M\|$$

$$\gamma M \leq R\sqrt{M}$$

$$\gamma \sqrt{M} \leq R$$

$$\gamma^2 M \leq R^2$$

$$\boxed{M \leq \left(\frac{R}{\gamma}\right)^2}$$

Ans

$\because \vec{a} \cdot \vec{b} = ab\cos\theta$

$\cos\theta$ is less than or equal to 1

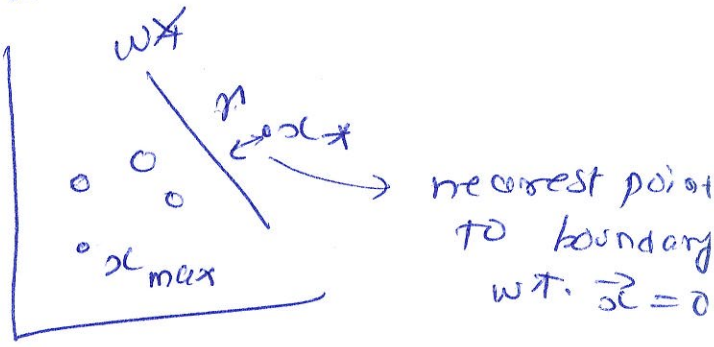
w_* is unit vector of $\vec{w}_M$

$\|w_*\| = 1$

Hence,

Let the data (training) sample S is,

x_1	t_1
x_2	t_2
x_3	t_3
⋮	⋮
x_N	t_N



$||x_{max}|| = R$

example,

x_1: 0 0 1 2 3 $\quad t_1 = +$

x_2: 5 0 3 2 6 $\quad t_2 = -$

x_N: 10 20 30 40 $\quad t_N = +$

then, if we use Perceptron to train our model, the total number of mistakes (M) made is,

$$M \leq \left(\frac{R}{\gamma}\right)^2$$

Q N 7 Let $U \in R_{K \times m}$

$X \in R_{n \times m}$

u_i, x_i = i^{th} column of U and X resp.

$1 \le i \le m$.

prove $UX^T = \sum_{i=1}^{m} u_i x_i^T$

Solution:

$$U = U_{K,m} = \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1m} \\ u_{21} & u_{22} & \cdots & u_{2m} \\ \vdots & & & \\ u_{K1} & u_{K2} & \cdots & u_{Km} \end{pmatrix}_{K,m} = \underset{1,m}{(u_1 \; u_2 \cdots u_m)}$$

K rows
m cols

$$X = X_{n,m} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & & & \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{pmatrix}_{n,m} = \underset{1,m}{(x_1 \; x_2 \cdots x_m)}$$

n rows
m cols

x_1 ↑ column vector

$$V = X^T = \begin{pmatrix} x_{11} & x_{21} & \cdots & x_{n1} \\ x_{12} & x_{22} & & x_{n2} \\ \vdots & \vdots & & \vdots \\ x_{1m} & x_{2m} & & x_{nm} \end{pmatrix} = \begin{pmatrix} x_1^T \\ x_2^T \\ \\ x_m^T \end{pmatrix}_{m,1}$$

From definition of matrix multiplication

if $U = U_{Km}$

and $V = V_{mn}$

then,

$$(UV)_{pq} = \sum_{i=1}^{m} U_{pi} V_{iq} \qquad \forall\ 1 < p < K \text{ and } 1 < q < n$$

$1 < p < K$

$1 < q < n$

where $X = \begin{pmatrix} x_{11} & x_{12} & x_{1n} \\ x_{21} & x_{22} & x_{2n} \\ x_{m1} & x_{m2} & x_{mn} \end{pmatrix}_{m,n}$

m rows

n cols

$$= \begin{pmatrix} u_{11} & u_{12} & u_{1m} \\ u_{21} & u_{22} & u_{2m} \\ u_{K1} & u_{K2} & u_{Km} \end{pmatrix}_{K,m} \begin{pmatrix} x_{11} & x_{21} & \cdots & x_{n1} \\ x_{12} & x_{22} & & x_{n2} \\ x_{1m} & x_{2m} & & x_{nm} \end{pmatrix}_{m,n} \qquad X_1 = \begin{pmatrix} x_{11} \\ x_{21} \\ x_{m1} \end{pmatrix}_{n,1}$$

$$= \begin{pmatrix} u_{11}x_{11} + u_{12}x_{12} + u_{1m}x_{1m} & u_{11}x_{21} + u_{12}x_{22} + u_{1m}x_{2m} & u_{11}x_{n1} + u_{12}x_{n2} + u_{1m}x_{nm} \\ u_{21}x_{11} + u_{22}x_{12} + u_{2m}x_{1m} & u_{21}x_{21} + u_{22}x_{22} + u_{2m}x_{2m} & u_{21}x_{n1} + u_{22}x_{n2} + u_{2m}x_{nm} \\ u_{K1}x_{11} + u_{K2}x_{12} + u_{Km}x_{1m} & u_{K1}x_{21} + u_{K2}x_{22} + u_{Km}x_{2m} & u_{K1}x_{n1} + u_{K2}x_{n2} + u_{Km}x_{nm} \end{pmatrix}_{K,n}$$

$$= \begin{pmatrix} u_{11}x_{11} & u_{11}x_{21} & u_{11}x_{n1} \\ u_{21}x_{11} & u_{21}x_{21} & u_{21}x_{n1} \\ u_{K1}x_{11} & u_{K1}x_{21} & u_{K1}x_{n1} \end{pmatrix}_{K,n} + \begin{pmatrix} u_{12}x_{12} & u_{12}x_{22} & u_{12}x_{n2} \\ u_{22}x_{12} & u_{22}x_{22} & u_{22}x_{n2} \\ u_{K2}x_{12} & u_{K2}x_{22} & u_{K2}x_{n2} \end{pmatrix}_{K,n} + \begin{pmatrix} u_{1m}x_{1m} & u_{12}x_{2m} & u_{1m}x_{nm} \\ u_{2m}x_{1m} & u_{22}x_{2m} & u_{2m}x_{nm} \\ u_{1m}x_{1m} & u_{K2}x_{2m} & u_{Km}x_{nm} \end{pmatrix}_{K,n}$$

$$= U_1 X_1^T + U_2 X_2^T + U_m X_m^T$$

10 ✓

since,

$$u_1 x_1^T = \begin{pmatrix} u_{11} \\ u_{21} \\ u_{k1} \end{pmatrix} \begin{pmatrix} x_{11} \\ x_{21} \\ x_{n1} \end{pmatrix}^T$$

$$= \begin{pmatrix} u_{11} \\ u_{21} \\ u_{k1} \end{pmatrix}_{k,1} \overrightarrow{\quad} \quad \underset{\downarrow}{(x_{11} \; x_{21} \; x_{n1})_{1,n}}$$

$$u_1 x_1^T = \begin{pmatrix} u_{11}x_{11} & u_{11}x_{21} & u_{11}x_{n1} \\ u_{21}x_{11} & u_{21}x_{21} & u_{21}x_{n1} \\ u_{k1}x_{11} & u_{k1}x_{21} & u_{k1}x_{n1} \end{pmatrix}_{k,n} \quad \text{——— (a)}$$

similarly,

$$u_2 x_2^T = \begin{pmatrix} u_{12}x_{12} & u_{12}x_{22} & u_{12}x_{n2} \\ u_{22}x_{12} & u_{22}x_{22} & u_{22}x_{n2} \\ u_{k2}x_{12} & u_{k2}x_{22} & u_{k2}x_{n2} \end{pmatrix}_{k,n} \quad \text{——— (b)}$$

and,

$$u_m x_m^T = \begin{pmatrix} u_{1m}x_{1m} & u_{12}x_{12} & u_{1m}x_{nm} \\ u_{2m}x_{1m} & u_{22}x_{22} & u_{2m}x_{nm} \\ u_{km}x_{1m} & u_{k2}x_{22} & u_{km}x_{nm} \end{pmatrix} \quad \text{——— (c)}$$

$\therefore \quad UX^T = u_1 x_1^T + u_2 x_2^T + \cdots + u_m x_m^T$

$$\boxed{UX^T = \sum_{i=1}^{m} u_i x_i^T}$$ Q.E.D.

Bhishan Poudel

225/250

HW Assignment 7 (Due by 10:30am on Nov 30)

1 Theory (140 points)

50 1. [**Normalized Kernels**, 50 + 25 points]
Let $K : X \times X \to R$ be a kernel function defined over a sample space X.

(a) Prove that the function below (a *normalized kernel*) is a valid kernel.

$$\frac{K(x,y)}{\sqrt{K(x,x)K(y,y)}}$$

(b) Why it would not make sense to normalize a Gaussian kernel?

(c) [**Bonus**] Which types of input data would benefit from normalizing the kernel function? Explain why, and provide real world examples.

50 2. [**Support Vector Machines**, 50 points]
Prove that the sum of slacks $\sum \xi_n$ from the objective function of the SVM formulation with soft margin is an upper bound on the number of misclassified training examples.

20 3. [**Max Margin Hyperplanes**, 20 points]
Consider the constrained optimization SVM problem for the separable case shown on slide 12. Show that, if the 1 on the right-hand side of the inequality constraint is replaced by some arbitrary constant $\gamma > 0$, the resulting maximum margin hyperplane is unchanged.

40 4. [**Kernel Techniques**, 20 + 20 points]
(a) Show that if $k_1(\mathbf{x}, \mathbf{y})$ and $k_2(\mathbf{x}, \mathbf{y})$ are valid kernel functions, then $k(\mathbf{x}, \mathbf{y}) = k_1(\mathbf{x}, \mathbf{y})k_2(\mathbf{x}, \mathbf{y})$ is also a valid kernel.
(b) **(*)** Show that if A is a symmetric positive semidefinite matrix, then $k(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T A\mathbf{y}$ is a valid kernel. *Hint: Inderjit Dhillon's Linear Algebra Background describes some useful properties of symmetric positive semidefinite matrices.*

20 5. [**Positive Definite Matrices (*)**, 20 points]
Show that a diagonal matrix $\mathbf{W}$ whose elements satisfy $0 < W_{ii} < 1$ is positive definite. Show that the sum of two positive definite matrices is itself positive definite.

50 30 6. [**Large Margin Perceptron (*)**, 30 points]
Let $\mathbf{u}$ be a current current vector of parameters and $\mathbf{x}$ and $\mathbf{y}$ two training examples such that $\mathbf{u}^T(\mathbf{x} - \mathbf{y}) < 1$. Use the technique of Lagrange multipliers to find a new vector of parameters $\mathbf{w}$ as the solution to the convex optimization problem below:

minimize:
$$J(\mathbf{w}) = \tfrac{1}{2}\|\mathbf{w} - \mathbf{u}\|^2$$

subject to:
$$\mathbf{w}^T(\mathbf{x} - \mathbf{y}) \geq 1$$

2 Text Classification (100 points)

Train and test the SVM algorithm on the *Spam vs. Non-spam* and *Atheism vs. Religion* classification problems, using the datasets provided for the previous assignment. Use a linear kernel, with the cost parameter $C = 5$. Report and compare the accuracy of the trained SVM models with the perceptron and average perceptron accuracies from the previous assignment.

3 Digit Recognition (200 points)

In this exercise, you are asked to run an experimental evaluation of SVMs and the perceptron algorithm, with and without kernels, on the problem of classifying images representing digits.

1. The UCI Machine Learning Repository at *www.ics.uci.edu/~mlearn* maintains datasets for a wide variety of machine learning problems. For this assignment, you are supposed to work with the Optical Recognition of Handwritten Digits Data Set. The webpage for this dataset is at:
   http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits
   The actual dataset is located at:
   http://archive.ics.uci.edu/ml/machine-learning-databases/optdigits/

   Read the description of the dataset. Download the training set `optdigits.tra` and the test set `optdigits.tes`. Use the first 1000 examples in `optdigits.tra` for *development* and the rest of 2823 examples for *training*. Use all 1797 examples in `optdigits.tes` for *testing*. Scale all the features between [0, 1], as discussed in class, using the min and max computed over the training examples. Create training files for each of the 10 digits, setting the class to 1 for instances of that digit, and to -1 for instances of other digits, i.e. *one-vs-rest* scenario.

2. Train first the linear perceptron, with the number of epochs set to $T \in \{1, 2, ..., 20\}$. After training each linear perceptron, normalize the learned weight vector. Select for T the value that obtains the best overall accuracy on the development data, and use this value for the remaining perceptron experiments.

   Run experiments with the linear and kernel perceptron algorithms. For the kernel perceptron, experiment with polynomial kernels $k(\mathbf{x}, \mathbf{y}) = (1 + \mathbf{x}^T\mathbf{y})^d$ with degrees $d \in \{2, 3, 4, 5, 6\}$, and with Gaussian kernels $k(\mathbf{x}, \mathbf{y}) = exp(-||\mathbf{x} - \mathbf{y}||^2/2\sigma^2)$ with the width $\sigma \in \{0.1, 0.5, 2, 5, 10\}$. For each hyper-parameter value, you will have trained 10 models, one for each digit. In order to compute the label for a test or development example, you will run the 10 trained models and output the label that obtains the highest score. Compute the accuracy on the development data and identify the hyper-parameter value that obtains the best accuracy. Use the tuned hyper-parameter (d for poly-kernel, σ for Gaussian) to compute the overall performance on the test data.

   For each of the three perceptrons (linear, poly kernel and Gaussian kernel) report the total training time, the overall accuracy, and the number of support vectors. Show and compare the corresponding 4 confusion matrices. Which digit seems to be the hardest to classify? Which perceptron / kernel combination achieves the best performance? Which algorithms are slower at training time, and why?

3. Run the same experiments using SVMs instead of perceptrons, i.e. linear SVMs and SVMs with polynomial and Gaussian kernels. Use the same tuning scenarios for the hyper-parameters of the polynomial and Gaussian kernels. Use $C = 1$ in all SVM experiments. Report the same types of results and analysis as above, and compare with the perceptron results.

4 Tools

You are free to use MATLAB, R, or packages written in C++/Java/Python such as SVM-LIGHT (C), LIBSVM (C++, Java), or SCIKIT-LEARN (Python) to complete the implementation part of this assignment. Their web sites contain plenty of documentation on how to use them. If you use SCIKIT-LEARN, the following functionality from the `sklearn.svm` will be useful:

1. `SVC()`: This is the main class used for SVM classification models. Its implementation is based on LIBSVM. Make sure that you properly map the SVM hyper-parameters to the parameters in the constructor of this class. For example, the *gamma* parameter in the constructor corresponds to our $1/2\sigma^2$ coefficient in the Gaussian kernel. The formulas for the kernels implemented by SVC are described in this User Guide.

2. `decision_function(x)`: Once the classifier is trained, this will compute the distance between a sample `x` and the decision hyperplance. This is the quantity that you can use to determine the highest scoring class when training the 10 $one - vs - rest$ classifiers: once a classifier is trained for all 10 digits, given a sample `x` you compute this quantity for all 10 classifiers and select the class that corresponds to the classifier with largest decision function value.

3. `fit()`: This is the function used to train the classifier.

4. `predict(x)`: This is used to calculate the (binary) label for sample `x`.

LIBSVM, and therefore SCIKIT-LEARN too, already implement the *one-vs-rest* classification scheme. In this scheme, you can directly use the training dataset with the 10 original labels, and SCIKIT-LEARN will train the 10 binary classifiers for you. You can use this capability for this assignment, however bonus points will be given if you train the 10 binary classifiers directly, as described for the perceptron algorithm above, by creating a binary training dataset for each class.

5 Submission

Turn in a hard copy of your homework report at the beginning of class on the due date. Electronically submit on Blackboard a hw07.zip file that contains the hw07 folder in which you place the code and the datasets. Make sure you include a README.txt file explaining how the code is supposed to be used to replicate the results included in the report. The screen output produced when running the code should be redirected to (saved into) an **output.txt** file.

On a Linux system, creating the archive can be done using the command:

```
> zip -r hw07.zip hw07
```

Please observe the following when handing in homework:

1. Structure, indent, and format your code well.
2. Use adequate comments, both block and in-line to document your code.
3. **Do not submit third-party ML packages on Blackboard!** Just explain in the REAMDE file how you use external packages.
4. Make sure your code runs correctly when used in the directory structure shown above.
5. **Type and nicely format the project report**, including discussion points, tables, graphs etc. so that it is presentable and easy to read.
6. Working code and/or correct answers is only one part of the assignment. The project report, including discussion of the specific issues which the assignment asks about, is also a very important part of the assignment. Take the time and space to make an adequate and clear project report. On the non-programming learning-theory assignment, clear and complete explanations and proofs of your results are as important as getting the right answer.

HW 07

Bhishan Poudel

65/50

A1a

Normalized kernel is a valid kernel.

5/5

Let $K(x,y)$ is a valid kernel, then, there exist a feature map $\phi(\cdot)$ in some Hilbert space that,

$$K(x,y) = \phi(x) \cdot \phi(y)$$

Now, $$\hat{K}(x,y) = \frac{K(x,y)}{\sqrt{K(x,x) \cdot K(y,y)}}$$

$$= \frac{\phi(x) \cdot \phi(y)}{\|\phi(x)\| \, \|\phi(y)\|} = \hat{\phi}(x) \cdot \hat{\phi}(y)$$

where $\hat{\phi}(x) = \frac{\phi(x)}{\|\phi(x)\|}$

since $K(x,y)$ is kernel $\Rightarrow \phi(x) \cdot \phi(y) \geq 0$

also, denominator $\Rightarrow \|\phi(x)\| \, \|\phi(y)\| \geq 0$

$\therefore \hat{K}(x,y)$ is non-negative.

also $\hat{K}(x,y)$ is symmetric.

$\therefore$ $\hat{K}(x,y)$ is a valid kernel.

25/25

Q1N1b Gaussian function,

Gaussian fn

$$g(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\|x-\mu\|^2}{2\sigma^2}}$$

(normalized gaussian function when $y=\mu$)

the gaussian kernel is,

gaussian kernel

$$K(x,y) = e^{-\frac{\|x-y\|^2}{2\sigma^2}} = e^{-\gamma\|x-y\|^2}$$

where $\gamma = \frac{1}{2\sigma^2}$

$$K(x,x) = e^{-0} = 1$$

$K(x,x) = 1$ for gaussian kernel.

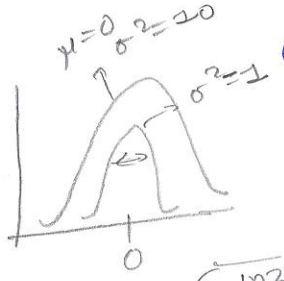
Since gaussian kernel is already normalized ($K(x,x)=1$), it would make no sense to normalize gaussian kernel.

10 some of the kernels used in practice are

linear kernel $K(x,y) = x^T y + c = \langle x, y \rangle + c = \vec{x} \cdot \vec{y} + c$

polynomial kernel $K(x,y) = (\alpha x^T y + c)^d$ α = slope, d = degree, c = const

Gaussian kernel $K(x,y) = e^{-\frac{1}{2\sigma^2}\|x-y\|^2} = e^{-\gamma\|x-y\|^2}$



(σ or γ determines the width of the gaussian curve.

overestimate σ → behave like linear kernel

underestimate σ → lack regularization and high sensitive to noise

tuning of σ → very important, otherwise very low performance

exponential kernel $K(x,y) = e^{-\frac{1}{\sigma}\|x-y\|}$

sigmoid kernel $K(x,y) = \tanh(\alpha x^T y + c)$

Gaussian kernels are already normalized and we need to normalize polynomial kernels.

cosine Normalization →
- Reduces dimension of data by 1 (since it projects all the data inside a unit radius sphere)
- → Good for high dimensional input data e.g. optdigits with 64 features
- → Not good for low dimensional data e.g. Iris data with only ~~two~~ four features.

ON2

prove: sum of slacks $\leq \xi_n$ is an upper bound on the number of misclassified training examples.

solution:

Input data X

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_i \\ \vdots \\ x_N \end{bmatrix}$$

Target t

$$\begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_i \\ \vdots \\ t_N \end{bmatrix} \qquad t \in \{-1, 1\}$$

$x_i \in \mathbb{R}^d$

$x_i \in \{x_{i1}, x_{i2}, \cdots, x_{id}\}$ d = dimension of features.

hypothesis $h_i = w^T x_i + b$

$\forall i \in \{1, \cdots N\}$

algorithm : constraint $t_i (w^T x_i + b) \geq 1 - \xi_i$

$$J(w, b) = \frac{1}{2} \|w\|^2$$

minimization objective

minimize $J(w, b) = \frac{1}{2} \|w\|^2$

with constraint $t_n (w^T x_n + b) \geq 1 - \xi_n$

consider $t_i = +1$

(correct classification)

$w^T x_i + b \geq 0$

$t_i (w^T x_i + b) \geq 0 \quad \because t_i = +1$

$t_i (w^T x_i + b) \geq 1 - \xi_i \; ; \; \xi_i \geq 0$

$\xi_i \geq 1$

$+ve \geq 1 - \xi_i$

since constraint want $\xi_i \geq 0$

ξ_i cannot be greater than 1

$0 \leq \xi_i \leq 1$

$t_i (w^T x_i + b) \geq 1$ if $\xi_i = 0$

(hard margin)

aside:

for perceptron

$h_i = sgn(w^T x_i + b)$

$h_i = t_i (w^T x_i + b)$

and if

$h_i \neq t_i$ we

update w, b

consider $t_i = +1$

(misclassification)

$w^T x_i + b < 0$

$t_i (w^T x_i + b) < 0 \quad \because t_i = +1$

constraint:

$t_i (w^T x_i + b) \geq 1 - \xi_i$

$-ve \geq 1 - \xi_i$

$\xi_i \geq 1 + (+ve)$

$\xi_i \geq 1$

consider $t_i = -1$ (correct classification)

$w^T x_i + b \leq 0$

$t_i (w^T x_i + b) \geq 0 \quad \because t_i = -1$

constraint:

$t_i (w^T x_i + b) \geq 1 - \xi_i$

$+ve \geq 1 - \xi_i$

$\xi_i \geq 1 - (+ve)$

$0 \leq \xi_i \leq 1$

and $t_i (w^T x_i + b) \geq 1$ if $\xi_i = 0$

$\Rightarrow$ for correct classification

$0 \leq \xi_i \leq 1$

$t_i = -1$ (misclassified)

$w^T x_i + b \geq 0$

$t_i (w^T x_i + b) < 0 \quad \because t_i = -1$

constraint:

$t_i (w^T x_i + b) \geq 1 - \xi_i$

$-ve \geq 1 - \xi_i$

$\xi_i \geq 1 + (+ve)$

$\xi_i \geq 1$

$\Rightarrow$ for misclassification

$\xi_i \geq 1$

now, for correct classification

$\xi_i \geq 0$

$\sum_{i \in \text{correct}} \xi_i \geq 0$

for misclassified

$\xi_i > 1$

$\sum_{i \in \text{misclassified}} \xi_i > \sum_{i \in \text{misclassified}} 1$

$\sum_{\text{misclassified}} \xi_i > \#\text{ of misclassified}$

$\therefore \sum_{i \in \text{correct}} \xi_i + \sum_{i \in \text{misclassified}} \xi_i > \#\text{ of misclassified}$

$\therefore \boxed{\sum_{i=1}^{N} \xi_i > \#\text{ of misclassified}}$

Q.E.D.

This shows that sum of slacks (ξ_i) over all the N training examples is greater than total number of misclassified examples, hence, sum of slacks is an upper bound to the number of misclassified examples.

Aside: The constrained optimization of SVM algorithms $t_i(w^T x_i + b) \geq 1$ is called Hard margin SVM, and, $t_i(w^T x_i + b) \geq 1 - \xi_i$ is called Soft margin SVM. They both are non-trivial algorithm (unlike vanilla (perceptron) and use Lagrange multipliers to solve the optimization.

Q3 max margin hyperplane

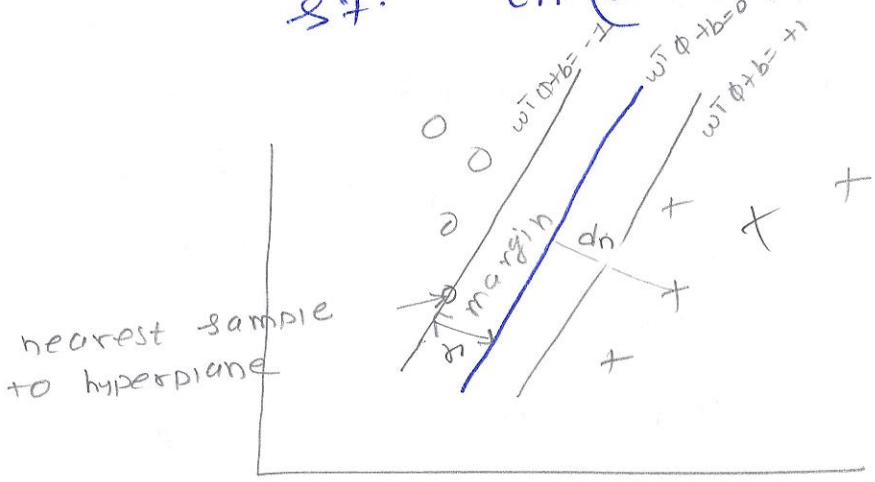
20/20

For the linearly separable dataset, the constrained optimizer for SVM is given by

$$\min \; J(w,b) = \frac{1}{2}\|w\|^2$$

(hard margin SVM)

$$\text{s.t.} \quad t_n\left(w^T \phi(x_n) + b\right) \geq 1 \quad \forall n \in \{1, \dots, N\}$$



distance to any sample,

$$d_n = \frac{t_n\left(w^T \phi(x_n) + b\right)}{\|w\|}$$

from slides, (lecture 07, page 5)

$$\text{margin} = \min_n \frac{t_n\, w^T \phi(x_n) + b}{\|w\|}$$

we find parameters w and b that maximizes margin,

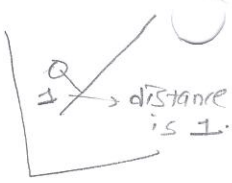
$$w^*, b^* = \underset{w,b}{\text{argmax}} \; \frac{1}{\|w\|} \cdot \min_n \; t_n\left(w^T \phi(x_n) + b\right)$$

Since w^*, b^* are obtained from argmax, they do not depend on rescaling of w, b.

So, without loss of generality, we can say that the nearest distance to the sample from hyperplane is 1.

i.e. for closest point(s),

$$t_n(w^T \phi(x_n) + b) = 1$$



then, our optimizing constraint becomes,

$$t_n(w^T \phi(x_n) + b) \geq 1 \quad \forall n \{1, \cdots, N\}$$

[Aside: previously we used

$t_n(w^T \phi + b) = 1$

$t_n h_n = 1$]

(for perceptron)

$w^T \phi(x_n) + b > 0$ if this +1

$w^T \phi(x_n) + b < 0$ if this −1

$h_n = \text{sgn}(w^T \phi(x_n) + b)$

now, with inclusion of this constraint, the new optimization problem for SVM becomes,

minimize

$$J(w, b) = \frac{1}{2} \|w\|^2$$

$$\text{s.t.} \quad t_n(w^T \phi(x_n) + b) \geq 1 \quad \forall n \in \{1, \cdots N\}$$

here we have use the margin $\gamma = 1$.

we have to show that if $\gamma \neq 0$, then also the decision hyperplane is unchanged.

Solution :

the distance to the nth sample from the decision hyperplane is,

$$d_n = \frac{t_n(w^T \phi(x_n) + b)}{\|w\|} \quad \} \text{ —— (1)}$$

if we rescale the weight vector to 1, $\|w\| = 1$, then,

$$d_n = t_n(w^T \phi(x_n) + b) \quad \} \text{ —— (2)}$$

then, maximizing margin algorithm becomes,

$$\max_{w,b} \min_n d_n$$

$$\text{s.t. } \|w\| = 1 \quad \} \text{ —— (3)}$$

let γ be the maximum margin to closest training example,

$$\forall_n \quad t_n(w^T \phi(x_n) + b) \geq \gamma$$

$$\Rightarrow \quad t_n\left(\frac{w^T}{\gamma}\phi + \frac{b}{\gamma}\right) \geq 1$$

$$\Rightarrow \quad t_n(V^T \phi + B) \geq 1 \quad \} \text{ —— (4)}$$

where $V = \frac{w}{\gamma}$, $B = \frac{b}{\gamma}$, γ = margin to the closest example.

then, $V = \frac{w}{\gamma}$

$$\|V\|^2 = \left\|\frac{w}{\gamma}\right\|^2 = \frac{1}{\gamma^2}\|w\|^2$$

$$\boxed{\|V\|^2 = \frac{1}{\gamma^2}} \quad \text{since } \|w\| = 1$$

Hence, our original optimization problem,

$$\max_{w,b} \min_n d_n = t_n(w^T \phi(x_n) + b)$$

reduces to maximizing $\frac{1}{\gamma^2}$

i.e. minizing $\|V\|^2$

i.e. minizing $\frac{1}{2}\|V\|^2$

which we can write as,

$$\min \frac{1}{2}\|V\|^2$$

$$s.t. \quad t_n(V^T \phi(x_n) + B) \geq 1. \quad \text{(from 4)}$$

which is same as,

$$\min J(w,b) = \frac{1}{2}\|w\|^2$$

$$s.t. \quad t_n(w^T \phi(x_n) + b) \geq 1 \quad \forall n \in \{1, N\}$$

Q.E.D.

Q N 4 a 30/40 product of two kernels is a kernel.

let ϕ_1, ϕ_2 are the feature map of kernels K_1, K_2 then,

20/20

$$K_1(x_1, x_2)\, K_2(x_1, x_2) = (\phi_1(x_1)\, \phi_1(x_2))\, (\phi_2(x_1)\, \phi_2(x_2))$$

$$= \left(\sum_{i=1}^{\infty} f_i(x_1)\, f_i(x_2)\right) \left(\sum_{j=1}^{\infty} g_j(x_1)\, g_j(x_2)\right)$$

$$= \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} f_i(x_1)\, f_i(x_2)\, g_j(x_1)\, g_j(x_2)$$

$$= \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} (f_i(x_1)\, g_j(x_1))\, (f_i(x_2)\, g_j(x_2))$$

$$= \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} h_{i,j}(x_1)\, h_{i,j}(x_2)$$

where $h_{i,j}(x)$ is a feature vector in feature space ϕ_3 such that

$$h_{i,j}(x) = f_i(x)\, g_j(x)$$

so, $K_1(x_1, x_2) \cdot K_2(x_2, x_3) = \phi_3(x_1)^T \phi_3(x_2)$

is a valid kernel.

~~Q4b~~ $x^T A y$ is valid kernel if A is sym PSD matrix.

Soln: Given $A \rightarrow$ symmetric positive semi-define

$A^T A$ is also sym PSD (from internet Shillan's reference)

Now,

Let, $K(x,y) = x^T A^T A y$

$= (Ax)^T (Ay)$ $\because (AB)^T = B^T A^T$

$= \phi(x)^T \phi(y)$

where we define feature map

$\phi(3) = A3$

$\therefore K(x,y)$ is a valid kernel.

Q.NO 5.a

20/20

Diagonal matrix with positive elements is posisitve definite matrix.

soln: Let $W = W_{n,n} \in \mathbb{R}^{n \times n}$ is a diagonal matrix then.

$$W = \begin{bmatrix} w_{11} & & 0 \\ & w_{22} & \\ 0 & & \ddots \; w_{nn} \end{bmatrix}_{n,n}$$

Let $\vec{x}$ be a n-dimensional column vector,

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}_{n,1}$$

then, the inner product of diagonal matrix W associated with $\vec{x}$ is,

$$x^T W x = \underset{1,n}{[x_1 \; x_2 \cdots x_n]} \begin{bmatrix} w_{11} & & 0 \\ & \ddots & \\ 0 & & w_{nn} \end{bmatrix}_{n,n} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}_{n,1}$$

$$= \sum_{j=1}^{n} w_j x_j^2$$

where $w_j = w_{ii}$ (diagonal elements)

$$x^T W x = \sum_{j=1}^{n} w_j x_j^2 \quad = +ve \text{ number}$$

here all the diagonal elements w_j are positive ($0 < w_{ii} < \infty \quad \forall\, i \in (1, n)$).

then the RHS quantity is positive

and hence W is a positive define matrix.

Alliter: $W = \text{diag}(w_{11}, w_{22}, \ldots, w_{nn})$, $w_{ii} > 0$

let $\sqrt{W} = \text{diag}(\sqrt{w_{11}}, \sqrt{w_{22}}, \ldots, \sqrt{w_{nn}})$

$$x^T W x = (x^T \sqrt{W}^T)(\sqrt{W} x)$$
$$= (\sqrt{W} x)^T (\sqrt{W} x)$$
$$= \| \sqrt{W} x \|^2 \geq 0$$

$\therefore$ W is PSD.

Ans 6 Sum of two PSD matrices is PSD matrix.

Let W is a positive semi-definite matrix

$\Rightarrow x^T W x \geq 0 \quad \forall x \in \mathbb{R}^n$

Let V is a positive semi-def matrix

$\Rightarrow x^T V x \geq 0$

now, $x^T W x + x^T V x \geq 0$

or, $x^T (W+V) x \geq 0$ (distributive property of matrix)

this means that sum of two matrices $W+V$ is also positive semi-definite.

proved

Q1.6 $w = u + \alpha(x-y)$

then, $J(w) = \frac{1}{2}\|w-u\|^2$

$= \frac{1}{2}\|\alpha(x-y)\|^2$

constraint: $w^T(x-y) = u^T(x-y) + \alpha\|x-y\|^2 \geq 1$

for, $\alpha\|x-y\|^2 \geq 1 - u^T(x-y)$

$\alpha \geq \frac{1-u^T(x-y)}{\|x-y\|^2}$

Then, optimization problem:

$\min_{\alpha} J(\alpha) = \frac{1}{2}\alpha^2\|x-y\|^2$

s.t. $\alpha \geq \frac{1-u^T(x-y)}{\|x-y\|^2}$

Again, $J(\alpha)$ will be minimum when α will be minimum.

But $\min \alpha = \frac{1-u^T(x-y)}{\|x-y\|^2}$

so, $\alpha = \frac{1-u^T(x-y)}{\|x-y\|^2}$

then, the w is given by (eliminate α)

$w = u + \alpha(x-y)$

$$w = u + \frac{1-u^T(x-y)}{\|x-y\|^2}(x-y)$$

Ans

Q6 Lagrange multipliers

Here, the optimization problem is,

$$\underset{w}{\text{minimize}} \quad J(w) = \frac{1}{2}\|w-u\|^2$$

$$\text{s.t.} \quad w^T(x-y) \geq 1$$

$$wx - wy \geq 1$$

$$1 - wx + wy \leq 0$$

primal Lagrangian

$$L_p(w, x, y, \alpha) = \frac{1}{2}(w-u)^2 + \alpha(1 - wx + wy)$$

$\alpha \geq 0$ is Lagrange multiplier

Set gradients to zero

$$0 = \frac{\partial L_p}{\partial w} = w - u - \alpha x + \alpha y \Rightarrow w = u + \alpha x - \alpha y = u + \alpha(x-y)$$

$$0 = \frac{\partial L_p}{\partial \alpha} = 1 - wx + wy \Rightarrow 1 = w(x-y) \Rightarrow x - y = \frac{1}{w}$$

then, $w = u + \frac{\alpha}{w} \Rightarrow w^2 = uw + \alpha \Rightarrow w^2 - u \cdot w - \alpha = 0$

$$w^* = \frac{u \pm \sqrt{u^2 + 4\alpha}}{2}$$

This is NOT solution look LHS

In gradient descent algorithm we initialize $\vec{u} = 0$ and $\alpha \geq 0$ and find the optimum value of w^*.

Bhishan poudel

HW Assignment 8 (Due by 10:30am on Dec 7)

1 Theory (150 points)

1. [**Kernel Nearest Neighbor**, 50 points]
   The nearest-neighbour classifier 1-NN assigns a new input vector $\mathbf{x}$ to the same class as that of the nearest input vector $\mathbf{x}_n$ from the training set, where in the simplest case, the distance is defined by the Euclidean metric $\|\mathbf{x} - \mathbf{x}_n\|^2$. By expressing this rule in terms of scalar products and then making use of kernel substitution, formulate the nearest-neighbour classifier for a general nonlinear kernel.

2. [**Distance-Weighted Nearest Neighbor**, 50 points]
   We have seen how to use kernels to formulate a distance-weighted nearest neighbor algorithm, when the labels are binary. Formulate a kernel-based, distance-weighted nearest neighbor that works for K classes, where $K \geq 2$.

3. [**Naive Bayes**, 50 points]
   The Naive Bayes algorithm for text categorization presented in class treats all sections of a document equally, ignoring the fact that words in the title are often more important than words in the text in determining the document category. Describe how you would modify the Naive Bayes algorithm for text categorization to reflect the constraint that words in the title are K times more important than the other words in the document for deciding the category, where K is an input parameter (include pseudocode).

4. [**Logistic Regression (*)**, 50 points]
   Assume that a binary feature x_i is equal to 1 for all training examples $\mathbf{x}$ belonging to a particular class C_k, and zero otherwise (i.e. x_i perfectly separates examples from class C_k from all other examples). Show that in this case the magnitude of the ML solution for $\mathbf{w}_k$ goes to infinity, thus motivating the use of a prior over the parameters (Hint: use the fact that the gradient on slide 24 must vanish at the solution).

2 Submission

Turn in a hard copy of your homework report at the beginning of class on the due date. On this theory assignment, **clear and complete explanations and proofs of your results are as important as getting the right answer.**

HW08

Bishan Poudel

Q.N.1 K-Nearest Neighbor (kernel based classifier)

(a) 1 nearest neighbor

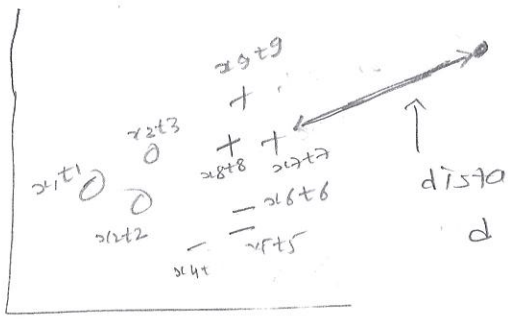
training dataset : $(x_1, t_1), (x_2, t_2), \cdots, (x_n, t_n)$

test dataset : x

required output : $\hat{y}$ (label of x or class of x)

$t_1, t_2, \cdots, t_n$ are classes from $c_1, c_2, \cdots, c_m$

for example { circle plus minus }
c_1 c_2 c_m



How to find $\hat{y}$?

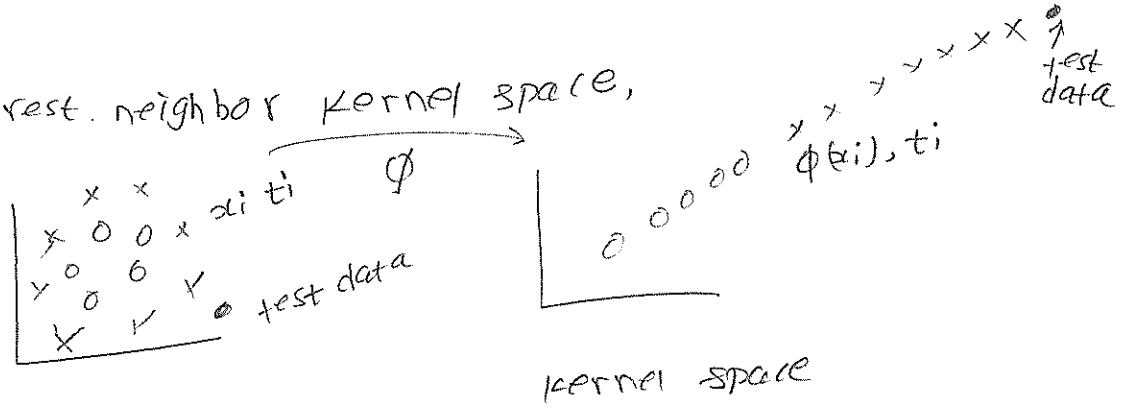
$\Rightarrow$ For 1-nearest neighbor, calculate the distance (maybe Euclidean, cosine, edit, etc) from test point $\vec{x}$ to all the data points x_i, $1 < i < n$ then find the minimum distance data point x'.

Assign label of x' as the predicted label to $\vec{x}$.

$\hat{y}$ = class of x'

= class of x_i such that $d(x, x_i)$ is minimum

① 1-nearest neighbor kernel space,



X-space

Kernel space

Here, when calculating distance, we use kernel method,

X-space: $d(x, x') = \|x - x'\| = \left|\sqrt{\|x - x'\|^2}\right| = \left|\sqrt{(x-x')^T (x-x')}\right|$

(Euclidean distance)

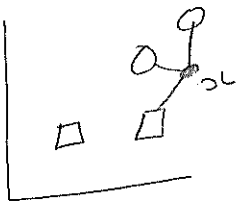
Kernel-space: $d(x, x') = \left|\sqrt{(\phi(x) - \phi(x'))^T (\phi(x) - \phi(x'))}\right|$

$= \left|\sqrt{K(x, x')}\right|$

→ we compute the distance of test $\vec{x}$ to the all the training examples x_i using kernel method then choose the minimum distance training point x' and then,

label of x = label of x'

⑥ K-nearest neighbor kernel method.



3-nearest neighbor
two are circle.
∴ label of x is circle.

Kernel ⇒ distances are calculated using kernel trick

$$d(x, x_i) = \sqrt{K(x, x_i)}$$

Kernel can be polynomial kernel, gaussian kernel and so on.

Let $x_1, x_2, \cdots, x_k$ are nearest of x

then,

$$y(x) = \underset{t \in T}{\operatorname{argmax}} \sum_{i=1}^{K} \delta_t(t_i)$$

Here, $t_i = \{$ circle, square $\}$

$K = 3$

for x_1 and x_2 $t = t_i =$ ○ number $= 2$

for x_3 $t = t_i =$ □ number $= 1$

argmax $= ○$

summary

input $(x_1, t_1), (x_2, t_2), \cdots (x_n, t_n)$ $t_n \in \{c_1, c_2, \cdots c_K\}$

test x, $t = ?$

map input from x-space to kernel-space

x-space input $\phi(x_1)$ $\phi(x_2)$ $\phi(x_n)$

kernel space test $\phi(x)$, $t = ?$

compute similarity $k(x, x_i)$ for all x_i, $1 \le i \le n$

(compute distance $= \sqrt{k(x, x')}$ is redundant since distance $\propto$ similarity)

then predicted class of x is,

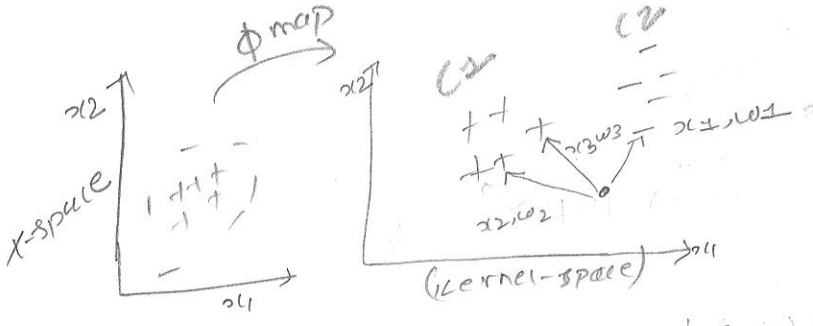
$$y(x) = \underset{t \in T}{\operatorname{argmax}} \sum_{i=1}^{k} \delta_t(t_i)$$

Q2 Distance-weighted Nearest Neighbor for classification

kernel space

(a) distance-weighted KNN binary classification

$t_i \in \{+1, -1\}$



3-nearest neighbor

contribution from +: $w_2 + w_3$

contribution from −: w_1

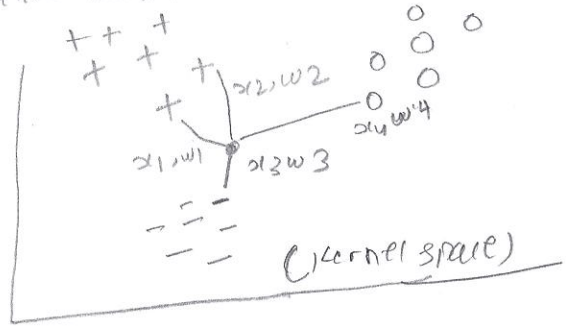
suppose $w_2 + w_3 > w_1$

then label of $x = +$

- x is test example
- $x_1, x_2, \dots, x_k$ are the nearest points to x, calculated using Euclidean distance in kernel space.
- $K(x, x_i)$ is the measure of similarity between point x and x_i in kernel space.

$$\hat{y} = \text{sign} \sum_{i=1}^{k} K(x, x_i)\, \delta_t(t_i)$$

(b) kernel-based distance weighted KNN for multiclass classification



Note: nearer points have more contribution

4-nearest neighbor

contribution from class $c_1 = +$
$= w_1 + w_2$

contribution from class $c_2 = -$
$= w_3$

contribution from class $c_3 = 0$
$= w_4$

suppose $w_1 + w_2 > w_3$ and $w_1 + w_2 > w_4$ $\Rightarrow \hat{y} = +$

$$\hat{y} = \underset{t \in T}{\text{argmax}} \sum_{i=1}^{k} K(x, x_i)\, \delta_t(t_i)$$

Ans

DW3 Naïve Bias

① simple text classification using Naive Bias

→ each document is an example vector

→ a document x has n words $w_1, w_2, \dots, w_n$

→ from all the documents $D = \{x_1, x_2, \dots, x_m\}$

we create vocabulary file (this is feature vector for whole dataset D)

→ vocabulary $V = \{w_1, w_2, \dots, w_{|V|}\}$, there are $|V|$ number of words in vocabulary.

→ each document x belongs to one of the catagories $C_K = \{c_1, c_2, \dots, c_K\}$

for example $D = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \text{line one} \\ \text{line two} \\ \text{line three} \end{bmatrix}$ (input data) $\begin{bmatrix} \text{news} \\ \text{sports} \\ \text{religion} \end{bmatrix}$ (target (class) label)

we want ⇒ given a document x find its catagory

$p(c_k|x)$ probability of document x belonging to catagory c_k (eg news)

Naive Bias gives ⇒

① $P(c_k)$ probability of each categories (priors)

② $P(w_i|c_k)$ conditional probability of data x w.r.t. category c_k.

then we use Bayes theorem to get $p(c^*|x)$.

$$p(c^*|x) = \frac{p(x|c^*)\,p(c^*)}{p(x)}$$

$$= p(c^*) \prod_{i=1}^{n} p(x_i|c^*)$$

where c^* corresponds to the $p(w_i|c_k)$ that has maximum probability

$$c^* = \underset{c_k}{\operatorname{argmax}}\; p(c_k|x)$$

NB for text classification when all words have equal importance

(A)
- D: Documents $x_1, x_2, \ldots x_n$
- $\{c_k\}$: classes $c_1, c_2, \ldots c_K$
- V: vocabulary (unique words in D − stopwords − below threshold words) $\{w_1, w_2, \ldots, w_V\}$
- D_k: subset examples having class c_k
- n_k: # of words in D_k

(B)
1. for each category c_k:
2. D_k = subset with category c_k
3. $P(c_k) = \frac{|D_k|}{|D|}$ (prior)
4. n_k = # words in D_k
5. for each word $w_i \in V$:
6. let n_{ki} = # of w_i in D_k
7. set $p(w_i|c_k) = \frac{n_{ki} + 1}{n_k + |V|}$ ← Laplace smoothing
   (cond prob)
8. Return priors $P(c_k)$ and cond prob $p(w_i|c_k)$

(C)
Then we find the posterior probability for each class c_k,

$$P(c_k|x) = \frac{p(x|c_k) \cdot P(c_k)}{p(x)}$$

$$\propto p(x|c_k) P(c_k)$$

$$\propto P(c_k) \cdot \prod_{i=1}^{n_V} p(w_i|c_k)$$ (w_i are words in test document x)

$$\propto \ln \left[P(c_k) \cdot \prod_i p(w_i|c_k) \right]$$

$$\propto \ln P(c_k) + \sum_i \ln (p(w_i|c_k))$$

(to prevent underflow)

(D)
Testing

$$c^* = \arg\max_{c_k} p(c_k|x)$$

$$c^* = \arg\max_{c_k} P(c_k) \prod_i p(w_i|c_k)$$

$$\propto \arg\max_{c_k} \left[\ln P(c_k) + \sum_i \ln (p(w_i|c_k)) \right]$$

Example of Naive Bias

train	Doc	words	class	Total
	1	chinese Beiging chinese	$c_1=c$	$n_1=8$ words $d_1=3$ examples $n_1+\lvert V\rvert = 8+6=14$
	2	chinese chinese shanghai	c_1	
	3	chinese macao	c_1	
	4	Tokyo Japan chinese	c_2	$n_2=3$ words $d_2=1$ example $n_2+\lvert V\rvert = 3+6=9$
test	1	chinese Chinese chinese Tokyo Japan ($\vec{x}$)	?	11 total words 6 unique words $\lvert V\rvert = 6$

vocabulary : { chinese (w_1), Beiging (w_2), shanghai (w_3), macao (w_4), Tokyo (w_5), Japan (w_6) } $\lvert V\rvert = 6$

	w_1	w_2	w_3	w_4	w_5	w_6
word freq in c_1 :	5	1	1	1	0	0
word freq in c_2 :	1	0	0	1	1	1
Laplace smoothing :	1	1	1	1	1	1
$\frac{n_{ki}+1}{n_k+\lvert V\rvert}$:	6, 2 14, 9	2 14, 9	1 14, 9	2 14, 9	1, 2 14, 9	1, 2 14, 9

② category $c_1=c$

- prior $P(c_1) = \lvert D_1\rvert / \lvert D\rvert = 3/4$ (3 documents belongs to class c_1)
- number of words in class, $n_1 = 8$
- conditional probabilities $\left(P(w_i \mid c_k) = \frac{n_{ki}+1}{n_k+\lvert V\rvert}\right)$

Chinese $P(w_1 \mid c_1) = \frac{5+1}{8+6} = \frac{6}{14} = \frac{3}{7}$ (n_{ki} → number of word w_i in class c)

Beiging $P(w_2 \mid c_1) = \frac{1+1}{14} = \frac{2}{7}$

shanghai $P(w_3 \mid c_1) = \frac{1+1}{14} = \frac{2}{7}$

macao $P(w_4 \mid c_1) = \frac{1+1}{14} = \frac{2}{7}$

Tokyo $P(w_5 \mid c_1) = \frac{0+1}{8+6} = \frac{1}{14}$

Japan $P(w_6 \mid c_1) = \frac{0+1}{14} = \frac{1}{14}$

category $c_2=j$

- prior $P(c_2) = \lvert D_2\rvert / \lvert D\rvert = 1/4$ (1 document belongs to class c_2)
- $n_2 = 3$
- conditional probabilities

$P(w_1 \mid c_2) = \frac{1+1}{3+6} = \frac{2}{9}$

$P(w_2 \mid c_2) = \frac{0+1}{3+6} = \frac{1}{9}$

$P(w_3 \mid c_2) = \frac{1}{9}$

$P(w_4 \mid c_2) = \frac{1}{9}$

$P(w_5 \mid c_2) = \frac{2}{9}$

$P(w_6 \mid c_2) = \frac{2}{9}$

③ choosing class

prior chinese tokyo japan

$P(c_1 \mid x) \propto P(c_1) \prod_{i \in c_1} P(w_i \mid c_1) = \frac{3}{4} \cdot \left(\frac{3}{7}\right)^3 \cdot \frac{1}{14} \cdot \frac{1}{14} = 0.0003$

$P(c_2 \mid x) \propto P(c_2) \prod_{i \in c_2} P(w_i \mid c_2) = \frac{1}{4} \cdot \left(\frac{2}{9}\right)^3 \cdot \frac{2}{9} \cdot \frac{2}{9} = 0.0001$

$c^* = \underset{c_k}{\operatorname{argmax}}\ P(c_k) \prod_{j=1}^{n} P(w_i \mid c_k) = \underset{c_k}{\operatorname{argmax}} \{0.0003, 0.0001\} =$ label of $0.0003 = c_1$

P.T.O.

⑤ give more importance to title of doc

① for each category c_k :

② bindclass prior $P(c_k) = |D_k| / |D|$ (where D_k is subset of D with class c_k)

③ # loop through title words
let n_k = number of title words in class c_k all examples
for each titleword $w_i \in V_1$:
let n_{ki} = number of words in title of all class c_k examples
. . . . # bind conditional prob of each title words
$P_1(w_i | c_k) = \frac{n_{ki}+1}{n_k + |V_1|} \times K$ ← multiplication factor for title words

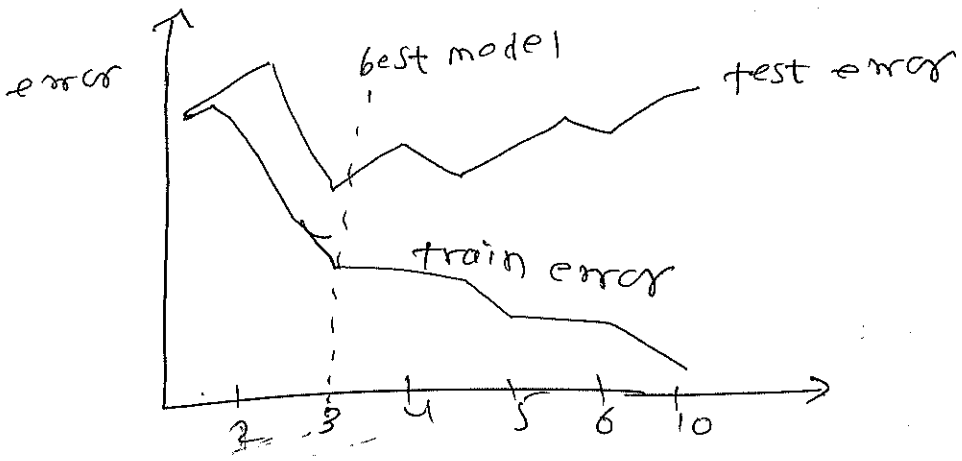
④ # loop through body words
let n_k = number of body words
for each bodyword $w_i \in V_2$:
let n_{ki} = number of words in body of all class c_k examples (for example c_k is class 1)
bind conditional prob of each body words
$P_2(w_i | c_k) = \frac{n_{ki}+1}{n_k + |V_2|}$

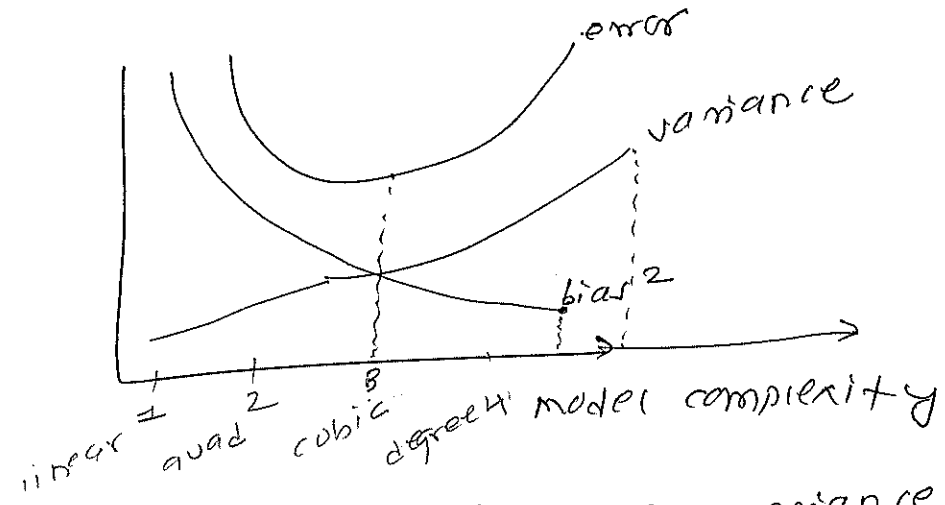
⟹ output $P(c_k)$, $P_1(w_i|c_k)$, $P_2(w_i|c_k)$

choosing class:
we again suppose title words and body words are independent,

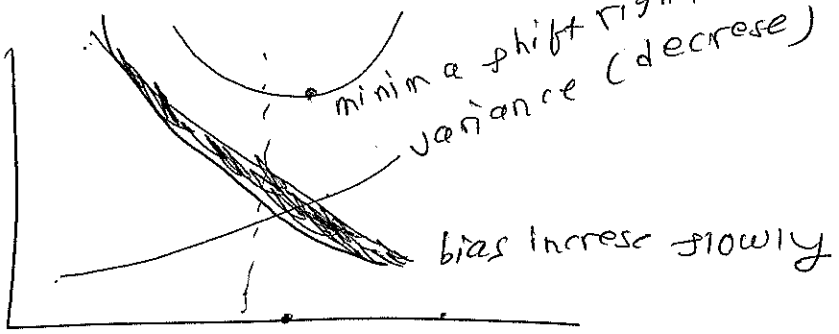
$P(c_k | x) \propto \underbrace{P_1(x|c_k)P(c_k)}_{\text{for title}} \cdot \underbrace{P_2(x|c_k)P(c_k)}_{\text{for body}}$

overfitting





when training data increase, variance is reduced and slightly more compricated model minimizes the expected error.



① Show that $x^T A y$ is a valid kernel if A is sym. psd.

$\Rightarrow A = Q^T \Lambda Q$ where $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ and $\lambda_1, \lambda_2, \ldots \geq 0$

define $F = \text{diag}(\sqrt{\lambda_1}, \sqrt{\lambda_2}, \ldots, \sqrt{\lambda_n})$ then $\Lambda = F^T F$

$\Rightarrow A = Q^T F^T F Q$

$\Rightarrow x^T A y = x^T Q^T F^T F Q y$

$= (FQx)^T (FQy)$

$= \phi(x)^T \phi(y)$

where $\phi(x) = FQx$

Aside:

$(AB)^T = B^T A^T$

$(ABC)^T = C^T B^T A^T$

$((AB)C)^T = C^T (AB)^T$

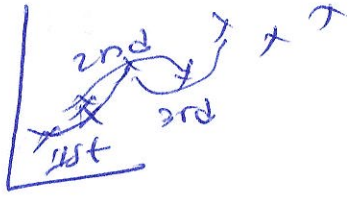
$= C^T B^T A^T$

⑥ cubic spline smoothing

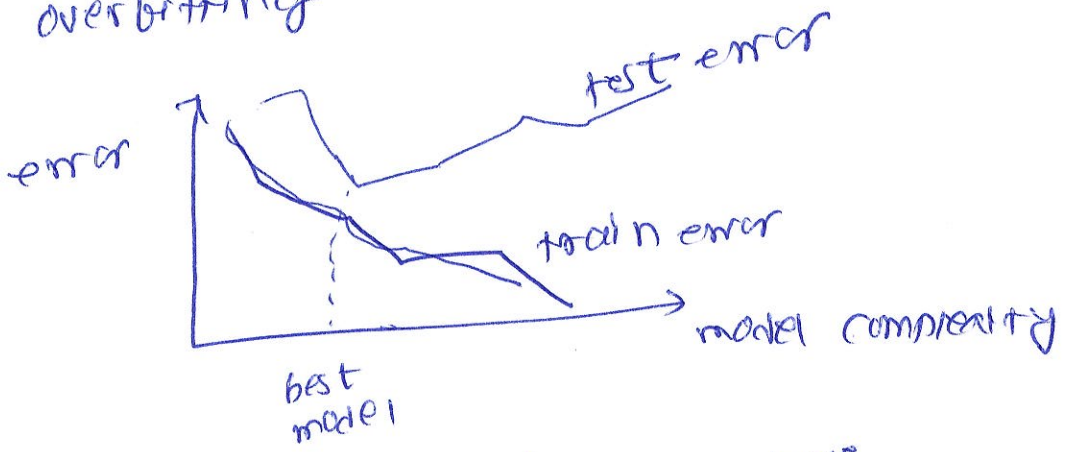
take 3 points x_{i-1} x_i x_{i+1}

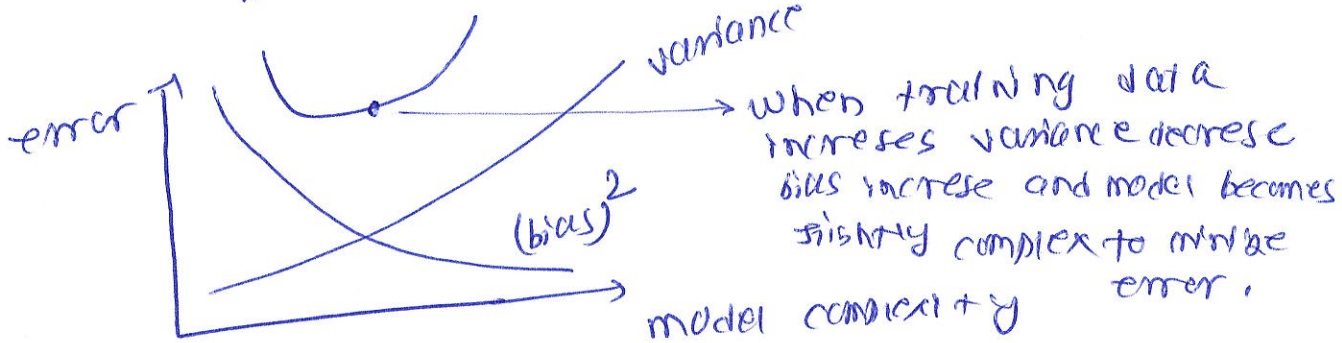
$$s_i(x) = a_i (x-x_i)^3 + b_i (x-x_i)^2 + c_i (x-x_i) + d_i$$

$$\forall x \in [x_i, x_{i+1}]$$



⑦ overfitting





① Likelihood function for Logistic Regression

$p(t_n|w) = h_n^{t_n} \cdot (1-h_n)^{1-t_n}$

$p(t|w) = \prod_{n=1}^{N} h_n^{t_n} (1-h_n)^{1-t_n}$

$$\text{cost } E = \frac{1}{N} \cdot (-\ln p) = -\frac{1}{N} \sum_{n=1}^{N} \left[t_n \ln h_n + (1-t_n) \ln(1-h_n) \right]$$

$$\text{grad} = \nabla_w J = \frac{1}{N} \cdot \sum_{n=1}^{N} (h_n - t_n) \cdot x_n$$

initialize $\bar{w} = 0, \tau = 1$

for $i = 1 : N$

$h_n = \text{sgn}(w^T x_n)$

if $h_n \neq t_n$ then

$w = w + t_n x_n$

$\tau = \tau + 1$

$\bar{w} = \bar{w} + w$

$\tau = \tau + 1$

return $\bar{w} / T$

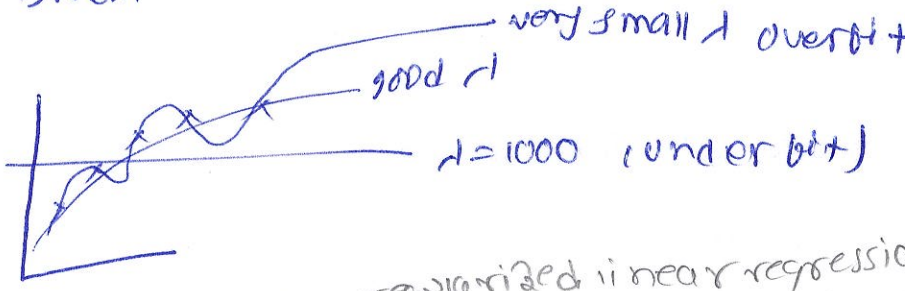
Test: $\text{sgn}(w^T x)$

② perceptron criterion

minimize $E_p(w) = -\sum_{n \in \text{Misclassified}} t_n w^T x_n$

① L2 → more penalty larger weights
does not drive weights to zero
weight vector is NOT sparse.

② $\lambda \uparrow \Rightarrow$ high bias, low variance, underfit, simpler model
drives weights closer to zero.
small change in training data $\Rightarrow$ big change in estimate



cost function for L2-regularized linear regression

N-examples
M-features

$$X = \begin{bmatrix} x_1^1 & x_1^2 & x_1^3 & x_1^M \\ x_2^1 & x_2^2 & x_2^3 & x_2^M \\ x_N^1 & x_N^2 & x_N^3 & x_N^M \end{bmatrix}$$

③ $$J = \frac{1}{2N} \sum_{n=1}^{N} (h_n - t_n)^2 + \frac{\lambda}{2} \|w\|^2$$

$$J(w) = \frac{1}{2N} \sum_{n=1}^{N} \left(w_0 + \sum_{j=1}^{M} w_j x_n^j - t_n \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{M} w_j^2$$

(we omit w_0 in regularization)

regularizing w_0 = shifting origin of target
$\Rightarrow$ same change in all target values $\not\Rightarrow$ similar change in estimates

④ Batch GD

for num-iters:

$$w^{\tau+1} = w^{\tau} - \eta \nabla J$$

η → learning rate

stochastic GD

for num-iters:
for sample in data:

$$w^{\tau+1} = w^{\tau} - \eta \nabla J$$

momentum GD

$$v^{\tau+1} = \eta \nabla J(w^{\tau})$$
$$w^{\tau+1} = w^{\tau} - v^{\tau+1}$$

$\gamma \approx 0.9$

Nesterov Accelerated gradient

$$w^{\tau+1} = w^{\tau} - \eta \nabla J(w^{\tau} - \gamma w^{\tau}) - \gamma w^{\tau}$$

(5) Gradient descent → $v^{\tau+1} = \begin{cases} \eta \nabla J(w^\tau) & \text{vanilla} \\ \eta \nabla J(w^\tau) + \gamma v^\tau & \text{with momentum} \\ \eta \nabla J(w^\tau - \gamma v^\tau) + \gamma v^\tau & \text{Nesterov} \end{cases}$

$w^{\tau+1} = w^\tau - v^{\tau+1}$

η = learning rate
γ = momentum hyperparameter
w^τ = weight vector at time stamp τ

$\gamma \approx 0.9$

Accelerated Gradient NAG

(6b) Types of GD based on data used:

Batch GD → use all data

stochastic GD → use only one example and update w after each iteration

minibatch GD → use constant number of examples and update w after each

$$J = \frac{1}{2N} \sum_{n=1}^{N} (h_n - t_n)^2$$

$$\nabla J = \frac{1}{N} \sum_{n=1}^{N} (h_n - t_n) \cdot x_n = \frac{1}{N} \cdot np.sum((h-t) \cdot x)$$

$$w = w - \eta \, grad$$

(5c) GD vs Normal eqn

GD: $w = w - \eta \, grad$

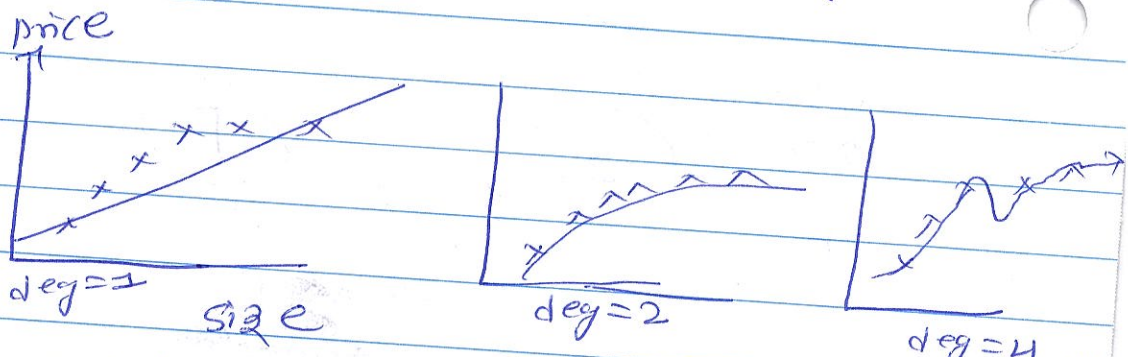
Normal eqn: $w = \underbrace{(x^T x)^{-1} x^T t}_{\text{pseudo inverse (pinv)}}$

$$w = \underbrace{(x^T x + \lambda N I)^{-1} x^T t}_{\text{for } L2 \text{ regularized}}$$

I is identity matrix of shape $x^T x$

$I[0][0] = 0$ for not to regularize bias term

Bias-variance Trade off



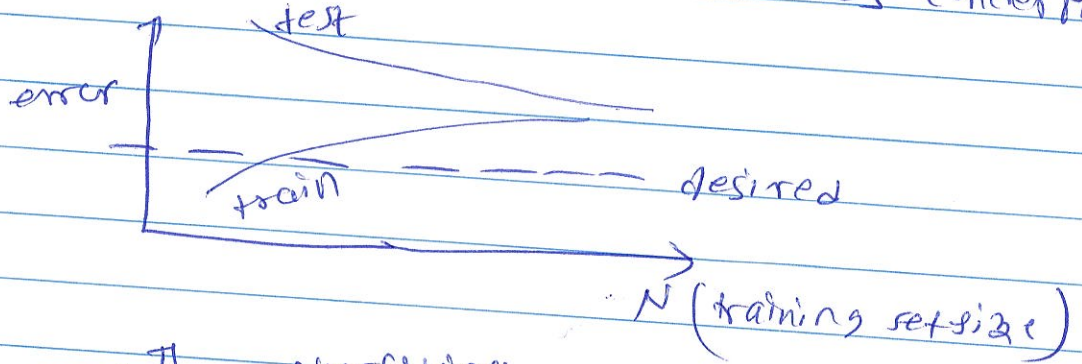
$w_0 + w_1 x$
high bias
(underfit)
(simple model)

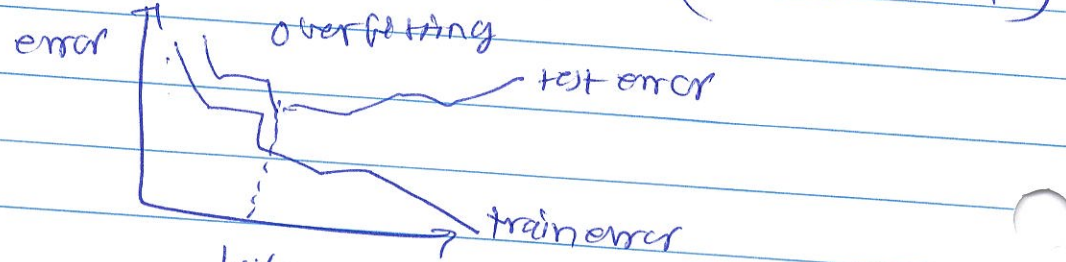
$w_0 + w_1 x + w_2 x^2$
just right

high var.
overfit)

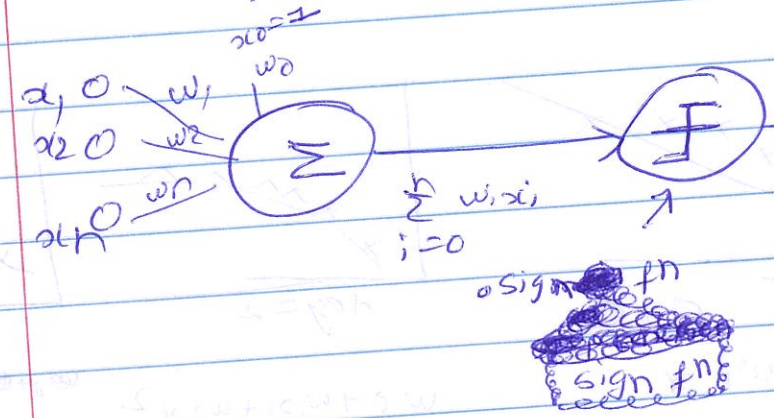
more training eg → fix high σ^2 (overfitting)
smaller feature set → fix high σ^2 (")
larger featureset → fix high bias (underfitting)

Learning curve for high bias (underfit)





perceptron



$$o = \begin{cases} 1 & \text{if } \sum_{i=0}^{n} w_i x_i > 0 \\ -1 & \text{otherwise} \end{cases}$$

$$= \begin{cases} 1 & \text{if } \vec{w} \cdot \vec{x} > 0 \\ -1 & \text{otherwise} \end{cases}$$

① cost for multivariate linear regression

$$J = \frac{1}{2N}\sum_{n=1}^{N}(h_n - t_n)^2$$

$$J = \frac{1}{2N}\sum_{n=1}^{N}\left(\sum_{j=0}^{M} w_j x_n^j - t_n\right)^2$$

$$h = w^T X \qquad h = X_1 @ w.T$$

$$h_n = \sum_{j=0}^{M} w_j x_n^j$$

$$h_1 = w_0 + w_1 x_1^{(1)} + w_2 x_1^{(2)} + w_M x_1^{(M)}$$ (first row of sample)

$$X_1 = \begin{bmatrix} 1 & 10 & 100 \\ 1 & 20 & 200 \\ 1 & 30 & 300 \\ 1 & 50 & 500 \end{bmatrix}_{50\times 4} \qquad t = \begin{bmatrix} 11K \\ 21K \\ 31K \\ 50K \end{bmatrix}_{50\times 1}$$

$$w = \begin{bmatrix} w_0 & w_1 & w_2 & w_3 \end{bmatrix}_{1\times 4}$$ (w_0: bias)

2 features: age, floorsize

$$h = (50,4)\cdot(4,1) = (50,1)$$

$$e = h - t = \begin{bmatrix} h_1 - t_1 \\ h_2 - t_2 \\ \vdots \\ h_{50} - t_{50} \end{bmatrix}_{50\times 1}$$

$$SSE = (h-t) ** 2$$

$$MSE = \frac{(h-t)**2}{N}$$

$$MSE = np.mean(SSE)$$

$$RMSE = np.sqrt(MSE)$$

$$J = np.sum((h-t)**2)/2/N$$

$$J = 0.5 * np.mean(np.sum((h-t)**2))$$

$$J = \frac{1}{2}\times MSE \quad (\text{single float})$$

$$\boxed{J = \frac{0.5}{len(t)}(h-t)^T(h-t)}$$

$$(h-t)^T(h-t) = \underset{1\times 50}{[h_1-t_1 \;\; h_2-t_2 \;\cdots]} \begin{bmatrix} h_1-t_1 \\ h_2-t_2 \\ \vdots \\ h_N-t_N \end{bmatrix}_{50\times 1} = \left[\sum_n (h_n-t_n)^2\right]_{1\times 1}$$

② gradient of J

$$\nabla_w J = \nabla_w \frac{1}{2N}\sum_n (h-t)^2 = \begin{bmatrix} \frac{\partial J}{\partial w_0} & \frac{\partial J}{\partial w_1} & \frac{\partial J}{\partial w_2} & \frac{\partial J}{\partial w_M} \end{bmatrix}_{4\times 1}$$

$$= (h-t).T @ X_1 \;/N$$

$$= (1,50)\cdot(50,4)$$

$$grad = \nabla_w J = (1,4)$$

$$\boxed{grad\text{-}ols = (h-t).T @ X_1}$$

$$\frac{\partial J}{\partial w_j} = \frac{1}{2N}\sum_{n=1}^{N}(x_n w_j - t)^2 = \frac{1}{2N}\sum_{n=1}^{N} 2(x_n w_j - t)\cdot x$$

$$= \frac{1}{2N}\sum_{n=1}^{N} 2(xw_j - t)\cdot x$$

[summation vanishes] we use matrix

$$\nabla_w J = \frac{1}{N}(h-t)\cdot X$$

④ X = design matrix $[\ ]_{N \times M}$ N = samples, m = features

X_1 = biased design matrix $[\ ]_{N, M+1}$

$t = [\ ]_{N \times 1}$ column vector

$w = [w_0\ w_1\ w_2\ w_M]_{1, M+1}$ row vector (2d array)

$w = np.array(w).reshape(1, X_1.shape[1]) = (1, M+1)$

linear regression

$h = X_1 @ w.T = (N, M+1)(M+1, 1) = (N, 1)$ same as t

$e = h - t = N, 1 = 50, 1$

$SSE = \sum_{n=1}^{N} (h-t)^2$

$MSE = \frac{1}{N} \sum_{n=1}^{N} (h-t)^2$

$J = \frac{1}{2N} \sum_{n=1}^{N} (h-t)^2 = np.sum((h-t)**2) / 2 / float(N)$

$RMSE = E_{RMS} = \sqrt{MSE}$

Normal eqn: $w = \underbrace{(X^T X)^{-1} @ X^T}_{\text{Moore-Penrose pseudoinverse of } X} @ t$

np.linalg.pinv(X)

multivariate linear

bias	X		t
	feature size	bed	
1	100	2	2K
1	200	3	3K
1	500	5	5K

t //

m features and one bias

$$X = \begin{bmatrix} x_0^{(1)}=1 & x_1^{(1)} & \cdots & x_m^{(1)} \\ x_0^{(2)} & x_1^{(2)} & \cdots & x_m^{(2)} \\ \vdots & \vdots & & \\ x_0^{(N)} & x_1^{(N)} & & x_m^{(N)} \end{bmatrix}_{N \times m+1} \quad \begin{bmatrix} t_1 \\ t_2 \\ t_3 \\ \vdots \\ t_N \end{bmatrix}_{N \times 1}$$

There are m features and one bias total m+1
There are N samples for each feature

$$J = \frac{1}{2N}|Xw - t|^2 = \frac{1}{2N}(Xw-t)^T(Xw-t)$$

$$\nabla_w J = \frac{1}{2} \frac{d}{dx}(Xw-t) \cdot 2(Xw-t)$$

$$\frac{d}{dx}(x^T x) = 2x$$

x is a matrix

$$0 = X^T(Xw-t)$$

$$X^T X w = X^T t$$

weights $$w = (X^T X)^{-1}(X^T t)$$

$$\frac{d}{dx}(Ax+b)^T(Ax+b) = 2A^T(Ax+b)$$

derivatives of product of matrix

L1 vs L2 norm

L2 more used

L2 → more penalty on large weights, but doesn't drive small weights to zero.

L1 → less penalty for large wt, but leads many weights to (or very very nose) to zero. leading to weight vector to be sparse.

Bias-variance Tradeoff

(a) Ridge: $\sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij}\right)^2$

$+ \lambda \sum_{j=1}^{p} \beta_j^2 = RSS + \lambda \sum_{j=1}^{p} \beta_j^2$

shrinkage parameter which has to be tuned via validation set → regularization strength

Ridge: $\lambda \uparrow \Rightarrow$ var ↓ bias ↑

meaning: small change in training data
big change in parameter estimates
effect will increase with no of parameters

(b) LASSO β_j^2 is $|\beta_j|$

Ridge shrinks wj but do not make 0
but LASSO makes them zero, makes less no of wtp.

Logistic Regression → classification (binary)

linear regression $h = w^T x$

logistic regression $h = \sigma(y) = \frac{1}{1+e^{-w^T x}} = h(x) = y(x)$

likelihood function $p(t|w) = \prod_n h_n^{t_n} (1-h_n)^{1-t_n}$

-ve log likelihood, E or $J = -\ln p(t|w)$

$$= -\sum_{n=1}^{N} [t_n \ln h_n + (1-t_n) \ln 1-h_n$$

∴ cost or Error or loss function

$$E_D = -\sum_n [t_n \ln h_n + (1-t_n) \ln (1-h_n)]$$

where $t_n \in \{0,1\}$ NOT $\{-1,1\}$

add regularizer,

$$E_w = \frac{\lambda}{2} w^T w$$

then L2 regularized logistic regression cost function

$$E = -\frac{1}{N} \sum_n [t_n \ln h_n + (1-t_n) \ln (1-h_n)] + \frac{\lambda}{2} w^T w$$

(cost) — this is regularizer

$$h(x) = \sigma(y) = \frac{1}{1+e^{-w^T x}}$$

$h_n = h(x_n) = \sigma(x_n) = \frac{1}{1+e^{-w^T x_n}}$ is sigmoid fn

Softmax Regression (multiclass)

training set: $(x_1, t_1), (x_2, t_2), \ldots (x_n, t_n)$

$X = [1, x_1, x_2, \ldots, x_n]$

$t_1, t_2, \ldots, t_n \in \{1, 2, \ldots, K\}$

$w_K = [w_{K0}, w_{K1}, w_{K2}, \ldots]^T$

one weight vector per class,

$$p(C_k|x) = \frac{e^{w_k^T x}}{\sum_j e^{w_j^T x}}$$

prob that nth example x_n has class t_n

$$p(t_n|x_n) = \frac{e^{w_{t_n}^T x_n}}{\sum_j e^{w_{t_j}^T x_n}}$$

likelihood is the joint probability of all classes,

$$L(w) = \prod_{n=1}^{N} p(t_n|x_n)$$

cost is the -ve log likelihood,

$$E_D(w) = -\frac{1}{N} \ln L(w)$$

$$= -\frac{1}{N} \ln \prod_{n=1}^{N} p(t_n|x_n)$$

$$= -\frac{1}{N} \sum_n \ln p(t_n|x_n)$$

$$= -\frac{1}{N} \sum_n \ln \frac{e^{w_{t_n}^T x_n}}{\sum_j e^{w_j^T x_n}}$$

w_{t_n} is weight vector for nth example

w_k is weight vector for all the examples belonging to class k

$$E_D(w) = -\frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} \delta_k(t_n) \ln \left(\frac{e^{w_k^T x_n}}{\sum_j e^{w_j^T x_n}} \right)$$

INLE

sample: $x_1, x_2, \ldots, x_N$

prob density of x is $p(x_n, w)$

joint prob of $x_1, x_2, \ldots, x_N$ is called likelihood $L(w)$

$L(w) = p(x_1, x_2, \ldots, x_N | w)$

$= p(x_1|w) p(x_2|w) \ldots$ (assuming independence)

$$L(w) = \prod_{n=1}^{N} p(x_n, w)$$

MAP solution for LR

$$p(t|w) = \prod_{n=1}^{N} y_n^{t_n} (1-y_n)^{1-t_n}$$

$$p(w) = \left(\frac{\alpha}{2\pi}\right)^{\frac{M+1}{2}} e^{-\frac{\alpha}{2} w^T w} \propto e^{-\frac{\alpha}{2} w^T w}$$

$$p(w|t) = \frac{p(t|w)\, p(w)}{p(t)} \propto p(t|w)\, p(w)$$

Now, $\vec{w}_{MAP} = \underset{w}{\operatorname{argmax}}\ p(w|t)$

$$= \underset{w}{\operatorname{argmax}}\ p(t|w)\, p(w)$$

$$= \underset{w}{\operatorname{argmin}}\ -\ln p(t|w) - \ln p(w)$$

$$= \underset{w}{\operatorname{argmin}}\ -\ln \prod_n y_n^{t_n} (1-y_n)^{1-t_n} - \ln e^{-\frac{\alpha}{2} w^T w}$$

$$= \underset{w}{\operatorname{argmin}}\ -\sum_n \ln\left(y_n^{t_n} (1-y_n)^{1-t_n}\right) + \frac{\alpha}{2} w^T w$$

$$= \underset{w}{\operatorname{argmin}}\ \underbrace{-\sum_n \left[t_n \ln y_n + (1-t_n)\ln(1-y_n)\right]}_{E_D(w)} + \underbrace{\frac{\alpha}{2} w^T w}_{E_W(w)}$$

$$\vec{w}_{MAP} = \underset{w}{\operatorname{argmin}}\ E_D(w) + E_W(w)$$

Ans

Softmax Regression

$$p(C_k|x) = \frac{e^{w_k^T x}}{\sum_j e^{w_j^T x}}$$

$$p(t_n|x_n) = \frac{e^{w_{tn}^T x_n}}{\sum_{j=1}^{N} e^{w_{tn}^T x_n}}$$

$$h(w_j) = \frac{1}{\sum_{k=1}^{K} e^{w_k^T x}} \begin{bmatrix} e^{w_1^T x} \\ e^{w_2^T x} \\ \vdots \\ e^{w_K^T x} \end{bmatrix}$$

$$E_D(w) = \frac{1}{N} \cdot -\ln \prod_n p(t_n|x_n)$$

$$= -\frac{1}{N} \ln \prod_n \frac{e^{w_{tn}^T x_n}}{\sum_j e^{w_{tn}^T x_n}}$$

$$= -\frac{1}{N} \cdot \sum_n \ln \frac{e^{w_{tn}^T x_n}}{\sum_j e^{w_{tn}^T x_n}}$$

$$\boxed{E_D(w) = -\frac{1}{N} \sum_n \sum_k \delta_k(t_n) \ln \left(\frac{e^{w_k^T x_n}}{\sum_k e^{w_k^T x_n}} \right)}$$

Regularizer

$$+ \frac{\alpha}{2} \sum_{k=1}^{K} w_k^T w_k$$

$$\frac{\partial E_w}{\partial w_k} = \alpha w_k$$

(no summation)

Let, $E_n = \sum_k \delta_k(t_n) w_k^T x_n - \sum_k \delta_k(t_n) \ln \left(\sum_k e^{w_k^T x_n} \right)$

$$\frac{\partial E_n}{\partial w_j} = \delta_j(t_n) x_n - \delta_j(t_n) \cdot \frac{1}{\sum_j e^{w_j^T x_n}} \cdot e^{w_j^T x_n} \cdot \delta_j(t_n) x_n$$

$$\frac{\partial E_n}{\partial w_k} = \delta_k(t_n) x_n - \frac{e^{w_k^T x_n}}{\sum_k e^{w_k^T x_n}} \cdot x_n$$

prevent overflow

$$p(C_k|x_n) = \frac{e^{w_k^T x_n}}{\sum_k e^{w_k^T x_n}}$$

$$c = \max_{i \in \{1,\dots,K\}} w_k^T x_n$$

$$p(C_k|x_n) = \frac{e^{(w_k^T x_n - c)}}{\sum_k \exp(w_k^T x_n - c)}$$

$$\frac{\partial E_n}{\partial w_k} = [\delta_k(t_n) - p(C_k|x_n)] x_n$$

$$\Rightarrow \boxed{\frac{\partial E_D(w)}{\partial w_k} = -\frac{1}{N} \sum_n [\delta_k(t_n) - p(C_k|x_n)] x_n}$$

there are K-such equations for each classes.

① MLE vs MAP

MLE = maximize $P(\text{data} | \text{params})$ by searching over parameters

MAP = maximize $P(\text{param} | \text{data})$ by searching over params and accounting for prior over params

MLE → finds w by maximizing likelihood $P(\text{data} | w)$

MAP → maximizes the posterior prob $P(w | d)$

② classification maps inputs to discrete outputs

Regression " " continuous ".

③ PCA & feature selection

Similarity: reduce the dimension of data

difference: feature selection finds a subset of features, PCA produces a smaller new set

perceptron

perceptron criterion: $w^T x_n > 0$ for $t_n = +1$

$w^T x_n < 0$ for $t_n = -1$

want: $t_n w^T x_n > 0$ for all patterns

minimize: $-w^T x_n t_n$ for all misclassified patterns M

$$E_P(w) = -\sum_{n \in M} w^T x_n t_n$$

E_P: perceptron; M: mistakes (misclassified)

Binary perceptron

initialize $\vec{w} = 0$

for $n = 1, \dots, N$

$h_n = \text{sgn}(w^T x_n)$

if $h_n \neq t_n$ then

$w = w + t_n x_n$

} repeat until convergence or given number of epochs

① Why Kernels are Symmetric?

Inner products are symmetric by definitions, so, therefore if the kernel function represent an inner product in some Hilbert Space, then the kernel function must be symmetric as well.

$$K(x,y) = \langle \phi(x), \phi(y) \rangle$$

$$= \langle \phi(y), \phi(x) \rangle \quad \text{(: property of inner product)}$$

$$= K(y,x)$$

② Show $K(x,z) = x^T A^T A z$ is a valid kernel.

Let $\phi(x) = Ax$,

then,

$$\langle \phi(x), \phi(z) \rangle = \phi(x)^T \phi(z)$$

$$= (Ax)^T (Az)$$

$$= x^T A^T A z$$

$$= K(x,z)$$

$\therefore K(x,z)$ is an inner product in some Hilbert Space.

(3) AA^T is PSD matrix (indirect)

Proof Let, $A \in \mathbb{R}^{m \times n}$

λ = eigen value of AA^T

q = eigen vector of λ

$(AA^T)q = \lambda q$ (premultiply by q^T)

$\Rightarrow q^T AA^T q = q^T \lambda q$ $\quad \because AA^T = A^T A$

$\Rightarrow \lambda = \frac{q^T AA^T q}{q^T q} = \frac{q^T A^T A q}{q^T q}$

$= (Aq)^T Aq / q^T q$

$= \frac{z^T z}{q^T q}$ where $z = A^T q$

here $z^T z \geq 0$

$q^T q \geq 0$

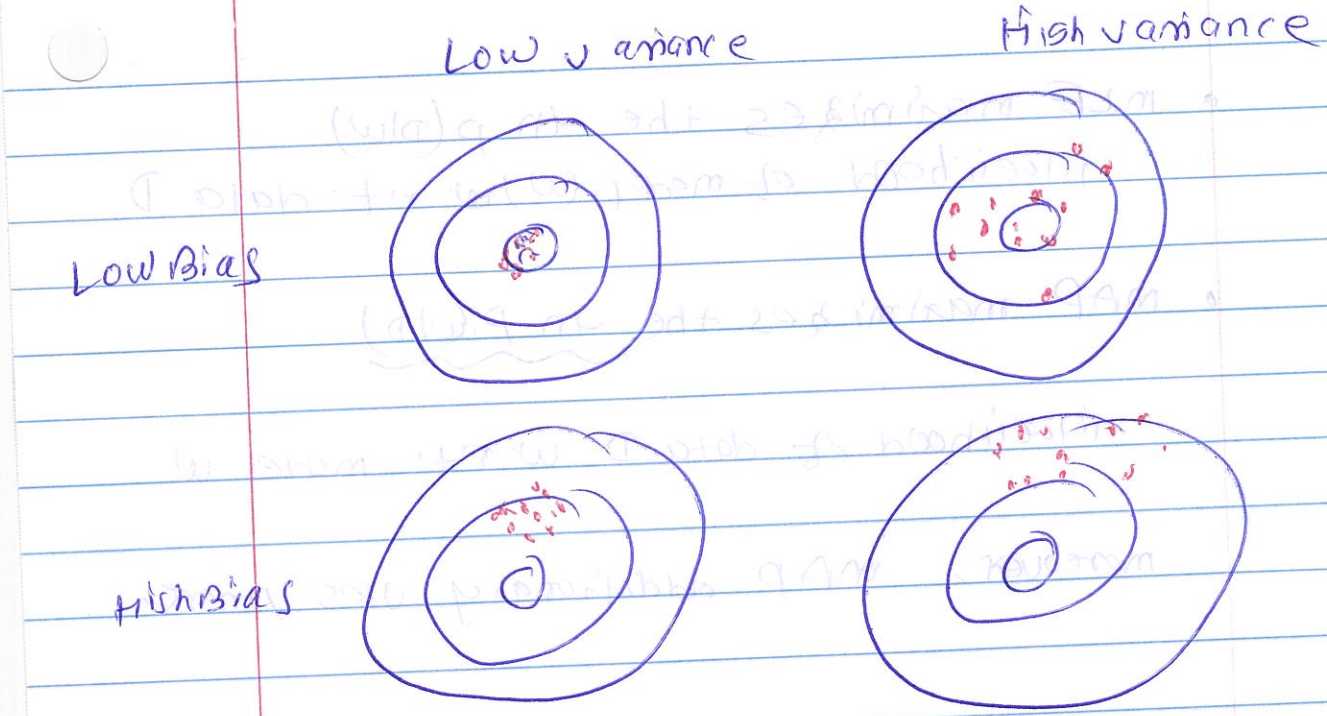
$\therefore \lambda \geq 0$ so AA^T is PSD.

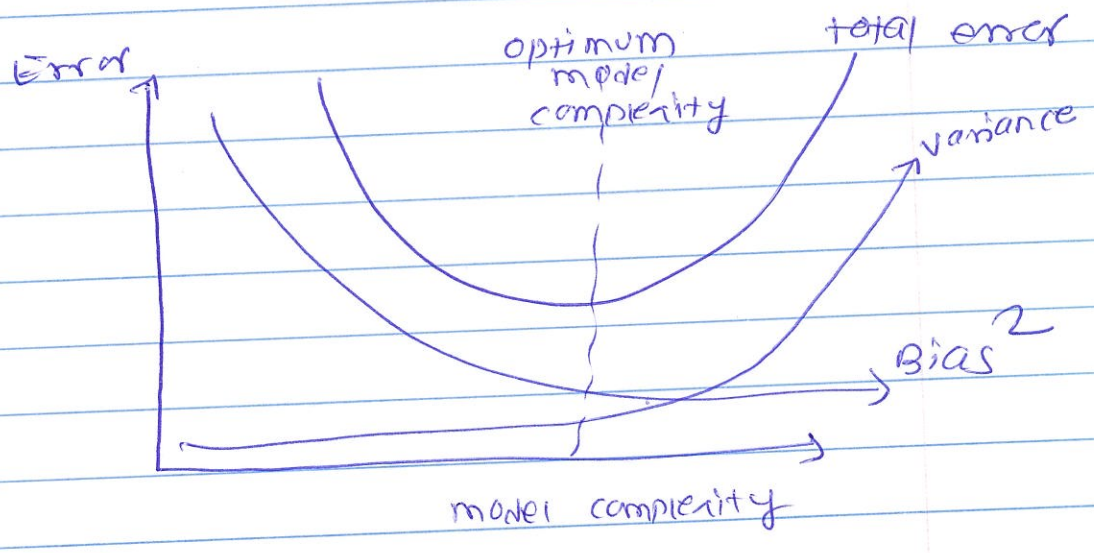
estimation maximum a priori

MLE vs MAP

- MLE maximizes the $\ln p(D|w)$ likelihood of model (w) w.r.t. data D
- MAP maximizes the $\ln p(w|D)$ likelihood of data D w.r.t. model w

moreover, MAP additionally uses priors.


Low variance
High variance
Low Bias
High Bias


Error
optimum model complexity
total error
variance
Bias²
model complexity

* A kernel will be said if there exists a space such that

$$K(x_1, x_2) = \phi(x_1)^T \phi(x_2) = \sum_{n=1}^{N} \phi_n(x_1) \phi_n(x_2)$$

* consider a quadratic kernel with $D = 2$,

$$K(x, z) = (x^T z)^2 = (x_1 z_1 + x_2 z_2)^2 \quad x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$= (x_1^2 z_1^2 + 2 x_1 z_1 x_2 z_2 + x_2^2 z_2^2) \quad z = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}$$

$$x^T = (x_1 \; x_2)$$

This can be expressed as an inner product of space where,

$$x^T z = (x_1 x_2) \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}$$

$$\phi(x) = x_1^2 + \sqrt{2} x_1 x_2 + x_2^2$$

this gives, $\phi(z) = z_1^2 + \sqrt{2} z_1 z_2 + z_2^2$

$$\phi(x)\phi(z) = x_1^2 z_1^2 + 2 x_1 x_2 z_1 z_2 + x_2^2 z_2^2$$

$$K(x, z) = \phi(x)^T \phi(z)$$

* A necessary and sufficient condition for a Kernel function to be "valid" is that the Gram matrix be positive and semidefinite for all choices of $\{\vec{x}_n\}$.

A Gram matrix of $\vec{x}$ is $x^T x$.

The linear vector $\vec{x}$ is projected into a quadratic surface.

If all the points in this surface are non-zero then our kernel is valid.

* Fisher's discriminant cost

$$J(w) = Tr\{(w S_w w^T)^{-1}\} \cdot (w S_B w^T)$$

— x — x — x — x — +ve oct ST

Least square perceptron ↙ does not work

$$E(w) = \frac{1}{2} \sum_{n=1}^{N} (h_n - t_n)^2$$

Nr of mistake → 0 $\frac{1}{2}$ .. $\frac{N-2}{2}$ $\frac{N}{2}$ ← N mistakes

N+1 values

min E = 0

max E = $\frac{N}{2}$

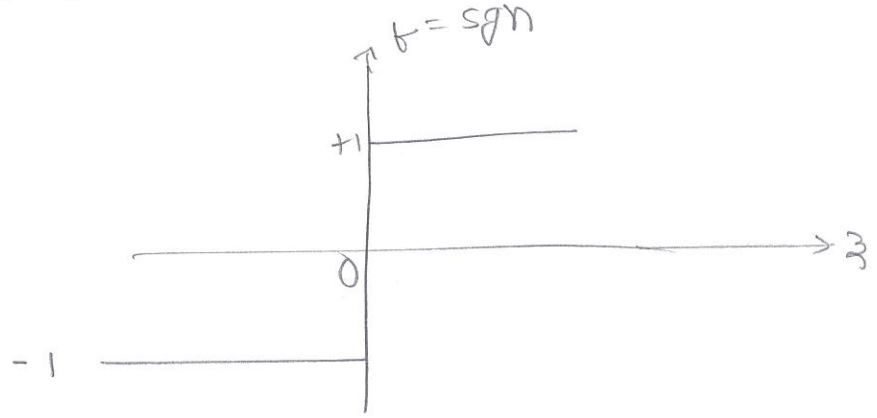
we cannot compute gradient, since functions discrete and not continuous.

cost = No. of misclassified patterns

= 0 for no misclay

$\frac{N}{2}$ for N mistakes

cost = [0 $\frac{1}{2}$ 2 ... $\frac{N-2}{2}$ $\frac{N}{2}$] discrete set.



Distance metrics

① Euclidean $d(x,y) = \|x-y\|_2 = \sqrt{(x-y)^T (x-y)}$

② Hamming $d(x,y)$ = # of different values in fixed length strings

③ Mahalanobis distance $d(x,y) = \sqrt{(x-y)^T S^{-1} (x-y)}$

S is sample cov. matrix

$S = I \rightarrow$ Euclidean

$S = \text{diag}(\sigma_1^{-2}, \sigma_2^{-2}, \ldots) \rightarrow$ normalized Euclidean dist.

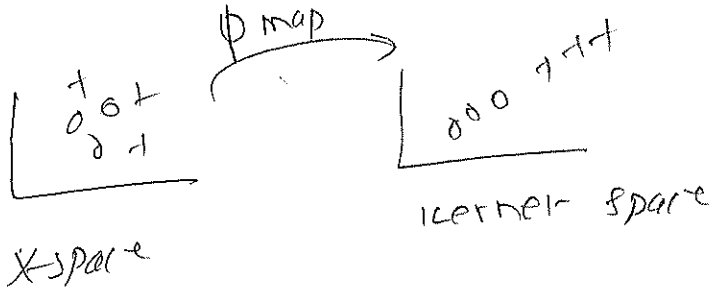
③ Cosine similarity

$$d(x,y) = 1 - \cos(x,y) = 1 - \frac{x^T y}{\|x\| \|y\|}$$

④ Levenstein distance (edit distance)

min # of basis operations (del, insert, juxtapose) between two strings

x = 'athens' y = 'hints' $d(x,y) = 4$



~~①~~ SVM with slack (soft margin SVM)

$$\min \; J(w,b,\xi) = \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N}\xi_n$$

with constraint

$$t_n(w^T\phi(x_n)+b) \geq 1-\xi_n$$

$$\xi_n \geq 0 \quad \text{and} \quad \sum_{i=1}^{N}\xi_i \leq Z \qquad \forall n \in \{1,\dots,N\}$$

i.e. $1-\xi_n - t_n w^T\phi(x_n) - b t_n \leq 0$ —— ① (convex fn)

$-\xi_n \leq 0$ —— ② (convex fn constraint) $\forall n = 1,2,\dots,N$

primal Lagrangian

$$L_p = L(w,b,\xi,\alpha,r) = \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N}\xi_n + \sum_{n=1}^{N}\alpha_n(1-\xi_n-t_n w^T\phi_n - b t_n) - \sum_{n=1}^{N} r_n\xi_n$$

dual Lagrangian

$$L_D(\alpha,r) = \inf_{w,b,\xi} L_p(w,b,\xi,\alpha,r)$$

NOTE! $0 = \frac{\partial L_p(w,b,\xi,\alpha,r)}{\partial w} = w - \sum_n \alpha_n t_n \phi(x_n) \Rightarrow \boxed{w = \sum_n \alpha_n t_n \phi_n}$

$0 = \frac{\partial L_p}{\partial b} = -\sum_n \alpha_n t_n \Rightarrow \boxed{\sum_n \alpha_n t_n = 0}$

NO summation! $0 = \frac{\partial L_p}{\partial \xi_n} = C - \alpha_n - r_n \Rightarrow \boxed{C = \alpha_n + r_n} \quad \forall n \in \{1,2,\dots,N\}$

then,

$$L_D(\alpha, r) = \frac{1}{2}\|w\|^2 + C\sum_n \xi_n + \sum_n \alpha_n(1 - \xi_n - t_n w^T \phi_n - t_n b) - \sum_n r_n \xi_n$$

$$L_D = \sum_n \alpha_n - \frac{1}{2}\sum_{m,n=1}^{N} \alpha_m \alpha_n t_m t_n k(x_m, x_n)$$

then the optimization problem in dual-space is,

$$\underset{\alpha}{\text{maximize}} \quad L_D(\alpha) = \sum_n \alpha_n - \frac{1}{2}\sum_{m,n} \alpha_m \alpha_n t_m t_n k(x_m, x_n)$$

with constraints $\quad 0 \le \alpha_n \le C \quad \forall n = 1, 2, \cdots, N$

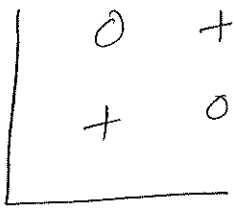
$$\sum_{n=1}^{N} \alpha_n t_n = 0$$

note: $\quad C\sum_n \xi_n - \sum_n \alpha_n \xi_n - \sum_n r_n \xi_n = 0$

since, $\quad C\sum_n \xi_n = \sum_n \alpha_n \xi_n + \sum_n r_n \xi_n$

$$= \sum_n (\alpha_n + r_n)\xi_n$$

(1)



- SVM (quad kernel) can achieve 3ero training error
- Logis Regr, 3-NN cannot

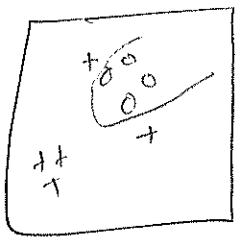
(2) If examples are iid,
increase training examples → may increase train error
↳ but decrease test error

(3) SVM effect of C

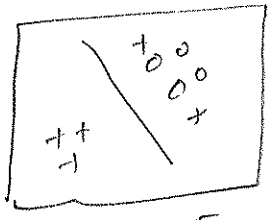
$$\min J(w,b) = \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N} \xi_n$$

$$s.t. \quad w^T \phi(x_n) + b \geq 1 - \xi_n \quad \forall n \in \{0, N\}$$

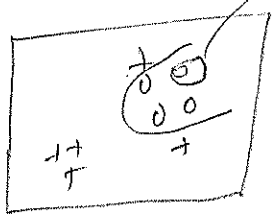
will not change decision bond.



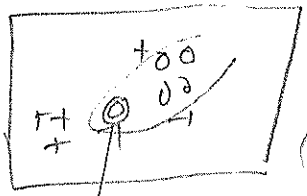
C = 10,000



C = 0.00005



C = 1



adding this change dec. boun. drastically

between C >> 1 and C ≈ 0 choose C ≈ 0 because it maximizes the margin between dominant cloud of points and we can not depend on any few data points which can be noise.

(4) Bias variance Tradeoff

	Bias	var
linear regr	high	low
d=2 poly	low	low
d=10 poly	low	high

python

① $X = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ append ones column to data X.

```
X = np.array([ [1,2,3], [4,5,6] ]) = np.arange(1,7).reshape(2,3)
ones = np.ones( X.shape[0] ).reshape(-1, 1)
X1 = np.append( ones, X, axis=1 ).astype(np.int)
   concatenate
X1 = np.c_[ np.ones( X.shape[0] ).reshape(-1,1) , X ]
X1 = np.column_stack(( , ))
X1 = np.c_[ np.ones ( X.shape[0] )[ :, np.newaxis ] , X ]
                    .reshape(-1,1)
   = np.c_[ np.atleast_2d( np.ones( X.shape[0] ).T , X ]
   = np.c_[ np.expand_dims( np.ones( X.shape[0] , axis=1) , X ]
```

(5) Given $\phi(x) = [1, x_1, x_2, x_1x_2]$

find the kernel $K(x, x')$.

$\Rightarrow K(x, x') = 1 + x_1x_1' + x_2x_2' + x_1x_2x_1'x_2'$

(6) L1 vs L2 loss

(a) False: L2 is more robust to outlier than L1

gradient of L2 loss can grow without bounds, but gradient of L1 loss is bounded, hence influence of outlier is limited.

(Note: L2 gives more power to misclassification than L1)

(b) L1 gives sparse solution & used in feature selection.

(c) Logistic loss is better than L2 loss in classification task.

(d) SVM small C,

For linearly separable data, small C can affect training accuracy.

A small C can allow some slacks, thus, the resulting classifier will have smaller w^2 and can have non-zero training error.

① pg ① solve the SVM problem without slack using Lagrange multiplier method

→ the optimization problem is

minimize $J(w,b) = \frac{1}{2}\|w\|^2$

s.t. $t_n(w^T\phi(x_n)+b) \geq 1 \quad \forall n \in \{1,\dots,N\}$

$1 \leq t_n(w^T\phi(x_n)+b)$

$1 \leq t_n w^T \phi(x_n) + t_n b$

$1 - t_n w^T \phi(x_n) - t_n b \leq 0$ (convex fn constraint)

compare $f_i(\alpha) \leq 0$ ~~then~~ $\forall i = 1,\dots,m$

primal Lagrangian,

$L_p(w,b,\alpha) = \frac{1}{2}\|w\|^2 + \sum_{n=1}^{N} \alpha_n C\,(1 - t_n w^T \phi_n - t_n b)$

where $\alpha_n \geq 0$ are Lagrange multipliers

dual Lagrangian,

$L_D(\alpha) = \inf_{w,b} L_p(w,b,\alpha)$

first find the infimum of L_p w.r.t w, b :

$\frac{\partial}{\partial w} L_p = 0 = w + \sum_n \alpha_n(-)t_n\phi_n \Rightarrow \boxed{w = \sum_n \alpha_n t_n \phi_n}$ — (1)

$\frac{\partial}{\partial b} L_p = 0 = \sum_n (-)\alpha_n t_n \Rightarrow \boxed{\sum_n \alpha_n t_n = 0}$ — (2)

look LHS

(12)

then dual Lagrangian is,

$$L_D(\alpha) = \frac{1}{2}\sum_{n,m} \alpha_m \alpha_n t_n t_m \phi_m \phi_n + \sum_n \alpha_n - \sum_n \alpha_n t_n \phi_n \cdot \sum_m \alpha_m t_m \phi_m$$

$$- \sum_n \alpha_n t_n b \quad (= 0)$$

$$\boxed{L_D(\alpha) = \sum_n \alpha_n - \frac{1}{2}\sum_{n,m} \alpha_m \alpha_n t_m t_n K(x_n, x_m)}$$

$$K(x, y) = \vec{\phi}(x) \cdot \vec{\phi}(y) = \phi^T(x)\,\phi(y)$$

then the optimization problem in Dual space is,

maximize $L_D(\alpha) = \sum_n \alpha_n - \frac{1}{2}\sum_n \sum_m \alpha_m \alpha_n t_m t_n K(x_m, x_n)$

s.t. $\alpha_n \geq 0 \quad \forall n \in \{1 \cdots N\}$

$$\sum_{n=1}^{N} \alpha_n t_n = 0$$

look here

Now, KKT conditions are,

note: we can write $y(x_n) = w^T \phi(x_n) + b$

① primal constraints $1 - t_n(w^T \phi(x_n) + b) \leq 0$ (convex constraint equation) $t_n(w^T \phi(x_n) + b) \geq 1$

② dual constraint $\alpha_n \geq 0 \quad \forall n \in \{1, \cdots N\}$

③ complementary slackness $\alpha_n \{1 - t_n w^T \phi(x_n) - t_n b\} = 0$

for any data point, either $\alpha_n = 0$

$1 - t_n w^T \phi(x_n) - t_n b = 0$

these x_n are called support vectors

pg 6

here $n = 1, 2, \cdots, N$

Suppose, out of N samples, there are m support vectors then N-m samples will have Lagrange parameter $\alpha = 0$ and m examples will have non-zero Lagrange parameters.

$$1 - t_m w^T \phi(x_m) - t_m b = 0$$

or, $$1 - t_m \phi(x_m) \sum_n \alpha_n t_n \phi(x_n) - t_m b = 0$$

or, $$t_m b = 1 - t_m \phi(x_m) \sum_n \alpha_n t_n \phi(x_n)$$

$$= 1 - t_m \sum_n \alpha_n t_n \phi(x_n) \phi(x_m)$$

$$b t_m = 1 - t_m \sum_n \alpha_n t_n \phi(x_n) \phi(x_m)$$

$$b = \frac{1}{t_m} - \sum_n \alpha_n t_n \phi(x_n) \phi(x_m) = t_m - \sum_n \alpha_n t_n \phi(x_n) \cdot \phi(x_m)$$

$$\boxed{b = t_m - \sum_n \alpha_n t_n K(x_n, x_m)} \quad \because \frac{1}{t_m} = t_m$$

$$\boxed{b = t_m - w \cdot \phi(x_m)}$$

$$\because 1 = \frac{1}{1} \text{ and } -1 = \frac{1}{-1}$$

this is true for all the m examples which have non-zero Lagrange parameter α.

For numerical stability we chose value of b as the mean of all b-values, then,

$$\boxed{b = \frac{1}{|S|} \sum_{m \in S} \left[t_m - \sum_{n \in S} \alpha_n t_n K(x_n, x_m) \right]}$$

where S is the subset of all the examples where Lagrange parameter α is non-zero.

$$S \subseteq D$$

$$S = \{ n \mid 1 - t_n w^T \phi(x_n) - t_n b = 0 \}$$

then, Linear discriminant function is,

$$\boxed{y(x) = w^T \phi(x) + b = \sum_{m \in S} \alpha_m t_m K(x_m, x) + \frac{1}{|S|} \sum_{m \in S} \left[t_m - \sum_{n \in S} \alpha_n t_n K(x_n, x_m) \right]}$$

KNN = memory-based (no model to fit)

① { find k nearest examples $x_1, x_2 \dots x_k$ from test x,
$y(x) = \underset{t \in T}{\operatorname{argmax}} \sum_{i=1}^{k} \delta_t(t_i)$

w_i gives distance-weighted k-NN

$w_i = \frac{1}{\|x - x_i\|^2}$

KNN usage

① Satellite image classification

② digit recognition

② Mahalanobis dist

$d(x,y) = \sqrt{(x-y)^T \Sigma^{-1} (x-y)}$

sample covariance matrix

if $\Sigma = I$ = Euclidean distance

$\Sigma = \text{diag}(\sigma_1^{-2}, \sigma_2^{-2}, \dots, \sigma_k^{-2})$

normalized Euclidean

③ cosine similarity

$d(x,y) = 1 - \frac{x^T y}{\|x\| \|y\|}$

④ kernel based distance weighted NN

→ binary classification $T \in \{+1, -1\}$

→ $y(x) = \text{sign}\left(\sum_{i=1}^{N} K(x, x_i) \cdot t_i\right)$

wrapper method

(1) greedy
Forward selection
F = all features
S = subset of features
start S = { } empty

for each feature f in F - S
find best performing feature f+ and add to S.

Repeat until performance does not increase
or performance good enough

(2) Recursive
Backward Elimination

F : {1, 2, ... K} is set of features
S = [ ] ranked set of features

Repeat until F - S is empty
train w using linear SVM and F - S
find feature f with minimum |w_f|
append f to S

Return S

④ distance-weighted KNN for regression

1. find K nearest points $x_1, x_2, \ldots, x_K$

2. $y(x) = \dfrac{\sum_{i=1}^{K} w_i t_i}{\sum_i w_i}$ where $w_i = \dfrac{1}{\|x - x_i\|^2}$

$K = N \rightarrow$ shepard's method

$y(x) = \sum_{i=1}^{N} k(x, x_i)\, t_i \;/\; \sum_{i=1}^{N} k(x, x_i)$ (kernel-based dist weighted)

⑤ Regression with KNN

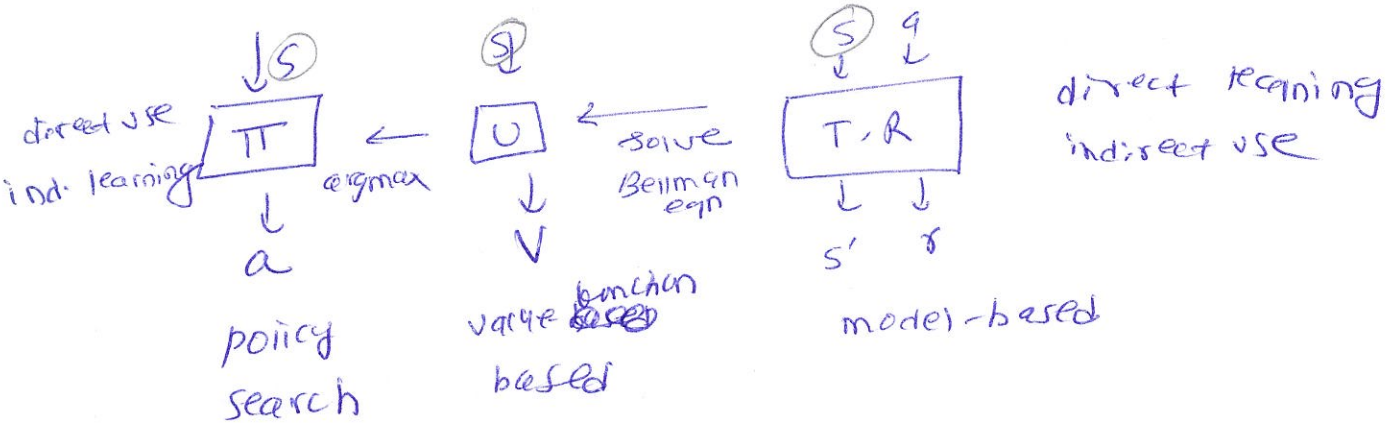
$y(x) = \frac{1}{K} \sum_{i=1}^{K} t_i$ (t_i are values, not classes)

⑥ distance-weighted KNN (regression)

primal: $y(x) = \sum_{i=1}^{K} w_i t_i \;/\; \sum_{i=1}^{K} w_i$ $\quad w_i = \dfrac{1}{\|x - x_i\|^2}$

kernel based: $y(x) = \sum_{i=1}^{N} k(x, x_i)\, t_i \;/\; \sum_{i=1}^{N} k(x, x_i)$ (t_i are values not classes)

3 Approaches to RL



Filter method of feature selection

(1) Mutual Information

$$MI(X,Y) = \sum_X \sum_Y p(x,y) \log \frac{p(x,y)}{p(x)\,p(y)}$$

$\Leftarrow$ 0 when x, y indep
max when x = y

(2)

Let there are K examples with 1 features.

O_{ij} = observed value for $x=i$, $y=j$

	1	...	j	...	1	
1						
⋮						
i			O_{ij}			} $N_{x=i}$
⋮						
K						
			$N_{y=j}$			

$$E_{ij} = \frac{N_{x=i}\, N_{y=j}}{N}$$

$$\chi^2 = \sum_{i=1}^{K} \sum_{j=1}^{1} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

classification

② probabilistic (Generative) models

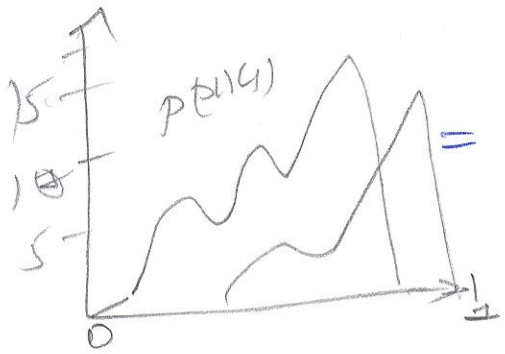
- Naive Bias
- Hidden Markov models

- ind. dec → separate
- can use p(x) for outlier or novelty detection
- need to model dependencies between features

NaiveBias → resilient to noise

- text classification with NB → 100K HW
- multiple classes
  posterior prob of class C_k given ~~test~~ data x is,

$$p(C_k|x) = \frac{p(x|C_k) \cdot p(C_k)}{\sum_j p(x|C_j)\, p(C_j)}$$



$$= \frac{\exp(a_k(x))}{\sum_j \exp(a_j(x))}$$

normalized exponential (softmax fn)

generative where $a_k(x) = \ln p(x|C_k) \cdot p(C_k)$

① SVM for ranking

optimization problem

minimize $J(w,b) = \frac{1}{2}\|w\|^2 + C\sum \xi_{k,i,j}$

s.t. $w^T \phi(q_k, d_i) \geq w^T \phi(q_k, d_j) + 1 - \xi_{k,i,j}$

$\xi_{k,i,j} \geq 0$

(for a query q_k we want document d_i be ranked higher than document q_j)

① Add ones column

```
= np.c_[np.ones(X.shape[0]).reshape(-1,1), X]
X = np.c_[np.ones(X.shape[0])[np.newaxis].T, X]
```

②

```
correct = np.sum(y_pred == y_test)
accuracy = correct / len(y_pred)
```

⑤ Gradient Descent (GD or B&D)

vanilla GD

$$v^{\tau+1} = \eta \nabla J(w^\tau)$$

$$w^{\tau+1} = w^\tau - v^{\tau+1}$$

GD with momentum

$$v^{\tau+1} = \gamma v^\tau + \eta \nabla J(w^\tau)$$

$$w^{\tau+1} = w^\tau - v^{\tau+1}$$

Batch Gradient Descent (Lect01, p.24)

$$J(w) = \frac{1}{2N} \sum_{n=1}^{N} (h_n - t_n)^2$$

$$w^{\tau+1} = w^\tau - \eta \nabla J(w^\tau)$$

$$= w^\tau - \eta \sum_{n=1}^{N} (h_n - t_n)\, x_n$$

$$= w^\tau - \text{learningRate} \cdot (h-t) \cdot X1$$

confusion matrix

Actual cats

predicted cats	cat	not cat	
cat	TP	FP	$\tilde{P}$
not cat	FN	TN	$\tilde{N}$
	true		
	P	N	

00 = TN
01 = FP
10 = FN
11 = TP

	1	0	
1	TP	FP	$\tilde{P}$
0	FN	TN	$\tilde{N}$
	P	N	

actual

	0	1	
0	TN	FN	$\tilde{N}$
1	FP	TP	$\tilde{P}$ → precision = $\frac{TP}{TP+FP}$
	N	P	

recall = $\frac{TP}{P} = \frac{TP}{TP+FN}$

(hit rate)
predicted true / actual true

F1 score is the harmonic mean of precision and recall.

$$F1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}}$$

$$= \frac{2 \text{ precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

* prove that the number of elements in x and y is also a kernel

$\Rightarrow$ $K(x,y) = |x \cap y|$ is a kernel.

<u>contd</u> (Nov 11)

$$w^T \phi(x) = \sum_{n=1}^{N} \alpha_n t_n K(x_n, x)$$

since $S = n$, $\alpha_n > 0$ (strict), $n = 1, \dots, N$; $n \notin S \Rightarrow \alpha_n = 0$

$$= \sum_{m \in S} \alpha_m t_m K(x_m, x)$$

$$+ \sum_{m \notin S} \alpha_m t_m K(x_m, x)$$

↑ but if $x_m \notin S$, then $\alpha_m = 0$

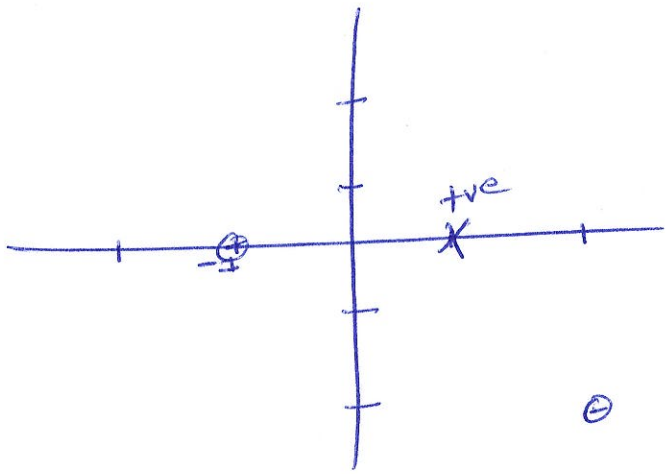
* package libsvm or svm lite,

input → $\{(x_n, t_n)\}$ $1 \le n \le N$

output → $\{\alpha_m, t_m, x_m\}$ $m \in S$

model.txt → [b ; $\alpha_m t_m$ x_m]

perceptron update



-1, 0 -ve
2, -2 -ve

1, 0 → +ve

final w = 0 1 2
let's say new point
x = 2, -1.01
xavg = 1 2 -1.0,
w xavg = 0 + 2 - 2.02
< 0
∴ -ve label

Augmented data

1	-1	0	-1
1	2	-2	-1
1	1	0	1

x_{avg} t

initial wt $\vec{w} = (0\ 0\ 0)$

note we can also use $w \cdot \vec{x} + b$ as original unaugmented x
$\eta = 0.9$
$b = 1$ etc

Test point	hypothesis	correct classified?	Updated weights
eg1 −: (1 −1 0)	wx = 0+0+0 ≤ 0	false	w = w − x = (−1 1 0)
eg2 −: (1 2 −2)	wx = −1+2+0 < 0	false	w = w − x = (−2 −1 2)
eg3 +: (1 1 0)	wx = −2−1+0 ≥ 0	false	w = w + x = (−1 0 2)
eg1 −: (1 −1 0)	wx = −1+0+0 < 0	true	w = w = (−1, 0 2)
eg2 −: (1 2 −2)	wx = −1+0+(−4) < 0	true	w = w = (−1, 0, 2)
eg3 +: (1 1 0)	wx = −1+0+0 ≥ 0	false	w = w + x = (0 1 2)
eg1 −: (1 −1 0)	wx = 0+(−1)+0 < 0	true	w = w = (0 1 2)
eg2 −: (1 2 −2)	wx = 0+2+(−4) < 0	true	w = w = (0 1 2)
eg3 +: (1 1 0)	wx = 0+1+0 ≥ 0	true	w = w = 0 1 2

Final weight

① constrained optimization
(Lagrange multipliers)

maximize $5 - (x_1-2)^2 - 2(x_2-1)^2$

s.t. $x_1 + 4x_2 = 3$

soln: If we ignore constraint we get $x_1 = 2, x_2 = 1$
then $x_1 + 4x_2 = 2 + 4 \cdot 1 = 6$ is too large
for the constraint.
consider
$L = L(x_1, x_2, \lambda) = 5 - (x_1-2)^2 - 2(x_2-1)^2 + \lambda(3 - x_1 - 4x_2)$

Look λ: $\lambda = 0$ $\quad 2, 1$
$\lambda = 1$ $\quad 3/2, 0$
$\lambda = \frac{2}{3}$ $\quad \left(\frac{5}{3}, \frac{1}{3}\right)$ $\quad \frac{5}{3} + 4 \cdot \frac{1}{3} = \frac{5}{3} + \frac{4}{3} = \frac{9}{3} = 3$ ✓

formal soln $\quad \frac{\partial L}{\partial x_1} = -2(x_1-2) - \lambda = 0$

$= -2x_1 + 4 - \lambda = 0 \Rightarrow 2x_1 = 4 - \lambda$

$2x_1 = 4 - \lambda$
$= 4 - \frac{2}{3}$
$= \frac{4 \cdot 3 - 2}{3} = \frac{10}{3}$
$x_1 = \frac{5}{3}$

$\frac{\partial L}{\partial x_2} = -4(x_2-1) - 4\lambda = 0$

$= -4x_2 + 4 - 4\lambda = 0$

$\Rightarrow 4x_2 = 4 - 4\lambda$
$x_2 = 1 - \lambda$

$\frac{\partial L}{\partial \lambda} = 3 - x_1 - 4x_2 = 0$

$\Rightarrow 6 - 2x_1 - 8x_2 = 0$

$\Rightarrow 6 - 4 + 2\lambda - 8(1 - \lambda) = 0$

$\Rightarrow 6 - 4 + 2\lambda - 8 + 8\lambda = 0$

$-6 + 10\lambda = 0$
$10\lambda = 6$
$\lambda = 2/3$

op1

$$\min_{w,b} \frac{1}{2}\|w\|^2$$

$$s.t.\ t_n(w^T\phi(x_n)+b) \geq 1$$

solution $= w_1^*, b_1^*$

op2

$$\min_{w,b} \frac{1}{2}\|w\|^2$$

$$s.t.\ t_n(w^T\phi(x_n)+b) \geq \gamma$$

$\Downarrow$ op3

$$\min_{w,b} \frac{1}{2}\left\|\frac{w}{\gamma}\right\|^2$$

$$s.t.\ t_n\left(\left(\frac{w}{\gamma}\right)^T\phi(x_n)+\frac{b}{\gamma}\right) \geq 1$$

op3 has solution

$w_2^* = w_1^*\gamma$

$b_2^* = b_1^*\gamma$

$\Uparrow$ op3

renaming

$w' = w/\gamma$

$b' = b/\gamma$

$$\min_{w,b} \frac{1}{2}\|w'\|^2$$

$$s.t.\ t_n(w'\phi(x_n)+b') \geq 1$$

Now, decision hyperplanes are,

$$H_1 = \{x \mid w_1^{*T}\phi(x) + b_1^* = 0\}$$

$$H_2 = \{x \mid w_2^{*T}\phi(x) + b_2^* = 0\}$$

$$= \{x \mid \gamma w_1^{*T}\phi(x) + \gamma b_1^* = 0\}$$

$$= \{x \mid w_1^{*T}\phi(x) + b_1^* = 0\}$$

$H_2 = H_1$ q.e.d

③ kmcp kernel multi-class perceptron

1 define $f(x) = \sum_{ij} \alpha_{ij} \left[\phi(x_i, t_i)^T \phi(x,t) - \phi(x_i, c_j)^T \phi(x,t) \right]$

(like $w^T x = \sum_n \alpha_n t_n x_n^T x = \sum_n \alpha_n t_n k(x_n, x)$)

2 initialize dual params $\alpha_{ij} = 0$

3 for $i = 1, \dots n$

4 $\quad c_j = \underset{t \in T}{\operatorname{argmax}} \; f(x_i, t)$ (hn = sgn(f(x)))

5 $\quad$ if $c_j \neq t_i$ then (hn ≠ tn)

6 $\quad\quad \alpha_{ij} = \alpha_{ij} + 1$ (add one)

} Repeat

Testing: $t^* = \underset{t \in T}{\operatorname{argmax}} \; f(x,t)$ ($h(x) = \operatorname{sgn}(f(x))$)

② mcp (concept of kernel comes from here)

initialize parameters $w = 0$

for $i = 1 \dots N$

$\quad c_j = \underset{t \in T}{\operatorname{argmax}} \; w^T \phi(x_i, t)$

$\quad$ if $c_j \neq t_i$ then

$\quad\quad w = w + \phi(x_i, t_i) - \phi(x_i, c_j)$

}

w is (up) invariant and is the weighted average

$$w = \sum_{ij} \alpha_{ij} \left(\phi(x_i, t_i) - \phi(x_i, c_j) \right)$$

$$f(x) = w^T \phi(x,t) = \sum_{ij} \alpha_{ij} \left(\phi(x_i, t_i)^T \phi(x,t) - \phi(x_i, c_j)^T \phi(x,t) \right)$$

① kp

define $f(x) = w^T x = \sum_n \alpha_n t_n k(x_n, x)$

initialize dual parameter $\alpha_n = 0$

for $i = 1 \dots N$

$\quad h_n = \operatorname{sign}(f(x_i))$

$\quad$ if $h_n \neq t_i$ then

$\quad\quad \alpha_n = \alpha_n + 1$

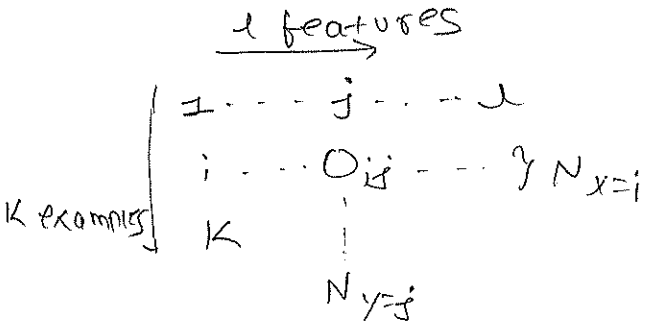
} Repeat

TEST: $y(x) = \operatorname{sign}(f(x))$

$MI(X,Y) = \sum_x \sum_y P(X,Y) \cdot \ln \frac{p(X,Y)}{p(X) \cdot p(Y)} = \begin{cases} 0 & \text{when } X, Y \text{ indep} \\ \max & \text{when } X = Y \end{cases}$

⑥ mutual information
- → works with nominal
- → biased towards high entropy feature
- → may choose redundant feature

⑦ chi-square



$$E_{ij} = \frac{N_{x=i} \cdot N_{y=j}}{N}$$

$$\chi_0^2 = \sum_{ij} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

⑧ pearson corr. coeff

$$\rho(X,Y) = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$$

⑨ SNR

$$\mu(X,Y) = \frac{|\mu_+ - \mu_-|}{\sigma_+ + \sigma_-} \quad \text{(binary)}$$

⑩ T-test

$$T(X,Y) = \frac{|\mu_+ - \mu_-|}{\sqrt{\frac{\sigma_+^2}{N_+} + \frac{\sigma_-^2}{N_-}}}$$

Filter	vs	wrapper
1 much faster since no need to train the model		• computationally expen.
2 use statistical method of evaluation		• uses cross-validation
3 might fail to find best subset		• finds best subset
4 less prone to overfitting		• more prone to overfitting

① SVM for regression

$$\min J(w,b) = \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N}(\xi_n + \hat{\xi}_n)$$

s.t. $t_n \le t_n(w^T\phi(x_n)+b) + \varepsilon + \xi_n$

$t_n \ge t_n(w^T\phi(x_n)+b) - \varepsilon - \hat{\xi}_n$

$\xi_n, \hat{\xi}_n \ge 0 \quad \forall \; n \le N$

② SVM for ranking

$$\min J(w,b) = \frac{1}{2}\|w\|^2 + C\sum_{k,i,j} \xi_{k,i,j}$$

s.t. $w^T\phi(q_k, d_i) \ge w^T(\phi(q_k, d_j) + 1 - \xi_{k,i,j}$

$\xi_{k,i,j} \ge 0$

MDP markov decision process

① policy (argmax)

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi\right]$$

(policy expectation)

$$\pi^* = \underset{a}{\operatorname{argmax}} \sum_{s'} T(s,a,s')\, U(s')$$

(best action policy)

R = Reward

$\gamma^t R$ = discounted Reward

② utility (expectation & max only)

$$U(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi,\ s_0 = s\right]$$

(expectation)

$$U(s) = R(s) + \gamma \max_{a} \sum_{s'} T(s,a,s')\, U(s') \leftarrow \text{Bellman eqn}$$

* Naive Bio :

3 Boolean input vectors $x_1 x_2 x_3$ and output y

$P(y=1)$

$P(x_1=1 \mid y=0)$ $P(x_1=1 \mid y=1)$

$P(x_2=1 \mid y=0)$ $P(x_2=1 \mid y=1)$

$P(x_3=1 \mid y=0)$ $P(x_3=1 \mid y=1)$

- # of parameters $= 2M+1 = 2\times 3+1 = 7$
- # of parameters if no conditional independence $= 1+2(2^M-1) = 1+2(8-1)$

$= 1+2(2^3-1) = 1+14 = 15$

Example

Train	Doc	words (w_i)	class (category label)	Total
3 documents for c_1	1	chinese Beijing chinese	c_1	$n_1 = 8$
	2	chinese chinese shanghai	c_1	
	3	chinese macao	c_1	
1 document for c_2	4	Tokyo japan chinese	c_2	$n_2 = 3$
Test	5	?	?	11 words, 6 unique
Total	$\lvert D \rvert = \lvert D_1 \rvert + \lvert D_2 \rvert = 3+1 = 4$			

1 chinese 1
2 beijing 1
3 shanghai 1
4 macao 1
5 tokyo 1
6 japan 1
} ignore

① vocabulary, $V = \{$ chinese, Beijing, shanghai, macao, Tokyo, japan $\}$

$|V| = 6$

② category $c_1 = c$ (china)

- prior $P(c_1) = \frac{|D_1|}{|D|} = \frac{3}{4}$
- # of words in class c_1, $n_1 = 8$
- $P(w_1|c_1) = P(\text{chinese}|c) = \frac{5+1}{8+6} = \frac{6}{14} = \frac{3}{7}$ ($n_{k1} = n_{11} = 5$; $n_k \to$ Laplace smoothing; $n \to 8+6$; 14)
- $P(w_2|c_1) = P(\text{Beijing}|c) = \frac{1+1}{14} = \frac{2}{7}$
- $P(w_3|c_1) = P(\text{shanghai}|c) = 2/7 = P$
- $P(w_4|c_1) = P(\text{macao}|c) = \frac{0+1}{8+6} = \frac{1}{14}$
- $P(w_5|c_1) = P(\text{Tokyo}|c) = \frac{0+1}{8+6} = \frac{1}{14}$
- $P(w_6|c_1) = P(\text{japan}|c) = \frac{0+1}{8+6} = \frac{1}{14}$

conditional probability

category $c_2 = j$ (japan)

- prior $P(c_2) = \frac{|D_2|}{|D|} = \frac{1}{4}$
- # of words in class c_2, $n_2 = 3$
- $P(w_1|c_2) = P(\text{chinese}|j) = \frac{1+1}{3+6} = \frac{2}{9}$
- $P(w_2|c_2) = P(\text{Tokyo}|j) = \frac{1+1}{3+6} = \frac{2}{9}$
- $P(w_3|c_2) = P(\text{japan}|j) = \frac{1+1}{3+6} = \frac{2}{9}$

③ choosing a class

$$P(c|d) \propto P(c_1) \prod_{i} P(w_i|c_1) = \frac{3}{4} \cdot \left(\frac{3}{7}\right)^3 \cdot \frac{1}{14} \cdot \frac{1}{14} \approx 0.0003$$

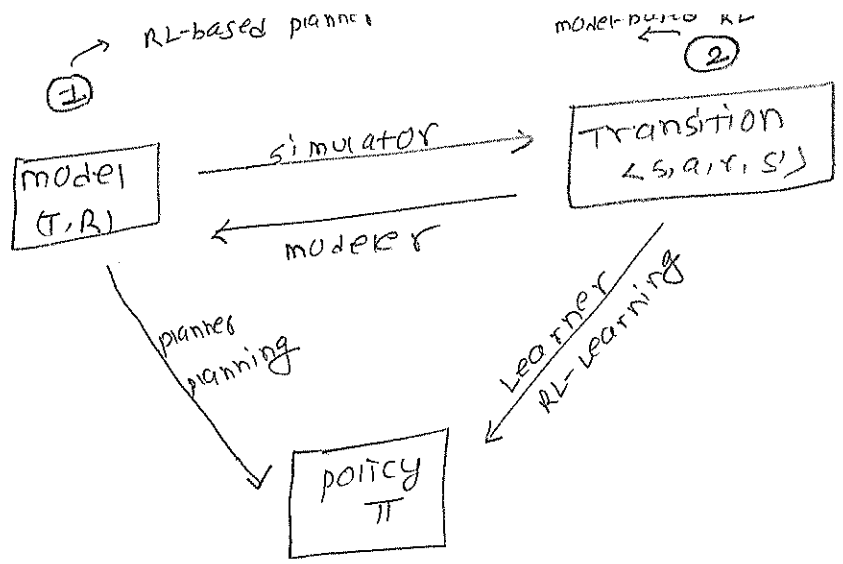
(prior, chinese, japan, tokyo)

$$P(c|d) \propto P(c_2) \prod_{i} P(w_i|c_2) = \frac{1}{4} \cdot \left(\frac{2}{9}\right)^3 \cdot \frac{2}{9} \cdot \frac{2}{9} \approx 0.0001$$

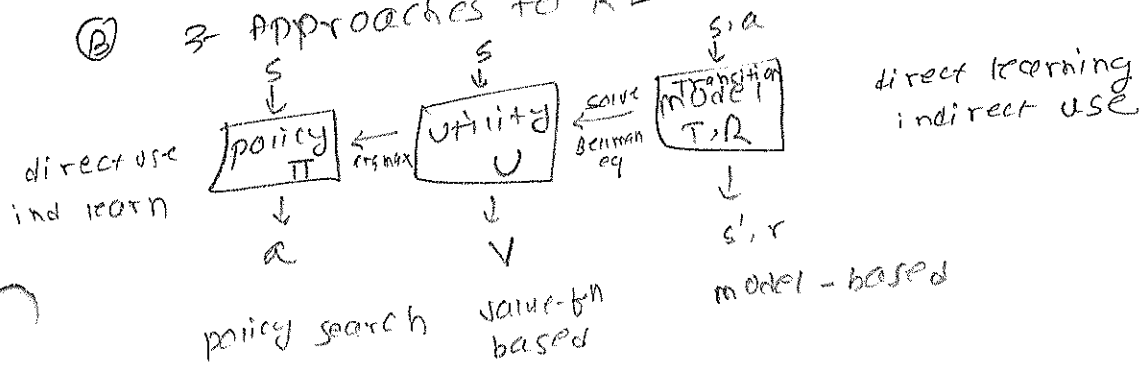
$$c^* = \underset{c_k}{\operatorname{argmax}} \; P(c_k) \prod_{j=1}^{n} P(w_j|c_k) = \underset{c_k}{\operatorname{argmax}} \{0.0003, 0.0001\}$$

$= \operatorname{argmax} 0.0003 = c_1$

(A)



(B) 3 approaches to RL



(C) Q function

$$U(s) = R(s) + \gamma \max_a \sum_{s'} T(s,a,s') U(s')$$ (Bellman eqn) utility is a scalar

change U to Q

$$\pi(s) = \arg\max_a \sum_{s'} T(s,a,s') U(s')$$ (policy gives an action)

$$Q(s,a) = R(s) + \gamma \sum_{s'} T(s,a,s') \max_{a'} Q(s',a')$$

quiz ⇒ $U(s) = \max_a Q(s,a)$

⇒ $\pi(s) = \arg\max_a Q(s,a)$

QN 4

① mutual information

$$MI(X,Y) = \sum_x \sum_y p(x,y) \cdot \ln\left(\frac{p(x,y)}{p(x)\,p(y)}\right)$$

$= 0$ when X, Y indp

$=$ max when $X = Y$

$$= KL\left[p(x,y) \,||\, p(x)\,p(y)\right]$$

bad → biased towards high entry features

bad → may choose redundant features

bad → works only with nominal features + labels

3 parametric approaches to

① discriminant function

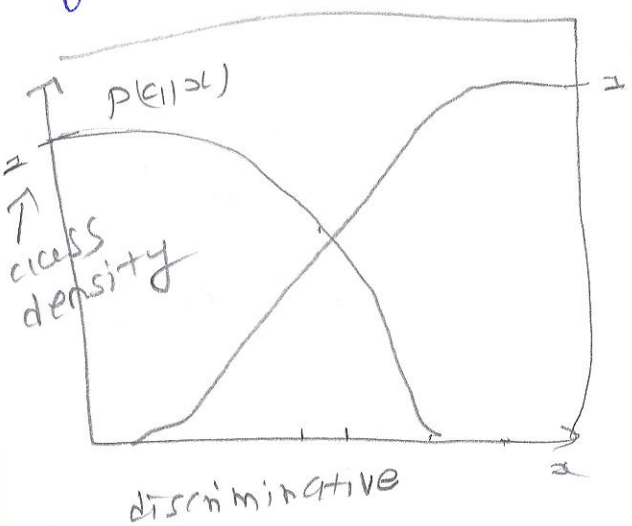
- fisher's linear disc.
- perceptron
- SVM

inference and decision are combined as single learning problem

② probabilistic discriminative models

- logistic regression
- conditional random field

- indirect → separate
- less data need to compute $p(C_k|x)$ than $p(x|C_k)$
- can accommodate many overlapping features



③ pearson corr coeff

$$\rho(x,y) = \frac{E[(x-\mu_x)(y-\mu_y)]}{\sigma_x \sigma_y} \quad \text{(population)}$$

$$\rho(x,y) = \frac{\sum_{i=1}^{n} (x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i-\bar{x})^2} \cdot \sqrt{\sum_{i=1}^{n} (y_i-\bar{y})^2}} \quad \text{(sample)}$$

$-1 \leq \rho(x,y) \leq 1$

④ SNR

binary classes = {+, −}
mean = {μ_+, μ_-}
examples are binary
$y \in \{+1, -1\}$

$$\mu(x,y) = \frac{|\mu_+ - \mu_-|}{\sigma_+ + \sigma_-}$$

μ_+, σ_+ = mean & std for +ve class samples

⑤ T-test

$$T(x,y) = \frac{|\mu_+ - \mu_-|}{\sqrt{\frac{\sigma_+^2}{\mu_+} + \frac{\sigma_-^2}{\mu_-}}}$$

CS 4900/5900: Machine Learning
Fall 2017

Class Meetings: Tue, Thu 10:30–11:50am, ARC 212
Instructor: Razvan Bunescu
Office: Stocker 341
Office Hours: Tue, Thu 12:00–12:30pm, or by email appointment
Email: bunescu @ ohio edu
Class Website: `http://ace.cs.ohio.edu/~razvan/courses/ml14900`

Prerequisites:

The students are expected to be comfortable with programming and familiar with basic concepts in linear algebra and statistics.

Textbook:

There is no textbook for this class. Slides and supplementary materials will be made available on the course website.

Supplementary Texts:

Machine Learning: The Art and Science of Algorithms that Make Sense of Data
by Peter Flach, Cambridge University Press, 2012

A Course in Machine Learning [free online]
by Hal Daume III

Machine Learning
by Tom Mitchell. McGraw Hill, 1997

Pattern Recognition and Machine Learning
by Christopher Bishop. Springer, 2007

Pattern Classification
by Richard O. Duda, Peter E. Hart, & David G. Stork. Wiley-IS, 2001

The Elements of Statistical Learning: Data Mining, Inference, and Prediction
by T. Hastie, R. Tibshirani, & J. H. Friedman. Springer Verlag, 2009

Course Description:

This course will give an overview of the main concepts, techniques, and algorithms underlying the theory and practice of machine learning. The course will cover the fundamental topics of classification, regression and clustering, and a number of corresponding learning models such as perceptrons, logistic regression, linear regression, Naive Bayes, nearest neighbors, and Support Vector Machines. The description of the formal properties of the algorithms will be supplemented with motivating applications in a wide range of areas including natural language processing, computer vision, bioinformatics, and music analysis. The topics covered in this course will also prepare students for taking more advanced courses in data mining and deep learning.

Grading:

50%: Homework Assignments
20%: Midterm Exam (Oct 12, in class) 1 hr 20 min
30%: Final Exam (Dec 12, 10:10am – 12:10pm)

Grading Scale:

A (> 92%) A−(> 90%) B+(> 87%) B(> 83%) B−(> 80%)
C+(> 77%) C(> 73%) C−(> 70%) D+(> 67%) D(> 63%) D−(> 60%)

Important Dates:

Friday, Sep 1: Last day to add class.
Tuesday, Oct 10: Reading Day, no class.
Friday, Nov 3: Last day to drop class.
Thursday, Nov 23: Thanksgiving break, no class.
Thursday, Dec 7: Last day of this class.

Course and Attendance policies:

Assignments: `All homework assignments are due before the class.` No late submissions will be accepted without prior approval.

Attendance: `It is in your best interest to attend the lectures.` Some of the material will not be found in the supplementary text or on the slides. Extra credit will be awarded for class activity. Also, be sure to check your OU email for important announcements on a regular basis.

Academic Dishonesty Policy:

`All work must be the student's own.` All external references used in reports must be properly cited. No credit will be given for duplicate or plagiarized work. Additional measures may be imposed by the University Judiciaries, when conditions warrant. Students may appeal academic sanctions through the grade appeal process. The OU Student Code of Conduct Policy is available online at:

`http://www.ohio.edu/communitystandards/academic/students.cfm`

Disability-based Accommodation:

Any student who suspects s/he may need an accommodation based on the impact of a disability should contact the class instructor privately to discuss the students specific needs and provide written documentation from the Office of Student Accessibility Services. If the student is not yet registered as a student with a disability, s/he should contact the Office of Student Accessibility Services.

Other Policies:

Be sure to notify the professor of any exam conflicts or other extenuating circumstances well in advance. `No missed exams will be made up without prior approval.` Medical excuse forms need to explicitly mention that the student could not have attended the exam at the specified time due to health concerns.