Robert Hecter
ELEC 378
5/3/2023

Final Exam

START TIME: 7:00 PM CST
STOP TIME: 10:00 PM CST

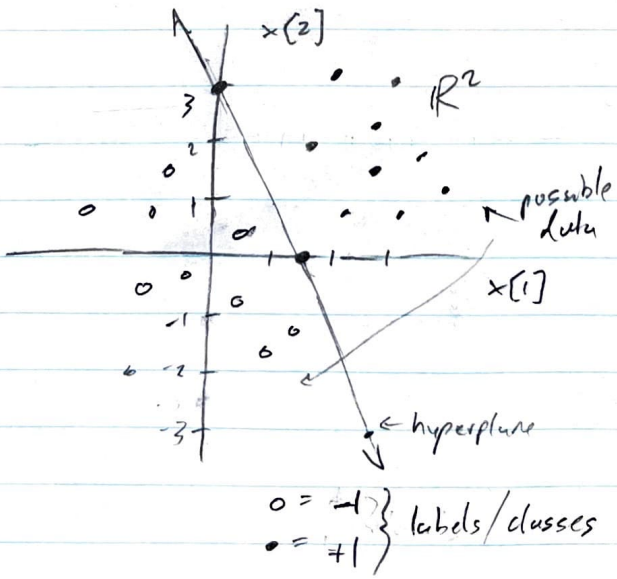
On my honor, I have neither given nor received any unauthorized aid on this exam.

Robert Hecter

1. a. Regression refers to the set of problems/methods that involve designing a function that assigns continuous labels $\hat{y}_i$ for input datapoints or vectors x_i. The labels $\hat{y}_i$ are continuous / in $\mathbb{R}$. Classification refers to a similar set of problems/methods, but where the labels $\hat{y}_i$ are discrete (not continuous) and fall in a set $\hat{y}_i \in \{ \_\_\_\_ \}$. While both regression & classification are (finite) types of supervised learning, classification methods require the finite set of labels to be predefined, whereas the labels in regression do not. Classification algorithms like logistic regression use special functions σ to map $\hat{y}_i \in \mathbb{R} \rightarrow \hat{y}_i \in \{\text{set of labels}\}$ (0, 1)

b. $\underline{w} = \begin{bmatrix} 6 \\ 3 \end{bmatrix}$ $\quad b = -9$ $\quad \hat{y}_i = \text{sign}(\langle \underline{w}, \underline{x}_i \rangle + b)$

hyperplane is $\mathbb{R}^1$ since $\underline{w}$ is in $\mathbb{R}^2$



intercepts: $\underline{x} = [1.5, 0]^T, [0, 3]^T$

hyperplane:

$[6, 3]^T \cdot \underline{x} - 9 = 0$

slope: -2

c. Next page

(1). c. PCA - principal components analysis — refers to the process of identifying the set of 1 dimensional subspaces of a data matrix that maximize the energy (spread/variance of the data which can then be used to transform the data matrix to lower dimensions that still retain the data's energy (spread/variance. The ~~principal components are the~~ ordered PCs are the leading eigenvectors of the data covariance matrix X^TX, where X is the data matrix, (in descending order) and the corresponding eigenvalues are the "energy" of that eigenvector (PC). In other words, to do PCA is to compute the PCs of the data matrix — starting from the eigenvectors of X^TX. This can be done with singular value decomposition $\rightarrow X = U\Sigma V^T$, where V's columns are PCs. The PC decomposition is given as $P = XV = U\Sigma$. The main PCs can be used to identify the most impactful features (dimensions in the original data matrix) since the size of the elements of each eigenvector corresponds to its weight — i.e. sorting the eigenvector gives most to least impactful features, some of which can be used to simplify the data matrix — reduce dimensions.

d. Multiclass support vector machines (SVM) can be performed in a "one vs-all" scheme. Here, a set of K 2-class SVMs are run to assign the data vectors to either one of the K classes or the rest of the classes. At the end, each data vector/point is labeled according to the SVM that classified the point with the largest margin.

e. A nonlinear function ϕ can be applied to the columns of the data matrix X to better separate the two classes (i.e. so that they are now linearly separable) before using the linear classifier on the transformed data.

$\underline{x}_i \rightarrow \phi(\underline{x}_i) \Rightarrow \underline{\underline{X}} \rightarrow \underline{\underline{\Phi}}(\underline{\underline{X}}) \Rightarrow$ use $\underline{\underline{\Phi}}$ for linear classifier since data is now linearly separable.

$\underline{x}_i$: a column of $\underline{\underline{X}}$

f. $\hat{y}_i = \sigma(\langle \underline{w}, \underline{x}_i \rangle + b)$

In SVM, σ is typically simply the sign function - i.e. $\hat{y}_i = \text{sign}(\langle \underline{w}, \underline{x}_i \rangle + b)$ since the decision hyperplane $0 = \langle \underline{w}, \underline{x}_i \rangle + b$ separates data that is $\langle \underline{w}, \underline{x}_i \rangle + b > 0$ and < 0. A single SVM decides between 2 classes by defining the optimal/maximum margin hyperplane while sometimes accounting for slight errors (i.e. slack variables that define the tradeoff).

In logistic regression, σ is a scalar nonlinearity - i.e. a function that maps the result from the regression to binary labels $[0, 1]$ (in $\mathbb{R}$)

The sigmoid function $\sigma(u) = \frac{1}{1+e^{-u}}$ is an example.

$$y = \begin{cases} 1 & \text{if } \sigma(w^T x + b) \geq 1/2 \\ 0 & \text{if } \sigma(w^T x + b) < 1/2 \end{cases}.$$

The hyperplane in logistic regression is defined only from the σ nonlinear function, while the hyperplane in SVM is based on the geometrical calculation of the decision plane's margins and error.

g. Cross validation is important to determine how accurate a trained model fits the overall data. It works by dividing the data up into a set of groups, and then, for each group, training the model with the remaining (non-chosen group) data/groups and then testing that trained model on the chosen group. This is repeated for all of the groups such that all of the data is used for testing and the average accuracy of the model is averaged from each train/test process for each group. Other metrics to determine the model's "success" or performance can be used (not just accuracy). For example, with our final project for this class, cross-validation can be run with the set of "test" labeled audio files - a function of the files can be saved for testing and the others used as training

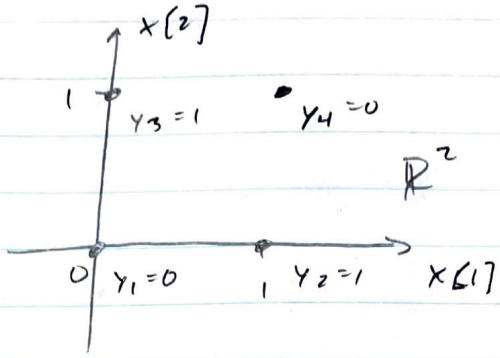
(i). (g), and the process can be repeated w/ another ~~set of~~ ~~labeled~~ new subset of the ~~testing~~ "test" data, until all of the audio files have been used for testing.

h. Kernel SVMs are advantageous in the fact that they ~~are~~ require less training data than neural networks to produce a reasonable decision boundary and because they are ~~simpler structurally~~ often structurally simpler, it is ~~easier to~~ ~~less~~ computationally ~~more~~ less intensive to train a kernel SVM compared to a neural network. ~~Thus~~ SVMs may be more useful for situations where there is not as ~~much~~ much data available, since they still maximize the decision boundary margins (i.e. builds a "buffer") while a neural network may have more trouble in generalizing to a larger data set/ real world situations.

Neural networks can be advantageous in situations where a lot of data is ~~available and there~~ and computational power is available and accuracy is important since they can account for very subtle features and patterns in the data using ~~the~~ gradual gradient descent

2. Next page

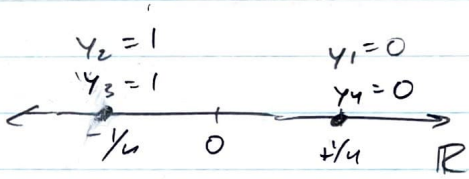
2. a.



$n = 4$ → 4 vectors
$p = 2$ → 2 dimensions/features per vector

b. It is not possible for a single layer perceptron to correctly predict the label of all of the data points because the data is not linearly seperable — i.e. a decision line boundary cannot be drawn that seperates the $y = 1$ and $y = 0$ points, without error.

c. i	$\underline{x}_i$	$\phi(\underline{x}_i)$
1	$[0\ 0]^T$	$-1/2 \cdot -1/2 = +1/4$
2	$[1\ 0]^T$	$+1/2 \cdot -1/2 = -1/4$
3	$[0\ 1]^T$	$-1/2 \cdot +1/2 = -1/4$
4	$[1\ 1]^T$	$+1/2 \cdot +1/2 = +1/4$



$n = 4$ → 4 vectors
$p = 1$ → 1 dimension/feature per vector

d. $\phi(\underline{x}_i)$ is a nonlinear transform of $\underline{x}_i$.

To be a linear transformation: $\begin{cases} F(\underline{u} + \underline{v}) = F(\underline{u}) + F(\underline{v}) \\ \text{and } c\,F(\underline{v}) = F(c\underline{v}) \end{cases}$

(fxn, vectors; c: constant scalar)

For example, $\underline{x}_1 = [0\ 0]^T$ $\quad \underline{x}_1 + \underline{x}_2 = [1\ 0]^T$

$\underline{x}_2 = [1\ 0]^T$

$\phi(\underline{x}_1 + \underline{x}_2) = \phi([1\ 0]^T) = -1/4$

$\phi(\underline{x}_1) + \phi(\underline{x}_2) = \phi([0\ 0]^T) + \phi([1\ 0]^T) = +1/4 + (-1/4) = 0$

$\therefore \phi(\underline{x}_1 + \underline{x}_2) \neq \phi(\underline{x}_1) + \phi(\underline{x}_2)$ and ϕ is not a linear transform of $\underline{x}_i$

$\therefore \phi$ is nonlinear transform of $\underline{x}_i$

(2). e. Yes, it is possible to use single layer perceptron to predict the label of the transformed data points.

$$\hat{z}_i = \sigma(v \cdot \phi(x_i) + c) \longrightarrow \hat{z}_i \in \{0, 1\}$$

Choose $v = -4$, $c = 0$, $\sigma(u) = \begin{cases} 0 \text{ if } u < 0 \\ 1 \text{ if } u \geq 0 \end{cases}$ (step function)

$\hat{z}_i = \sigma(-2\,\phi(x_i) + 0) \rightarrow$

i	x_i	$\phi(x_i)$	$\hat{z}_i$
1	$[0\ 0]^T$	$+1/4$	$-4(1/4) = -1 < 0 \rightarrow 0$
2	$[1\ 0]^T$	$-1/4$	$-4(-1/4) = 1 \geq 0 \rightarrow 1$
3	$[0\ 1]^T$	$-1/4$	$-4(-1/4) = 1 \geq 0 \rightarrow 1$
4	$[1\ 1]^T$	$+1/4$	$-4(1/4) = -1 < 0 \rightarrow 0$

$v = -4$
$c = 0$
$\sigma(u) =$ step function (see above)

predictions accurate $(\hat{z}_i = \hat{y}_i)$

Another option w/ linear σ is:

$v = -2$
$c = +1/2$
$\sigma(u) = u$

~~$\sigma(u) = \begin{cases} 0 \text{ if } u < 0 \\ 1 \text{ if } u \geq 0 \end{cases}$~~

3. Next page

3. a. Measuring the error as $e = \frac{1}{2}(y_i - \hat{y}_i)^2$ is preferable to defining the error as $|y_i - \hat{y}_i|$ since the $\frac{1}{2}(y_i - \hat{y}_i)^2$ error is continuously differentiable, which is required for gradient descent to optimize the $\underline{w}$ and b that produce the lowest error. $|y_i - \hat{y}_i|$ is not differentiable at $\hat{y}_i = y_i$ whereas $\frac{1}{2}(y_i - \hat{y}_i)^2$ is.

b. The sigmoid function $u(z) = \sigma(z) = \frac{1}{1+e^{-z}}$ may be preferred to the step function since it is continuously differentiable whereas the unit step function has a discontinuity at 0. This ~~would be pref~~ is preferred for gradient descent, which requires that the loss function be continuously differentiable, since the ~~derivat~~ derivative is required for descent.

~~c.~~

c. However, when $\sigma(z) = \frac{1}{1+e^{-z}}$, the predictions from $\hat{y} = \sigma(\langle \underline{x}, \underline{w} \rangle + b)$ are not binary, since $\sigma(z)$ does not return discrete values (i.e. not 1 or 0). The resulting predictions will never be truly binary (i.e. may take ~~values~~ values between 1 and 0).

d. $\hat{y} = \sigma(\langle \underline{x}, \underline{w} \rangle + b)$ ~~is~~ with $\sigma(z) = \frac{1}{1+e^{-z}}$ is different from the logistic regression model since the logistic regression model restricts the predictions using inequalities:

$$\hat{y}_i = \begin{cases} 0 & \text{if } \sigma(\langle \underline{x}_i, \underline{w} \rangle + b) < \frac{1}{2} \\ 1 & \text{if } \sigma(\langle \underline{x}_i, \underline{w} \rangle + b) \geq \frac{1}{2} \end{cases}$$

Also, ~~the~~ logistic regression uses negative cross-entropy as an objective/loss function, rather than the ~~square~~ square of the error (half error squared) as is used here. → i.e. Error $= \frac{1}{2}(y_i - \hat{y}_i)^2$

e. Next page

(3). i.e. $\nabla_{\underline{v}}\, \sigma(f(\underline{v})) = \sigma(f(\underline{v}))\,(1-\sigma(f(\underline{v})))\,\nabla_{\underline{v}} f(\underline{v})$

① ② ③

$\ell(\underline{w},b) = \sum_{i=1}^{n} \frac{1}{2}(y_i - \hat{y}_i)^2$

$\hat{y}_i = \sigma(\langle \underline{x}_i, \underline{w} \rangle + b) = \sigma(\underline{w}^T \underline{x} + b)$

$\ell(\underline{w},b) = \sum_{i=1}^{n} \frac{1}{2}\left(y_i - \sigma(\underline{w}^T \underline{x} + b)\right)^2$

chain rule:

$\nabla_{\underline{w}} \ell(\underline{w},b) = \sum_{i=1}^{n} \left[(y_i - \sigma(\underline{w}^T \underline{x} + b)) \left(0 - \sigma(\underline{w}^T \underline{x} + b)\left(1 - \sigma(\underline{w}^T \underline{x} + b)\right)\left(\frac{d}{d\underline{w}}(\underline{w}^T \underline{x} + b)\right)\right)\right]$

① ② ③

$\frac{d}{d\underline{w}} y_i = 0$

$\frac{d}{d\underline{w}}(\underline{w}^T \underline{x} + b) = \underline{x}$

$= \sum_{i=1}^{n} (y_i - \sigma(\underline{w}^T \underline{x}_i + b)) \left[-\underline{x}_i \sigma(\underline{w}^T \underline{x}_i + b)(1 - \sigma(\underline{w}^T \underline{x}_i + b)) \right]$

$\nabla_{\underline{w}} \ell(\underline{w},b) = -\sum_{i=1}^{n} \underline{x}_i (y_i - \sigma(\underline{w}^T \underline{x}_i + b)) \left[\sigma(\underline{w}^T \underline{x}_i + b)(1 - \sigma(\underline{w}^T \underline{x}_i + b)) \right]$

$\frac{d}{db} y_i = 0$

$\nabla_b \ell(\underline{w},b) = \sum_{i=1}^{n} \left[(y_i - \sigma(\underline{w}^T \underline{x} + b)) \left(0 - \sigma(\underline{w}^T \underline{x}_i + b)\left(1 - \sigma(\underline{w}^T \underline{x} + b)\right)\left(\frac{d}{db}(\underline{w}^T \underline{x} + b)\right)\right)\right]$

① ② ③ → 1

$\nabla_b \ell(\underline{w},b) = -\sum_{i=1}^{n} \left[y_i - \sigma(\underline{w}^T \underline{x}_i + b) \left[\sigma(\underline{w}^T \underline{x}_i + b)(1 - \sigma(\underline{w}^T \underline{x}_i + b)) \right] \right]$

u.f. Next page

(3). f. $\phi(\underline{x}_i) \in \mathbb{R}^P \quad P \gg p$

$\phi(\underline{x}_i) \in \mathbb{R}^P$

A nonlinear transformation ϕ allows the linear classifier to potentially form a prediction for data that is nonlinearly separable, given an appropriate transformation/nonlinear func. ϕ. Using ϕ to transform the data $\underline{x}_i \in \mathbb{R}^P$ and then calculate labels for the new matrix requires $O(P)$ which is more expensive than the $O(p)$ of using the original data. In some cases, the kernel trick can be used to avoid the costly computation of the P-dimensional transformed matrix and reduce the complexity back to $O(p)$, but the kernel matrix and nonlinear transformation must satisfy Mercer's Theorem $K(\underline{x}_i, \underline{x}) = \phi(\underline{x}_i) \cdot \phi(\underline{x})$. Not all ϕ can be used to make a kernel matrix and apply the kernel trick, in which case the more lengthy $O(P)$ calculation of the transformed data must be done.

?

g. $\alpha \in \mathbb{R}^n$ such that $\underline{w} = \underline{X}^T \alpha$ where $X = [x_1 \dots x_n]^T$

This assumption alone does not guarantee that the kernel trick can be applied, since there must exist an $\alpha \in \mathbb{R}^n$ such that $w = \underline{\Phi}^T \alpha$ where $\underline{\Phi}$ is the transformed data matrix X. This requires that the nonlinear function ϕ is appropriate to compute the transformation, and kernel matrix $K(\underline{x}_i, \underline{x}) = \phi(\underline{x}_i) \cdot \phi(\underline{x})$

?

$$\hat{y}_i = \sigma(\langle \underline{x}_i, \underline{w} \rangle + b) \rightarrow \hat{y}_i = \sigma\left(\underbrace{\alpha \underbrace{\phi(\underline{x})^T}_{\underline{\Phi}}}_{\underline{w}} \underbrace{\phi(\underline{x}_i)}_{\text{transformed } \underline{x}_i} + b\right)$$

$$\hat{y}_i = \sigma\left(\alpha \underbrace{\left(\phi(\underline{x}_i) \cdot \phi(\underline{x})\right)}_{= K} + b\right)$$

$\hat{y}_i$ can be found from kernel matrix K, without explicitly calculating transformed data matrix $\underline{\Phi}$ →

$$\boxed{\hat{y}_i = \sigma(\alpha K(\underline{x}_i, \underline{x}) + b)}$$

Robert Heeter

03/10/2023

ELEC 378

Midterm Cheat Sheet

1. Linear algebra, optimization

Unsupervised: PCA, k-means clustering, hier. clust.

Supervised: linreg, least squares, ridge/lasso reg.

2. Vector space V: α is scalar, $x, y \in V$ $\Rightarrow \alpha x \in V,\ x+y \in V$

Linear combination: $y = \sum_{m=1}^{M} \alpha_m x_m = X\alpha$ (y: vector; x_m: vector; α: vector; ↳ ie. mixing board)

Euclidean 2 norm: $\|x\|_2 = \sqrt{\sum_{i=1}^{p} |x[i]|^2}$ → energy $\|x\|_2^2$

Q-norm: $\|x\|_q = \left(\sum_{i=1}^{p} |x[i]|^q\right)^{1/q}$

∞-norm $\|x\|_\infty = \max_i |x[i]|$ ← peak value of x

Normalize vector $\|x\|_2 = 1$ → scale by $\frac{1}{\|x\|_2}$

Inner product: $x \cdot y = \langle x, y \rangle = y^T x = \sum_{i=1}^{p} x[i]y[i]$

Cosine similarity: $\cos(\theta)_{x,y} = \frac{x}{\|x\|_2} \cdot \frac{y}{\|y\|_2}$ $\quad x \cdot x = \|x\|_2^2$

Orthogonality: $x \cdot y = 0 = \langle x, y \rangle$

3. Cauchy-Schwarz inequality: $0 \le |x \cdot y| \le \|x\|_2 \|y\|_2$

x & y are: most diff when orthogonal; or most similar when collinear/lin. dependent

Basis: vectors $\{b_k\}$ are vector space V, lin. ind. & span V

dimension of V is p $\quad b_k \cdot b_k = 0,\ k \neq l$

Orthogonal basis: $\{b_k\}_{k=1}^{p}$ → elements orthogonal

Orthonormal basis → orthogonal basis & normalized $\|b_k\|_2 = 1$

Inverse of matrix $AA^{-1} = A^{-1}A = I$

↳ B contains orthonormal basis, then $B^{-1} = B^T$

and B is orthogonal matrix

$\underline{x} = \underline{\underline{B}}\underline{a} = \sum_{k=1}^{p} \alpha_k \underline{b}_k$ synthesis; $\underline{a} = \underline{\underline{B}}^T \underline{x}$, $\alpha_k = \underline{x} \cdot \underline{b}_k$ analysis

Data matrix $n \times p$ $[x]_{i,j}$ $\quad \underline{x}_i$: data point (vector) $\quad \underline{x}_i \in \mathbb{R}^p$ $p \times 1$ column vec.

$$X = \begin{bmatrix} \underline{x}_1^T \\ \underline{x}_2^T \\ \vdots \\ \underline{x}_n^T \end{bmatrix}$$ ← vectors are rows

Correlation matrix $X^T X$ → symmetric

$$X^T X = \sum_{k=1}^{n} \underline{x}_k \underline{x}_k^T$$

Centering & normalizing: $\mu_i = \frac{1}{n} \sum_{k=1}^{n} x_k[i]$

$x_j[i] \leftarrow \frac{x_j[i] - \mu_i}{\sigma_i}$ $\quad \sigma_i = \frac{1}{n} \sum_{k=1}^{n} |x_k[i] - \mu_i|^2$

3. $Av = \lambda v$ (λ: eigenvalues; v: eigenvector(s); must be square)

$A = V \Lambda V^{-1}$ → real symmetric matrix (correlation/covariance, for example) have real-valued eigenvals & e-vecs.

Hermitian transpose A^H

$A^H A = AA^H = I$ → unitary matrix A (collection of orthogonal)

SVD: $A = U \Sigma V^H$ (columns of U, U^H, V, V^H are orthonormal bases)

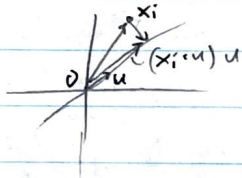
$n \times p$ = $n \times n$ unitary w/ eigenvectors of AA^H · $n \times p$ diagonal w/ singular values (non-neg real #s) · $p \times p$ unitary w/ eigenvals of $A^H A$

Singular value σ: $Av = \sigma u$, $A^H u = \sigma u$

Frobenius norm $\|A\|_F^2 = \sum_{i=1}^{p} \sum_{j=1}^{n} |a_{i,j}|^2$

Rank = # of nonzero singular vals.

4. PCA → Data centered at 0 & normalized



$\|u\|_2 = 1$

$\varepsilon = \sum_{i=1}^{n} (x_i \cdot u)^2$

PCA → $\max_{\|u\|_2 = 1} \sum_{i=1}^{n} (x_i \cdot u)^2 \equiv \|X_c u\|_2^2 = u^T X_c^T X_c u$

↓ maximize energy/spread/variance of data after orthogonal projection onto 1D subspace

$X^T X = V \Lambda V^T \rightarrow \max_{\|u\|_2 = 1} u^T V \Lambda V^T u$

covariance matrix; max is first eigenvector v_1 of $X^T X$

energy is eigenvalue → $\lambda_1 = \sum_{i=1}^{n} (x_i \cdot v_1)^2$

2nd PC is orthogonal to first & is 2nd leading eigenvector

PCA via SVD: $X = U \Sigma V^T$ (V: right singular vectors/columns of V are PCs)

σ_k^2 is energy in PCs

$P = XV = U\Sigma$

5. Objective (loss) fxn $\mathcal{L}$, w_0 is minimizer/max...

Convex fxn: for any x_1, x_2 in f and $0 \le t \le 1$

$f(tx_1 + (1-t)x_2) \le t f(x_1) + (1-t) f(x_2)$

Convex set: $tx_1 + (1-t)x_2 \in S$

Local min $\|w - w^*\| < \delta$ (constraint)

Lagrange multiplier λ: $\min_{w \in \mathbb{R}^p, \lambda \in \mathbb{R}} \mathcal{L}(w) + \lambda g(w)$

Gradient: $\nabla \mathcal{L}(w) = \left[\frac{\partial \mathcal{L}}{\partial w_1}, \frac{\partial \mathcal{L}}{\partial w_2}, \ldots \frac{\partial \mathcal{L}}{\partial w_p}\right]^T$

Quadratic X w $+e$

$$y\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{bmatrix}$$

$= w^t - \mu^t \sum_{t=1}^{T} \nabla \mathcal{L}_t(w)$ ← step size/learn rule

(5). Gradient descent: $w^{t+1} = w^t - \mu^t \nabla \mathcal{L}(w^t)$

↳ Issues: w saddle and not smooth

Stochastic GD: $\mathcal{L}(w) = \sum_{t=1}^{T} \mathcal{L}_t(w)$ ← sum of smaller obj. functions

↳ computationally less expensive; slow convergent & non monotonically

$w^{t+1} = w^t - \mu^t \sum_{t=1}^{T'} \nabla \mathcal{L}_t(w)$

↳ Epoch = all t used grads ← set of gradients picked each iteration (sequence or random)

6. Cluster → homogenous groups; unsupervised

K-means: $\min_{C_1 \dots C_K} \sum_{k=1}^{K} W(C_k)$ ← within cluster variation minimize

↳ $W(C_k)$ w/ 2 norm $= \frac{1}{|C_k|} \sum_{i,i' \in C_k} \|x_i - x_{i'}\|^2$

↳ computation inefficient

finds local min ← 1) Random cluster assignment, 2) Centroid $\frac{1}{|C_k|} \sum_{i \in C_k} x_i$, 3) Assign to closest centroid

Different "distance" metrics i.e. Euclidean

7. Hierarchical clustering → tree-based taxonomy

↳ Agglomerative (bottom up) ↳ Divisive (top-down)

↳ Similarity function → dendrogram tree

1) $\|\ \|_2$ 2-norm
2) $\|\ \|_2^2$ squared 2
3) $\|\ \|_1$ 1 norm
4) $\|\ \|_\infty$ ∞ norm (max dist)
5) Cosine distance

8. x_i → data point/vector y_i → label

$y_i = 1 w_1 + x_i w_2 + e_i$ ← minimize error strength $\sum_{i=1}^{n} |e_i|^2$ (least squares)

$y = Xw + e$

$\min_{w \in \mathbb{R}^2} \underbrace{\|y - Xw\|^2}_{e} \rightarrow \mathcal{L}(w) = \|y - Xw\|^2$ objective fxn

$\nabla \mathcal{L}(w^*) = 0$ @ minimizer w^*

$\nabla \mathcal{L}(w) = \left[\frac{\partial \mathcal{L}(w)}{\partial w_1}, \frac{\partial \mathcal{L}(w)}{\partial w_2}\right]^T$

$\mathcal{L}(w) = \|y - Xw\|^2 = (y - Xw)^T(y - Xw)$

$y^T Xw = w^T X^T y$ ← $= y^T y - y^T Xw - w^T X^T y + w^T X^T Xw$

$= y^T y - 2w^T X^T y + w^T X^T Xw$

$\nabla_z z^T a = a, \quad \nabla_z z^T A z = zA z$

$\nabla \mathcal{L}(w) = -2X^T y + 2X^T Xw = 0 \rightarrow X^T y = X^T Xw$

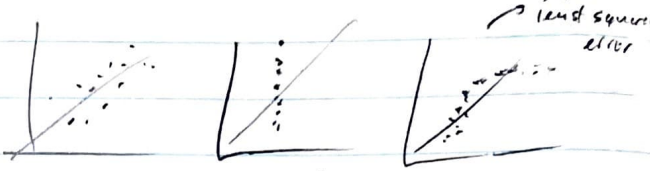
Normal eqns: $X^T y = X^T X w^*$

$w^* = \underbrace{(X^T X)^{-1} X^T}_{= X^+ \text{ Moore Penrose}} y$

$X = U \Sigma V^T, \quad X^+ = V \Sigma^+ U^T$

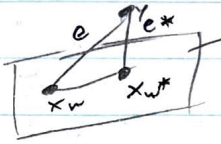
↑ reciprocal of nonzero entries of Σ

8. $w^* = (X^T X)^{-1} X^T y \qquad \hat{y} = Xw^*$



Orthogonality principle $X^T e^* = 0$

↳ optimal error vector e^* is orthogonal to each row of X^T and ∴ each column of X



col(X) → e^* orthogonal to any Xw

$\mathbb{R}^n$

Include column of 1s for y-intercept term w_0

Center data by subtracting mean of y and each column of X

$1^T y = 0, \quad 1^T X = 0^T$

→ w_1 is unidentifiable and ∞ w_1's solve normal eqns

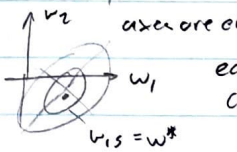
9 $(X^T X)^{-1}$ does not exist → $X^T X$ is singular or ill conditioned

1) $n \geq p$ (tall, narrow X) but rank(X) $< p$

↳ 2+ columns of X (and $X^T X$) are lin. dependent

2) $n < p$ (short, wide X) → Overfitting; more params w_j than data points (x_i, y_i) model fit to noise

LS Geometry



axes are eigenvectors

eccentricity is from condition #

tradeoff λ

variance ↓ | bias ↓

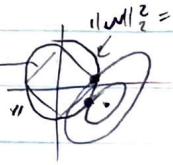
Regularization w/ constraint: $\min_{w \in \mathbb{R}^p, \lambda \in \mathbb{R}} \|y - Xw\|_2^2 + \lambda \|w\|_2^2$

$X^T y = (X^T X + \lambda I) w_{ridge} \rightarrow w_{ridge} = V(\Sigma + \lambda I)^{-1} U$

Ridge — always well-conditioned

Lasso → 1 norm penalty $\|w\|_1 = c$

$\|w\|_2^2 = c$



w_{lasso} is sparse, "variable selection"

Elastic net → combine 2 penalties

Need optimization alg. for lasso

Robert Heeter
5/1/2023
ELEC378

Final Cheat Sheet

11. SGD: $\mathcal{L}(w) = \sum_{m=1}^{n} \mathcal{L}_m(w)$
↳ select set of grads. (mini batch)
GD for ridge & lasso
$w^{t+1} = w^t + \mu^t \sum_{m \in M(t)} \nabla \mathcal{L}_m(w^t)$

① Ridge via GD:
$O(np)$ ← $\min_{w \in \mathbb{R}^p} \|y - Xw\|_2^2 + \lambda \|w\|_2^2 = \min_{w \in \mathbb{R}^p} H(w)$
$\nabla H(w) = -2X^T(y - Xw) + 2\lambda w$

② Lasso via GD: ← norm not differentiable @ $w=0$
$\min_{w \in \mathbb{R}^p} \|y - Xw\|_2^2 + \lambda \|w\|_1 = \min_{w \in \mathbb{R}^p} G(w)$
↳ added condition for when $w = 0$
$O(np)$ ← $\nabla G(w) = -2X^T(y - Xw) + \lambda \, \mathrm{sgn}^*(w)$

12. LMS
Centered data; minimize error energy $\|e\|^2$
Time signals → window p samples $= \|y - Xw\|^2$
Wiener Filter ← w = impulse response
$y = Xw + e$
↑ Toeplitz matrix, Xw = convolution
☆ LMS algorithm: $w^{t+1} = w^t + \mu x_t (y_t - (x_t \cdot w^t))$
$O(p)$ ← ↳ System ID, linear prediction, L.P.C.

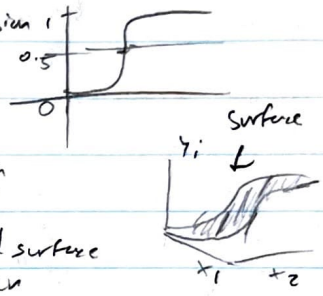
13. Classification
x_i → data point, feature, signal
y_i → label
$y_i = f(x_i)$ ← predictor
{ regression $y_i \in \mathbb{R}$; classification $y_i \in \{0,1\}$ or larger set }
- Decision boundaries
- Error rates TPR, FPR, TNR, FNR
- Confusion matrix

	predicted class	
actual class		

13 KNN
No optimization
1. Test data point x_{test}
2. Find K points closest to x_{test} by some metric
3. $f(x_{test}) =$ majority vote of the y_i's of K nearest neighbors
- Good performance, handles many classes
- Computationally expensive $O(np)$ for 1 test point
↳ Good for small data set

15. Logistic Reg
$P(y_i = 1) = \sigma(w \cdot x_i + b)$
↑ scalar nonlinearity $\sigma(u) = \frac{1}{1+e^{-u}}$
$y = \begin{cases} 0 & \text{if } P(y=1) = \sigma(w \cdot x + b) < 1/2 \\ 1 & \text{if } P(y=1) = \sigma(w \cdot x + b) \geq 1/2 \end{cases}$
b → shifts sigmoid; sets classification threshold

multiple logistic regression:
↳ w rules hyperplane level curves of sigmoid surface in $\mathbb{R}^{p+1}$
↳ b → shifts sigmoid surface in w direction



Objective fxn: $\|y_i - \sigma(x_i \cdot w + b)\|_2^2$ → non convex due to sigmoid
↳ use negative cross entropy
$\mathcal{L}(w,b) = \sum_{i=1}^{n} -y_i \log(\sigma(x_i \cdot w + b)) - (1 - y_i) \log(1 - \sigma(x_i \cdot w + b))$ ↳ convex!
Derivative of sigmoid
$\sigma(z) = \frac{1}{1+e^{-z}} \rightarrow \sigma'(u) = \sigma(u)(1 - \sigma(u))$
Gradient step: $w^{t+1} = w^t + \mu^t \sum_{i=1}^{n} x_i (y_i - \sigma(x_i \cdot w^t + b^t))$

- Binary classifier
↳ one vs all classifier for $k > 2$ classes
↳ classify each class vs rest of training

Multinomial log. reg. (softmax)
↳ softmax nonlinearity: ↳ convert K outputs to probabilities
$\frac{1}{\sum_{k=1}^{K} \exp(x_i \cdot w_k + b_k)} \begin{bmatrix} \exp(x_i \cdot w_1 + b_1) \\ \vdots \end{bmatrix}$
↓ measure distances b/w histograms w/ negative cross entropy
objective fxn: $\mathcal{L}(w_1, b_1, \ldots w_K b_K)$
$= -\sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1} \log \frac{\exp(x_i \cdot w_k + b_k)}{\exp(x_i \cdot w_j + b_j)}$
$\nabla_{w_k} \mathcal{L} = -\sum_{i=1}^{n} x_i \left(\mathbb{1}(y_i = k) - P(y_i = k) \right)$

Perceptron
$\hat{y} = \sigma(x \cdot w + b)$ ← bias
scalar nonlinearity/ activation function
↳ sigmoid, hyperbolic tangent, ReLU
↳ $\hat{y} = \sigma(Wx + b)$ ← applied component wise
weight matrix; offsets vector

(15) MLP → multiple layers

DNN → many params in $W_1 \dots W_L$ and $b_1 \dots b_L$

↳ constrain $W_1 \dots W_L$ to be circulant convolution matrices w/ short circular impulse response

↳ reduce # of params → CNN!

↳ Final activation σ^* → ReLU for regression

σ^* → softmax for classif.

optimize params W, b w/ SGD

backprop = chain rule in calc

16. SVM → 2 classes

hyperplane $0 = f(\underline{x}) = \underline{w} \cdot \underline{x} + b$

$y = \text{sign}(f(\underline{x}))$

#3 structural SVM complex data + labels

$\underline{w}, b$ not unique

#2 $k(k-1)/2$ SVMs b/w each pair of classes

↳ choose class selected by most SVMs

multiclass SVM:

#1 learn k one vs all 2-class SVMs

each class vs rest of data

given test data point $\underline{x}$ choose SVM w/ greatest margin

$\underline{w}$ orthogonal to hyperplane

$\frac{b}{\|\underline{w}\|_2}$ distance from origin to plane

$p-1$ dimensional hyperplane

maximum margin hyperplane

margin → maximize $\frac{\text{margin}}{2} = \frac{1}{\|\underline{w}\|}$

support vector $\underline{w}^* \cdot \underline{x} + b = \pm 1$

$\max_{w \in \mathbb{R}^p, b \in \mathbb{R}} \frac{2}{\|\underline{w}\|}$ subject to $\underline{w} \cdot \underline{x}_i + b \begin{cases} \geq +1 \text{ when } y_i = +1 \\ \leq -1 \text{ when } y_i = -1 \end{cases}$

↳ convex problem w/ unique global minimum

ξ slack variables → $\min_{w,b,\xi} \left(\|\underline{w}\|_2 + C \sum_{i=1}^{n} \xi_i \right)$

C = regularization param / makes constraints easy / hard to ignore if large / small

large → small margin but fewer errors

small → opposite

$\xi_i \geq 0$

$y_i (w \cdot x_i + b) \geq 1 - \xi_i$

Hinge loss $\xi_i = \max\{0, 1 - y_i(\underline{w} \cdot \underline{x}_i + b)\}$

↳ violates constraint = contribution to hinge L

driving hinge loss down drives margin up

now: $\min_{w,b} \|\underline{w}\|_2 + C \sum_{i=1}^{n} (1 - y_i(\underline{w} \cdot \underline{x}_i + b))$

$J(w) = \min_{w,b} \sum_{i=1}^{n} (1 - y_i(\underline{w} \cdot \underline{x}_i + b)) + \lambda \|\underline{w}\|_2$ ($\lambda \propto \frac{1}{C}$)

SVM GD. $O(np)$

$\nabla_w J(\underline{v}, b) = \sum_{i=1}^{n} s_i^w(\underline{w}) + 2\lambda \underline{w}$

$\nabla_b J(\underline{v}, b) = \sum_{i=1}^{n} s_i^b(b)$

Nonlinear separable:

replace $n \times p$ data matrix X w/ $n \times P$ data matrix $\Phi = \Phi(X)$

$\underline{x}_i \to \phi(\underline{x}_i)$ column of X

↳ adds new terms / dims to data matrix

17. Invariant to nuisance transformations

Feature engineering → edge detection, cepstrum, SIFT

Feature learning → sparse coding

Deep learning → DNN

Cepstrum features → cepstrum extracts elements from time signal x

mel-freq. cepstrum audio signals

$F^{-1}[\log(|Fx|^2)]^2$

F^{-1} ↑ Fourier transform

Edge detection

↳ steep gradients

SIFT → scale invariant feature transform → scale space, pyramid transform

dictionary matrix D → $\underline{x} = D\underline{w}$ (D $p \times m$, $m > p$)

find w w ridge / lasso

sparse coding

$\min_{D,w} \|X^T - Dw\|_F^2 + \lambda \|w\|_{1,1}$

$p \times n$ transpose of data matrix; coeff. matrix; 1-norm (sum of magnitudes); Frobenius norm $\Sigma\Sigma$ squares

overcomplete dictionary → bi convex

18. Ridge reg: $\min_{w \in \mathbb{R}^p} \|\underline{y} - \underline{X}\underline{w}\|_2^2 + \lambda \|w\|_2^2$

normal eqs $X^T y = (X^T X + \lambda I) w$

primal soln $w = X^T \left(\frac{1}{\lambda} (y - Xw) \right) = X^T a$ ← dual soln

$\min_{\alpha \in \mathbb{R}^n} \|y - XX^T \alpha\|_2^2 + \lambda \|X^T \alpha\|_2^2$

$K = XX^T$ ($n \times n$ dims (not $p \times p$))

↳ solve for α

$\hat{y} = \hat{f}(\underline{x}) = \underline{w}^* \cdot \underline{x} = (\underline{x}^T \alpha^*)^T \underline{x} = \alpha^{*T} \underline{x}_*$

$\hat{y}_i = \alpha_i^T (\underline{x}_i \cdot \underline{x}_*)$

$K = \Phi \Phi^T \to n \times n$

$[K]_{ij} = \phi(\underline{x}_i)^T \phi(\underline{x}_j) = \phi(\underline{x}_i) \cdot \phi(\underline{x}_j)$

$\hat{y} = \alpha^T \Phi \phi(\underline{x}) \to \hat{y}_i = \alpha_i \phi(\underline{x}_i) \cdot \phi(\underline{x})$

Mercel's Thm: class of fcns ϕ such that $\phi(\underline{x}_i) \cdot \phi(\underline{x}) = K(\underline{x}_i, \underline{x})$

↳ mapping fcn don't need to work in p-dim space

Kernels: Polynomials $K(\underline{u}, \underline{v}) = (\underline{u} \cdot \underline{v})^2$

Gaussian → radial basis fcn

sigmoid tanh

19 → Dropout

20. Feature selection → statistical tests, correlation matrix, PCA

Cross validation → divide matrix & average accuracies from each test group

overfitting as $p \to n$

↳ dropout, regularization, early stopping