

CSCE 509, Pattern Recognition, Spring 2019

Class Notes

Brad Burkman

Updated 22 April 2019



Contents

1	16 January	lecture 00.pdf	Syllabus and Conventions	3
1.1	Assignments			3
1.2	References			3
1.3	Conventions			3
1.4	Topics			3
1.5	Implementation			4
2	16 January	lecture 01.pdf	What is Pattern Recognition?	4
2.1	Basic Example I			4
2.2	Probability Vocabulary			4
2.3	Basic Example II			5
2.4	Pattern Recognition			6
2.5	Course Topics related to Basic Example II			7
3	28 January	lecture 02.pdf	Optimal Decision Theory, Slides #1-32	7
3.1	Review, Slides #1-12			7
3.2	Gaussian Density (Bell Curve)			8
3.3	Class Density and Posterior Probability			8
3.4	Bayes Rule			8
3.5	Vocabulary			9
3.6	Final Thoughts			9

4	30 January	9
4.1	Gaussian Functions	9
4.2	Covariance Matrix	11
4.3	Deriving the Optimum Decision Rule	11
4.4	Prior and Posterior Probability	11
4.5	Gaussian Density Function	12
4.6	1-D Gaussian Density Function	12
5	18 February Lecture	13
5.1	Things Missed Last Lecture	13
6	20 February Lecture	13
7	11 March Lecture	14
7.1	Review	14
7.2	New Stuff	14
8	13 March Lecture	15
8.1	k -means Clustering	15
8.2	Hierarchical Clustering	15
8.3	Self-Organizing Map	16
8.4	After Clustering	16
8.5	Next Week: Python Implementation	16
9	18 March Lecture: sklearn in Python	16
9.1	Python	16
9.2	k -nn Classifier	16
10	25 March Lecture: DBSCAN	17
10.1	Supervised Learning	18
11	27 March Lecture: Clustering	18
12	1 April Lecture: Classification by Supervised Learning	19
12.1	TensorFlow Solution	19
12.2	Batch Gradient Descent Algorithm	19
12.3	Assignment	19
12.4	Coming on Wednesday	19
13	3 April Lecture	19
13.1	Second Homework	19
13.2	Graph Descent Algorithms	19
13.3	TensorFlow Solution	20
13.4	Mini-Batch	20
13.5	Future	20
14	8 April Lecture	20
14.1	Review	20
14.2	Mini-Batch using TensorFlow	20

15 22 April Lecture: TensorFlow and Neural Networks	21
15.1 Random Notes as He Went Through the Code	21
15.2 “This is how this is going to work”	21

1 16 January lecture 00.pdf Syllabus and Conventions

1.1 Assignments

All coding assignments require a written report.

1.2 References

- Duda, Hart, and Stork, *Pattern Classification*, Second Edition, Wiley, 2001
- Ripley, *Pattern Recognition and Neural Networks*, Cambridge UP, 1996
- Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006

1.3 Conventions

A Upper-case bold Matrix

a Lower-case bold Vector Column vector

$$\underset{m \times p}{\mathbf{A}} = \underset{m \times n}{\mathbf{B}} \underset{n \times p}{\mathbf{C}} \quad \mathbf{a}_{ij} = \sum_{k=0}^{n-1} \mathbf{b}_{ik} \mathbf{c}_{kj}$$

In matrix-vector multiplication, the matrix is often considered an operator on the vector.

1.4 Topics

Up to ten years ago, these methods were known as “pattern recognition” methods. Today they’re called “data science” methods.

- Optimal Decision Theory: “Known Solutions”
- Unsupervised Methods
 - Projection Methods
 - * Principal component analysis
 - * Independent component analysis
 - Clustering
 - Self-organizing Maps
- Supervised Methods
 - Linear Classifiers
 - Artificial Neural Networks
 - * Deep Learning

- Kernel Methods
- Nearest Neighbor Classifier
- Learning Vector Quantizers



1.5 Implementation

Previously taught in R, this semester in Python.

2 16 January lecture 01.pdf What is Pattern Recognition?

The field of pattern recognition is concerned with the automatic discovery of regularities in data through the use of computer algorithms and with the use of these regularities to take actions such as classifying the data into different categories. (Bishop, 2006)

2.1 Basic Example I

1. Flip a coin
2. If (coin=="head"), choose one of these balls. 
3. If (coin=="tail"), choose one of these balls. 
4. Given the observation of the drawn ball, decide the outcome of the coin toss.
Observe blue, decide "head."
Observe red, decide "tail."
5. Suppose I pay you \$1 for every correct call, and you pay me \$1 for each incorrect call. After 100 calls, how much money do you think you'll have?
\$100.
6. Optimum Decision: What is the maximum amount that you can expect to have given the setup of the game?



2.2 Probability Vocabulary

- Event
A subset of the set of all possible outcomes
 - Head, tail
 - Blue ball, red ball
- Probability
A measure defined for each event
 $P(\text{head}) = P(\text{tail}) = \frac{1}{2}$
- Random variable
A real-valued function defined on an event.
 - Assigns a real value to an event

- Has no relationship to the likelihood of the event
- Assignment of the value has to do with the problem.
- Not random, despite the name.
- Decision Rule
 - To optimize the expected value, we can change the decision rule.
- Expected value (of a random variable)
 - Sum of all values taken on by the random variable weighted by the probability of the associated event.
 - An event has no expected value. Only a random variable has an expected value.
 - We will talk about “risk” in this course.
- Example

Event	Coin toss results in a tail.
Random variable	Assign “1” if the coin toss results in a head. $x(\text{head}) = 1$ $x(\text{tail}) = 0$
Expected value	$E(x) = 1 \times P(x = 1) + 0 \times P(x = 0)$ $= 1 \times P(\text{Head})$ $= 1 \times \frac{1}{2}$ $= \frac{1}{2}$
- Possible questions about this example
 - What are the events?
 - What are the probabilities?
 - What is the random variable?
 - Why do we care about the expected value?
 - What is the expected value of the coin toss? [Trick question]

2.3 Basic Example II

1. Flip a coin
 2. If (coin==“head”), choose one of these balls. 
 3. If (coin==“tail”), choose one of these balls. 
- Events
 - Head, tail
 - Blue ball, red ball

- Probability

$$P(\text{head}) = P(\text{tail}) = 1/2$$

$$P(\text{blue}|\text{head}) = 3/4$$

$$P(\text{red}|\text{head}) = 1/4$$

$$P(\text{blue}|\text{tail}) = 1/4$$

$$P(\text{red}|\text{tail}) = 3/4$$

$$P(\text{red}) = P(\text{red}|\text{head})P(\text{head}) + P(\text{red}|\text{tail})P(\text{tail})$$

$$= \frac{1}{4} \cdot \frac{1}{2} + \frac{3}{4} \cdot \frac{1}{2} = \frac{1}{2}$$

$$P(\text{blue}) = P(\text{blue}|\text{head})P(\text{head}) + P(\text{blue}|\text{tail})P(\text{tail})$$

$$= \frac{3}{4} \cdot \frac{1}{2} + \frac{1}{4} \cdot \frac{1}{2} = \frac{1}{2}$$

- Random variable

Suppose A pays B \$1 for every correct call, and B pays A \$1 for every incorrect call.

Define x to be the amount of money that B wins (loses).

Assign “1” if the call is correct.

Assign “-1” if the call is incorrect.

$$x(\text{correct}) = 1$$

$$x(\text{incorrect}) = -1$$

$$x = \begin{cases} +1 & \text{if the decision is correct} \\ -1 & \text{if the decision is incorrect} \end{cases}$$

- Decision Rule

$$d(\text{observation}) = \begin{cases} \text{head} & \text{if the observation is blue} \\ \text{tail} & \text{if observation is red} \end{cases}$$

$$P(\text{correct}) = P(\text{blue}|\text{head}) \times P(\text{head}) + P(\text{red}|\text{tail}) \times P(\text{tail})$$

$$= \frac{3}{4} \cdot \frac{1}{2} + \frac{3}{4} \cdot \frac{1}{2} = \frac{3}{4}$$

$$P(\text{incorrect}) = 1 - P(\text{correct}) = \frac{1}{4}$$

- Expected Value

$$E(x) = +1 \times P(\text{correct decision}) - 1 \times P(\text{incorrect decision})$$

$$= +1 \times \frac{3}{4} - 1 \times \frac{1}{4} = \frac{1}{2}$$

2.4 Pattern Recognition

Based on the observed color of the drawn ball, and knowing the probability of the coin toss and composition of each of the buckets, decide the outcome of the coin toss.

In practice, we do not know the exact composition of each of the buckets. That is why we need to do learning (supervised or unsupervised).

Coin is usually not fair. Example: Airport security. There is not an even chance that a suitcase contains contraband.

2.5 Course Topics related to Basic Example II

In Optimum Decision Theory, we know the composition of the balls.

Sometimes you don't know whether it's a two-class problem. Coin flipping is a two-class problem; dice rolling is a six-class problem. Use **unsupervised methods** when you don't know how many classes there are.

If you know things about the data (not from observing the data, but from the outside), then it's a supervised learning problem.

Pattern Recognition: Come up with decision rules that minimize the risk (expected loss).

Use **linear classifiers** when you know the data distribution is Gaussian. Also use it when you don't have enough data to use learning methods; you have to assume the distribution is Gaussian.

Use **kernel methods** if you know the distribution is not Gaussian. Major pain to implement. Statistical density estimation (?)

Nearest neighbor classifiers are an approximation of kernel methods.

Learning vector quantizers Precursor to self-learning maps.

3 28 January lecture 02.pdf Optimal Decision Theory, Slides #1-32

3.1 Review, Slides #1-12

The field of pattern recognition is concerned with the automatic discovery of regularities in data through the use of computer algorithms and with the use of these regularities to take actions such as classifying data into different categories. (Bishop, 2006)

How do we make the classifications?

What is the optimal decision rule?

It's more realistic that we're trying to avoid penalties for incorrect decisions, rather than trying to get rewards to correct decisions.

What are we optimizing? Candidates:

- Maximize the probability of a correct decision for a particular class
- Maximize the probability of a correct decision for all classes.

Process

- Define a loss function for an incorrect decision for each class.
- *Risk* is then the expected value of the total loss function.

- The optimal decision rule is the one that minimizes the total risk.

Slide #16: Events can be discrete or continuous

3.2 Gaussian Density (Bell Curve)

- Probability density function
- Always talk about an interval and the area under the curve over that interval.
- Area under a Gaussian curve does not have an analytical solution.
- Distributions with the same mean may happen
- Multiple densities: If the distributions are Gaussian, the solution is known. The problem is that you may not know the distributions.

3.3 Class Density and Posterior Probability

Class Density	Posterior Probability
$p(x k)$	$p(k x)$
Problem-specific	Derive from class densities and prior probabilities
Outcome given class	Class given outcome

3.4 Bayes Rule

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Prior probabilities	Class densities	Posterior probability
Class k is chosen by the source with probability	Class density k drives the observed value x .	Given that the value x is observed, the probability that it was from class k .
π_k for $k \in \{0, 1, \dots, K-1\}$	$p(x k) = p_k(x)$ for $k \in \{0, 1, \dots, K-1\}$	$p(k x)$ for $k \in \{0, 1, \dots, K-1\}$
	$p(k x) = \frac{p(k, x)}{p(x)} = \frac{p(x k)\pi_k}{p(x)}$	

Question: Why do we use π_k instead of $p(k)$?

Question: What does the notation $p(k, x)$ mean?

Maximum a posteriori rule:

Choose class k if $p(k, x) > p(k'|x)$ for all $k' \neq k$

Choose class k if $p(x|k)\pi_k > p(x|k')\pi_{k'}$ for all $k' \neq k$.

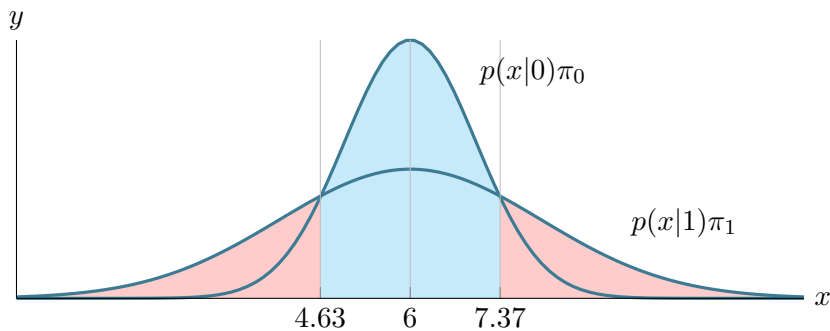
NOT if $p(x|k) > p(x|k')$. DO NOT ASSUME $\pi_k = \pi_{k'}$.

The $p(x)$ in the denominator doesn't matter because it's the same for all classes.

3.5 Vocabulary

Random source	Like dice or coin
Prior probability	Known before you see the outcome. Dangerous to assume that they're equally likely.
Class	Each outcome is a class. Number of classes is always discrete.
Class densities	Can be a density; in practice, can be a probability.
Posterior probability	

If posterior probability is higher, call that class.



$$d = \begin{cases} k = 0 & \text{if } 4.63 < x < 7.37 \\ k = 1 & \text{otherwise} \end{cases}$$

3.6 Final Thoughts

- Gaussian distribution is not required
- The choice that minimizes the risk is optimum
- The posterior rule is also optimum
- Loss function can be really complicated

Finished at Slide #36.

4 30 January

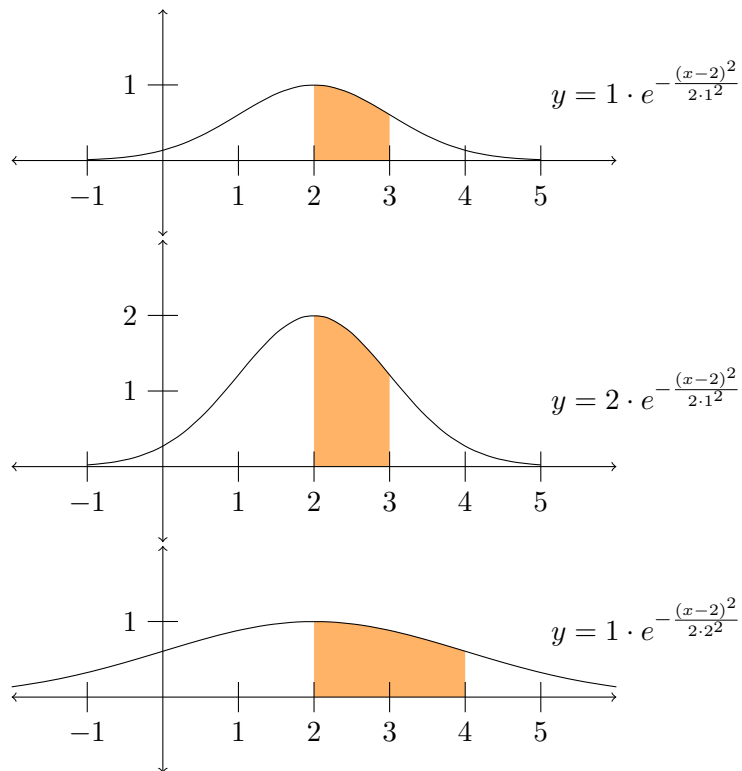
To Do:

- Work through Wikipedia page on “Gaussian Function.”
- Learn what a covariance matrix is.
- Work through slides and figure out the math.

4.1 Gaussian Functions

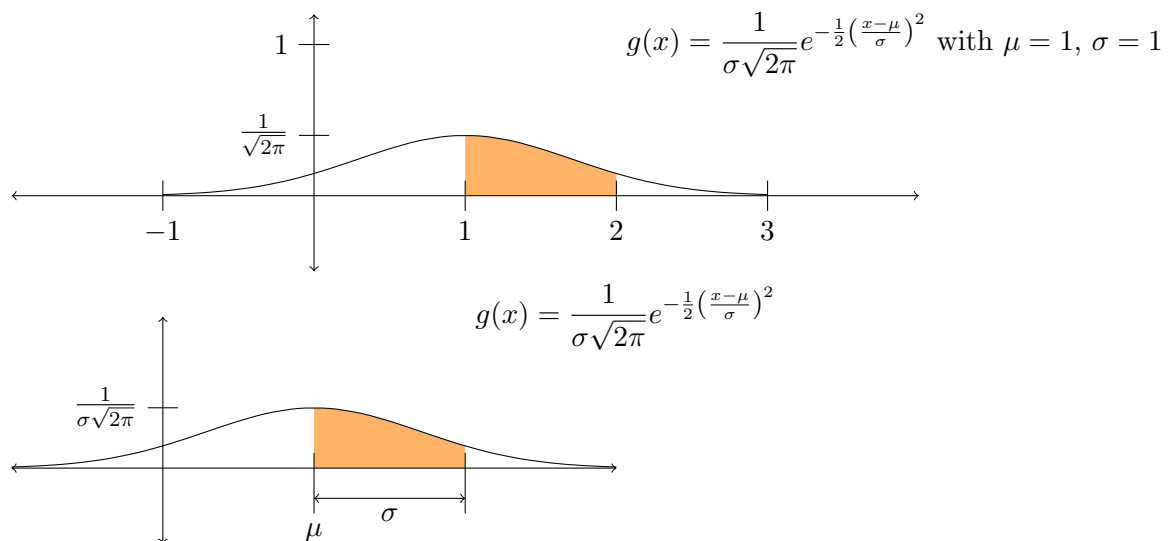
$$f(x) = ae^{-\frac{(x-b)^2}{2c^2}}$$

- a height of the curve's peak
- b position of the center of the peak
- c standard deviation



For a probability density function of a normally distributed random variable with expected value $\mu = b$ and variance $\sigma^2 = c^2$, we get

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$



4.2 Covariance Matrix

In probability theory and statistics, a **covariance matrix**, also known as auto-covariance matrix, dispersion matrix, variance matrix, or variance-covariance matrix, is a matrix whose element in the i, j position is the covariance between the i -th and j -th elements of a random vector. [Wikipedia]

In probability theory and statistics, covariance is a measure of the joint variability of two random variables.[1] If the greater values of one variable mainly correspond with the greater values of the other variable, and the same holds for the lesser values, (i.e., the variables tend to show similar behavior), the covariance is positive.[2] In the opposite case, when the greater values of one variable mainly correspond to the lesser values of the other, (i.e., the variables tend to show opposite behavior), the covariance is negative. The sign of the covariance therefore shows the tendency in the linear relationship between the variables. The magnitude of the covariance is not easy to interpret because it is not normalized and hence depends on the magnitudes of the variables. The normalized version of the covariance, the correlation coefficient, however, shows by its magnitude the strength of the linear relation. [Wikipedia]

4.3 Deriving the Optimum Decision Rule

If we know the prior probabilities and class densities, then the optimum decision rule is known.

One approach:

- Define a loss function for an incorrect decision for each class.
- Define the total risk as the expected value of the loss function over all classes.
- The optimum decision rule is the one that minimizes the total risk.

4.4 Prior and Posterior Probability

Prior probabilities Class k is chosen by the source with probability π_k for $k \in \{0, 1, \dots, K-1\}$.

Class densities Class density k drives the observed value x .

$$p(x|k) = p_k(x) \text{ for } k \in \{0, 1, \dots, K-1\}$$

Posterior probability Given that the value x is observed, the probability that it was from class k .

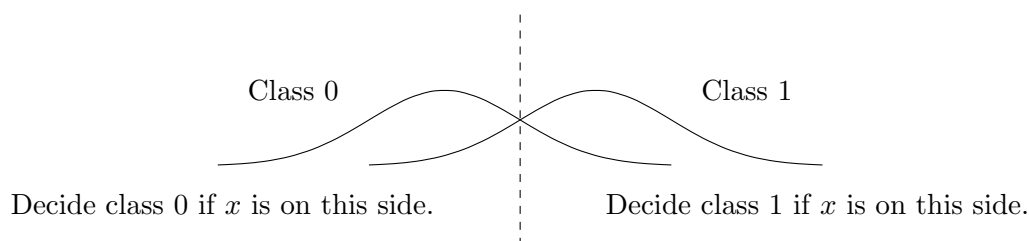
$$p(k|x) \text{ for } k \in \{0, 1, \dots, K-1\}.$$

$$p(k|x) = \frac{p(k, x)}{p(x)} = \frac{p(x|k)\pi_k}{p(x)}$$

Maximum a posteriori rule Choose class k if $p(k|x) > p(k'|x)$ for all $k' \neq k$.

i.e. $p(x|k)\pi_k > p(x|k')\pi_{k'}$ for all $k \neq k'$.

CAUTION: Not $p(x|k) > p(x|k')$ for all $k \neq k'$. You cannot assume that the π_k 's are equal.



4.5 Gaussian Density Function

$$p(x) = \frac{1}{(2\pi)^{p/2}} \cdot \frac{1}{|\Sigma|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\mathbf{m})^T \Sigma^{-1}(\mathbf{x}-\mathbf{m})}$$

Notes:

p is the number of dimensions.

\mathbf{x} is the observation.

\mathbf{m} is the mean.

Σ is the covariance matrix.

$\mathbf{x} - \mathbf{m}$ is a column vector.

$\Sigma^{-1}(\mathbf{x} - \mathbf{m})$ is a column vector.

$(\mathbf{x} - \mathbf{m})^T$ is a row vector.

$(\mathbf{x} - \mathbf{m})^T \Sigma^{-1}(\mathbf{x} - \mathbf{m})$ is a scalar.

The Gaussian density for a p -dimensional vector x has two parameters, the mean vector and the covariance matrix.

The density is centered at the mean.

The spread of the density is controlled by the covariance matrix.

4.6 1-D Gaussian Density Function

Relating $p(x) = \frac{1}{(2\pi)^{p/2}} \cdot \frac{1}{|\Sigma|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\mathbf{m})^T \Sigma^{-1}(\mathbf{x}-\mathbf{m})}$ to $g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$

p is the dimensionality; in this case, $p = 1$.

$\mathbf{m} = \mu$

Since $\Sigma = E[(\mathbf{x} - \mathbf{m})(\mathbf{x} - \mathbf{m})^T]$, in the 1-D case, $|\Sigma| = (x - m)^2 = (x - \mu)^2 = \sigma^2$

In the 1-D case, $\mathbf{x} - \mathbf{m}$ is a 1-vector, so $(\mathbf{x} - \mathbf{m})^T = \mathbf{x} - \mathbf{m}$, and we can treat them as scalars.

Similarly, Σ is a 1×1 matrix, so we can treat it as a scalar, so $\Sigma^{-1} = 1/\Sigma$, and the multiplication commutes.

$$(\mathbf{x} - \mathbf{m})^T \Sigma^{-1}(\mathbf{x} - \mathbf{m}) = (x - \mu) \frac{1}{\sigma^2} (x - \mu) = \left(\frac{x - \mu}{\sigma} \right)^2$$

5 18 February Lecture

5.1 Things Missed Last Lecture

Two linear classifiers, there's a weight update.

Perceptron algorithm: Target is -1 or 1, not 0 or 1. No weight update if target is correct.

η sometimes written as $\eta/2$.

Geometric understanding: Moving perpendicular to the boundary.

Talked about what the Widrow Hoff and Perceptron algorithms assume and guarantee, but I didn't get it all. Look it up.

Widrow Hoff does not guarantee that it will minimize the error.

Widrow Hoff solves the Least Squares problem. Computationally easy way to get a least squares solution.

Use the rectified data set to derive the Fischer Linear Discriminant.

"Do not fear the messy terms."

Solved for w_p in terms of \mathbf{w} . Don't know either yet.

The $S = S_1 + S_0$ equation is worth understanding.

Fisher linear discriminant frequently used in making classifications.

Covariance matrix is the same as the FLD, within a constant multiple.

How do we use the FLD if so many things are unknown?

FLD gives a weight vector, but we will not use it for classification. It reduces the dimensionality to a 1-D classification problem.

1D Classification Problem

If distribution is linear, then FLD is called Linear Discriminant Analysis.

6 20 February Lecture

`studentweb.cmix.louisiana.edu`

`python` defaults to 2.7.

Use `python3` to use Python 3

Libraries

- `numpy`
- `scipy`
- `scikit-learn`

```
pip install -U scikit-learn
```

In Python, be careful as to what is a list and what is an array. Matrices are also different.

Use Numpy arrays.

In numpy, have to explicitly say that it's an array. Python lists won't work.

Note on the slides: There is no `xrange` in Python3. You can use `range` in Python3 like `xrange` in Python2.

Dr. Chu suggests Python2 instead of Python3.

For fun in homework, make a data set that is linearly separable, and watch Perceptron flop.
Homework: Finish Perceptron example, make another data set.

.cov covariance matrix

Homework to be posted by Thursday morning.

Due in three weeks, 13 March

7 11 March Lecture

7.1 Review

What do we mean by “discussion” in assignments? Analyze whether the solution is correct.

When data is not linearly separable, Perceptron will not give a good solution. When Dr. Chu changed `tvec` to `tbad` by swapping two points, he created a lot of other errors.

Important to know what **cross validation** is.

7.2 New Stuff

We’ve done parametric methods. Next moving to non-parametric methods (kernel methods, etc).
Then massively parametric (neural networks and deep learning).

For now, doing unsupervised learning.

- Too many features.
- Need to reduce dimensionality.
- Projection.
- Fischer says that his method gives the most interesting projection.
- Fischer linear projection more difficult than the projection methods here.

$$\underset{q \times 1}{\mathbf{y}} = \underset{q \times p}{\mathbf{M}} \underset{p \times 1}{\mathbf{x}}$$

- “Centered data”: Column means are zero.
- In his slides, the @ means matrix multiplication.
- Think of projection as rotating and scaling.
- PCA takes a slanted elliptical, rotates it to align with one of the axes, scales it to a circle, and rotates it back.
- “Sphering” or “whitening” the data: All column means are zero.
- Running PCA does not solve everything. Several examples in the slides of where PCA would fail.
- k -means clustering, where k is the number of clusters.

- If you continue the algorithm based on the distance from each point in a cluster to the cluster center, you get n clusters for n points.
- Transformed distortion, peak of jump, may give the best indication of the best number of clusters.
- k -means algorithm is easy. Choosing k is not so easy.

8 13 March Lecture

8.1 k -means Clustering

k -means clustering algorithm. Assume you know what k is.

Iterate until the change is less than some tolerance.

Probably the best approach is to start with one cluster center, split into two, split each of those in two, until you get the best 2^n number of clusters.

Need to be able to measure the “goodness” of the cluster.

Candidates:

- Mean squared distance to the cluster center.
- Measures of within-cluster scatter matrix.

To find the mean squared distance of all samples, find the mean squared distance for each cluster, and weight by the number of points in each cluster.

Determinant of Within-Cluster Scatter Matrix is a useful way to compare results for different k .

Scatter matrix is a type of *distortion measure*.

To find the best value of k , find the jump value for each k and find the largest jump value.

Want clusters to be compact and separated. Often can't have both.

Ways to measure clustering results. All are based on a ratio of the compactness and separation.

- Calinski-Harabasz Index is easiest to understand.
- Dunn's Index fairly famous.
- Generalized Dunn's Index
- Silhouette Index

8.2 Hierarchical Clustering

Organized in a tree

Agglomerative: Bottom-up, starting with n clusters for n data points.

Divisive: Top-down, splitting clusters

Need to decide how to measure the distance between two clusters: Minimum distance, Average distance, Maximum distance

Dendrogram: Visualization of the tree.

8.3 Self-Organizing Map

Lost interest over the last twenty years, but coming back into vogue with machine learning.

8.4 After Clustering

Use the clusters as “labeled” data for supervised learning.

Estimate the density if you can’t assume it’s Gaussian.

Kernel function. Nonparametric method. Expensive because you have to use all of the class samples, and to make a decision you have to go through both lists for classes 0 and 1.

The kernel width needs to be bigger in areas with fewer samples. In overlapping areas, trouble.

One option is to make the kernel width large enough to include 3 data samples. Problem is that you’re no longer using density estimation.

8.5 Next Week: Python Implementation

No class next Wednesday 20 March

9 18 March Lecture: sklearn in Python

9.1 Python

Each row is an observation.

Each column is a feature.

`x,y = dataMat.T` is a way to separate the data into a list of x values and a list of y values to pass into pyplot.

`KMeans` is a class in `sklearn`

Assignment coming soon: Evaluating performance.

9.2 k -nn Classifier

Supervised Learning

Find k nearest neighbors from the entire training set; take majority vote.

In the interior of a class, can we ignore some of them so we’re memorizing fewer data points?

Need fast algorithms for finding nearest neighbors.

Edit the data to reduce classification time and storage.

Two classes of editing algorithms. Won’t give the same sets, but hopefully get the same results.

- Multiedit

Remove unhelpful points.

Eventually will remove all controversial points. Like clustering.

- Condensing Algorithm

Pick those points that help you to make classification.

Finds the boundary points.

Can result in a fairly small set.

Multiedit and Condensing will both give subsets of the original set. What if we want to find a new set that represents the original set but is not a subset of it? Learning Vector Quantization (which has nothing to do with vector quantization algorithms).

Like cluster centers. Somehow originally place the learning vectors, then iteratively check whether each of the original data elements is correctly classified in the learning, and move the learning vectors accordingly.

If α is constant, then you'll get perpetual vibration. If you let α go to zero, then it will converge.

Trying to consider two code vectors at a time requires incredibly complicated math and code.

Will you get a good solution? Depends on whether you have enough code vectors.

There is a perfect way to adjust the LVQ1 learning rate. If you want to become an LVQ expert, learn it.

Usually the front end of LVQ is a self-organizing map, which will give the number of code vectors.

LVQ developed by Finns who believe that this is how the brain works.

Next assignment on clustering.

He gave us a toy example of k-means clustering.

10 25 March Lecture: DBSCAN

So far we have learned two algorithms: k -means and ...

Strength of k -means: Calculates the distance to a number of centers. Good at convex sets.

k -means does not take into account the connectivity of points.

New standard algorithm:; DBSCAN

Good because it isolates background noise.

Based on connectivity.

MinPts gives the minimum number of points in an ϵ neighborhood to say whether the density is high or low.

Labels trickier in DBSCAN.

If label is -1 , then it's a noise point, not connected to any other.

Also a way to identify core points, to distinguish them from border points.

DBSCAN makes a great homework assignment, because there are a lot of things to play with.

How do we pick ϵ and MinPts? Start with ϵ , do a histogram, and pick the value that gives you the most connected points. Then play with MinPts. Look at how many points fall into the

neighborhood, do a histogram. If MinPts is too high, most points will be noise. If MinPts too low, no noise. Don't want to set MinPts too high.

Silhouette score: Smaller is better. More on Wednesday.

10.1 Supervised Learning

Where we've been: Linear classifier, unsupervised learning, nonparametric classifiers, clustering.

Today: Massively parametric

Most of what we've done is to use a training set to learn how to set the weight vector.

Nonlinearity - Making a local decision.

Nonlinearity is critical.

Back-propagation algorithm.

Big jump from artificial neural networks to deep learning.

Theoretically, two layers is sufficient, so no need for deep learning.

Hyperbolic tangent function falls off quickly, so soon you have no learning.

Wednesday: sklearn, artificial neural networks, deep learning.

Homework posted by Wednesday.

11 27 March Lecture: Clustering

k -means is basically a distance-based algorithm.

DBSCAN deals well with noise.

Is density-reachable symmetric and transitive?

Known weakness of DBSCAN to deal well with clusters with different densities.

How do we choose the parameters for DBSCAN, ϵ and `min_samples`?

```
a = np.array([-3.0, -4.0])
```

```
b = np.array([3.0, 4.0])
```

```
import scipy
```

```
scipy.spatial.distance.euclidean(a,b)
```

```
scipy.spatial.distance.cdist(x,x)
```

Finds the distance between each point in x and every other point in x . If x has n points, then it returns an $n \times n$ array of distances with 0 on the main diagonal.

Assignment: Fiddle with the parameters.

Scikit neural network implementation: Assignment for our entertainment.

Wants us to work on Tensorflow.

12 1 April Lecture: Classification by Supervised Learning

Special case of a regression problem.

Target is observed value.

In linear classification, the function only has values 0 and 1.

`np.c_` numpy operation to do the stuff he did longhand.

Pseudoinverse of x $(X^T X)^{-1}$

12.1 TensorFlow Solution

From Google, uses machine learning.

TensorFlow solution to the gradient descent.

Matrix operations built in (of course).

12.2 Batch Gradient Descent Algorithm

$$\nabla_w = -\eta \frac{2}{N} X^T (z - Xw)$$

Gradient Descent Algorithms: Three given plus Woodrow-Hoff.

12.3 Assignment

Get into TensorFlow on cmix.

12.4 Coming on Wednesday

TensorFlow can do everything for you.

13 3 April Lecture

13.1 Second Homework

Gradient Descent problem.

13.2 Graph Descent Algorithms

Batch, Stochastic, Mini-batch, and Woodrow-Hoff

Stochastic can be done in SciLearn.

Mini-batch is a hybrid of Batch and Stochastic.

Woodrow-Hoff is equivalent to Mini-batch with batch size set to one.

13.3 TensorFlow Solution

Construction phase and Execution phase.

Example is a batch solution.

Finds the gradient automatically by symbolic differentiation, `tf.gradients(mse, [weights])`, returns a list of length corresponding to the number of weights.

13.4 Mini-Batch

Update weights by using a randomly selected subset of the training set at each iteration.

Define a batch size. `//` gives integer quotient.

13.5 Future

Homework will be mimicking examples. Do in python and in TensorFlow, not in sklearn.

No midterm. Will see about finals.

14 8 April Lecture

14.1 Review

Creating mini-batches.

Which of these four should we use?

PseudoInverse and MiniBatch may give the same answer for small data sets.

PseudoInverse requires taking the inverse of a matrix; may not be best for large data sets.

14.2 Mini-Batch using TensorFlow

`tf.constant` returns a TensorFlow data structure, not numpy.

`tf.placeholder` Don't need to specify length of array, but need to specify width.

Difference between `Variable` and `placeholder`

Feed in mini-batches as a dictionary.

Number of mini-batches depends on data set size. Choosing the number is more of an art than a science.

Difference b/n stochastic gradient descent and mini-batch: Stochastic picks one of them. Mini-batch uses the entire training set, often faster than stochastic.

Plot graphs using TensorBoard to visualize the training process.

Correction: `tensorboard --logdir tf_logs`

To keep everyone from using the same port, don't use 6006. Use the last four digits of my ULID.

15 22 April Lecture: TensorFlow and Neural Networks

15.1 Random Notes as He Went Through the Code

Calling Scikitlearn to do Neural Networks is about as hard as calling Perceptron.

Implementing Neural Networks in TensorFlow: Only different in code syntax.

Multilayer neural network.

MNIST characters

There will be another assignment.

Minibatches - Use placeholders.

Scale data into $[0,1]$.

Input later, two hidden layers, output layer.

reul - Rectified Linear Units Rectifier kills negative values by setting them to zero. Linearly scales the positive values.

Cost function. Cross entropy. Convenient function for training neural networks.

Dropped all of the things that make TensorBoard look pretty.

`print(epoch, acc_train, acc_val)` in my code, graph in TensorBoard.

If validation accuracy is not improving, then stop; we're overfitting.

15.2 “This is how this is going to work”

Slightly more challenging for last assignment.

Toy example given that will run as an example.

We will come up with a slightly more complicated problem.

Use the toy problem as a skeleton. Add TensorBoard.

He's tired of handwritten characters problem.

Generate input data.