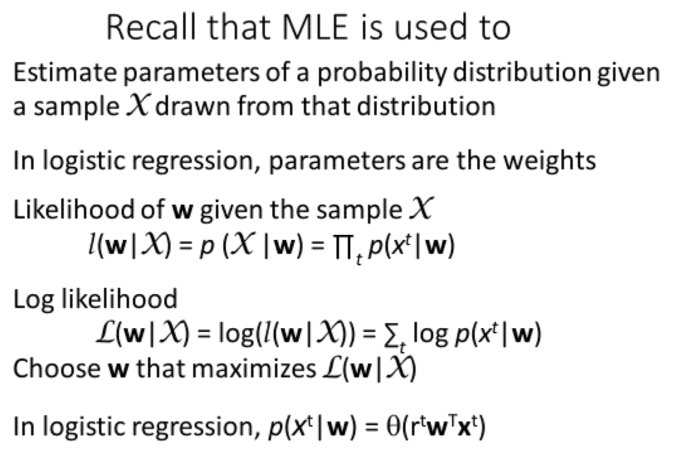
Logistic regression



Recall that MLE is used to

Estimate parameters of a probability distribution given a sample $\mathcal{X}$ drawn from that distribution

In logistic regression, parameters are the weights

Likelihood of **w** given the sample $\mathcal{X}$

$$l(\mathbf{w}|\mathcal{X}) = p(\mathcal{X}|\mathbf{w}) = \prod_t p(x^t|\mathbf{w})$$

Log likelihood

$$\mathcal{L}(\mathbf{w}|\mathcal{X}) = \log(l(\mathbf{w}|\mathcal{X})) = \sum_t \log p(x^t|\mathbf{w})$$

Choose **w** that maximizes $\mathcal{L}(\mathbf{w}|\mathcal{X})$

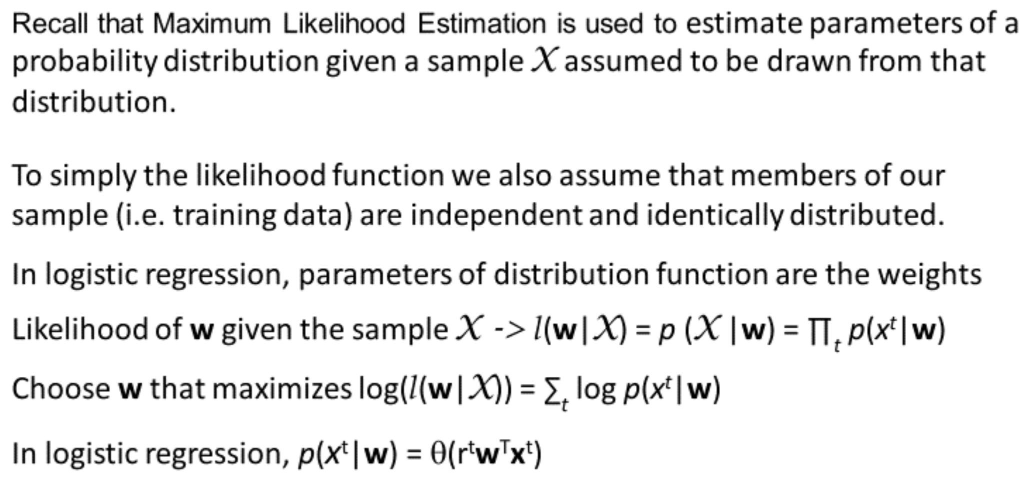
In logistic regression, $p(x^t|\mathbf{w}) = \theta(r^t\mathbf{w}^T\mathbf{x}^t)$

For application of MLE, why is it important to assume that samples used to estimate parameters of the distribution function are independent and identically distributed?

The I.I.D. assumption allows us to simplify the likelihood function as the product of the individual probability of each example in the dataset:

$$p(X|\boldsymbol{w}) = \prod_t p(x^t|\boldsymbol{w})$$

We take the log of the likelihood, so the product becomes a sum, enabling us to easily calculate derivatives.



Recall that Maximum Likelihood Estimation is used to estimate parameters of a probability distribution given a sample $\mathcal{X}$ assumed to be drawn from that distribution.

To simply the likelihood function we also assume that members of our sample (i.e. training data) are independent and identically distributed.

In logistic regression, parameters of distribution function are the weights

Likelihood of **w** given the sample $\mathcal{X}$ -> $l(\mathbf{w}|\mathcal{X}) = p(\mathcal{X}|\mathbf{w}) = \prod_t p(x^t|\mathbf{w})$

Choose **w** that maximizes $\log(l(\mathbf{w}|\mathcal{X})) = \sum_t \log p(x^t|\mathbf{w})$

In logistic regression, $p(x^t|\mathbf{w}) = \theta(r^t\mathbf{w}^T\mathbf{x}^t)$

Extending linear models by features

The XOR data set is not linearly separable in attribute space.
What is the basic rationale for getting linearly separable features using either a Gaussian transformation or an ANN with 1 hidden layer containing 2 nodes and a bias?

We make a transformation that moves two of the attribute vectors to essentially the same point.

Therefore, we have three points in a two-dimensional feature space; three points in a two-dimensional feature space are always linearly separable.

This is the Gaussian result.

This is the idealized ANN result; we don't quite realize that, but we get fairly close.

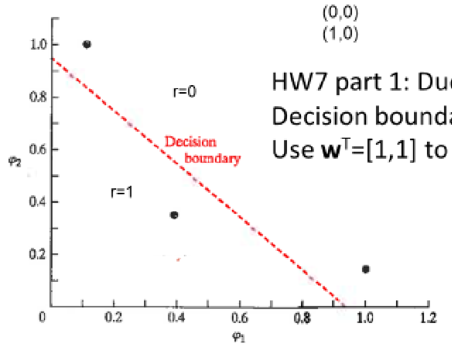
XOR data

x_1	x_2	r
0	0	0
0	1	1
1	0	1
1	1	0

XOR in Gaussian feature space

$\phi_1 = \exp(-|\mathbf{X} - [1,1]|^2)$
$\phi_2 = \exp(-|\mathbf{X} - [0,0]|^2)$

X	ϕ_1	ϕ_2
(1,1)	1	0.1353
(0,1)	0.3678	0.3678
(0,0)	0.1353	1
(1,0)	0.3678	0.3678

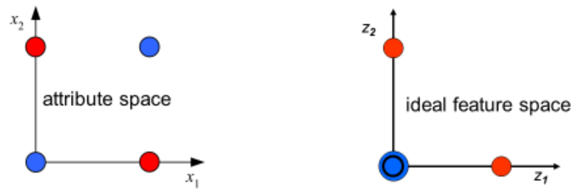


HW7 part 1: Due 10/22/19
Decision boundary is $\phi_1+\phi_2=0.94$
Use $\mathbf{w}^T=[1,1]$ to calculate bias and margins

Solution of XOR by ANN: Add a "hidden" layer with 2 hidden nodes and a bias. Input to hidden nodes is sigmoid($\mathbf{w}_h{}^T\mathbf{x}$)

Consider hidden nodes, z_h, as features. Choose weight vectors $\mathbf{w}_h$ so that in feature space examples at (0,0) and (1,1) in attribute space are close to the same point in feature space.

$$z_h = \frac{1}{1+\exp[-w_h{}^T x]} = \text{sigmoid}(w_h{}^T x)$$



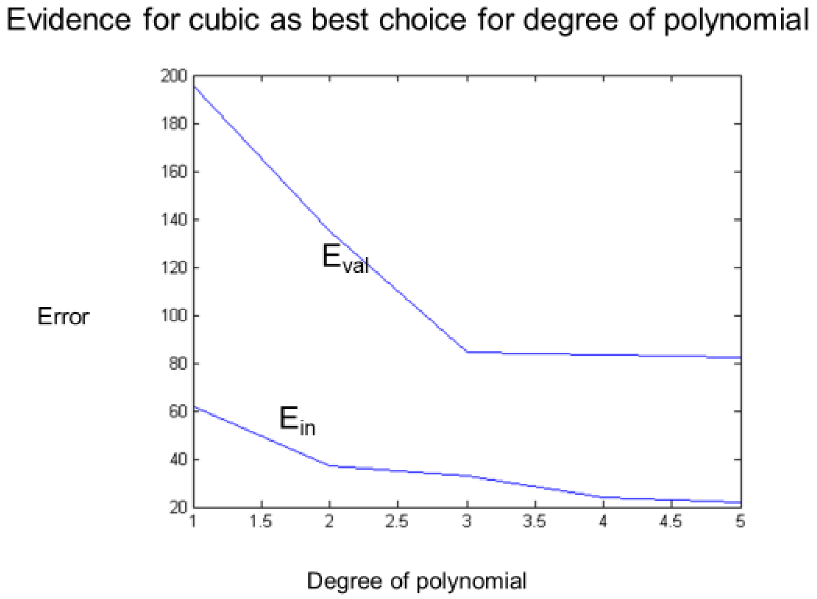
Explain how a validation set can be used to find the best degree of a polynomial feature with sacrificing any training data.

How is the training set used?

The training set is used to optimize the weights, the linear combination of the polynomial features, by minimizing an error function (e.g. sum of squared deviations).

How is the validation set used?

We don't determine weights. The training set was used to minimize the in-sample error, now we take those weights and calculate the validation error to get the degree of the polynomial.



How do we use a plot of validation error vs degree of polynomial?

We're looking for the point where the slope changes radically (elbow), in the above case the best choice is a 3rd degree polynomial. The in-sample error doesn't help us at all because it keeps going down with the higher degree of polynomial.

What do we do after be degree of polynomial has been chosen?

Once we have the degree (the best complexity for the problem at hand), we use the full dataset (combine the training and validation sets) to get the best estimation of the weights.

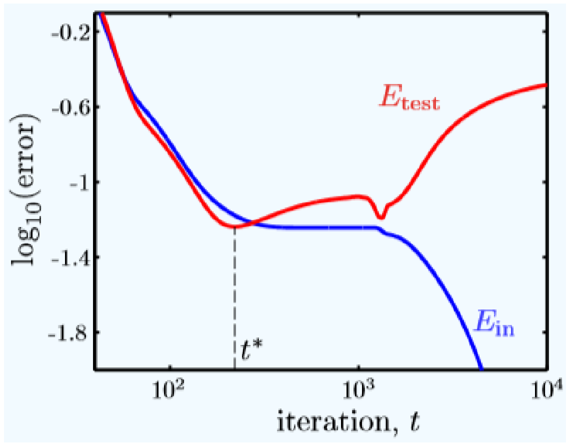
How does the validation-set approach have to be modified when applied to ANNs?

In polynomial regression the validation subset can be returned to training set after the best degree has been determined.

Not true with MLP non-linear regression. With MLP, we cannot expect the results revealed by a validation set to hold if back propagation is repeated with new training set that is old training set + validation set.

With MLP back propagation, E_{val} is more like E_{test}, examples never used in training

---



What's wrong with these approaches to back propagation:

Combine test and training sets and retrain for t* iterations of back propagation.

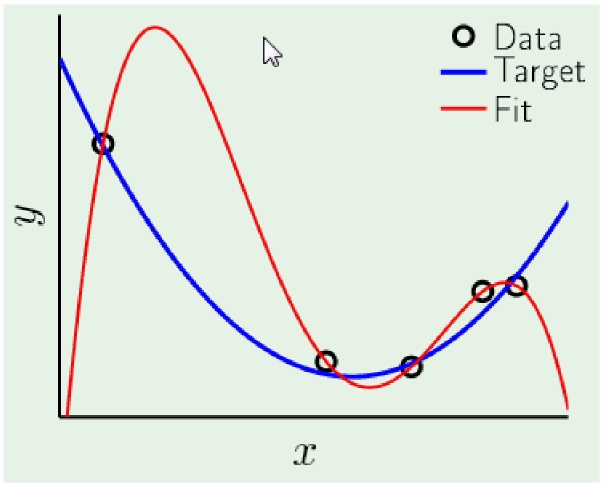
That would not necessarily give us anything like the result we got with just the training set.

Combine test and training sets and retrain until $E_{in} = E_{test}(t^*)$

That might not even be possible or desirable. Could cause no training cycles if they're initially equal and could cause infinite cycles if they never intersect.

The results we got at t cycles of back propagation are undoubtable the best result to use.*

Overfitting



What's wrong with this result that has $E_{in} = 0$?

It won't generalize well because the fit is absolutely nothing like the target. Although we have zero in sample error, we can't say this 4th order polynomial has a shape in the range of the data that is anything like the target function.

What should I expect from another set of 5 points in the same range of x?

Something totally different because the shape of this 4th order polynomial result is just the result of the error that has been assigned to these data points extracted out of the target function. So basically, it's an expression of the fit to the error in those data points.

Regularization by weight decay

$$E_{aug} = E_{in} + ?$$

What do we add to E_{in} to get an augmented error function that will penalize selection of large weight vectors?

$$E_{aug} = E_{in} + \lambda \mathbf{w}^T\mathbf{w}$$

A constant that has yet to be determined and multiply it by the square modulus of the weight vector. Obviously, large weight vectors will be penalized by that form.

With polynomial features, how do I find the best choice of λ and the best model with weight-decay regularization.

We have to take the data set, divide it into training samples and validation samples, determine the weights for a given value of λ, take those weights and see what the error in the validation set is, do that for various values of λ, plot the validation error and see what value of λ produces an elbow. Then we combine the training and validation set and get the weights using the optimum value of λ.

What happens when the value of λ gets very large?

At $\lambda = 0$ isn't any help and too large of a λ causes the model to forget about the data (all it cares about is λ). Somewhere between the two is the optimum choice of λ.

$$\Delta w_j = -\eta \frac{\partial E^t}{\partial w_j} = \eta \left(r^t - y^t \right) x_j^t$$

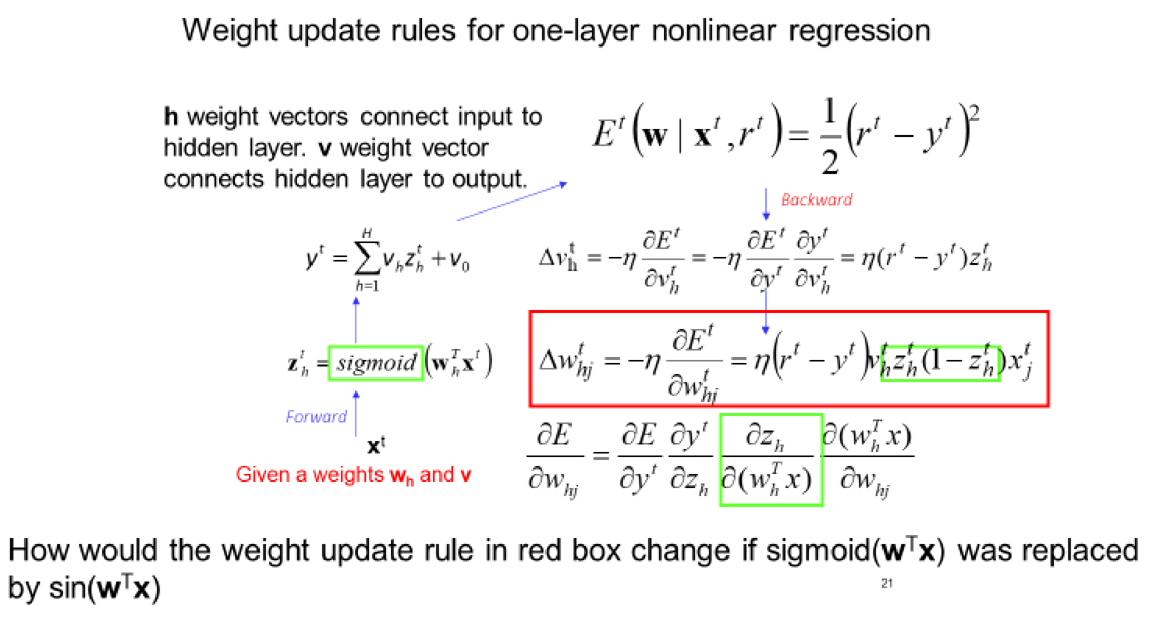
How do we change weight update rules of back propagation to penalize selection of large weight vectors in back propagations?

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i} - \lambda w_i$$

Obviously, we change the weight update rule to not only involve the learning weight and the gradient of the error function with respect to the weight, but also take minus lambda times the weight itself. This $-\lambda w_i$ term is basically going to always cause Δw_i to make the weight smaller.

With hidden layer features, how do I find the best choice of λ?

For neural networks, find the value of lambda that minimizes error in the test set.



Weight update rules for one-layer nonlinear regression

h weight vectors connect input to hidden layer. **v** weight vector connects hidden layer to output.

$$E^t(\mathbf{w} \mid \mathbf{x}^t, r^t) = \frac{1}{2}(r^t - y^t)^2$$

Backward

$$\Delta v_h^t = -\eta \frac{\partial E^t}{\partial v_h^t} = -\eta \frac{\partial E^t}{\partial y^t}\frac{\partial y^t}{\partial v_h^t} = \eta(r^t - y^t) z_h^t$$

$$y^t = \sum_{h=1}^{H} v_h z_h^t + v_0$$

$$\mathbf{z}_h^t = \boxed{sigmoid}(\mathbf{w}_h^T \mathbf{x}^t)$$

Forward

$\mathbf{x}^t$

Given a weights $\mathbf{w}_h$ and $\mathbf{v}$

$$\boxed{\Delta w_{hj}^t = -\eta \frac{\partial E^t}{\partial w_{hj}^t} = \eta(r^t - y^t) v_h \boxed{z_h^t(1 - z_h^t)} x_j^t}$$

$$\frac{\partial E}{\partial w_{hj}} = \frac{\partial E}{\partial y^t}\frac{\partial y^t}{\partial z_h}\boxed{\frac{\partial z_h}{\partial(w_h^T x)}}\frac{\partial(w_h^T x)}{\partial w_{hj}}$$

How would the weight update rule in red box change if sigmoid($\mathbf{w}^T\mathbf{x}$) was replaced by sin($\mathbf{w}^T\mathbf{x}$)

21

The derivative of sigmoid with respect to $w_h^T x^t$ (it's argument)

$z_h^t(1 - z_h^t)$ *is the derivative of the sigmoid function with respect to it's argument* $w_h^T x^t$

The derivative of sin is cos, but we can't just use that because the weight update rule is written in terms of the value of z

The z is defined as the sin of the dot product, so we can't just replace that by cos

We have to express the cos as a function of the sin

$$sin^2 + \cos^2 = 1$$
$$z^2 + \cos^2 = 1$$
$$\sqrt{1 - z^2} = cos$$

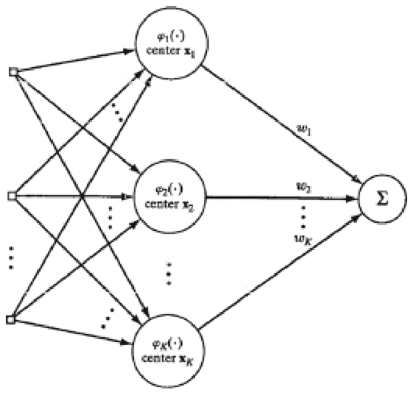
The answer is to calculate this derivative $\frac{\delta z_h}{\delta(w_h^T x)}$

Whatever nonlinear function I use to calculate z, calculate the derivative and express the result as a function of z.

https://www.wolframalpha.com

Radial basis function network

Application of K-means clustering to RBF-ANN


$\varphi_1(\cdot)$ center $\mathbf{x}_1$
$\varphi_2(\cdot)$ center $\mathbf{x}_2$
$\varphi_K(\cdot)$ center $\mathbf{x}_K$
w_1
w_2
w_K
Σ

Given converged K-means centers, estimate variance for RBFs by $\sigma^2 = d^2_{max}/2K$, where d_{max} is the largest distance between clusters.

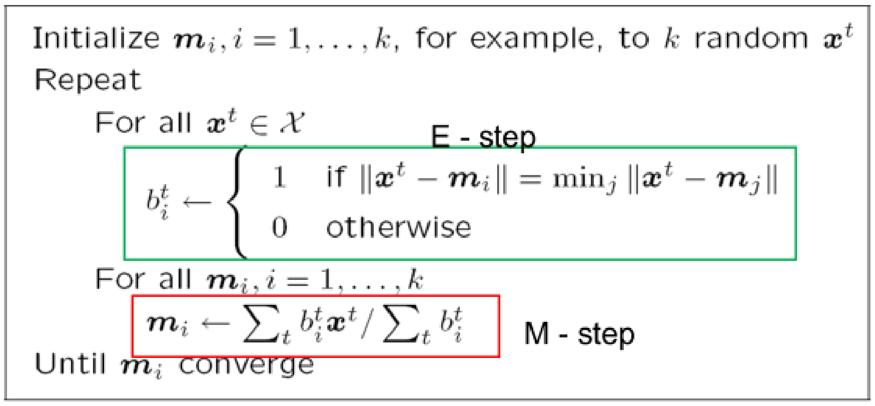
If K=2 how do we find σ^2

I need to have vectors for the mean values of the two clusters. Then take the difference of the mean values (d) and divide it by 4.

What if we had 3 clusters?

I have 3 mean values. That describes where each cluster is in attribute space. I need to find the largest distance between the 3 vectors. So calculate the 3 differences, get the magnitudes, compare and take the max.

Describe in words the E-step and M-step of K-means

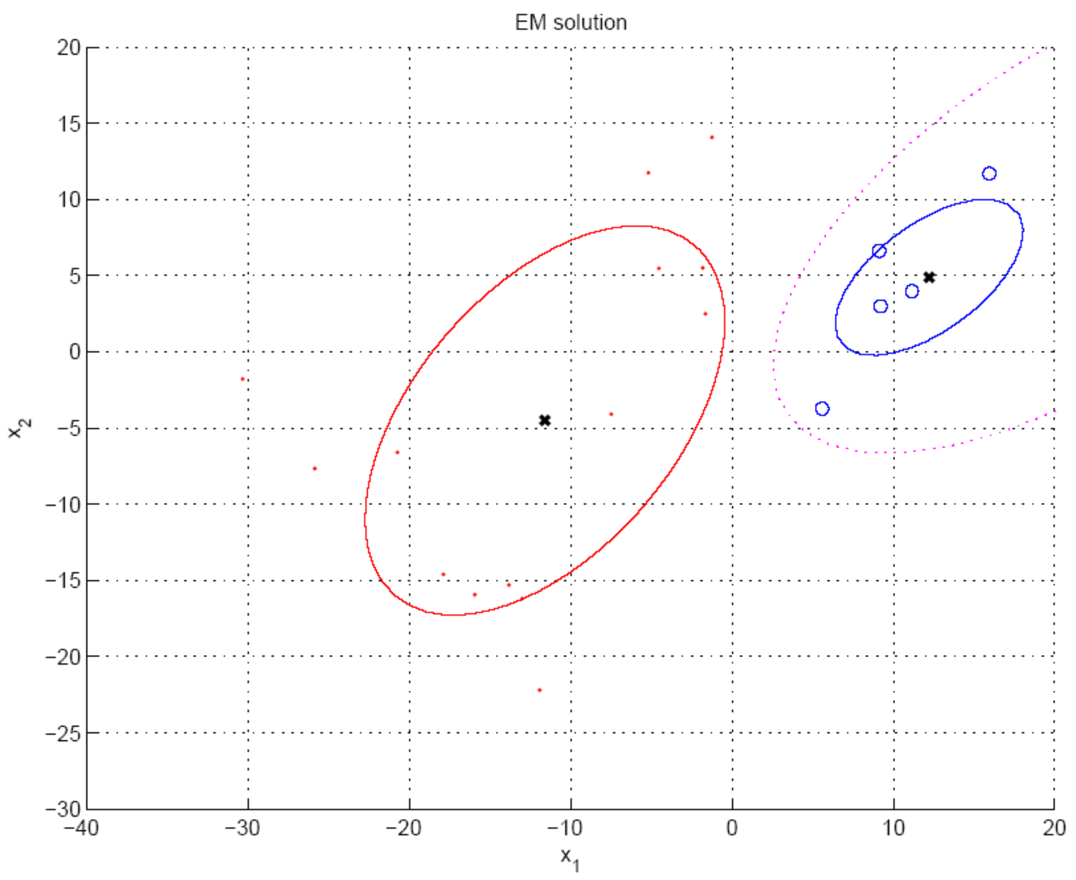


```
Initialize $\boldsymbol{m}_i, i = 1, \dots, k$, for example, to $k$ random $\boldsymbol{x}^t$
Repeat
    For all $\boldsymbol{x}^t \in \mathcal{X}$
        $b_i^t \leftarrow \begin{cases} 1 & \text{if } \|\boldsymbol{x}^t - \boldsymbol{m}_i\| = \min_j \|\boldsymbol{x}^t - \boldsymbol{m}_j\| \\ 0 & \text{otherwise} \end{cases}$
    For all $\boldsymbol{m}_i, i = 1, \dots, k$
        $\boldsymbol{m}_i \leftarrow \sum_t b_i^t \boldsymbol{x}^t / \sum_t b_i^t$
Until $\boldsymbol{m}_i$ converge
```

E - step
M - step

An E-step is assigning labels to clusters. If attribute vector t belongs in cluster i, it's label is 1, and the label for every other cluster is 0. I need to know the means because that's how I know if attribute t belongs in cluster i, how close it is to the mean of all the clusters, and which is closest.

The M-step uses the labels of the E-step to calculate new means.



What is the meaning of soft label h_i^t in Gaussian mixtures

This label is the probablity that attribute vector t belongs to cluster i. This soft label has a value every cluster, being the probably

Each attribute vector has a soft label for every cluster indicating the probability that it belongs to that cluster.

What is the advantage of Gaussian mixtures over K-means as a clustering method for RBF ANN?

K-means only determines the mean, so you need to use another method to estimate the variance. Gaussian mixtures give both means and variances, both parameters we need for the Gaussian basis functions of the regular basis function ANN.

James Jessen 2019-11-05 Quiz 2

Genetic Algorithm

Going from fitness to discrete probability and selecting chromosome for modification to create the next generation.

Method to select chromosomes for refinement

Calculate fitness $f(x_i)$ for each chromosome in population
Assign each chromosome a discrete probability by

$$p_i = \frac{f(x_i)}{\sum_{i=1}^{n} f(x_i)}$$

Divide number line between 0 and 1 into segments of length *1/population size*

Get ***r***, random number uniformly distributed between 0 and 1

Choose the chromosome of the line segment containing ***r***

1st generation: 5-bit binary numbers chosen randomly

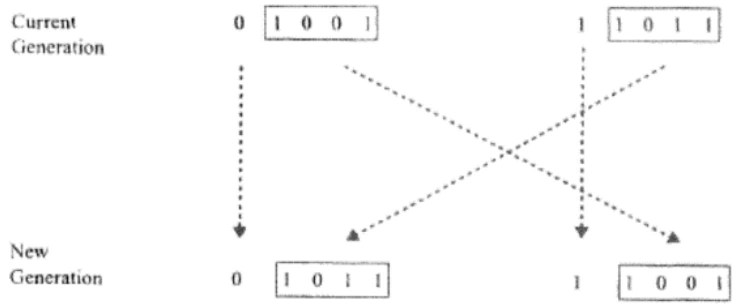
00100 = 4	fitness = 0.0011	pi = 0.044
01001 = 9	fitness = 0.0216	pi = 0.861
11011 = 27	fitness = 0.0023	pi = 0.091
11111 = 31	fitness = 0.0001	pi = 0.004

$\Sigma_i f(x_i) = 0.0251$

Assume the pair with greatest fitness (01001 and 11011) are selected for replication

Crossover selected to induce change

Assume mixing point (aka locus) is chosen between first and second bit.



Mutation is rejected as method to induce change

Evaluate fitness of new population

00100 = 4	fitness = 0.0011	pi = 0.0201
01011 = 11	fitness = 0.0457	pi = 0.8339
11001 = 25	fitness = 0.0079	pi = 0.1442
11111 = 31	fitness = 0.0001	pi = 0.0018

$\Sigma_i f(x_i) = 0.0548$ about 2 times that of the 1st generation

Repeat until fitness of population is almost uniform

Integer values of all chromosomes should be near 16

Principal Component Analysis

How do I obtain the correlation matrix from the covariance matrix?
What is the trace of the correlation matrix equal to?

$$\underset{d\times d}{\Sigma} = E\left[\underset{d\times 1}{(\mathbf{x}-\mu)}\underset{1\times d}{(\mathbf{x}-\mu)^T}\right] \qquad \Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1d} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2d} \\ \vdots & & & \\ \sigma_{d1} & \sigma_{d2} & \cdots & \sigma_d^2 \end{bmatrix}$$

Dividing elements covariance matrix by the product of standard deviations produces the “correlation” matrix with 1’s on diagonal. Off-diagonals are “correlation coefficients”

$$\mathbf{Corr}(x_i, x_j) \equiv \rho_{ij} = \frac{\sigma_{ij}}{\sigma_i \sigma_j} \qquad \text{range -1 to 1}$$

Prove that λ_k is the variance of z_k

$\mathbf{z_k} = \mathbf{w_k}^T\mathbf{x}$

$\Sigma \boldsymbol{w}_k = \lambda_k \boldsymbol{w}_k$

$\text{var}(z_k) = \boldsymbol{w}_k^T \Sigma \boldsymbol{w}_k = \boldsymbol{w}_k^T \lambda_k \boldsymbol{w}_k = \lambda_k \boldsymbol{w}_k^T \boldsymbol{w}_k = \lambda_k$

Σ is the covariance matrix

Proving that the variance of the k^{th} principal component is equal to its eigenvalue.

How do we convert attributes to z-scores = $(x_k - \mu_k)/\sigma_{kk}$

Estimate the mean μ_k and the square of the standard deviation σ_{kk}

What is another name for the covariance matrix of z-scores?

Correlation Matrix

What is the value of the total variance captured if all the PCs of z-scores are retained?

It's the number of attributes.

The trace of a matrix is invariant under diagonalization, which means that the sum of the eigenvalues of the correlation matrix will be the same before and after diagonalization.

The diagonal element of the correlation matrix is 1, so if you add them up it's the number of attributes.

Proportion of Variance (PoV)

$$\text{PoV}(k) = \frac{\lambda_1 + \lambda_2 + \cdots + \lambda_k}{\lambda_1 + \lambda_2 + \cdots + \lambda_k + \cdots + \lambda_d}$$

Given k and d eigenvalues, calculate PoV(k)

Use the characteristic polynomial to calculate eigenvalues of a 2x2 matrix

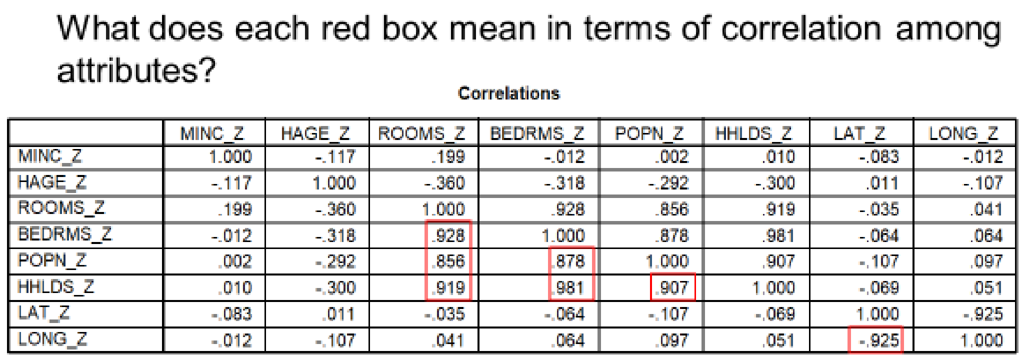
calculate eigenvalues of $\begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}$

$\det(\mathbf{A} - \lambda\mathbf{I}) = \det\left(\begin{bmatrix} 3-\lambda & 1 \\ 1 & 3-\lambda \end{bmatrix} \right)$

$(3-\lambda)(3-\lambda) - 1 = \lambda^2 - 6\lambda + 8 = 0$

by quadratic formula $\lambda_1 = 4$ and $\lambda_2 = 2$

https://www.wolframalpha.com



What does each red box mean in terms of correlation among attributes?

Correlations

	MINC_Z	HAGE_Z	ROOMS_Z	BEDRMS_Z	POPN_Z	HHLDS_Z	LAT_Z	LONG_Z
MINC_Z	1.000	-.117	.199	-.012	.002	.010	-.083	-.012
HAGE_Z	-.117	1.000	-.360	-.318	-.292	-.300	.011	-.107
ROOMS_Z	.199	-.360	1.000	.928	.856	.919	-.035	.041
BEDRMS_Z	-.012	-.318	$\boxed{.928}$	1.000	.878	.981	-.064	.064
POPN_Z	.002	-.292	$\boxed{.856}$	$\boxed{.878}$	1.000	.907	-.107	.097
HHLDS_Z	.010	-.300	$\boxed{.919}$	$\boxed{.981}$	$\boxed{.907}$	1.000	-.069	.051
LAT_Z	-.083	.011	-.035	-.064	-.107	-.069	1.000	-.925
LONG_Z	-.012	-.107	.041	.064	.097	.051	$\boxed{-.925}$	1.000

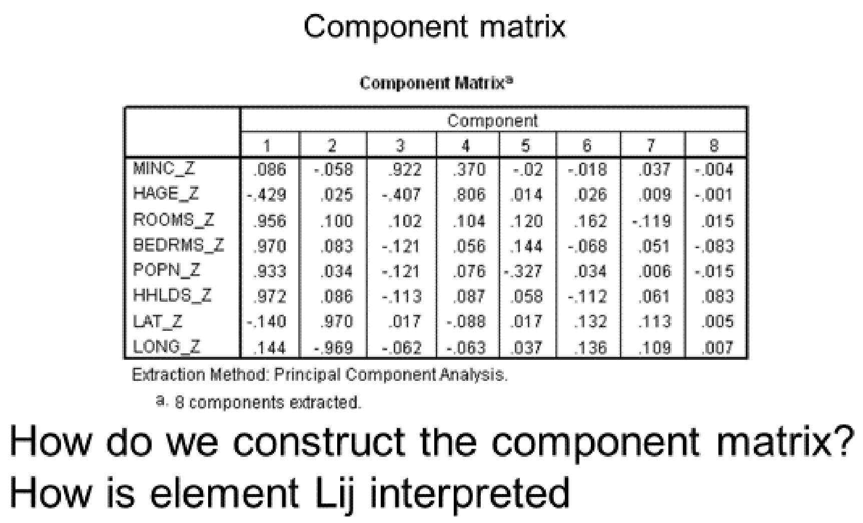
ROOMS is strongly correlated with BEDRMS, POPN, HHLDS

BEDRMS is strongly correlated with POPN, HHLDS

POPN is strongly correlated with HHLDS

LAT is strongly correlated with LONG

This is because the values are close to 1. If close to -1, strong negative correlation.



Component matrix

Component Matrix[a]

	Component							
	1	2	3	4	5	6	7	8
MINC_Z	.086	-.058	.922	.370	-.02	-.018	.037	-.004
HAGE_Z	-.429	.025	-.407	.806	.014	.026	.009	-.001
ROOMS_Z	.956	.100	.102	.104	.120	.162	-.119	.015
BEDRMS_Z	.970	.083	-.121	.056	.144	-.068	.051	-.083
POPN_Z	.933	.034	-.121	.076	-.327	.034	.006	-.015
HHLDS_Z	.972	.086	-.113	.087	.058	-.112	.061	.083
LAT_Z	-.140	.970	.017	-.088	.017	.132	.113	.005
LONG_Z	.144	-.969	-.062	-.063	.037	.136	.109	.007

Extraction Method: Principal Component Analysis.

a. 8 components extracted.

How do we construct the component matrix?
How is element Lij interpreted

For every principal component, there is a weight vector that is the solution of the eigenvalue equation. That weight vector tells us how to take linear combinations of attributes in order to get PCs. If I take the weight vector and multiply by the sqrt of the eigenvalue, I get a column of the component matrix.

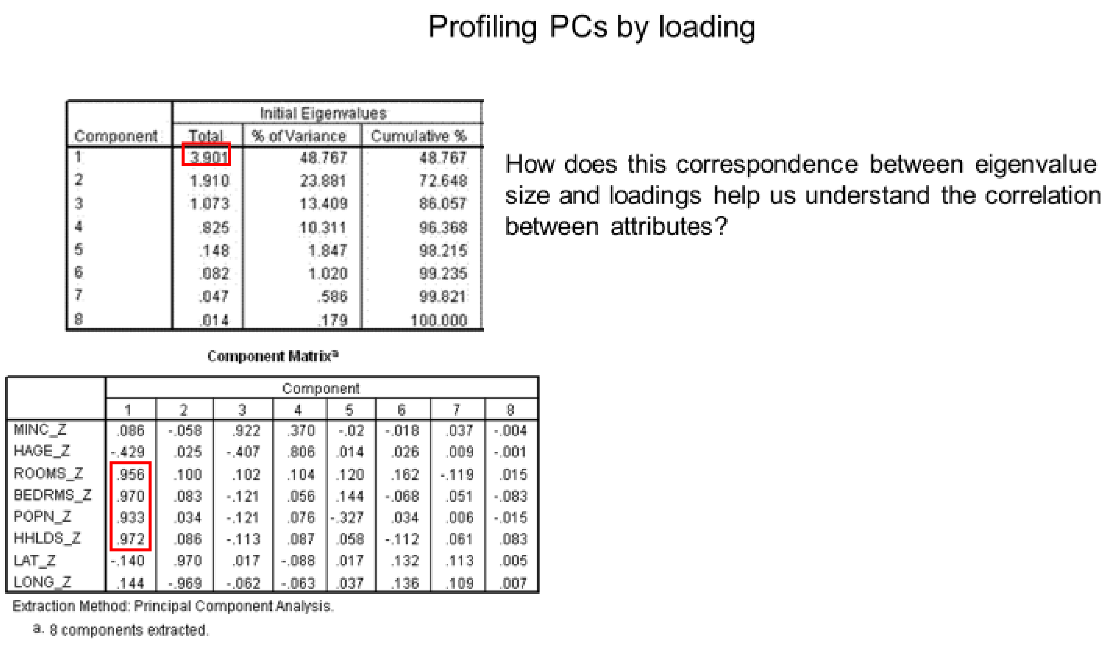
Column 2 of the component matrix is the weight vector solution of the eigenvalue equation for the next smallest eigenvalue and I multiply that weight vector by the sqrt of the eigenvalue and I get this particular column.

The entries in this matrix are called loadings. The rows are z-scores of attributes and the columns are PCs. So a row shows how that attribute relates to each PC.

The attribute that loads PC 1 the most is HHLDS because .972 is the max in the column.

MINC loads PC 3 more than any of the others.

Column 1 shows how much each attribute loads PC 1.



Profiling PCs by loading

Component	Initial Eigenvalues		
	Total	% of Variance	Cumulative %
1	3.901	48.767	48.767
2	1.910	23.881	72.648
3	1.073	13.409	86.057
4	.825	10.311	96.368
5	.148	1.847	98.215
6	.082	1.020	99.235
7	.047	.586	99.821
8	.014	.179	100.000

How does this correspondence between eigenvalue size and loadings help us understand the correlation between attributes?

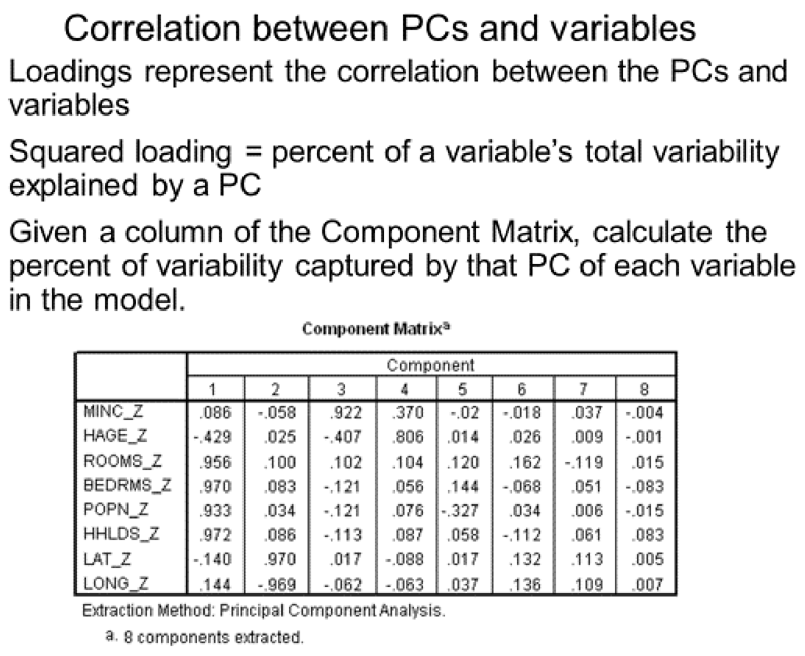
Component Matrix[a]

	Component							
	1	2	3	4	5	6	7	8
MINC_Z	.086	-.058	.922	.370	-.02	-.018	.037	-.004
HAGE_Z	-.429	.025	-.407	.806	.014	.026	.009	-.001
ROOMS_Z	.956	.100	.102	.104	.120	.162	-.119	.015
BEDRMS_Z	.970	.083	-.121	.056	.144	-.068	.051	-.083
POPN_Z	.933	.034	-.121	.076	-.327	.034	.006	-.015
HHLDS_Z	.972	.086	-.113	.087	.058	-.112	.061	.083
LAT_Z	-.140	.970	.017	-.088	.017	.132	.113	.005
LONG_Z	.144	-.969	-.062	-.063	.037	.136	.109	.007

Extraction Method: Principal Component Analysis.

a. 8 components extracted.

Note that the total eigenvalue is almost equal to 4, meaning that PC 1 is probably the equivalent of 4 attributes. Looking at the Component Matrix, there are 4 significant loadings, so there is 1 factor that contributes to the strong correlation between those 4 variables. That factor is represented by PC 1, in this case block size.



Correlation between PCs and variables

Loadings represent the correlation between the PCs and variables

Squared loading = percent of a variable's total variability explained by a PC

Given a column of the Component Matrix, calculate the percent of variability captured by that PC of each variable in the model.

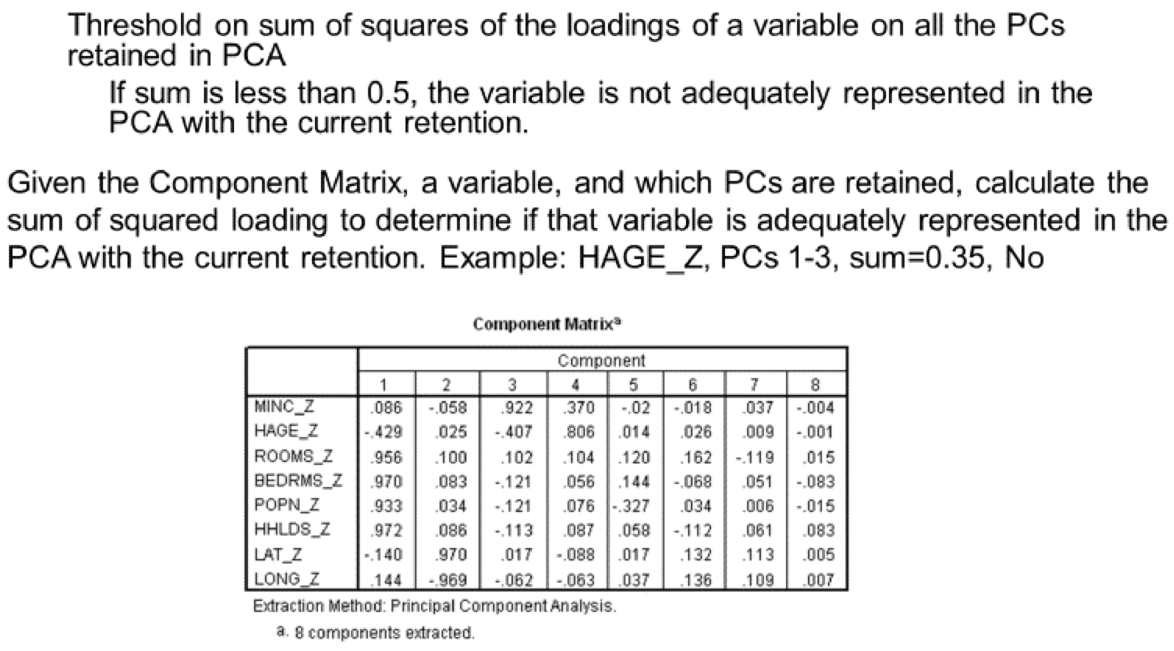
Component Matrix[a]

	Component							
	1	2	3	4	5	6	7	8
MINC_Z	.086	-.058	.922	.370	-.02	-.018	.037	-.004
HAGE_Z	-.429	.025	-.407	.806	.014	.026	.009	-.001
ROOMS_Z	.956	.100	.102	.104	.120	.162	-.119	.015
BEDRMS_Z	.970	.083	-.121	.056	.144	-.068	.051	-.083
POPN_Z	.933	.034	-.121	.076	-.327	.034	.006	-.015
HHLDS_Z	.972	.086	-.113	.087	.058	-.112	.061	.083
LAT_Z	-.140	.970	.017	-.088	.017	.132	.113	.005
LONG_Z	.144	-.969	-.062	-.063	.037	.136	.109	.007

Extraction Method: Principal Component Analysis.

a. 8 components extracted.

Use quiz2-helper.xlsx



Threshold on sum of squares of the loadings of a variable on all the PCs retained in PCA

If sum is less than 0.5, the variable is not adequately represented in the PCA with the current retention.

Given the Component Matrix, a variable, and which PCs are retained, calculate the sum of squared loading to determine if that variable is adequately represented in the PCA with the current retention. Example: HAGE_Z, PCs 1-3, sum=0.35, No

Component Matrix[a]

	Component							
	1	2	3	4	5	6	7	8
MINC_Z	.086	-.058	.922	.370	-.02	-.018	.037	-.004
HAGE_Z	-.429	.025	-.407	.806	.014	.026	.009	-.001
ROOMS_Z	.956	.100	.102	.104	.120	.162	-.119	.015
BEDRMS_Z	.970	.083	-.121	.056	.144	-.068	.051	-.083
POPN_Z	.933	.034	-.121	.076	-.327	.034	.006	-.015
HHLDS_Z	.972	.086	-.113	.087	.058	-.112	.061	.083
LAT_Z	-.140	.970	.017	-.088	.017	.132	.113	.005
LONG_Z	.144	-.969	-.062	-.063	.037	.136	.109	.007

Extraction Method: Principal Component Analysis.

a. 8 components extracted.

Use quiz2-helper.xlsx