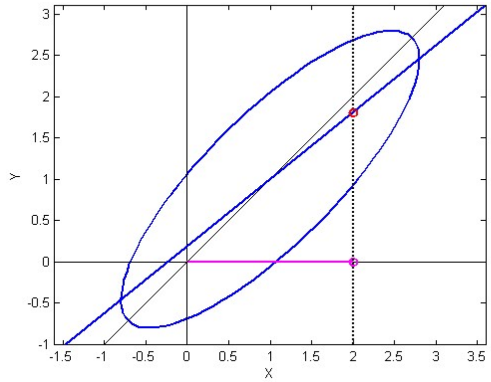
MLSP

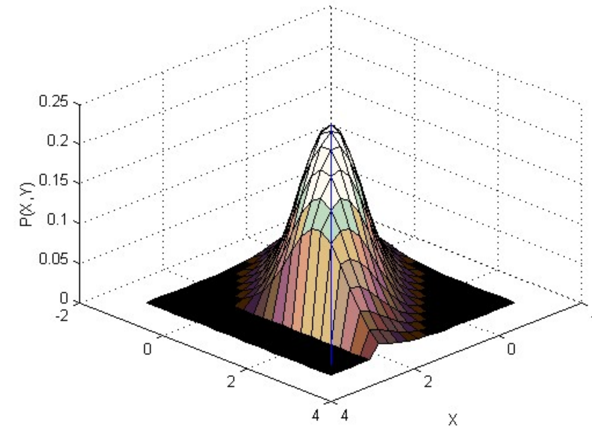
Factor Analysis

Preliminaries : $P(y|x)$ for Gaussian

- If $P(x,y)$ is Gaussian:

$$P(\mathbf{x},\mathbf{y}) = N\left(\begin{bmatrix}\mu_{\mathbf{x}} \\ \mu_{\mathbf{y}}\end{bmatrix}, \begin{bmatrix} C_{\mathbf{xx}} & C_{\mathbf{xy}} \\ C_{\mathbf{yx}} & C_{\mathbf{yy}}\end{bmatrix}\right)$$



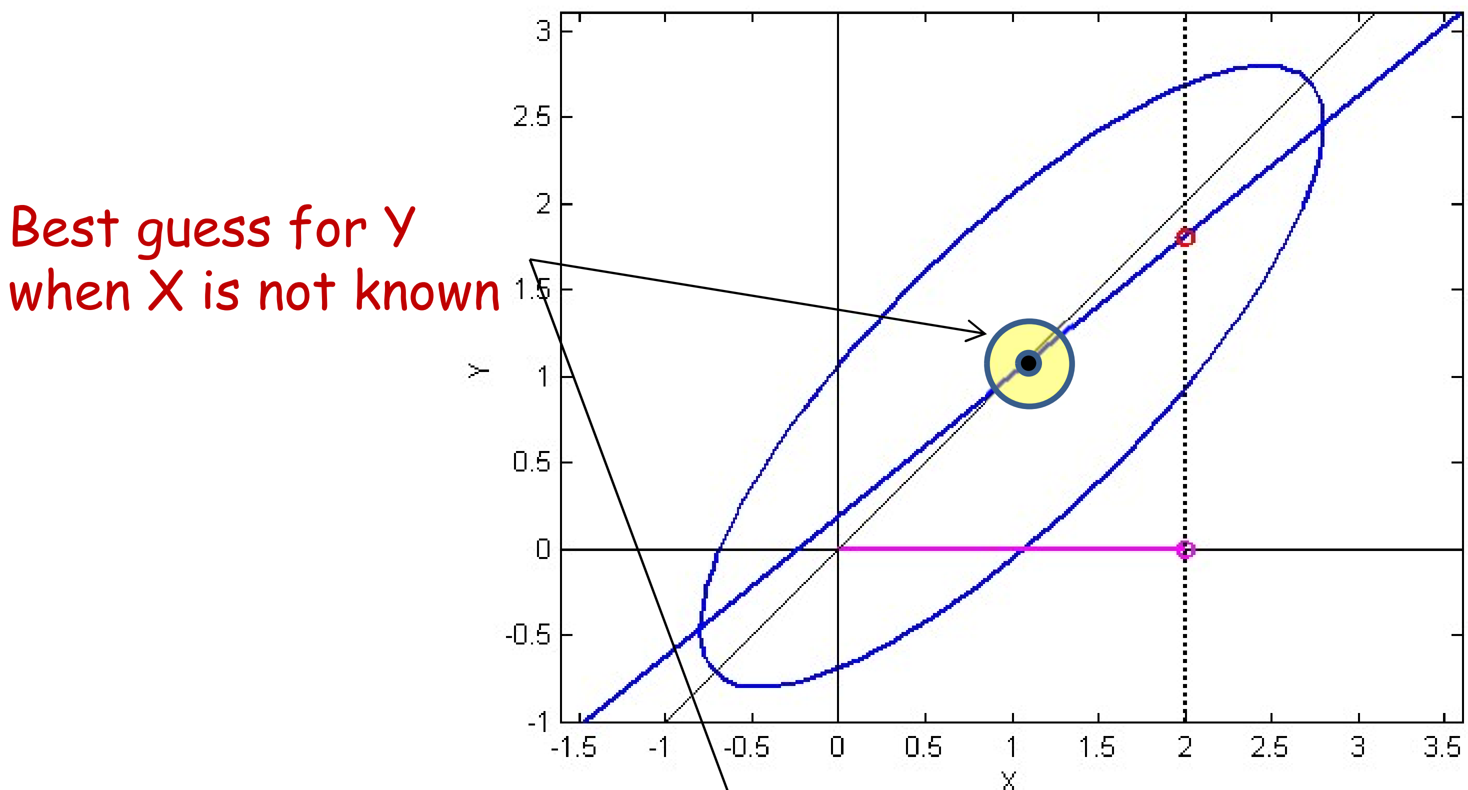


- The conditional probability of y given x is also Gaussian
  - The slice in the figure is Gaussian

$$P(y\,|\,x) = N(\mu_y + C_{yx}C_{xx}^{-1}(x-\mu_x), C_{yy} - C_{yx}C_{xx}^{-1}C_{xy})$$

- The mean of this Gaussian is a function of x
- The variance of y reduces if x is known
  - Uncertainty is reduced

Preliminaries : P(y|x) for Gaussian



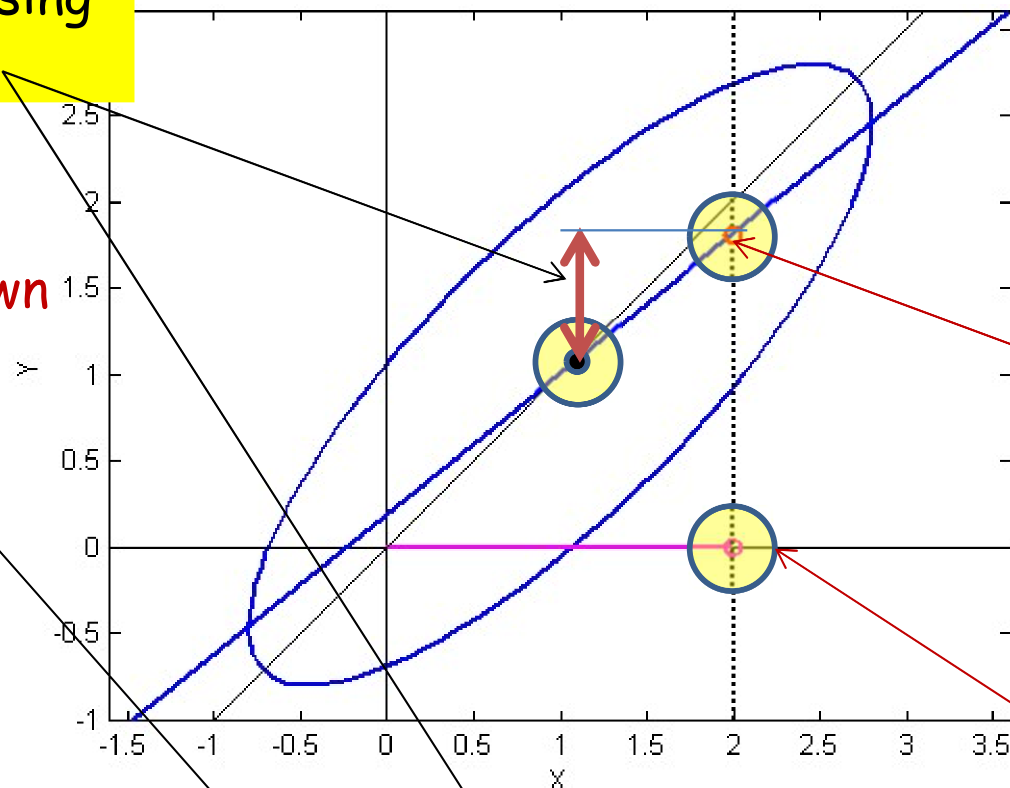
$$P(y \mid x) = N(\mu_y + C_{yx}C_{xx}^{-1}(x - \mu_x), C_{yy} - C_{yx}C_{xx}^{-1}C_{xy})$$

Preliminaries : P(y|x) for Gaussian

Update guess of Y based on information in X
Correction is 0 if X and Y are uncorrelated, i.e $C_{yx} = 0$

Correction of Y using information in X

Best guess for Y when X is not known



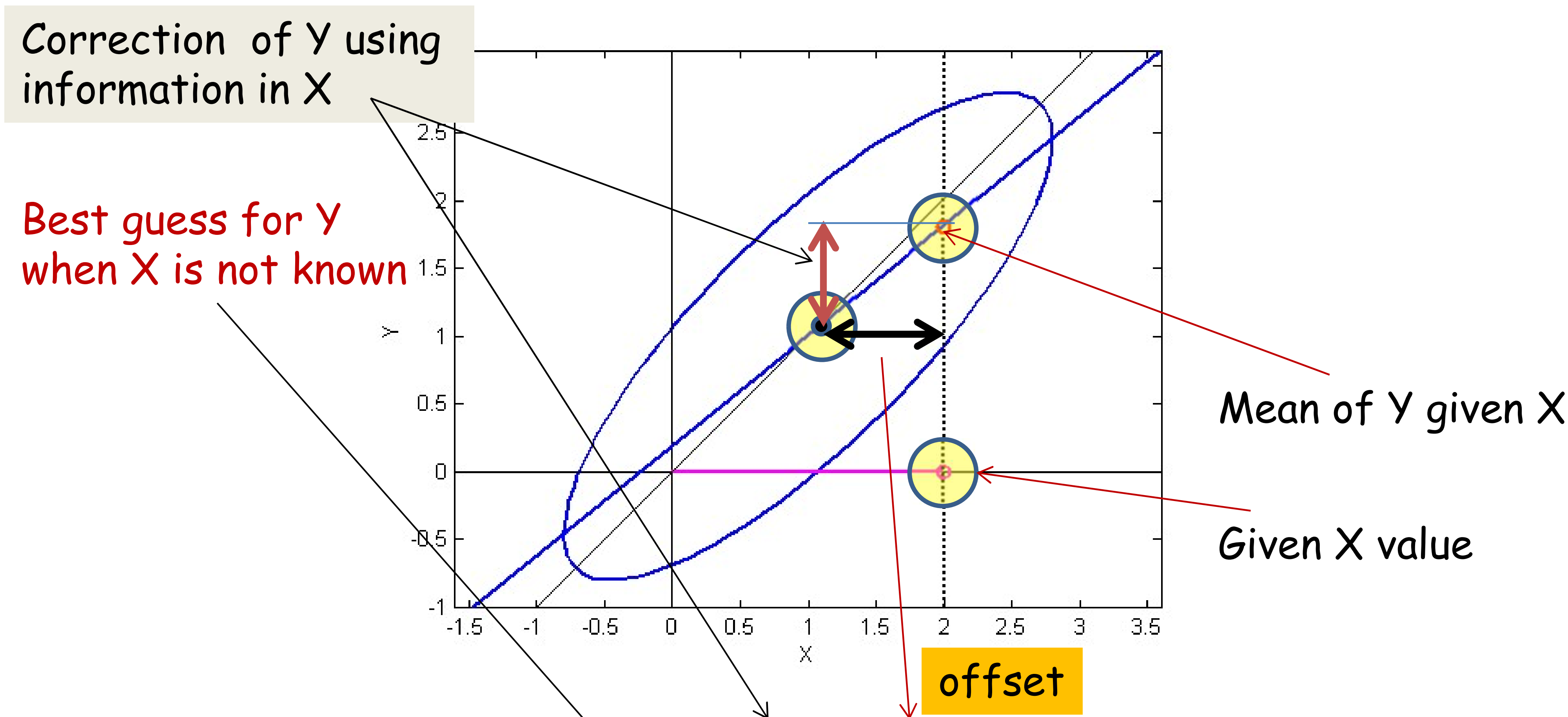
Mean of Y given X

Given X value

$$P(y\,|\,x) = N(\mu_y + C_{yx}C_{xx}^{-1}(x - \mu_x), C_{yy} - C_{yx}C_{xx}^{-1}C_{xy})$$

Preliminaries : P(y|x) for Gaussian

Correction to Y = slope * (offset of X from mean)



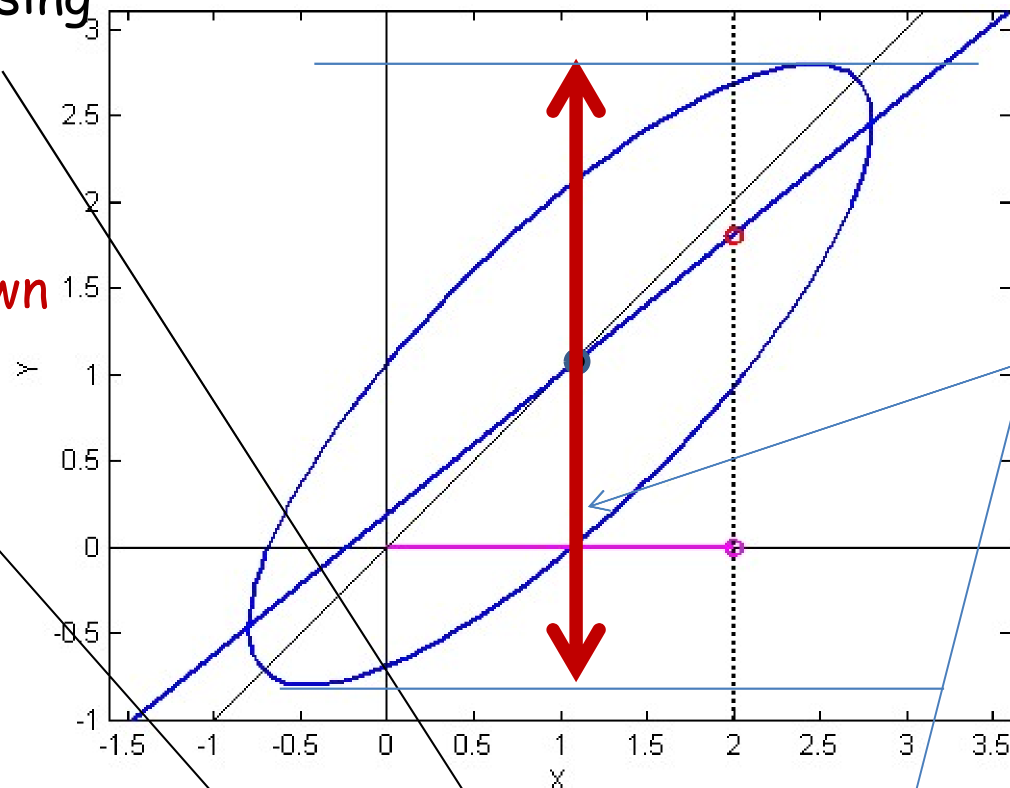
$$P(y \mid x) = N(\mu_y + C_{yx} C_{xx}^{-1} (x - \mu_x), C_{yy} - C_{yx} C_{xx}^{-1} C_{xy})$$

Slope

Preliminaries : P(y|x) for Gaussian

Correction of Y using information in X

Best guess for Y when X is not known



Uncertainty in Y when X is not known

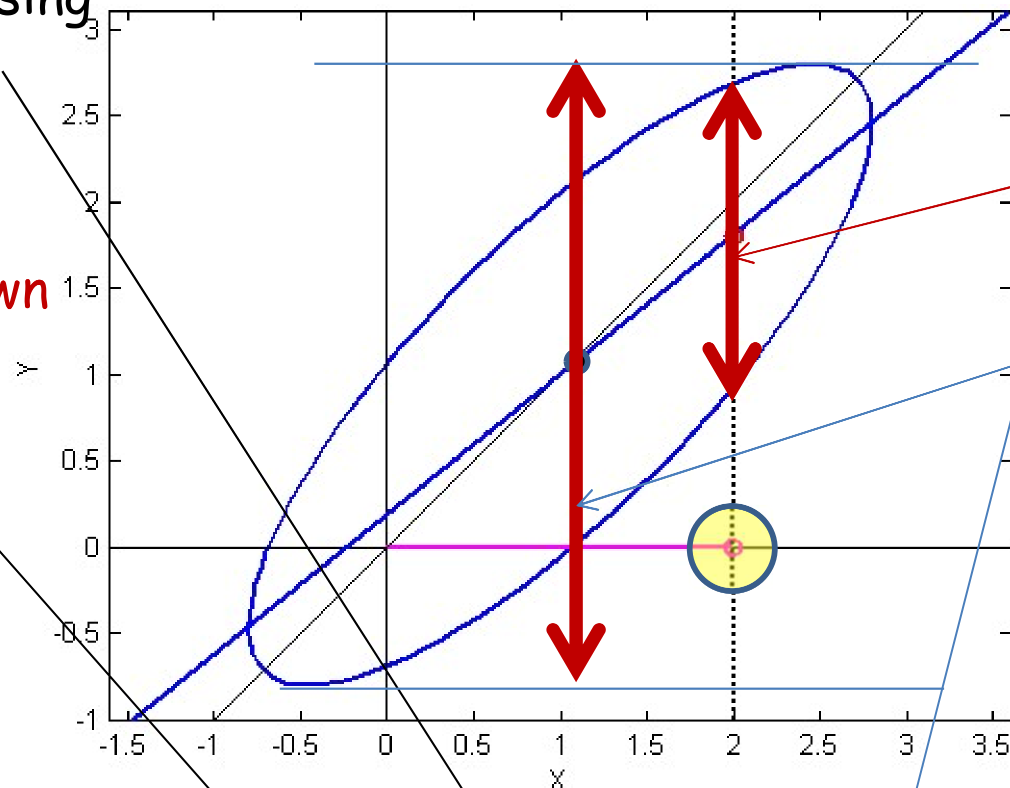
$$P(y|x) = N(\mu_y + C_{yx}C_{xx}^{-1}(x-\mu_x), C_{yy} - C_{yx}C_{xx}^{-1}C_{xy})$$

Preliminaries : P(y|x) for Gaussian

Shrinkage of variance is 0 if X and Y are uncorrelated, i.e C_{yx} = 0

Correction of Y using information in X

Best guess for Y when X is not known



Reduced uncertainty from knowing X

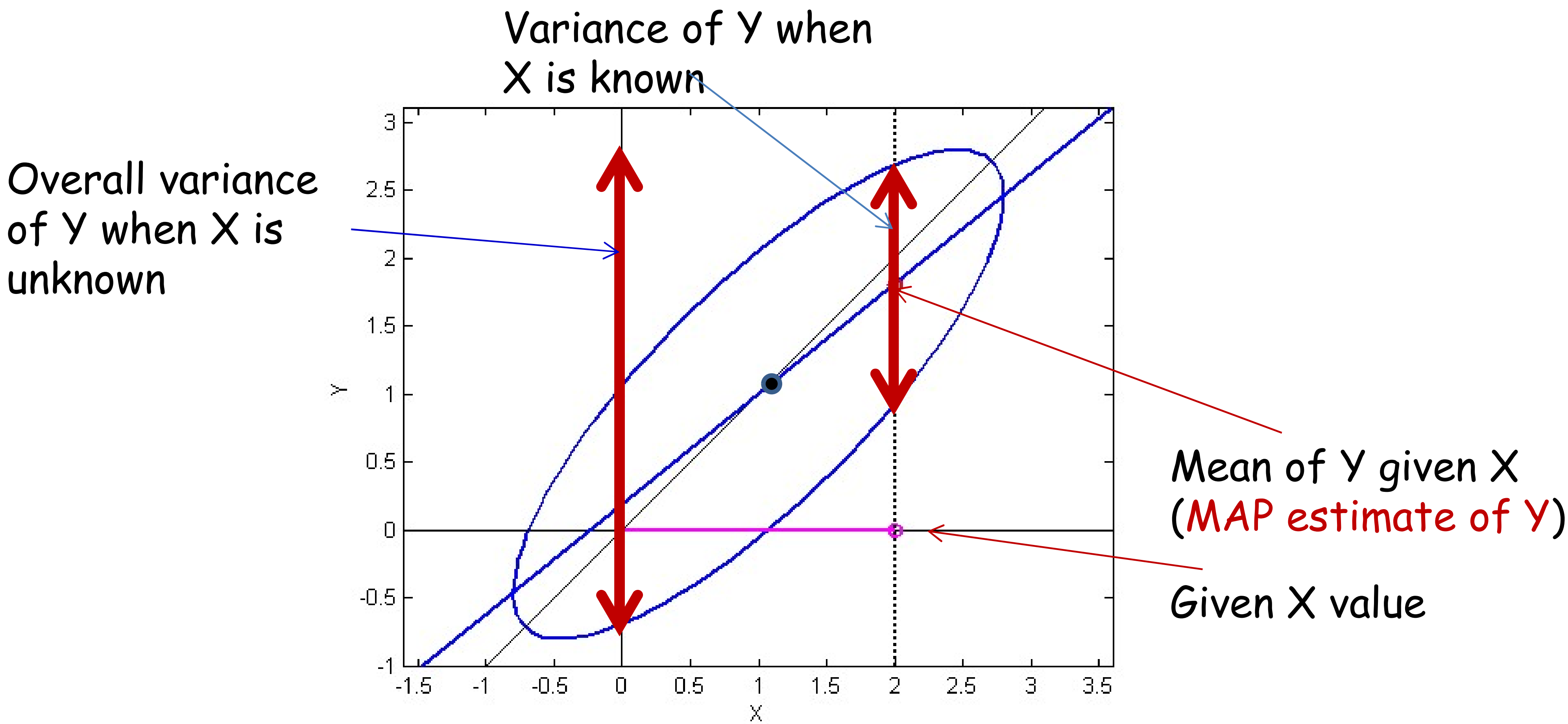
Uncertainty in Y when X is not known

Shrinkage of uncertainty from knowing X

$$P(y\,|\,x) = N(\mu_y + C_{yx}C_{xx}^{-1}(x-\mu_x),\, C_{yy} - C_{yx}C_{xx}^{-1}C_{xy})$$

Preliminaries : P(y|x) for Gaussian

Knowing X modifies the mean of Y and shrinks its variance



$$P(y \mid x) = N(\mu_y + C_{yx} C_{xx}^{-1}(x - \mu_x), C_{yy} - C_{yx} C_{xx}^{-1} C_{xy})$$

Background: Sum of Gaussian RVs

$$O = AS + \varepsilon$$

$$S \sim N(\mu_s, \Theta_s) \qquad \varepsilon \sim N(\mu_\varepsilon, \Theta_\varepsilon)$$

- Consider a random variable *O* obtained as above
- The expected value of *O* is given by

$$E[O] = E[AS + \varepsilon] = A\mu_s + \mu_\varepsilon$$

- Notation:

$$E[O] = \mu_O$$

Background: Sum of Gaussian RVs

$$O = AS + \varepsilon$$

$$S \sim N(\mu_S, \Theta_S) \qquad \varepsilon \sim N(\mu_\varepsilon, \Theta_\varepsilon)$$

- The variance of O is given by

$$Var(O) = \Theta_O = E\left[(O - \mu_O)(O - \mu_O)^T\right]$$

- This is just the sum of the variance of AS and the variance of ε

$$\Theta_O = A\Theta_S A^{\mathrm{T}} + \Theta_\varepsilon$$

Background: Sum of Gaussian RVs

$$O = AS + \varepsilon$$

$$S \sim N(\mu_S, \Theta_S) \qquad \varepsilon \sim N(\mu_\varepsilon, \Theta_\varepsilon)$$

- The conditional probability of *O*:

$$P(O|S) = N(AS + \mu_\varepsilon, \Theta_\varepsilon)$$

- The overall probability of *O*:

$$P(O) = N(A\mu_s + \mu_\varepsilon, A\Theta_S A^{\mathrm{T}} + \Theta_\varepsilon)$$

Background: Sum of Gaussian RVs

$$\boldsymbol{O} = \boldsymbol{AS} + \boldsymbol{\varepsilon}$$

$$\boldsymbol{S} \sim N(\boldsymbol{\mu_s}, \boldsymbol{\Theta_s}) \qquad \boldsymbol{\varepsilon} \sim N(\boldsymbol{\mu_\varepsilon}, \boldsymbol{\Theta_\varepsilon})$$

- **The *cross-correlation* between *O* and *S***

$$\boldsymbol{\Theta_{OS}} = \boldsymbol{E}\left[(\boldsymbol{O} - \boldsymbol{\mu_O})(\boldsymbol{S} - \boldsymbol{\mu_s})^T\right] = \boldsymbol{E}\left[(\boldsymbol{A}(\boldsymbol{S} - \boldsymbol{\mu_s}) + (\boldsymbol{\varepsilon} - \boldsymbol{\mu_\varepsilon}))(\boldsymbol{S} - \boldsymbol{\mu_s})^T\right]$$

$$= \boldsymbol{E}\left[\boldsymbol{A}(\boldsymbol{S} - \boldsymbol{\mu_s})(\boldsymbol{S} - \boldsymbol{\mu_s})^T + (\boldsymbol{\varepsilon} - \boldsymbol{\mu_\varepsilon})(\boldsymbol{S} - \boldsymbol{\mu_s})^T\right]$$

$$= \boldsymbol{AE}\left[(\boldsymbol{S} - \boldsymbol{\mu_s})(\boldsymbol{S} - \boldsymbol{\mu_s})^T\right] + \boldsymbol{E}\left[(\boldsymbol{\varepsilon} - \boldsymbol{\mu_\varepsilon})(\boldsymbol{S} - \boldsymbol{\mu_s})^T\right]$$

$$= \boldsymbol{AE}\left[(\boldsymbol{S} - \boldsymbol{\mu_s})(\boldsymbol{S} - \boldsymbol{\mu_s})^T\right]$$

- $= \boldsymbol{A}\ \boldsymbol{\Theta_s}$
- The cross-correlation between *O* and *S* is

$$\boldsymbol{\Theta_{OS}} = \boldsymbol{A\Theta_S}$$

$$\boldsymbol{\Theta_{SO}} = \boldsymbol{\Theta_S A^T}$$

Background: Joint Prob. of O and S

$$O = AS + \varepsilon$$

$$Z = \begin{bmatrix} O \\ S \end{bmatrix}$$

- The joint probability of *O* and *S* (i.e. P(*Z*)) is also Gaussian

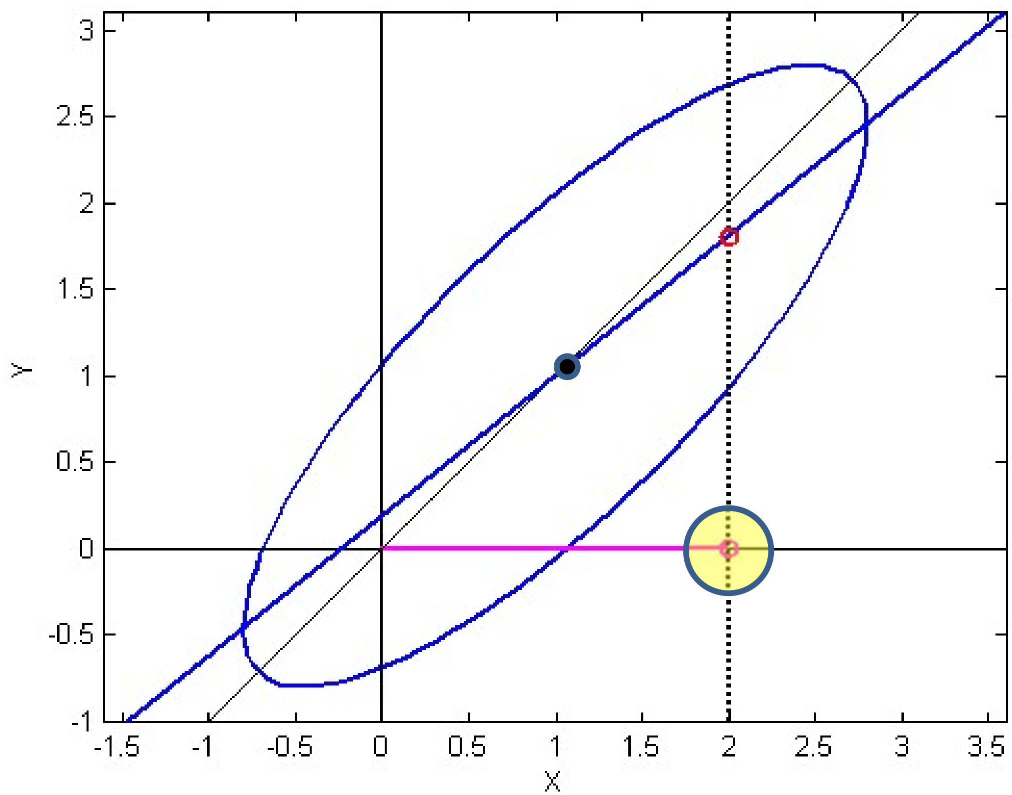
$$P(Z) = P(O, S) = N(\mu_Z, \Theta_Z)$$

- Where

$$\mu_Z = \begin{bmatrix} \mu_O \\ \mu_S \end{bmatrix} = \begin{bmatrix} A\mu_s + \mu_\varepsilon \\ \mu_S \end{bmatrix}$$

- $\Theta_Z = \begin{bmatrix} \Theta_O & \Theta_{OS} \\ \Theta_{SO} & \Theta_S \end{bmatrix} = \begin{bmatrix} A\Theta_S A^T + \Theta_\varepsilon & A\Theta_S \\ \Theta_S A^T & \Theta_S \end{bmatrix}$

Preliminaries : Conditional of S given O: P(S|O)



$$O = AS + \varepsilon$$

$$P(S|O) = N(\mu_S + \Theta_{SO}\Theta_O^{-1}(O - \mu_O),\ \ \Theta_S - \Theta_{SO}\Theta_O^{-1}\Theta_{OS})$$

$$P(S|O) = N(\mu_S + \Theta_S A^{\mathrm{T}}\left(A\Theta_S A^{\mathrm{T}} + \Theta_\varepsilon\right)^{-1}(O - A\mu_s - \mu_\varepsilon),$$
$$\Theta_S - \Theta_S A^{\mathrm{T}}\left(A\Theta_S A^{\mathrm{T}} + \Theta_\varepsilon\right)^{-1} A\Theta_S)$$

Poll 1

- If P(x, y) is joint Gaussian distribution, then the conditional probability of y given x P(y |x) is also Gaussian.
  - True
  - False

- If X is a Gaussian random variable, than a linear transformation of X is also a Gaussian variable
  - True
  - False

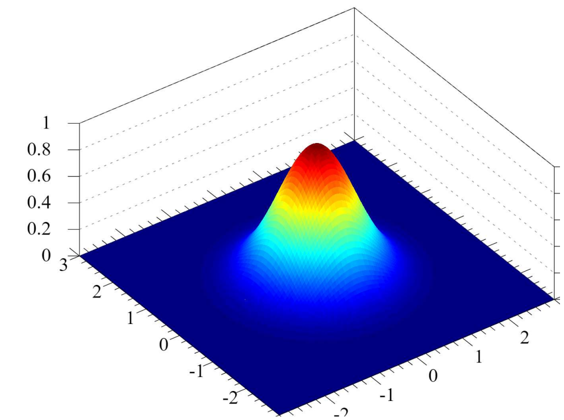
Poll 1

- If P(x, y) is joint Gaussian distribution, then the conditional probability of y given x P(y |x) is also Gaussian.
  - **True**
  - False

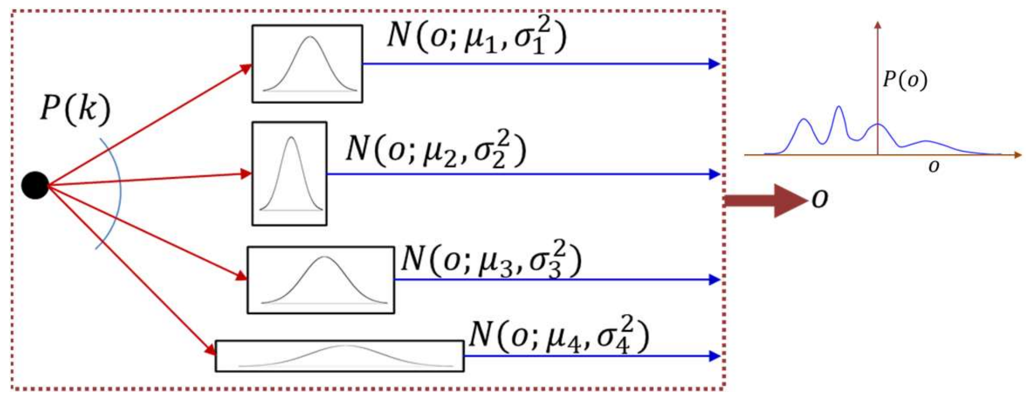
- If X is a Gaussian random variable, than a linear transformation of X is also a Gaussian variable
  - **True**
  - False

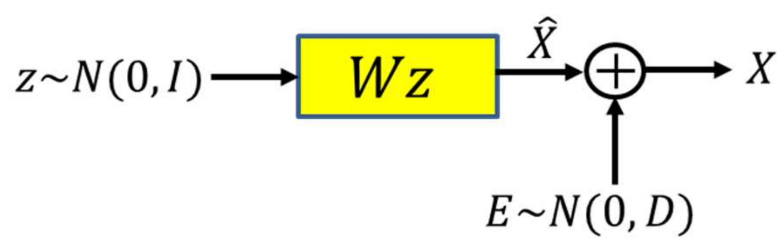
Recap: Examples of Generative Models

- Generative models can be simple, one step models of the generating
  - E.g. Gaussians, Multinomials



- Or a multi-step generating process
  - E.g. Gaussian Mixtures
  - E.g. Linear Gaussian Models





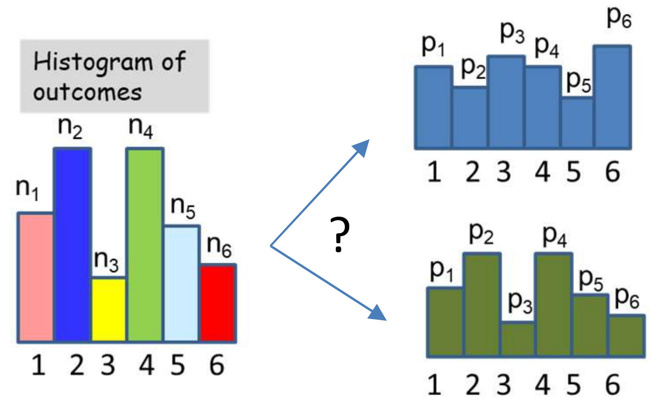
Recap: ML Estimation of Generative Models

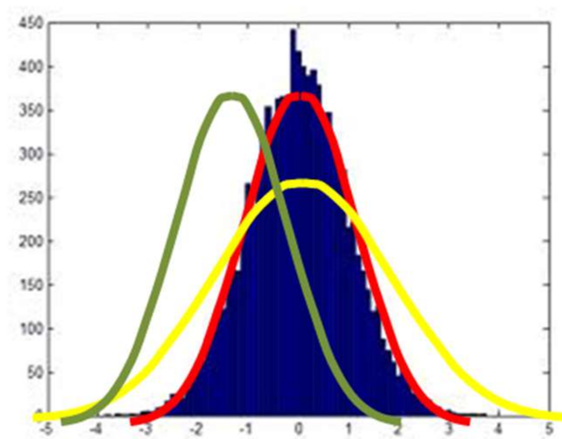
- Must estimate the parameters of the model from observed data

- Maximum likelihood estimation: Choose parameters to maximize the (log) likelihood of observed data

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \log(P(X;\theta))$$

$$= \underset{\theta}{\operatorname{argmax}} \sum_{x \in X} \log(P(x;\theta))$$





Recap: ML estimation from incomplete data

- In many situations, our observed data are missing information
  - E.g. components of the data
  - E.g. "inside" information about how the data are drawn by the model

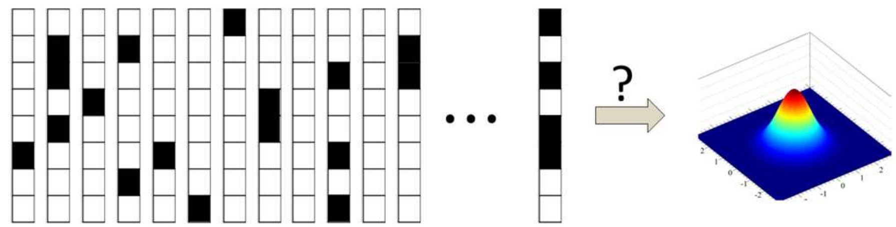
- In these cases, the ML estimate must only consider the *observed* data O

$$\underset{\theta}{\text{argmax}} \sum_{o \in O} \log P(o;\theta)$$

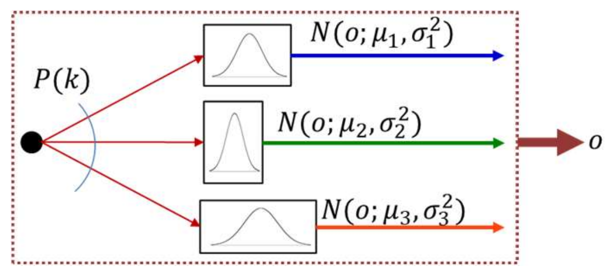
  - But the observed data are incomplete

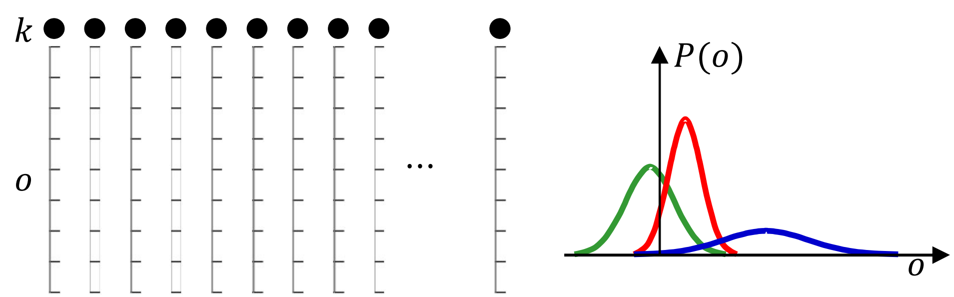
- Observation probability $P(o)$ must be obtained from the *complete* data probability, by marginalizing out missing components
  - This can cause ML estimation to become challenging

Observation O consists of only the unblackened components



Blacked-out components are missing from data





Recap: The Expectation Maximization Algorithm

- Define the *auxiliary* function:

$$Q(\theta, \theta^k) = \sum_{o \in O} \sum_{h} P(h|o; \theta^k) \log P(h, o; \theta)$$

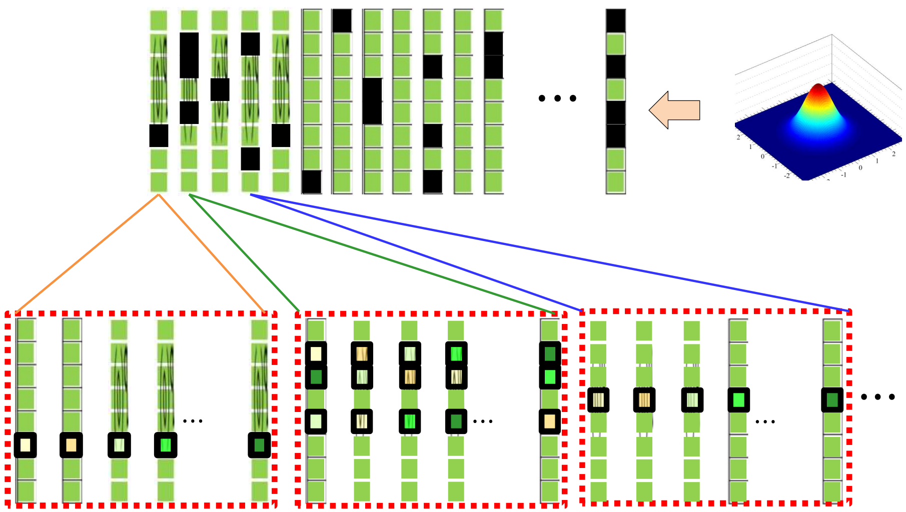
  - Which is the ELBO plus a term that doesn't depend on θ

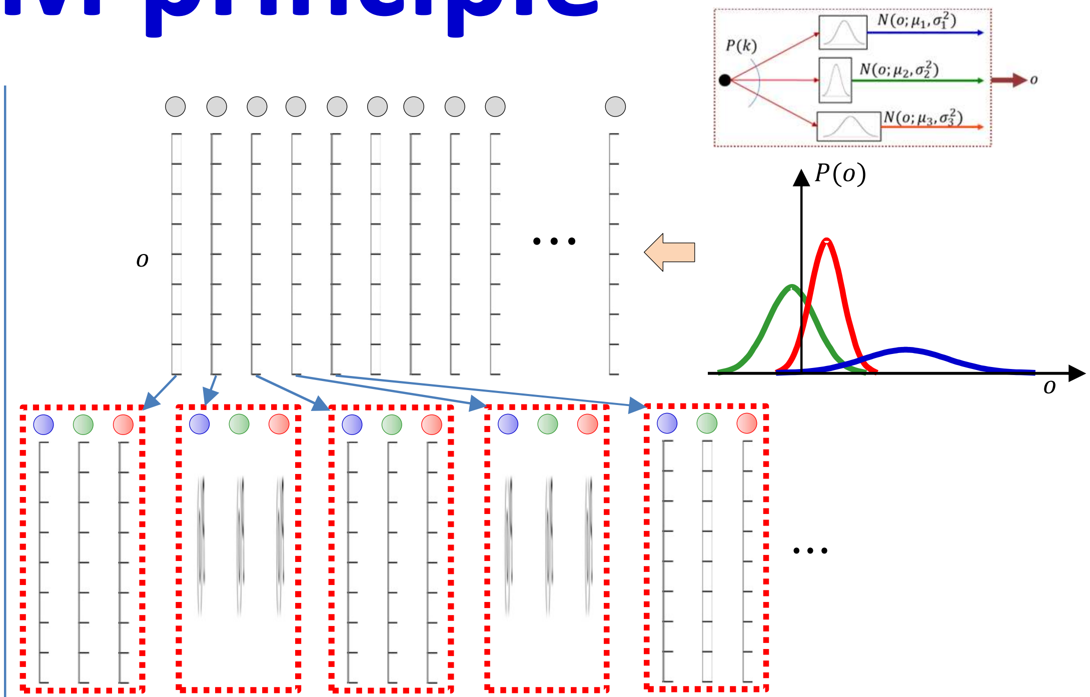
- Iteratively compute

$$\theta^{k+1} \leftarrow \underset{\theta}{\operatorname{argmax}}\, Q(\theta, \theta^k)$$

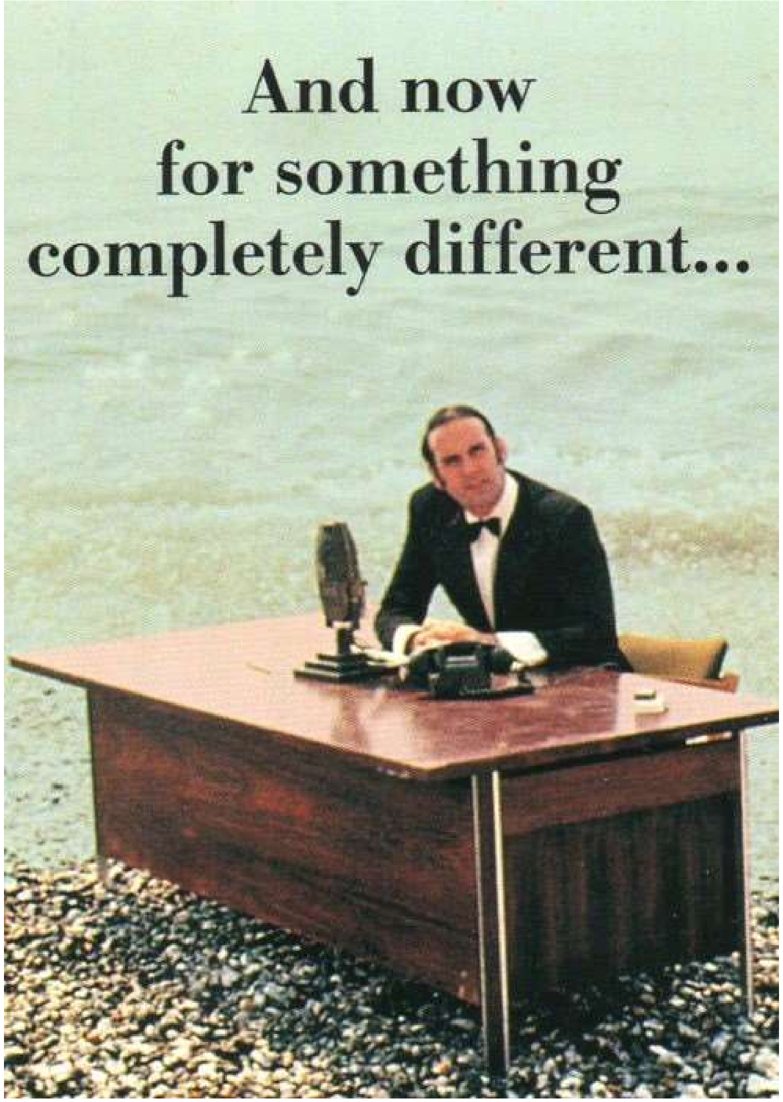
- Guaranteed to increase $\log P(o)$ with every iteration

Recap: EM principle





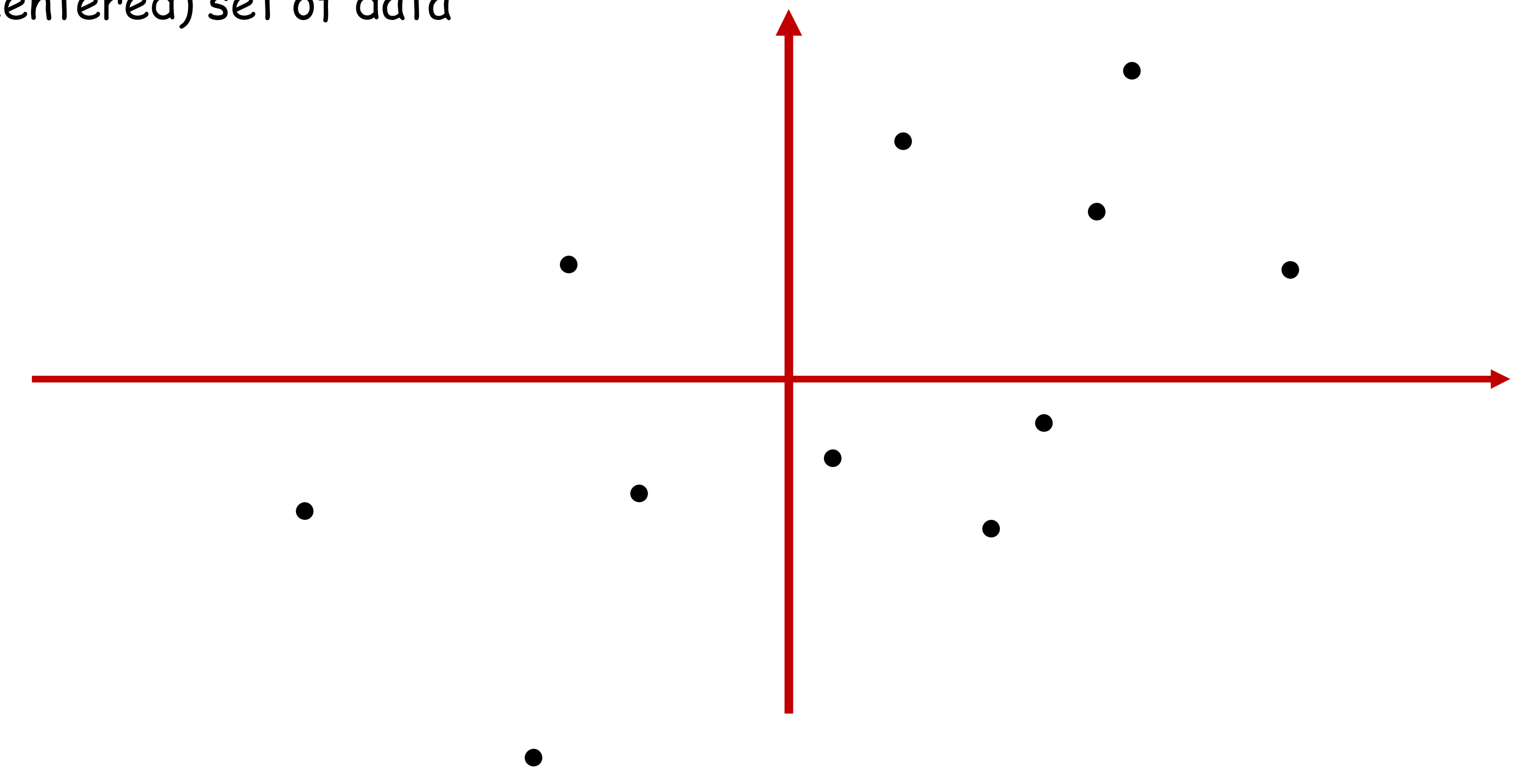
- Iteratively:
- **Complete the data according to the posterior probabilities $P(m|o)$ computed by the current model**
  - By explicitly considering every possible value, with its posterior-based proportionality
  - Or by sampling the posterior probability distribution $P(m|o)$
    - Upon completion each incomplete observation implicitly or explicitly becomes many (potentially infinite) complete observations
- ***Reestimate the model from completed data***



Principal Component Analysis

Principal Component Analysis

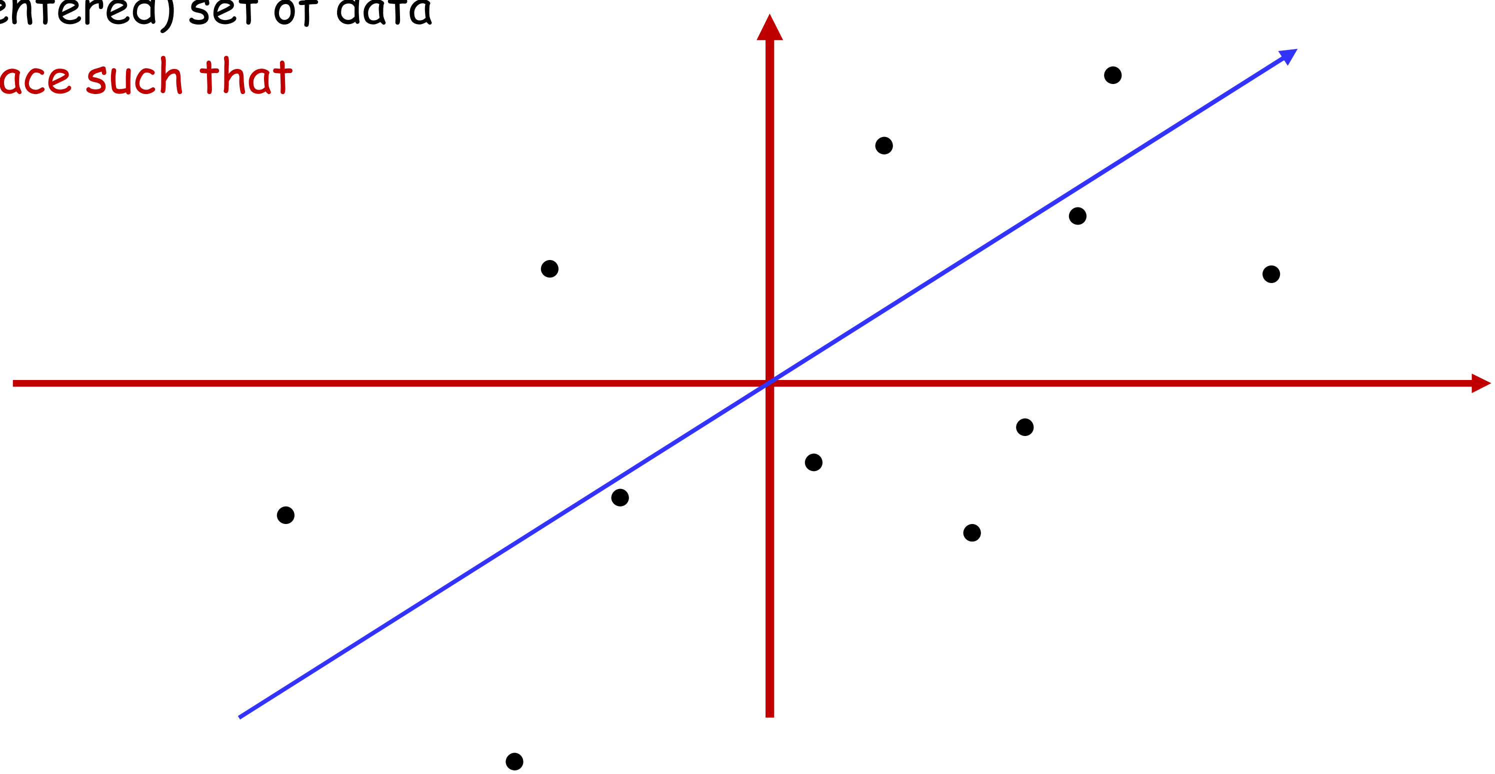
Given a (centered) set of data



- Find the principal subspace such that when all vectors are approximated as lying on that subspace, the approximation error is minimal
  - Assuming "centered" (zero-mean) data

Principal Component Analysis

Given a (centered) set of data
find subspace such that

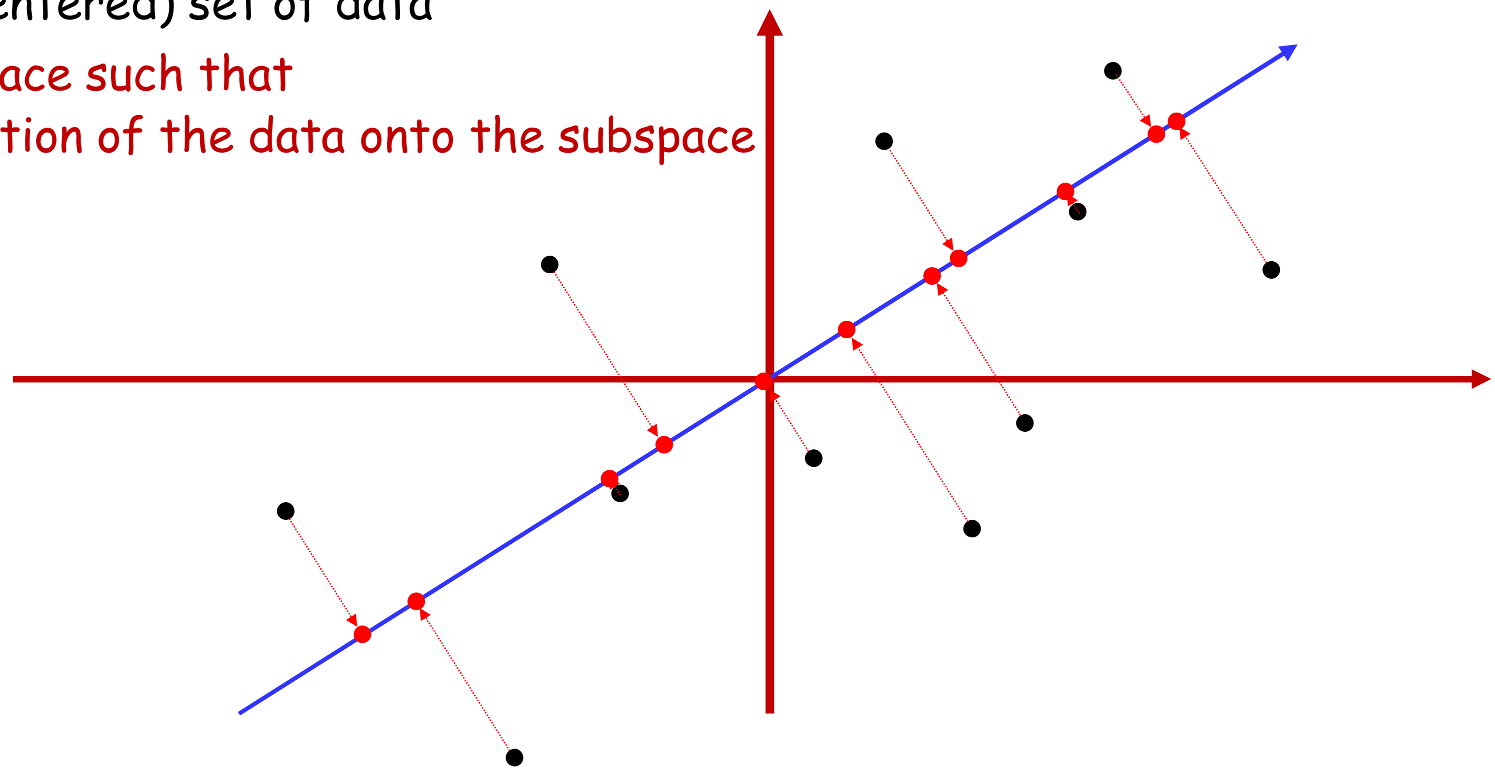


- Find the principal subspace such that when all vectors are approximated as lying on that subspace, the approximation error is minimal
  - Assuming "centered" (zero-mean) data

Principal Component Analysis

Given a (centered) set of data

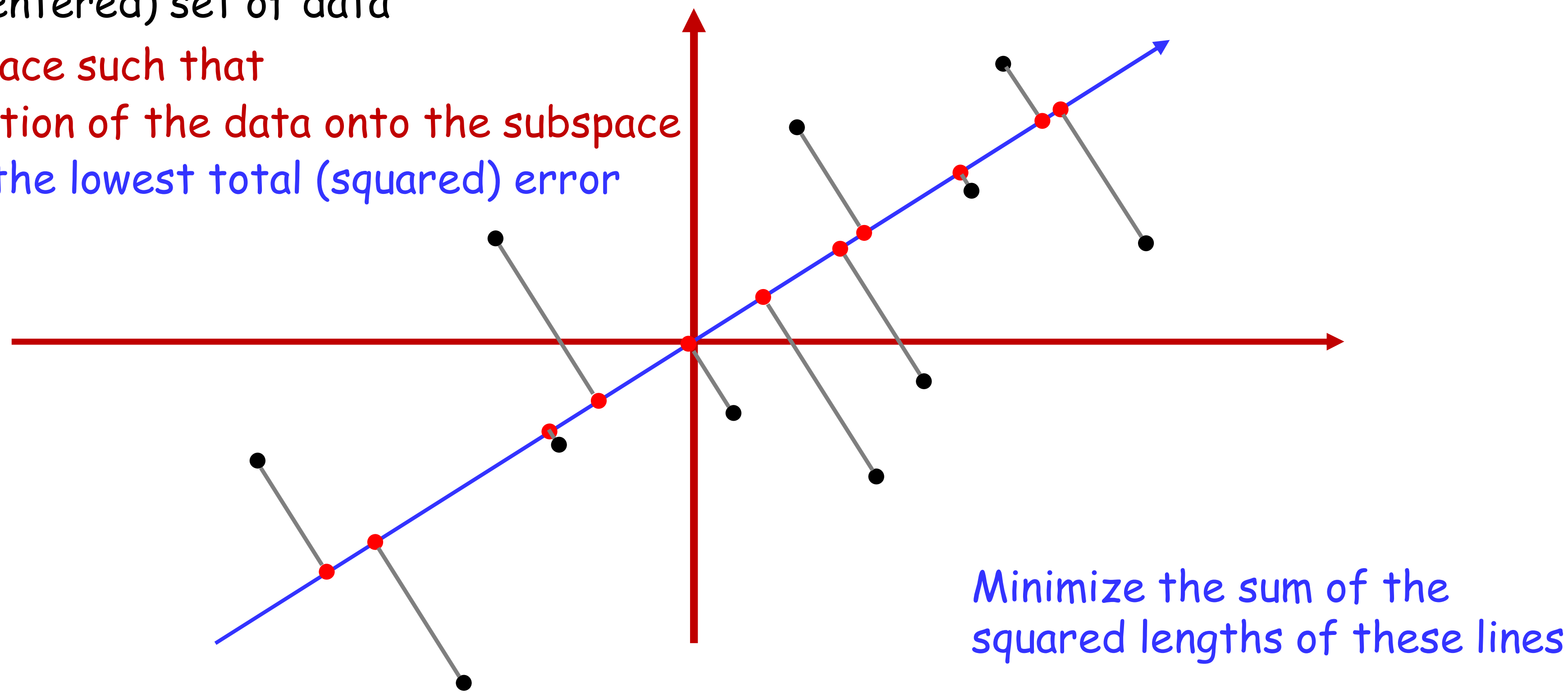
find subspace such that
the projection of the data onto the subspace



- Find the principal subspace such that when all vectors are approximated as lying on that subspace, the approximation error is minimal
  - Assuming "centered" (zero-mean) data

Principal Component Analysis

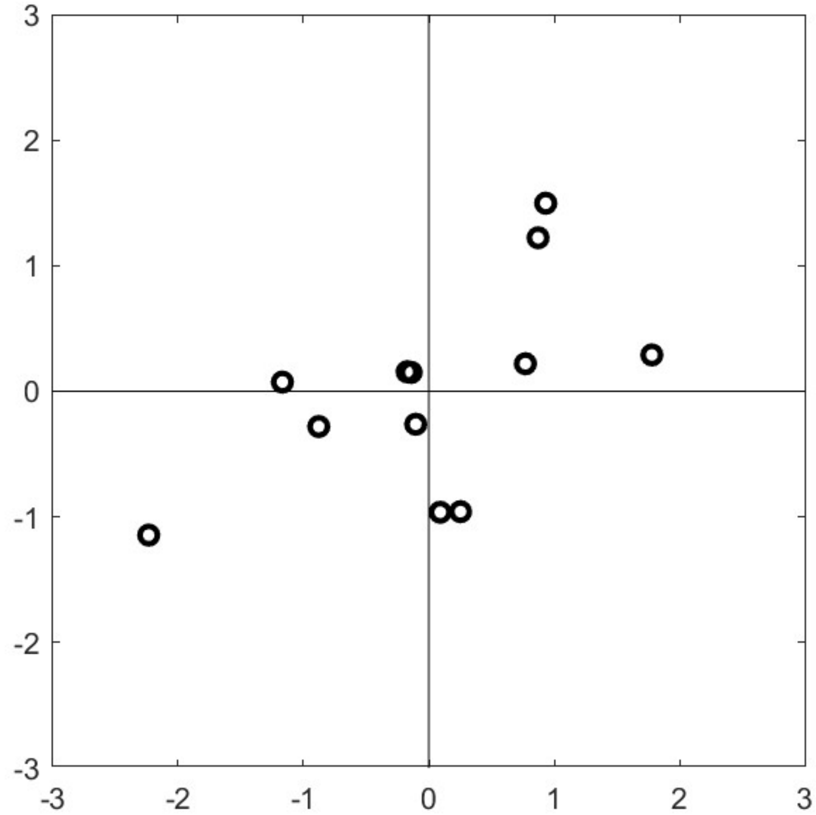
Given a (centered) set of data
find subspace such that
the projection of the data onto the subspace
results in the lowest total (squared) error



- Find the principal subspace such that when all vectors are approximated as lying on that subspace, the approximation error is minimal
  - Assuming "centered" (zero-mean) data

Principal Component Analysis

Animation:
Original centered data

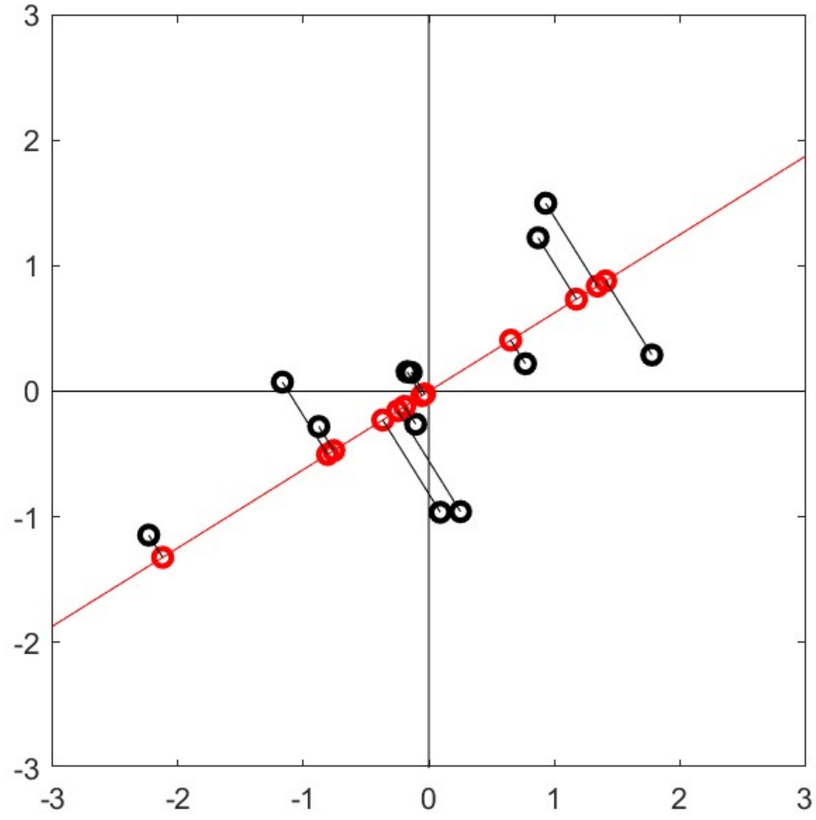


- Find the principal subspace such that when all vectors are approximated as lying on that subspace, the approximation error is minimal
  - Assuming "centered" (zero-mean) data

Principal Component Analysis

Animation:
Original centered data

Principal axis we're searching for

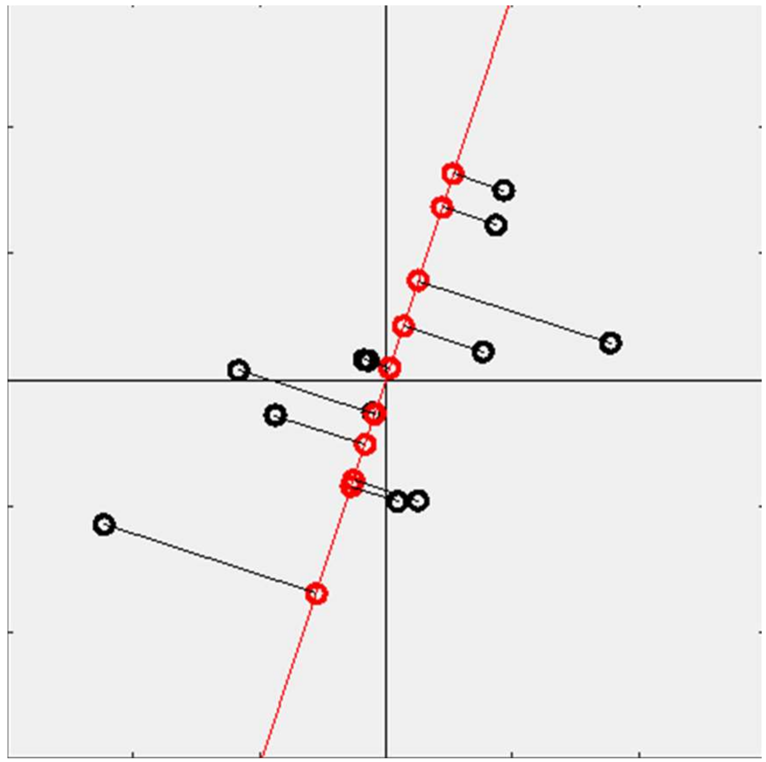


- Find the principal subspace such that when all vectors are approximated as lying on that subspace, the approximation error is minimal
  – Assuming "centered" (zero-mean) data

Principal Component Analysis

Animation:
Original centered data

Principal axis we're searching for

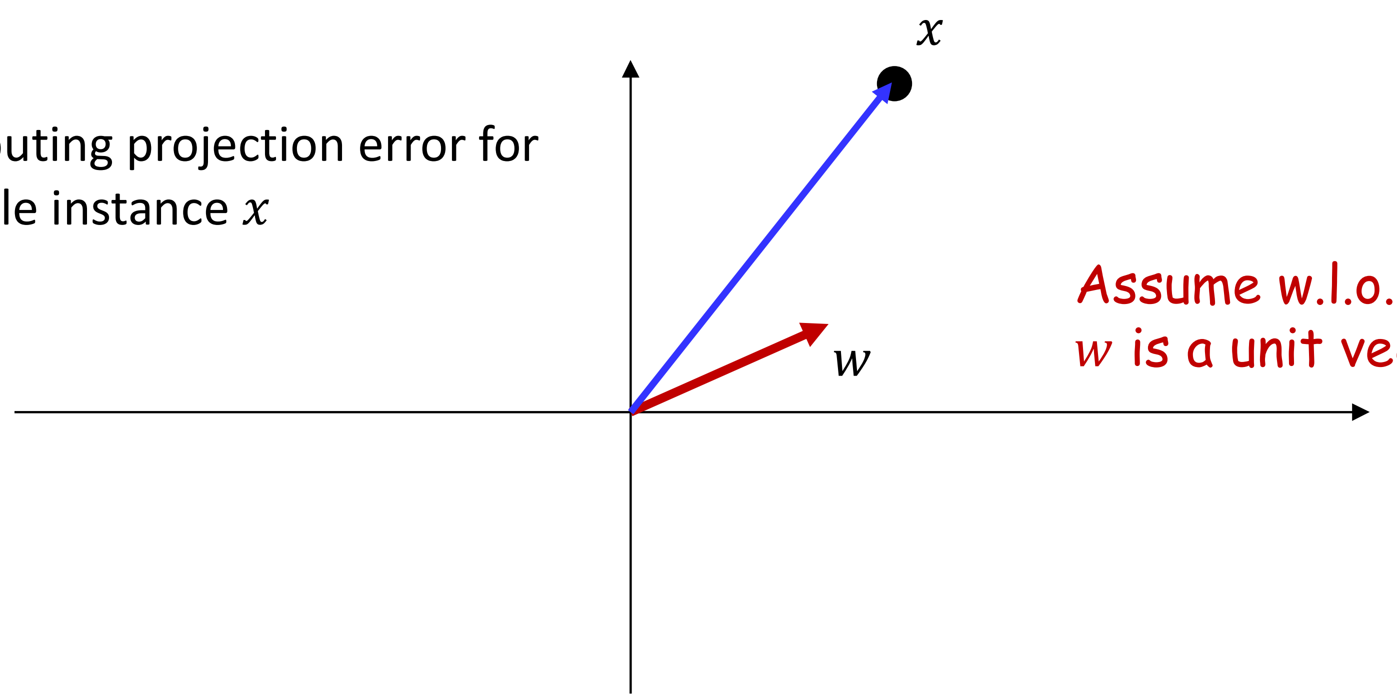


Search through all subspaces to find the one with minimum projection error

- Find the principal subspace such that when all vectors are approximated as lying on that subspace, the approximation error is minimal
  - Assuming "centered" (zero-mean) data

Can be done in closed form

Computing projection error for a single instance x

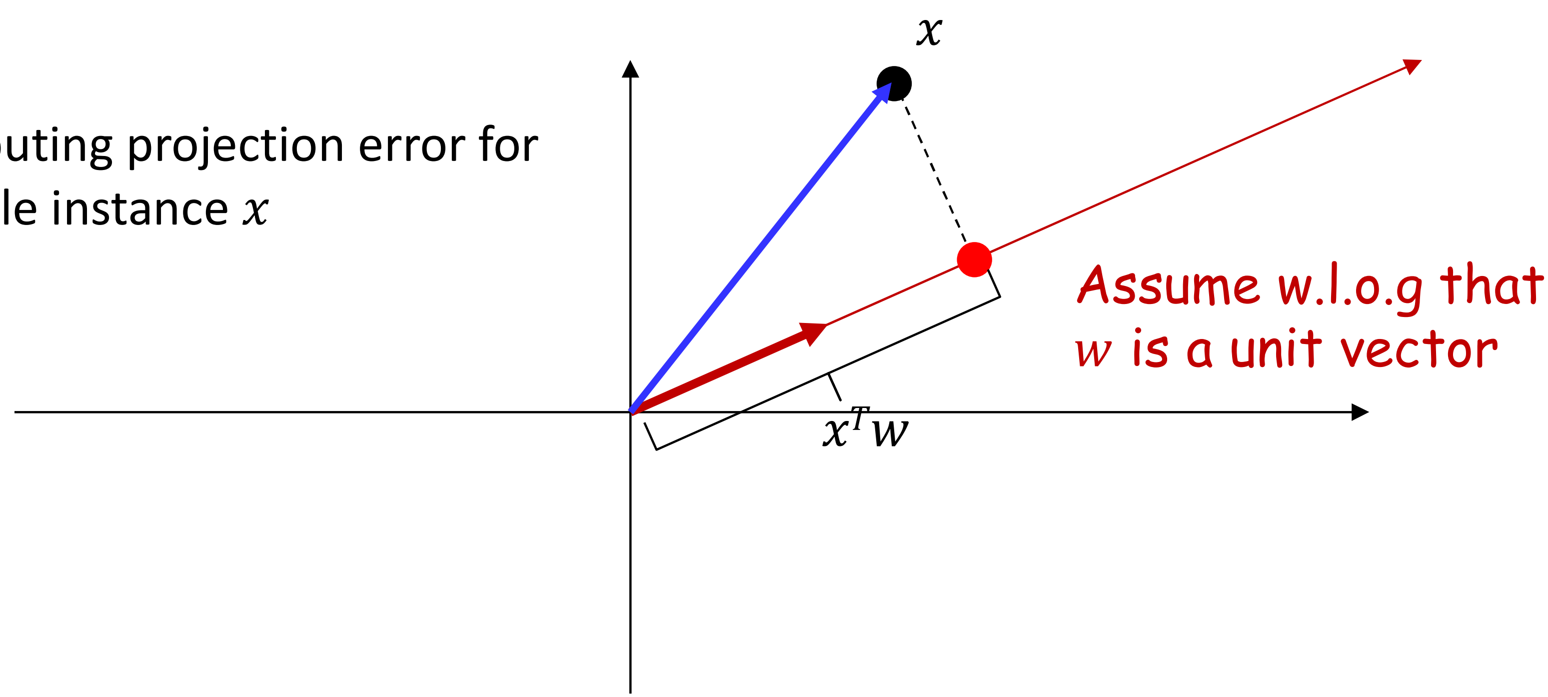


Assume w.l.o.g that w is a unit vector

- Since we're minimizing quadratic L_2 error, we can find a closed form solution

Can be done in closed form

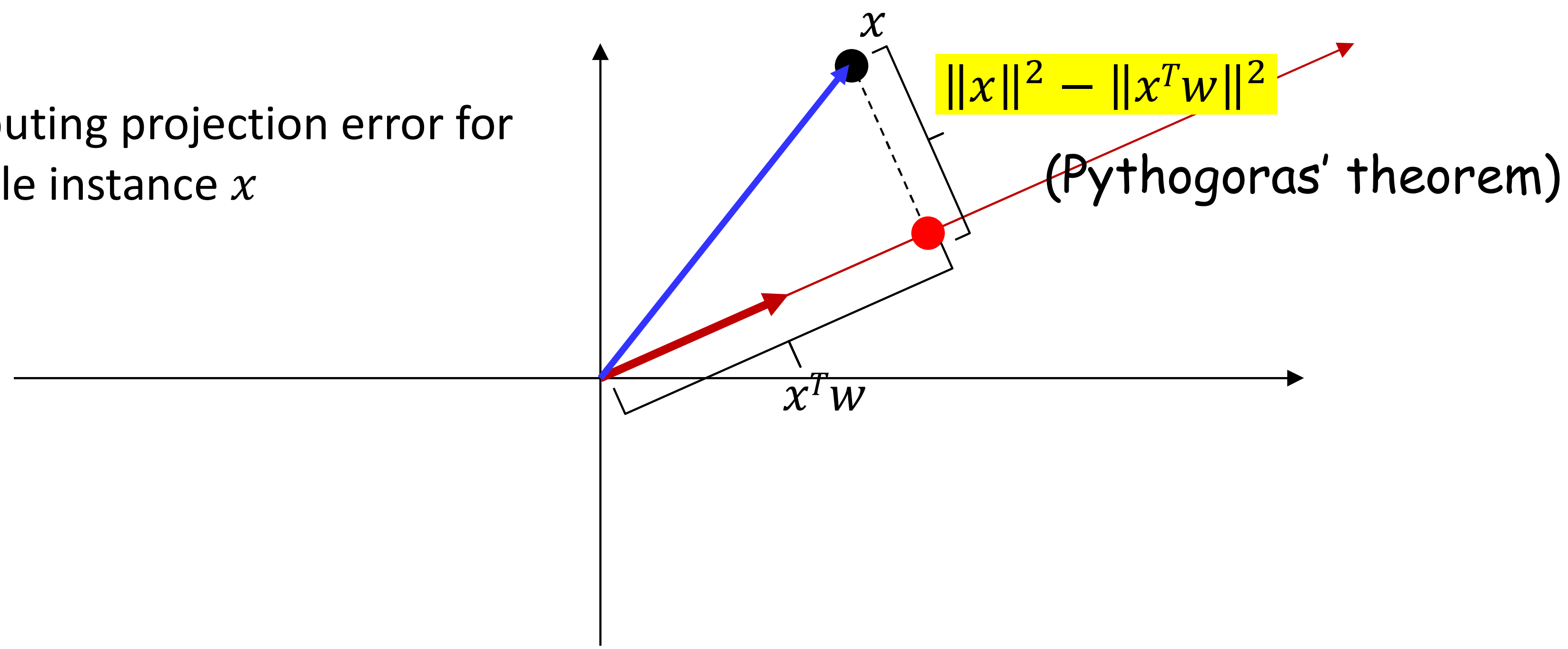
Computing projection error for a single instance x



- Since we're minimizing quadratic L_2 error, we can find a closed form solution

Can be done in closed form

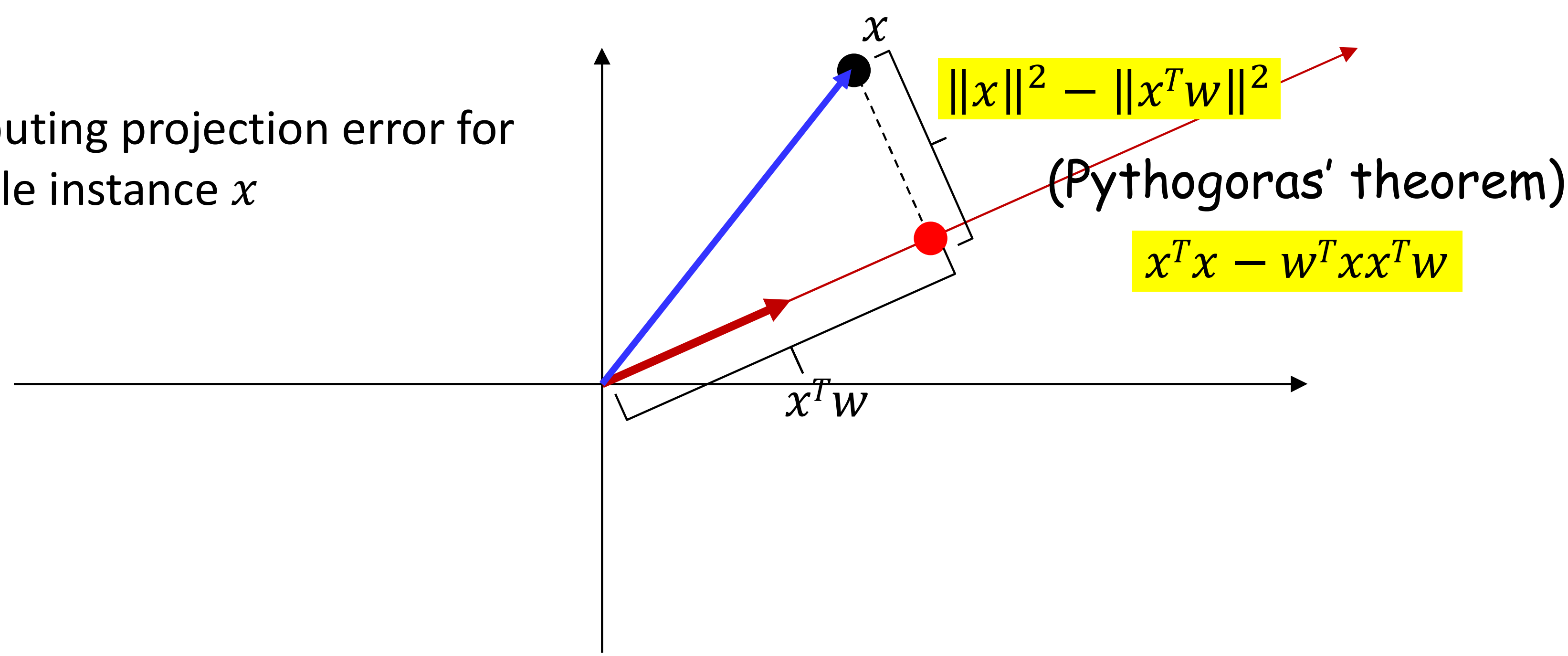
Computing projection error for a single instance x



- Since we're minimizing quadratic L_2 error, we can find a closed form solution

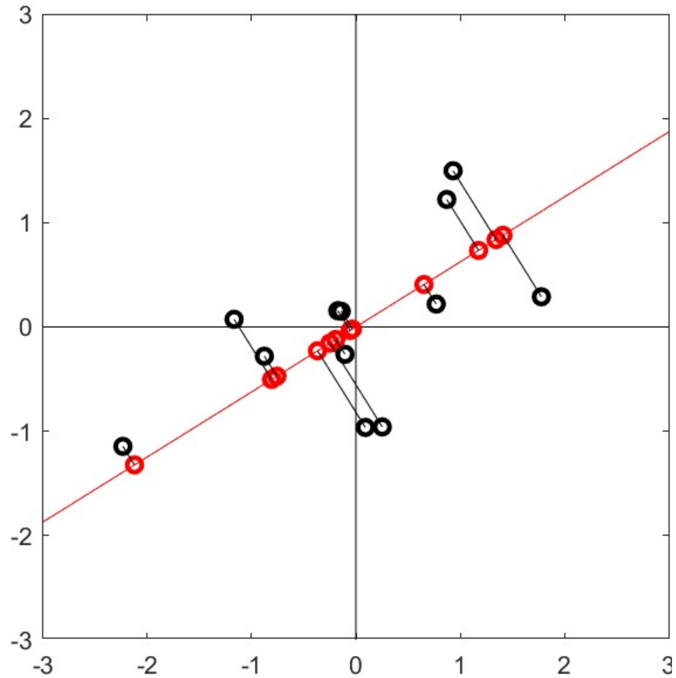
Can be done in closed form

Computing projection error for a single instance x



- Since we're minimizing quadratic L_2 error, we can find a closed form solution

Can be done in closed form



- Since we're minimizing quadratic L_2 error, we can find a closed form solution
- Total projection error for all data:

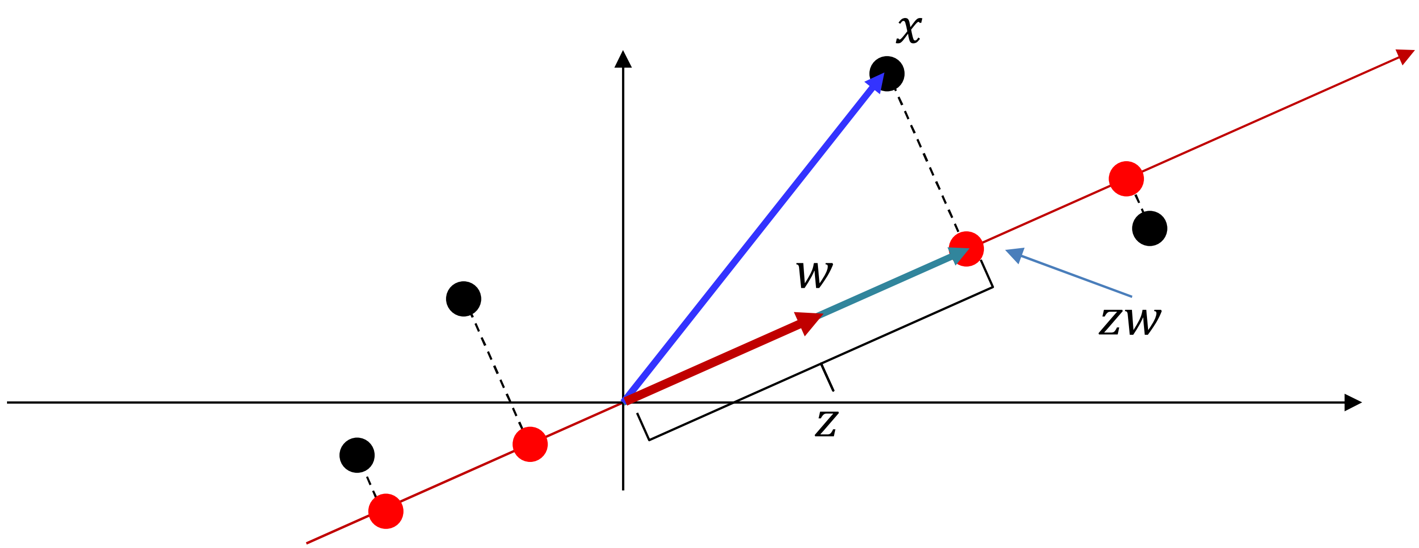
$$L = \sum_x x^T x - w^T x x^T w$$

- Minimizing this w.r.t w (subject to w = unit vector) gives you the Eigenvalue equation

$$\left(\sum_x x x^T \right) w = \lambda w$$

- This can be solved to find the principal subspace

There’s also an iterative solution

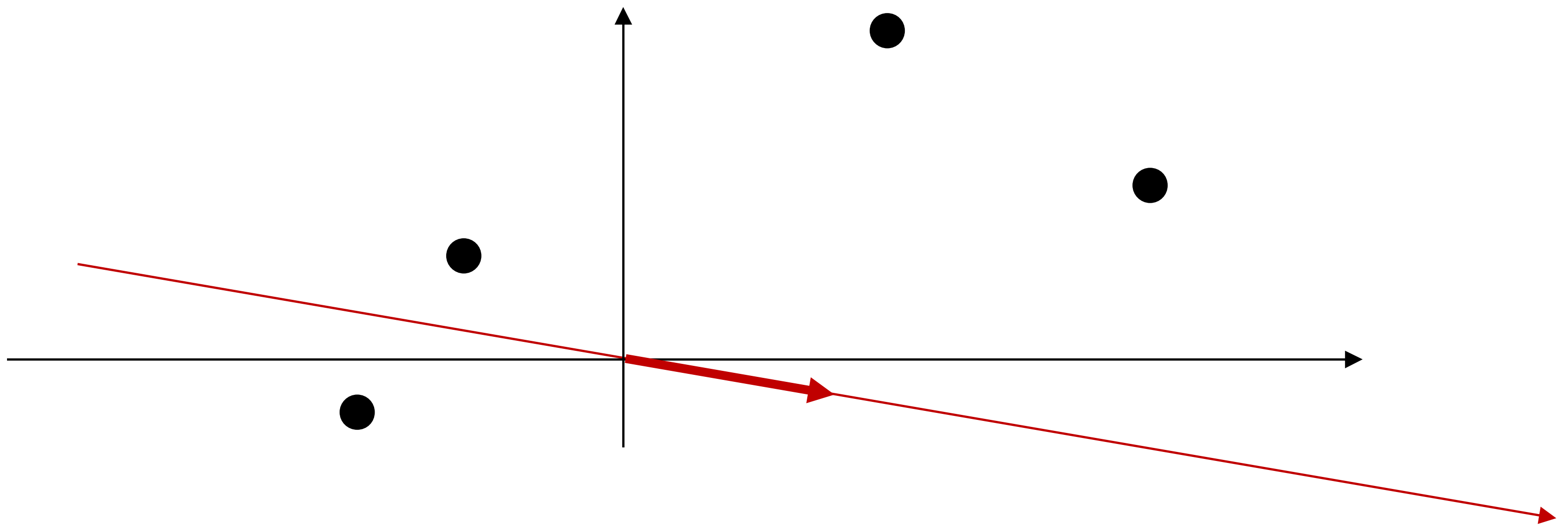


- Objective: find a vector (subspace) w and a *position* z on w such that $zw \approx x$ most closely (in an L_2 sense) for the entire (training) data

- Let $X = [x_1 x_2 \dots x_N]$ be the entire training set (arranged as a matrix)
  - Objective: find vector bases (for the subspace) W and the set of *position vectors* $Z = [z_1 z_2 \dots z_N]$ for all vectors in X such that $WZ \approx X$

- Initialize W
- Iterate until convergence:
  - Given W, find the best position vectors Z: $Z \leftarrow W^{+}X$
  - Given position vectors Z, find the best subspace: $W \leftarrow XZ^{+}$
  - Guaranteed to find the principal subspace

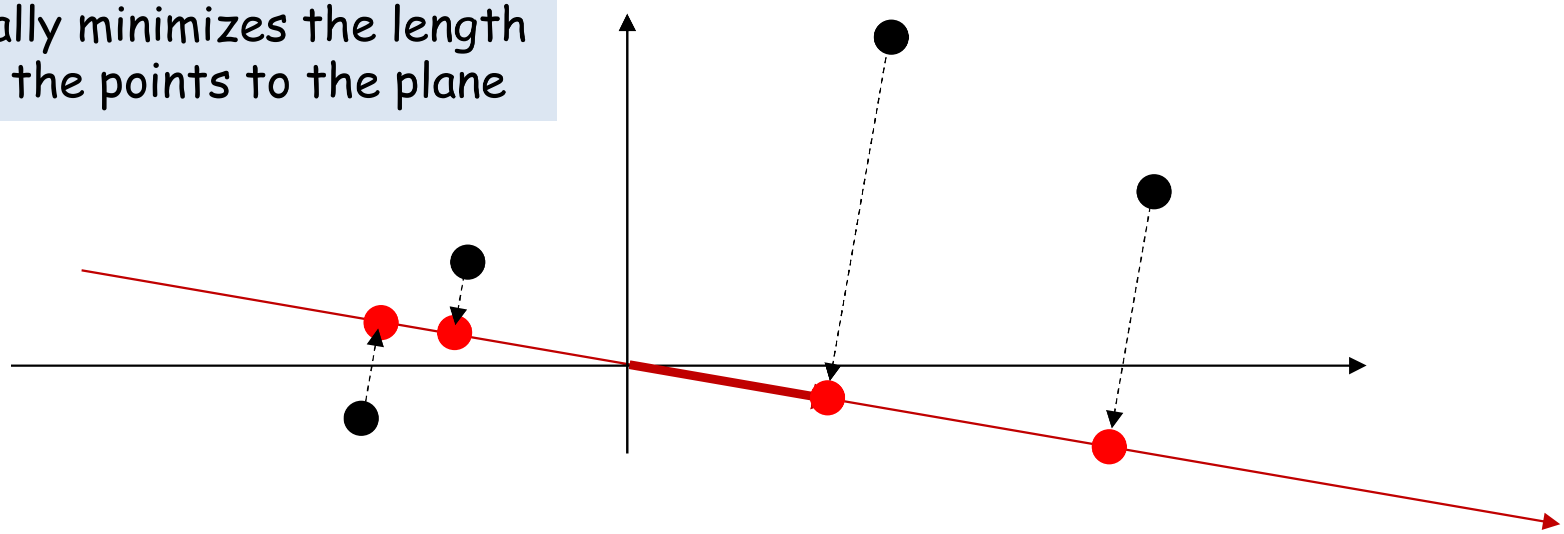
The iterative algorithm



- Initialize a subspace (the basis w)

The iterative algorithm

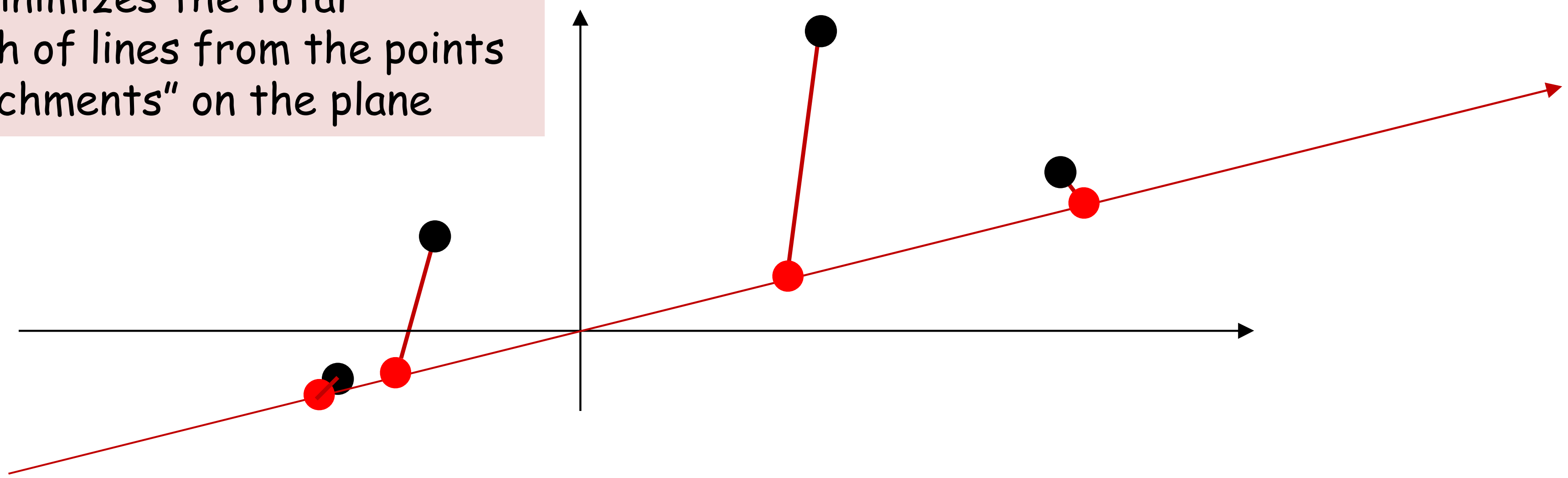
This individually minimizes the length of lines from the points to the plane



- Initialize a subspace (the basis w)
- Iterate until convergence:
  - Find the best position vectors Z on the W subspace for each training instance
    - Find the location on W that is *closest* to each instance, i.e. the perpendicular projection

The iterative algorithm

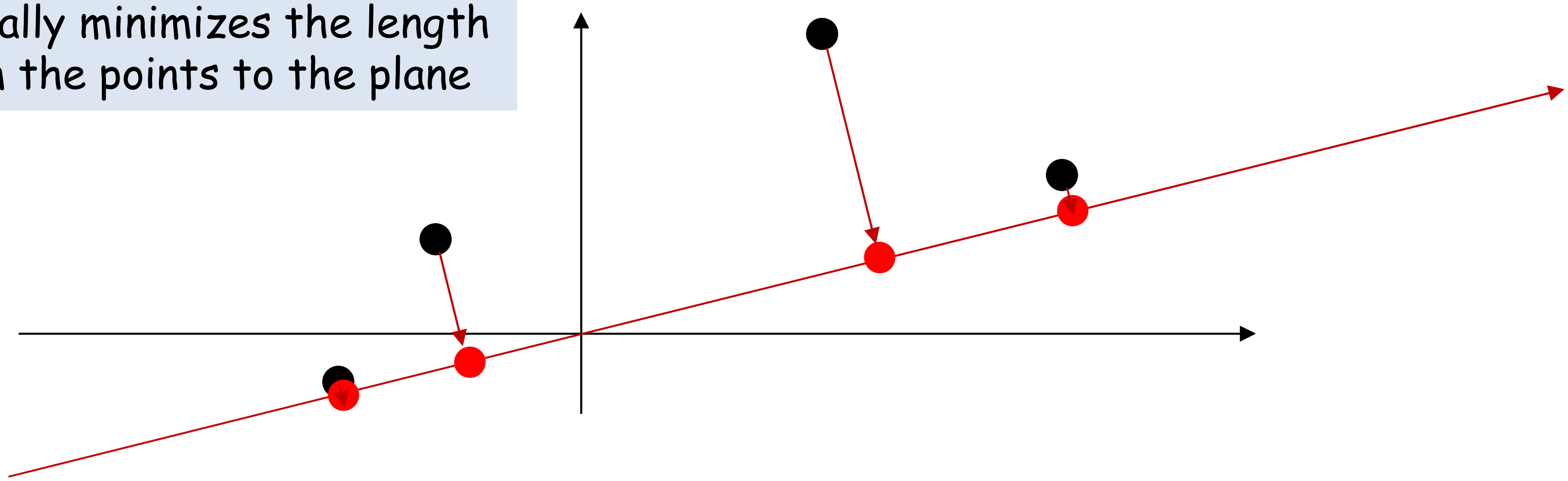
This *jointly* minimizes the total squared length of lines from the points to their "attachments" on the plane



- Initialize a subspace (the basis w)
- Iterate until convergence:
  - Find the best position vectors Z on the W subspace for each training instance
    - Find the location on W that is *closest* to each instance, i.e. the perpendicular projection
  - Let W rotate and stretch/shrink, keeping the arrangement of Z locations fixed
    - Minimize the total square length of the lines attaching the projection on the place to the instance

The iterative algorithm

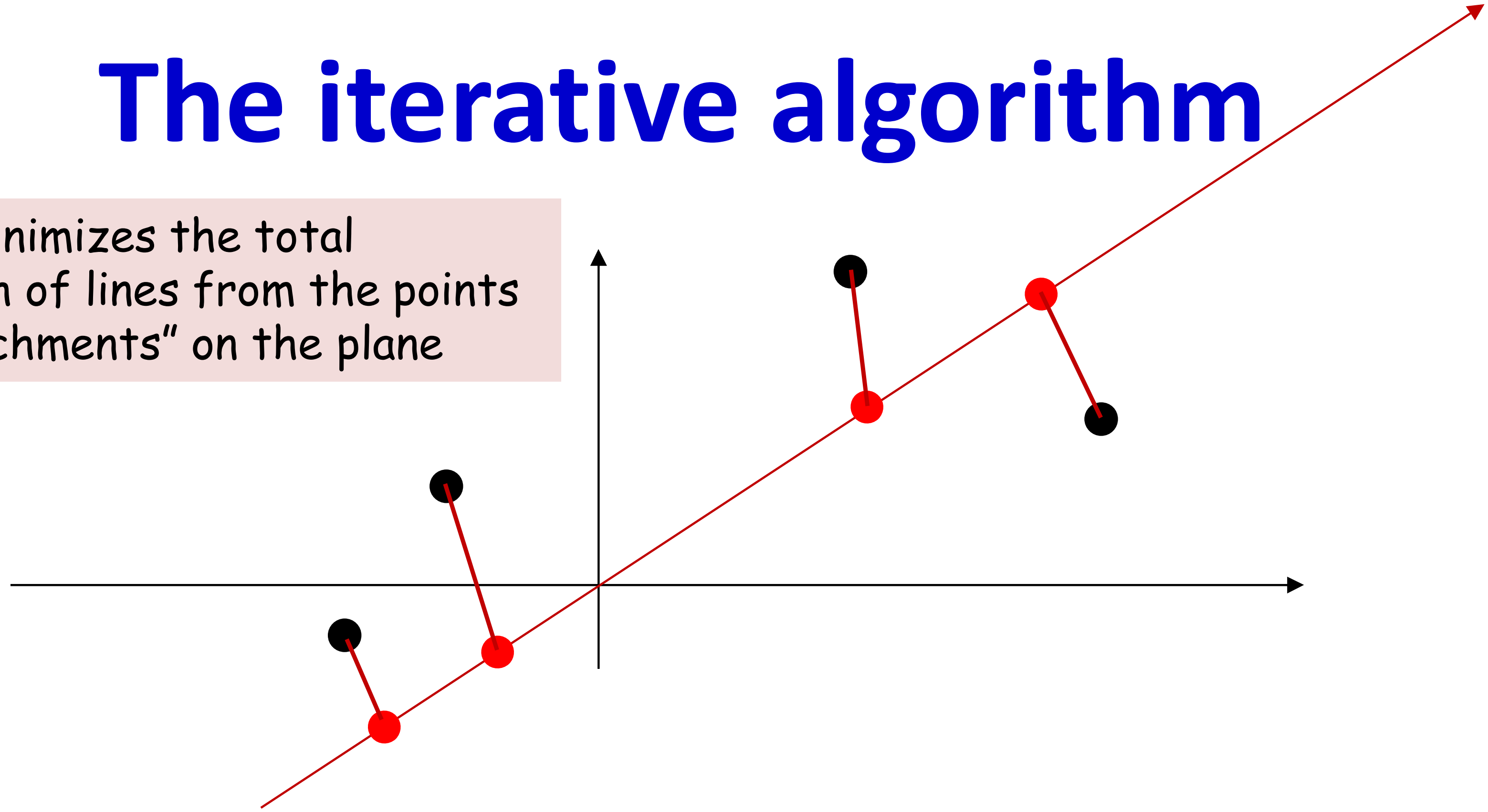
This individually minimizes the length of lines from the points to the plane



- Initialize a subspace (the basis w)
- Iterate until convergence:
  - Find the best position vectors Z on the W subspace for each training instance
    - Find the location on W that is *closest* to each instance, i.e. the perpendicular projection
  - Let W rotate and stretch/shrink, keeping the arrangement of Z locations fixed
    - Minimize the total square length of the lines attaching the projection on the place to the instance

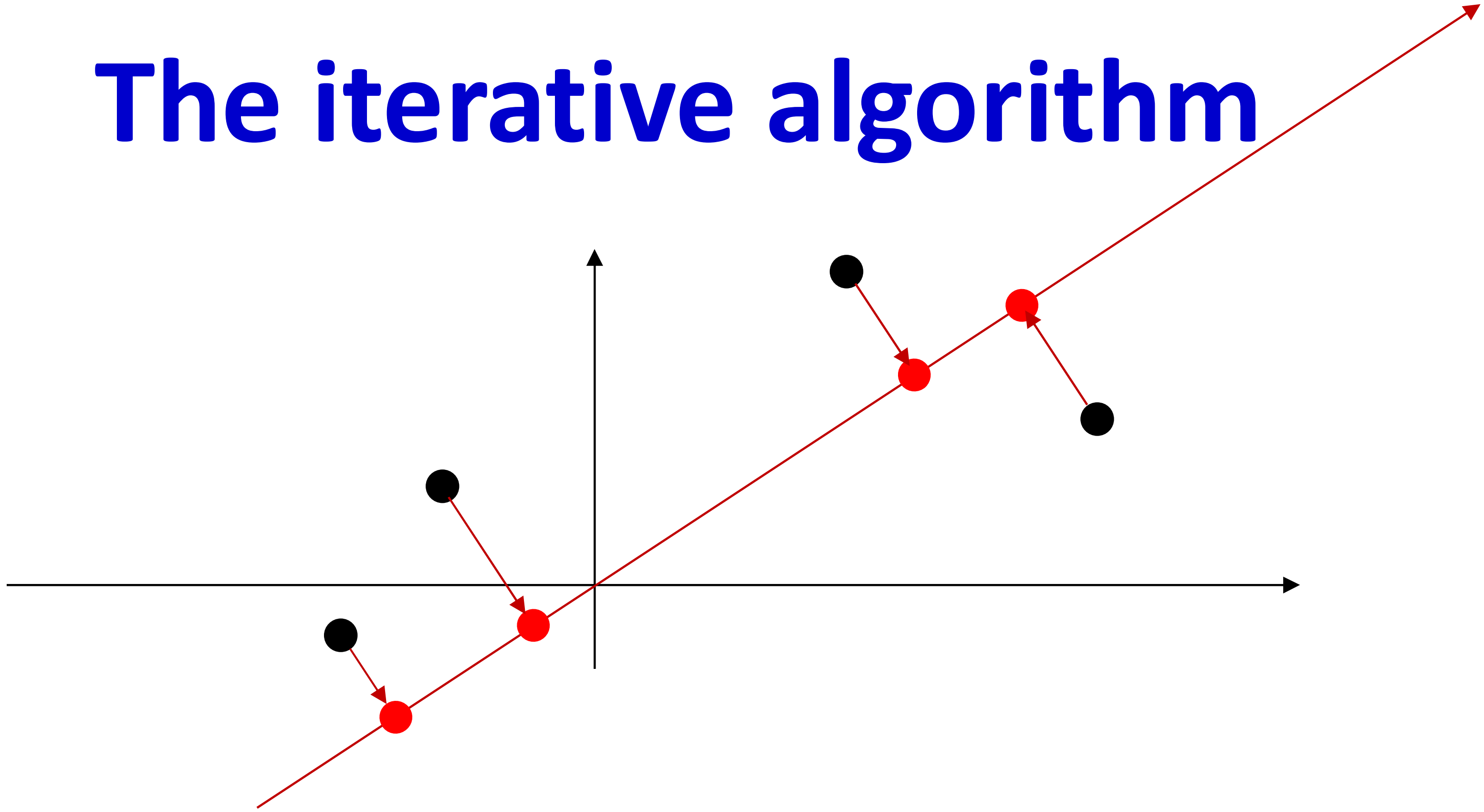
The iterative algorithm

This *jointly* minimizes the total squared length of lines from the points to their "attachments" on the plane



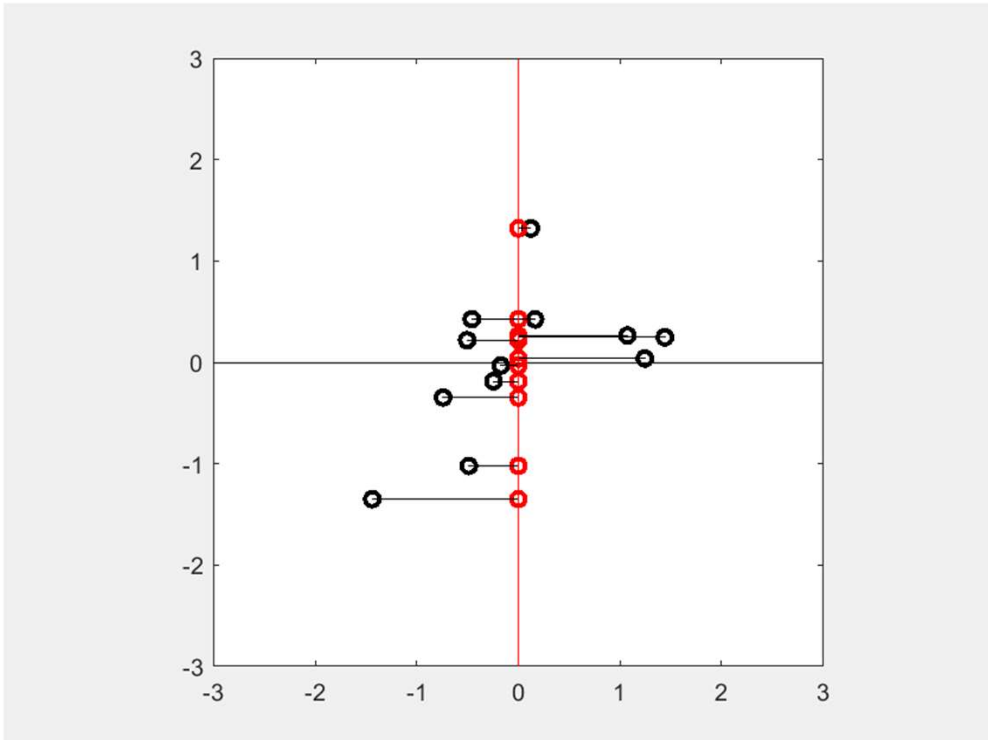
- Initialize a subspace (the basis w)
- Iterate until convergence:
  - Find the best position vectors Z on the W subspace for each training instance
    - Find the location on W that is *closest* to each instance, i.e. the perpendicular projection
  - Let W rotate and stretch/shrink, keeping the arrangement of Z locations fixed
    - Minimize the total square length of the lines attaching the projection on the place to the instance

The iterative algorithm



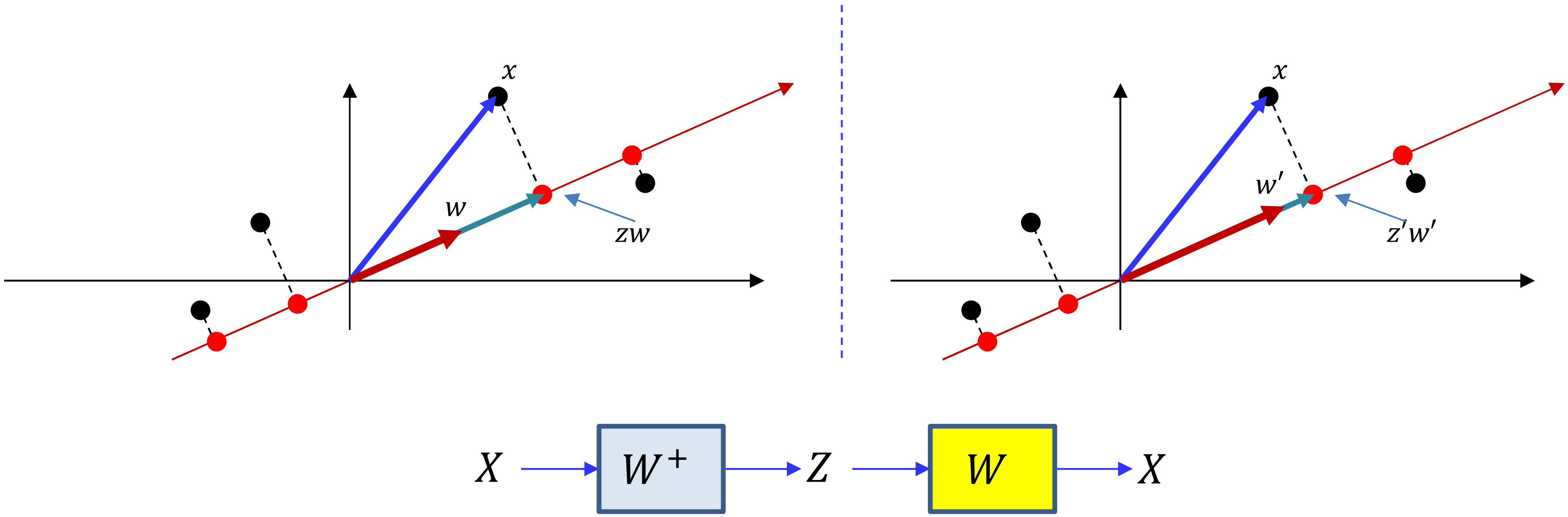
- Initialize a subspace (the basis w)
- Iterate until convergence:
  - Find the best position vectors Z on the W subspace for each training instance
    - Find the location on W that is *closest* to each instance, i.e. the perpendicular projection
  - Let W rotate and stretch/shrink, keeping the arrangement of Z locations fixed
    - Minimize the total square length of the lines attaching the projection on the place to the instance

A failed attempt at animation



- Someone with animated-gif generation skills, help me…

A minor issue: Scaling invariance



- The estimation is scale invariant
- We can increase the length of w, and compensate for it by reducing z
  - Can shrink the coordinate values by lengthening the bases and vice versa
- The solution is not unique!

Rotation/scaling invariance

$v = aw_1 + bw_2$

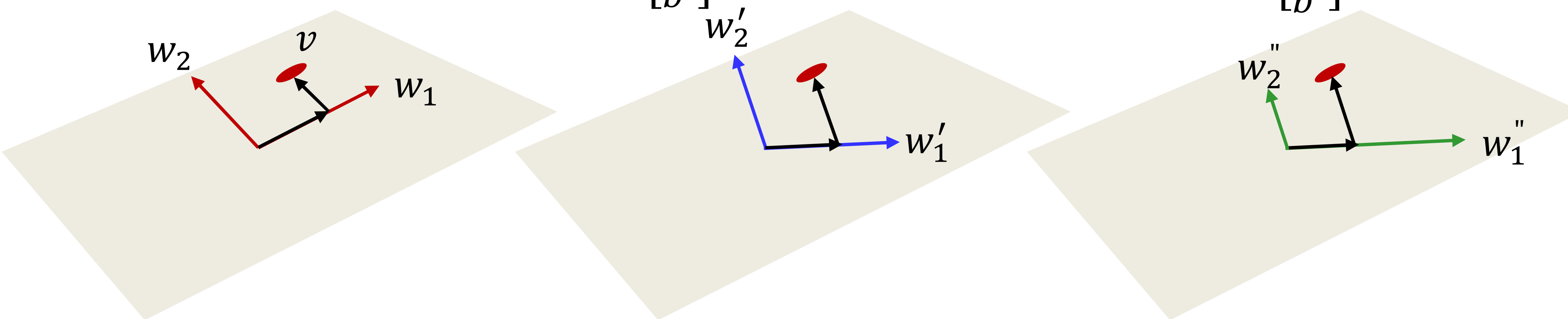
$z = \begin{bmatrix} a \\ b \end{bmatrix}$

$v = a'w_1' + b'w_2'$

$z = \begin{bmatrix} a' \\ b' \end{bmatrix}$

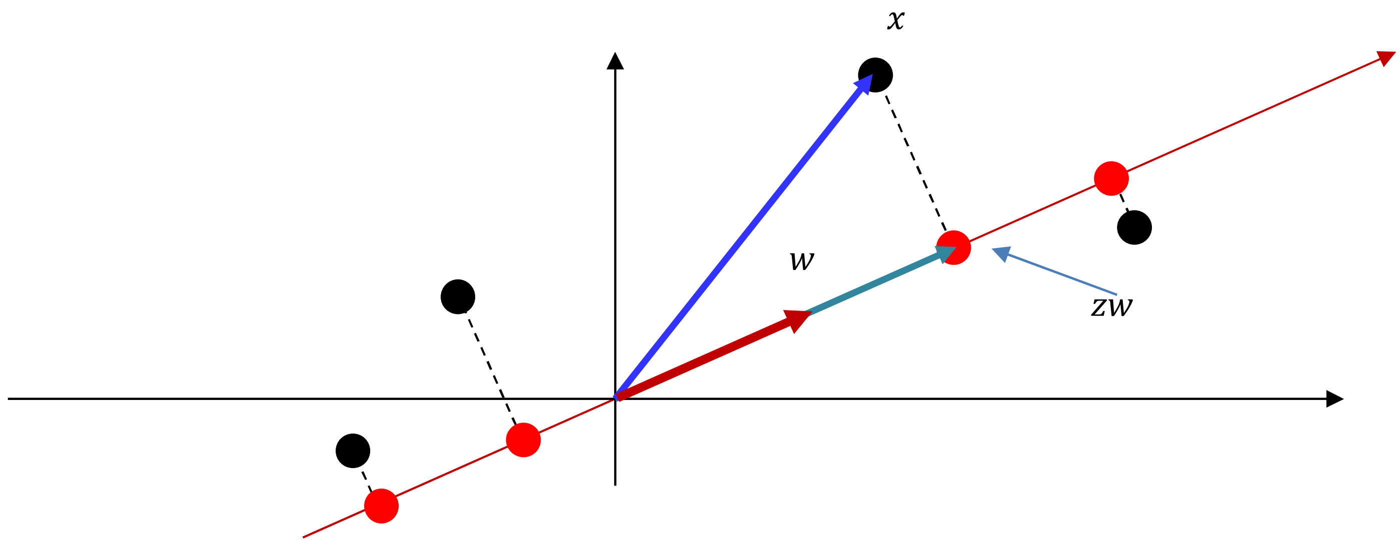
$v = a''w_1'' + b''w_2''$

$z = \begin{bmatrix} a'' \\ b'' \end{bmatrix}$



- We can rotate and scale the vectors in W without changing the actual subspace they compose
- The representation of any point in the hyperspace in terms of these vectors will also change
  - The zs in the two cases will be related through a linear transform
- The subspace is invariant to transformations of z

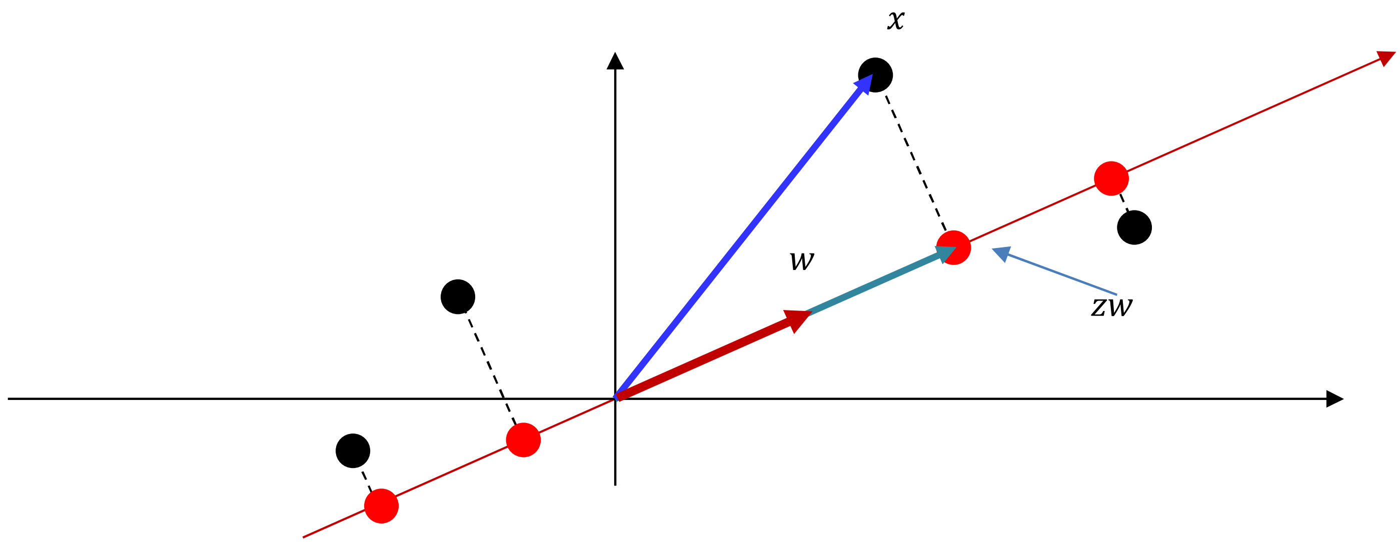
Resolving this issue



- A unique solution can be found by either
  - Requiring the vectors in W to be unit length and orthogonal
    - Standard "closed" form PCA
  - Constraining the variance of Z to be unity (or the identity matrix)

- While the Ws estimated with the two solutions will be different, the resulting discovered principal subspace will be the same

Resolving this issue

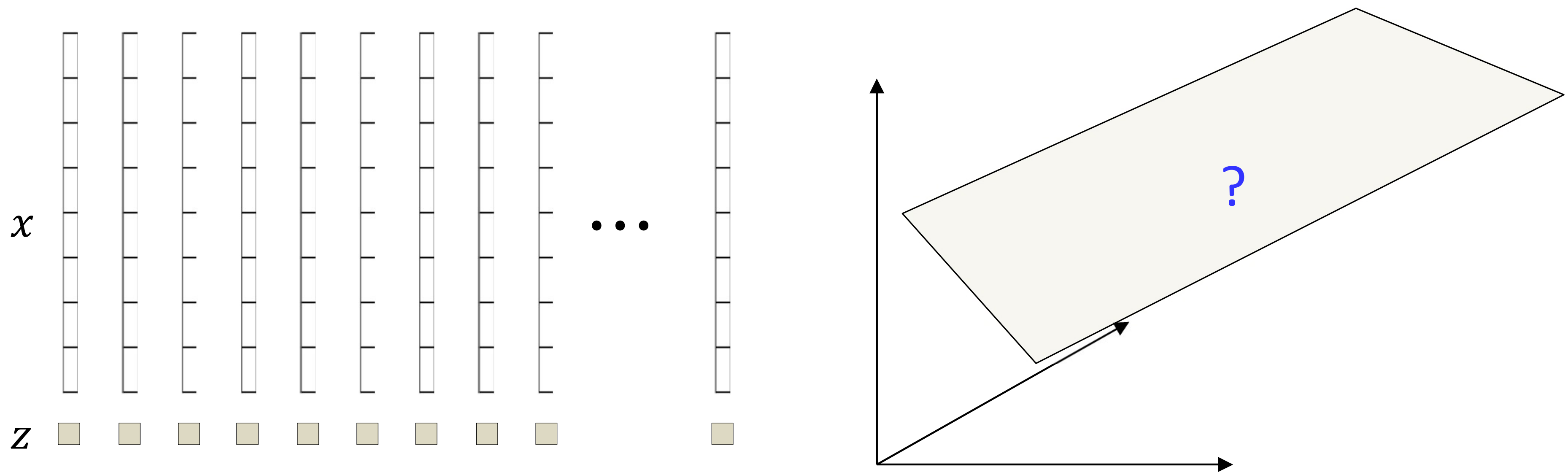


- A unique solution can be found by either
  - Requiring the vectors in W to be unit length and orthogonal
    - Standard “closed” form PCA
  - Constraining the variance of Z to be unity (or the identity matrix)

How?

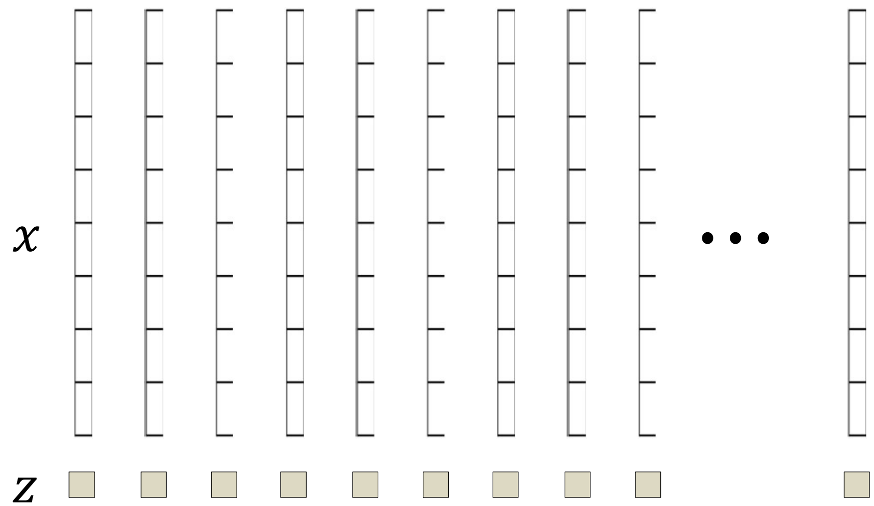
- While the Ws estimated with the two solutions will be different, the resulting discovered principal subspace will be the same

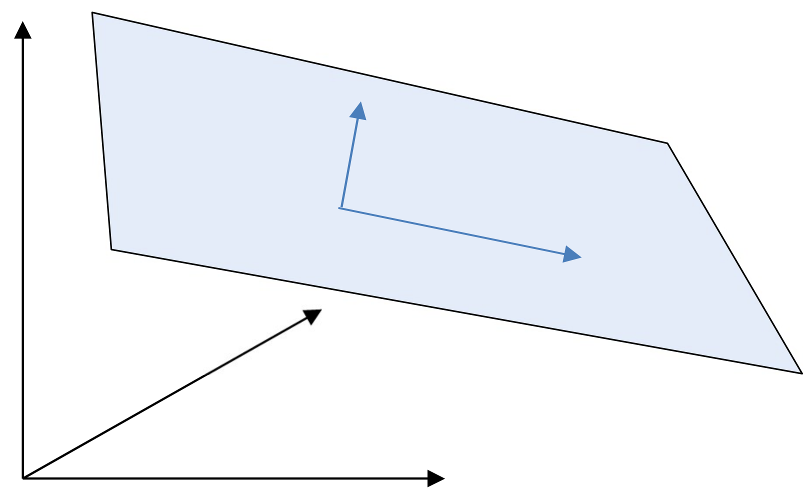
So what are we doing in the iterative solution?



- For every training vector x, we are missing the information z about where the vector lies on the principal subspace hyperplane
- If we had z, we could uniquely identify the plane

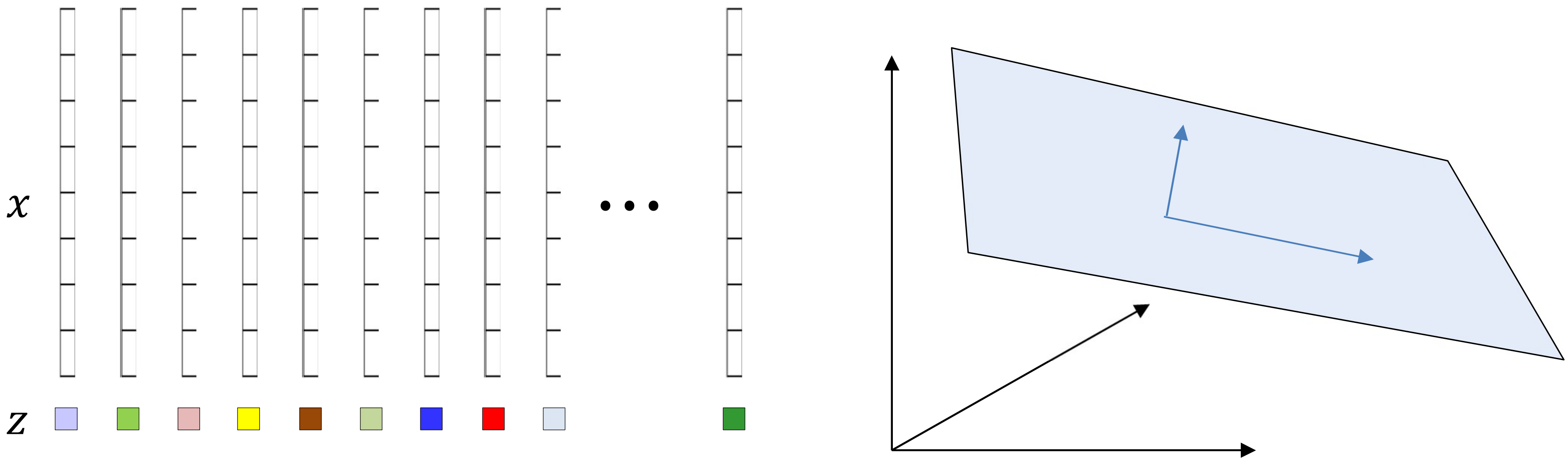
Iterative solution





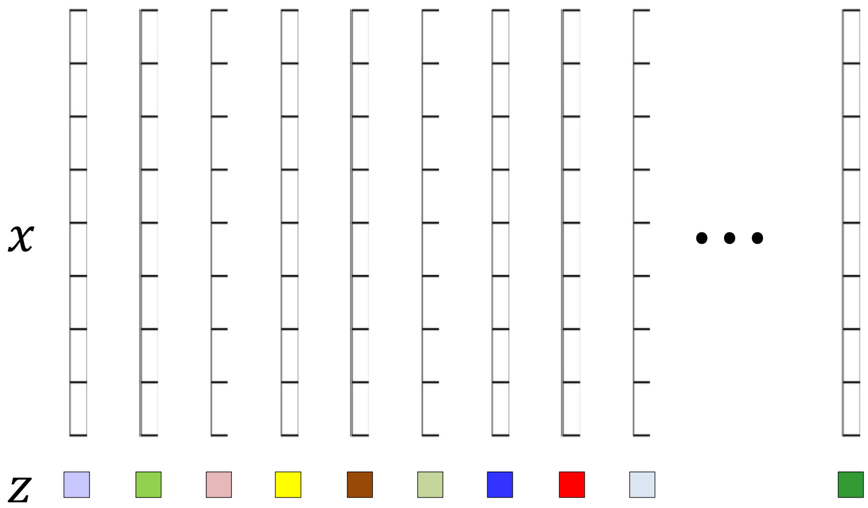
- Initialize the plane
  - Or rather, the bases for the plane

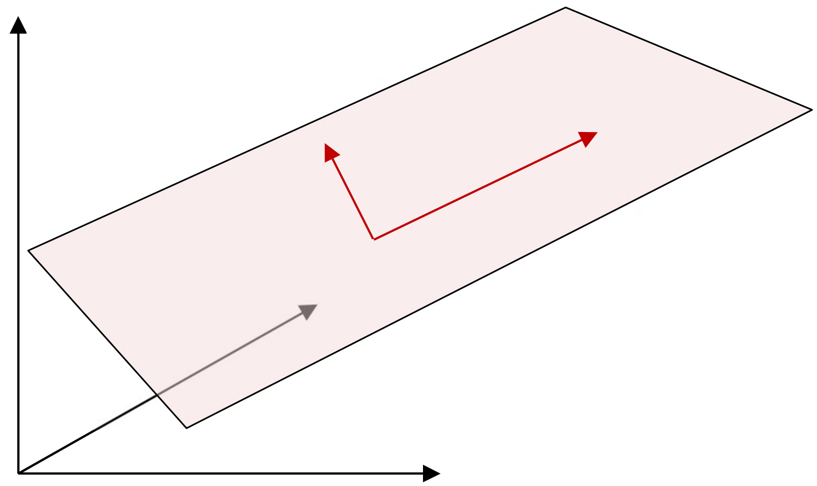
Iterative solution



- Initialize the plane
  - Or rather, the bases for the plane
- "Complete" the data by computing the appropriate zs for the plane

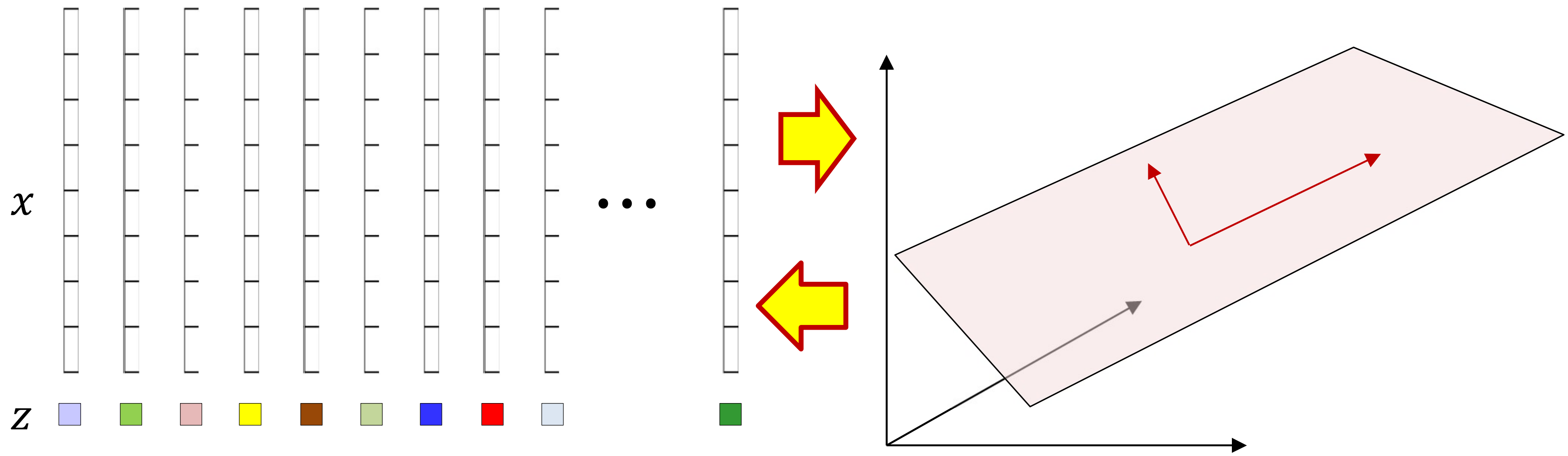
Iterative solution





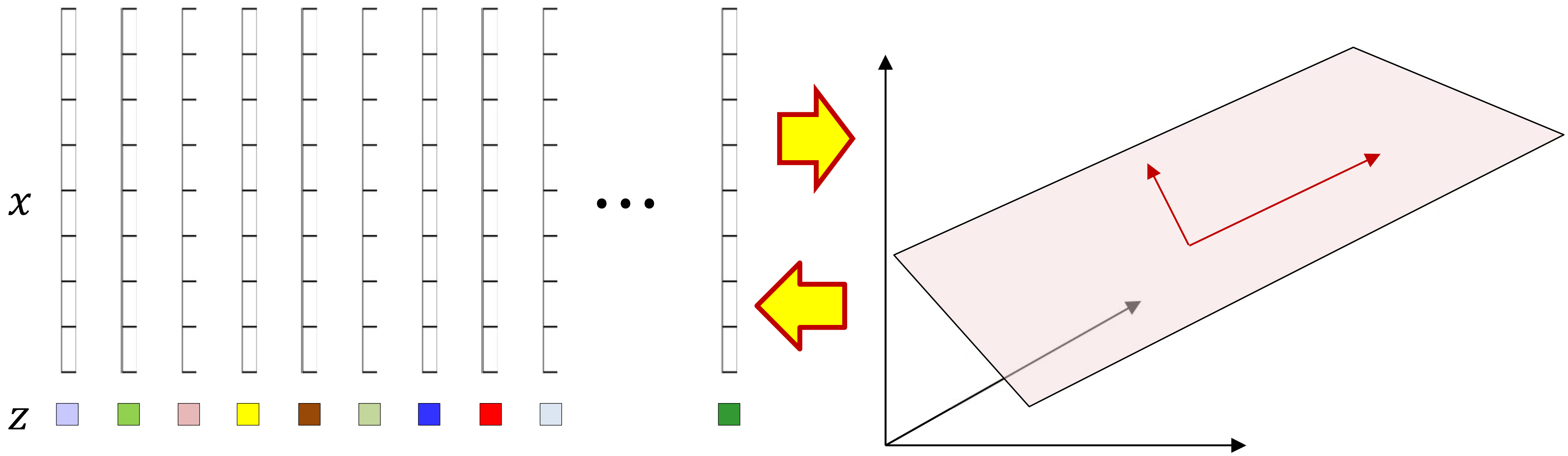
- Initialize the plane
  - Or rather, the bases for the plane
- “Complete” the data by computing the appropriate zs for the plane
- Reestimate the plane using the zs

Iterative solution



- Initialize the plane
  - Or rather, the bases for the plane
- "Complete" the data by computing the appropriate zs for the plane
- Reestimate the plane using the zs
- Iterate

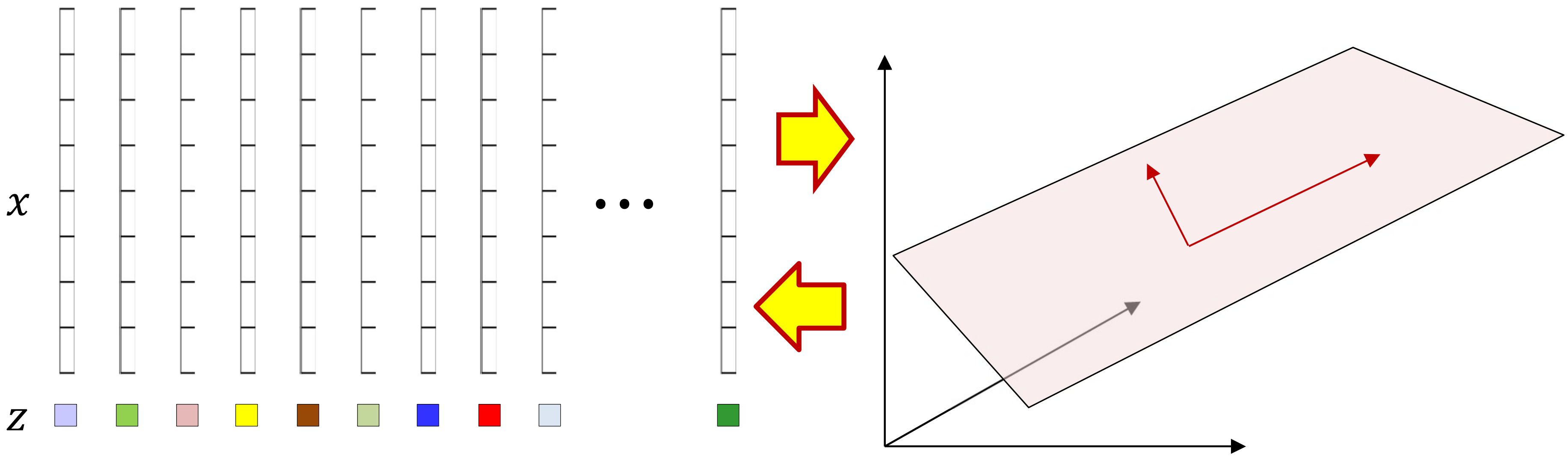
Iterative solution



- Initialize the plane
  - Or rather, the bases for the plan
- “Complete” the d
- Reesti
- Iterate

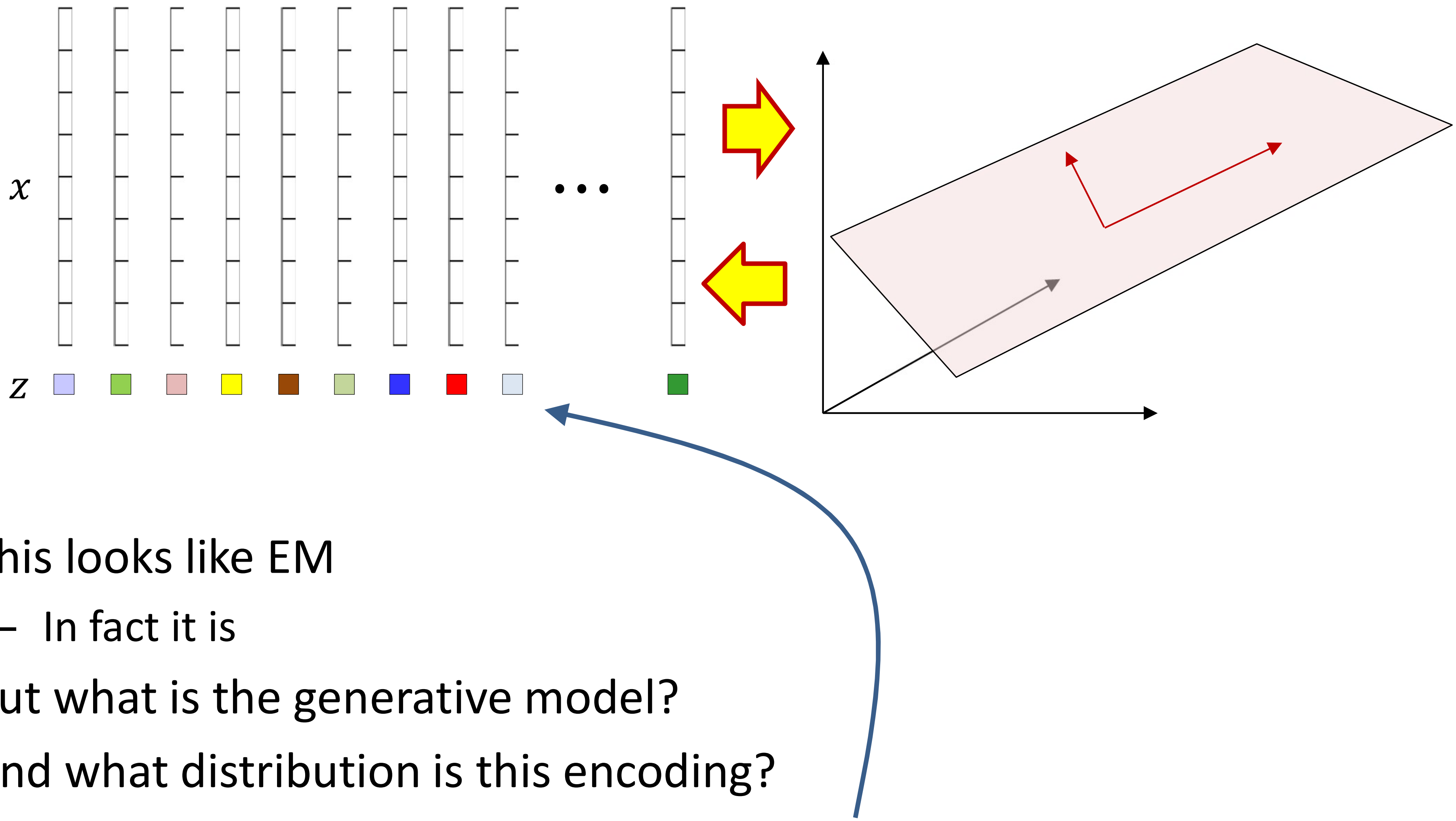
Look Familiar?

Iterative solution



- This looks like EM
  - In fact it is
- But what is the generative model?
- And what distribution is this encoding?

Iterative solution



- This looks like EM
  - In fact it is
- But what is the generative model?
- And what distribution is this encoding?
  - If we assume the zs have Gaussian distribution

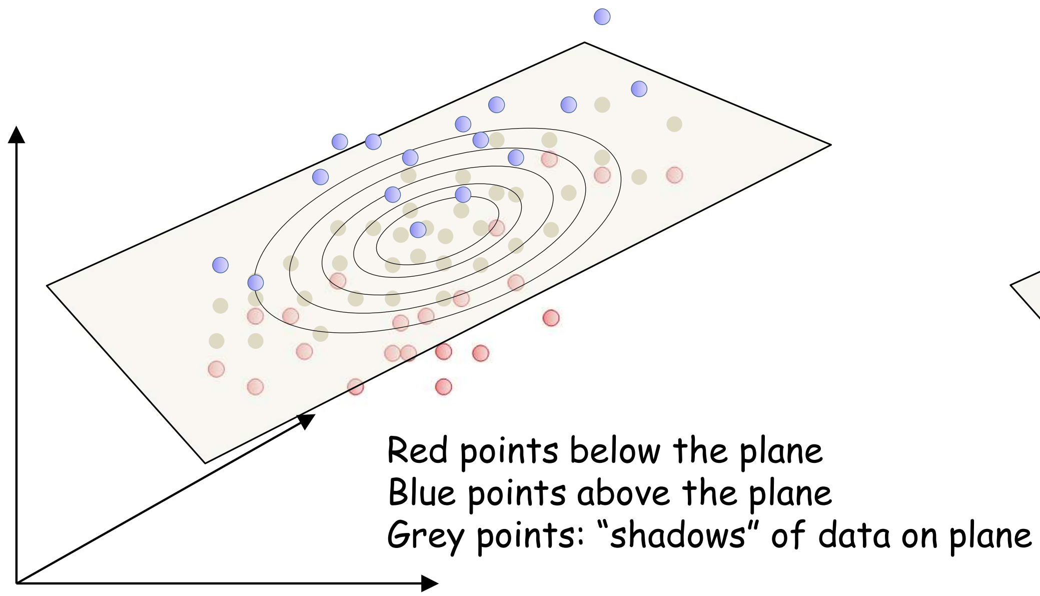
Poll 2

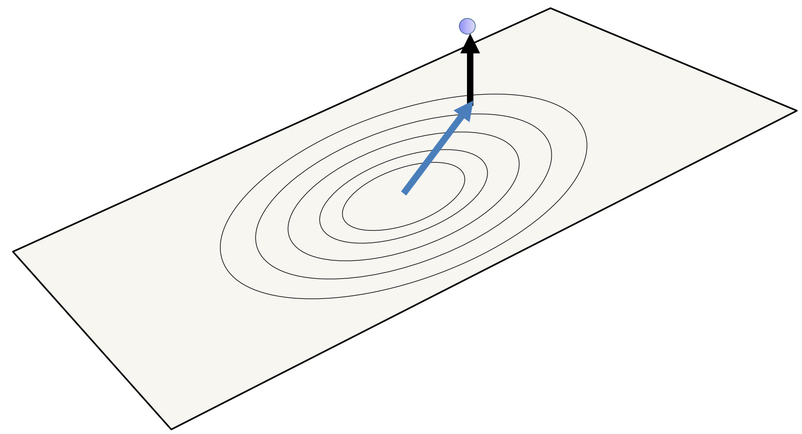
- Mark true statements
  - Generative models require a large amount of example/training data to learn properly
  - The amount of training data required is smaller if the Maximum-likelihood Estimator has closed form formulae
  - EM algorithm can be used to learn the parameters of generative model for which closed form Maximum Likelihood Estimators are not available
  - PCA can be implemented in an iterative way, by alternately estimating the principal component bases, and the coordinates of the data vectors in terms of these bases

Poll 2

- Mark true statements
  - Generative models require a large amount of example/training data to learn properly
  - The amount of training data required is smaller if the Maximum-likelihood Estimator has closed form formulae
  - **EM algorithm can be used to learn the parameters of generative model for which closed form Maximum Likelihood Estimators are not available**
  - **PCA can be implemented in an iterative way, by alternately estimating the principal component bases, and the coordinates of the data vectors in terms of these bases**

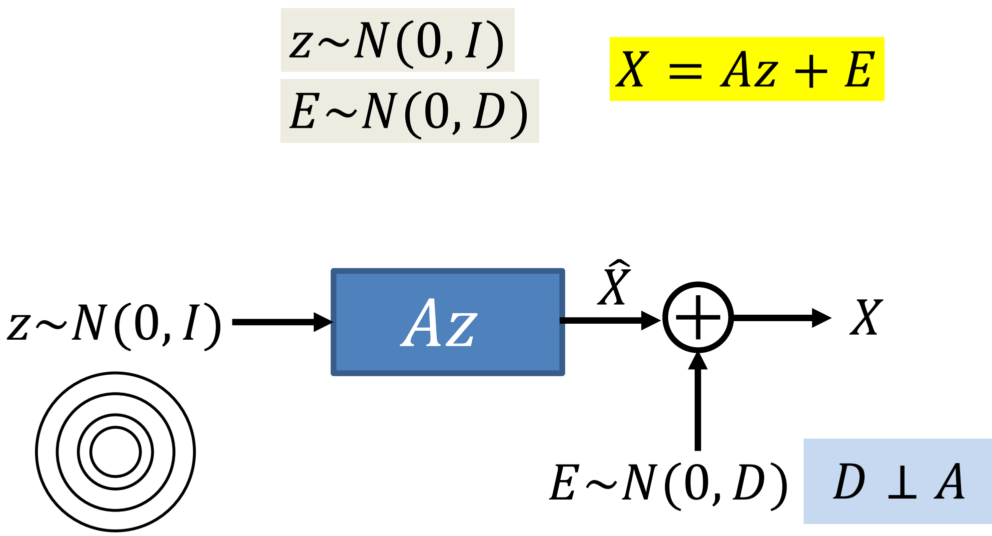
The *generative* story behind PCA

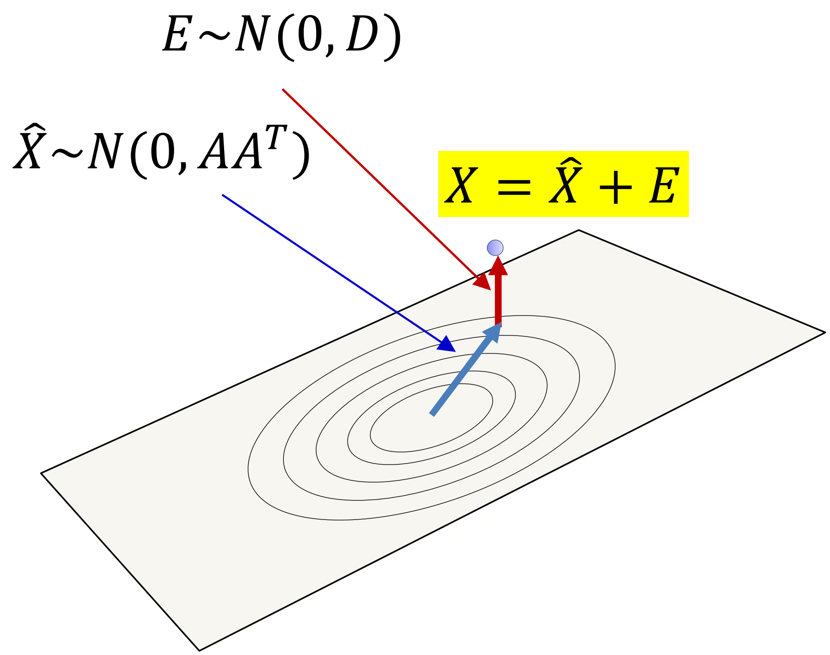




- PCA actually has a generative story
- In order to generate any point
  - We first take a Gaussian step on the principal plane
  - Then we take an orthogonal *Gaussian* step from where we land to generate a point
  - PCA finds the plane and the characteristics of the Gaussian steps from the data

The *generative* story behind PCA





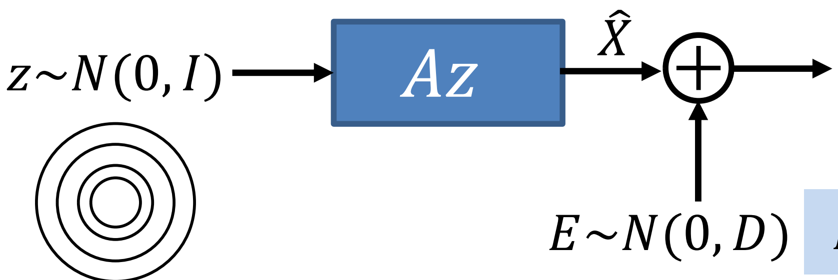
- **Generative story for PCA:**
  - z is drawn from a K-dim isotropic Gaussian
    - K is the dimensionality of the principal subspace
  - A is "basis" matrix
    - Matrix of principal Eigen vectors scaled by Eigen values
  - E is a 0-mean Gaussian noise that is orthogonal to the principal subspace
    - **The covariance of the Gaussian is low-rank and orthogonal to the principal subspace!**

The *generative* story behind PCA

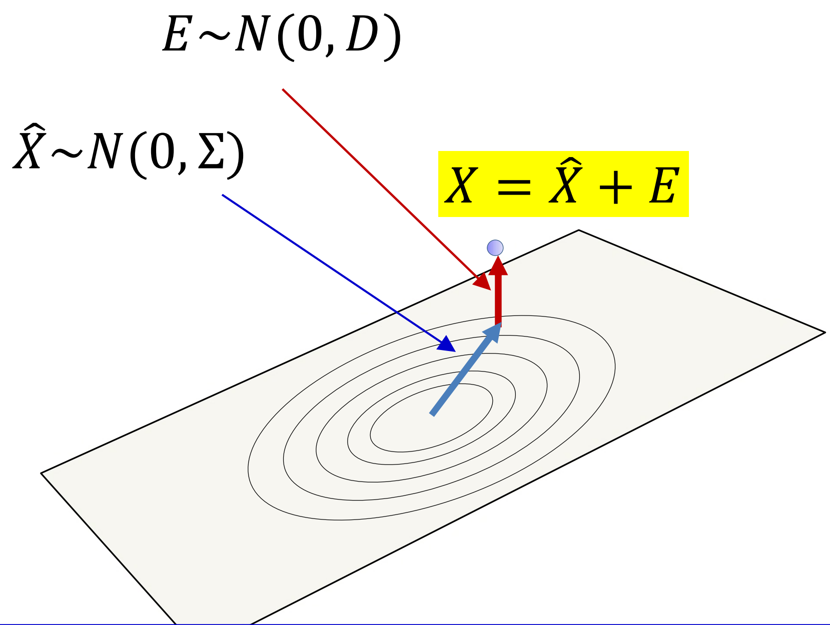
$z \sim N(0, I)$

$E \sim N(0, D)$

$X = Az + E$



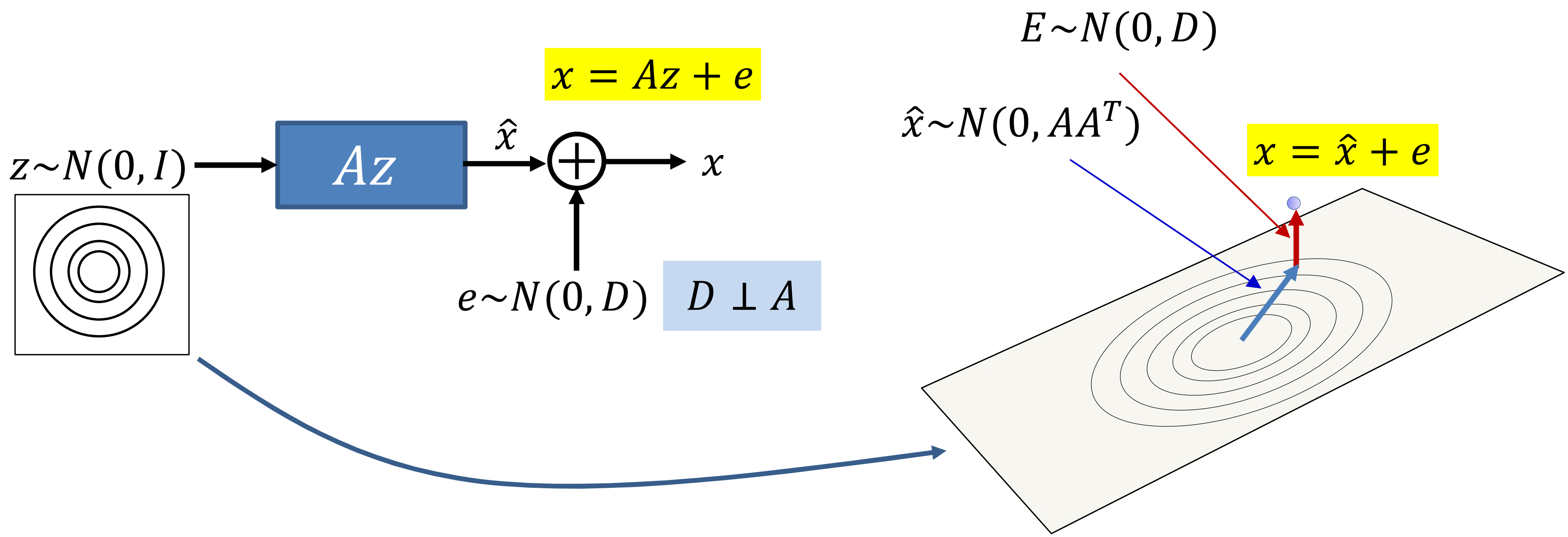
$D \perp A$



PCA implicitly obtains maximum likelihood estimate of A and D, from training data X

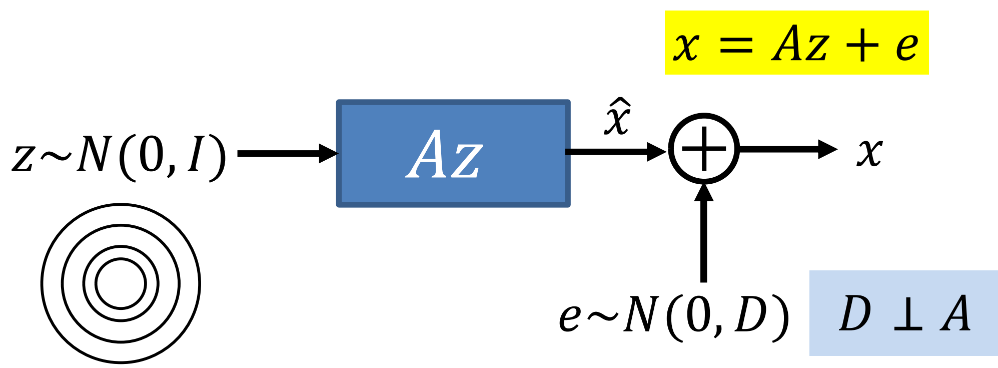
- **Generative story for PCA:**
  - z is drawn from a K-dim isotropic Gaussian
    - K is the dimensionality of the principal subspace
  - A is "basis" matrix
    - Matrix of principal Eigen vectors scaled by Eigen values
  - E is a 0-mean Gaussian noise that is orthogonal to the principal subspace
    - **The covariance of the Gaussian is low-rank and orthogonal to the principal subspace!**

Recap: The *generative* story behind PCA



- Alternate view: Az stretches and rotates the K-dimensional planar space of z into a K-dimensional planar subspace (manifold) of the data space
- The circular distribution of z in the K-dimensional z space transforms into an ellipsoidal distribution on a K-dimensional hyperplane the data space
- Samples are drawn from the ellipsoidal distribution on the hyperplane, and noise is added to them

The probability modelled by PCA



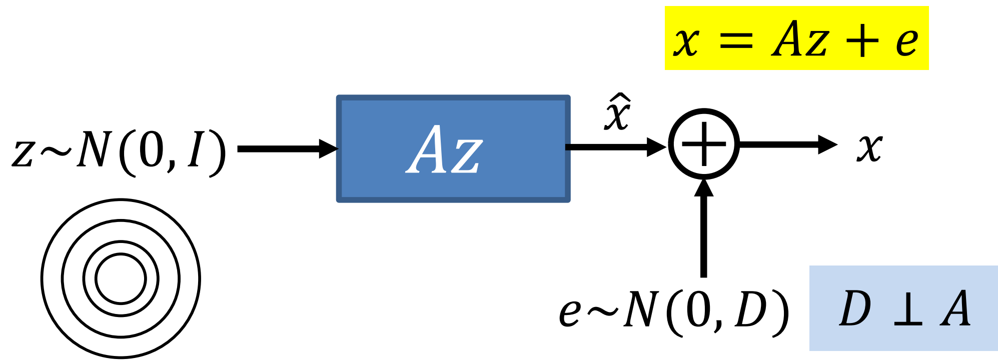
- **PCA models a Gaussian distribution:**

$$\hat{x} = Az \Rightarrow \qquad P(\hat{x}) = N(0, AA^T)$$
$$x = \hat{x} + E \Rightarrow \qquad P(x) = N(0, AA^T + D)$$

- The probability density of x is Gaussian lying mostly close to a hyperplane
  - With correlated structure on the plane
  - And uncorrelated components orthogonal to the plane
- Also

$$P(x|z) = N(Az, D)$$

The probability modelled by PCA



- **PCA models a Gaussian distribution:**

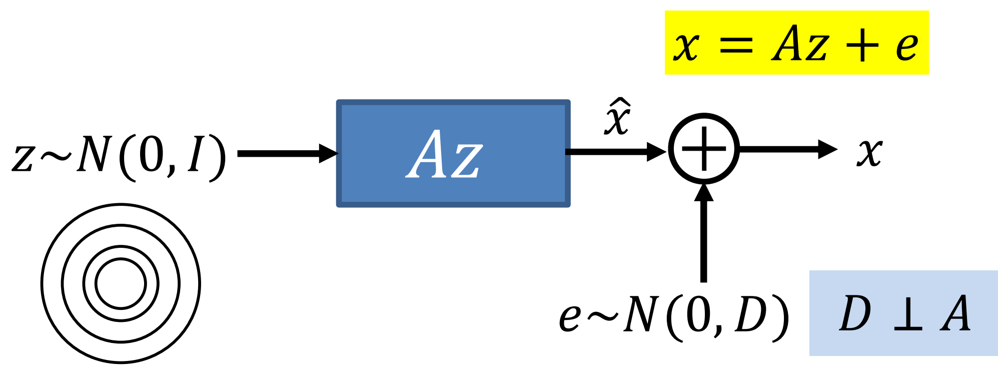
$$\hat{x} = Az \Rightarrow \qquad P(\hat{x}) = N(0, AA^T)$$

$$x = \hat{x} + E \Rightarrow \qquad P(x) = N(0, AA^T + D)$$

- The probability density of x is Gaussian lying mostly close to a hyperplane
  - With correlated structure on the plane
  - And uncorrelated components orthogonal to the plane
- Also

$$P(x|z) = N(Az, D)$$

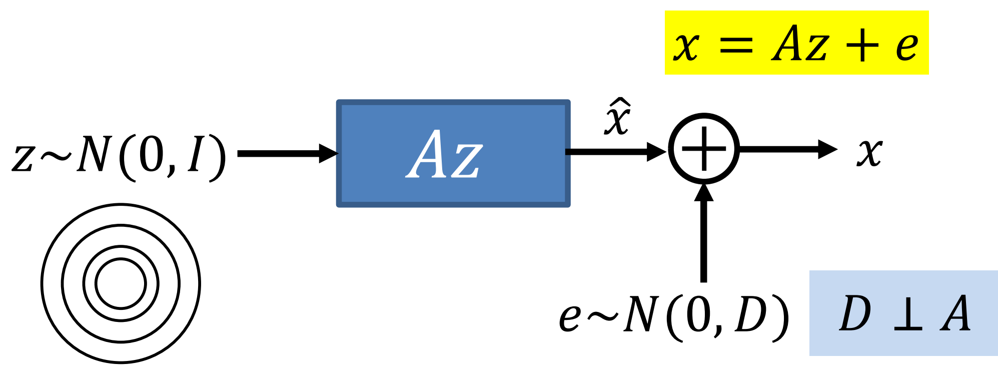
The probability modelled by PCA



$$P(x|z) = N(Az, D)$$

- How?

ML estimation of PCA parameters



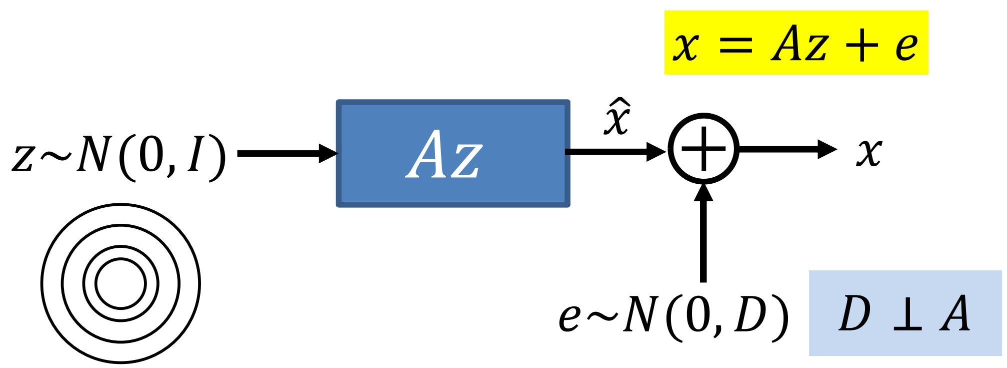
$$P(x) = N(0, AA^T + D)$$

- The parameters of the PCA generative model are A and D
- The ML estimator is

$$\underset{A,D}{\operatorname{argmax}} \sum_x \log \frac{1}{\sqrt{(2\pi)^d |AA^T + D|}} \exp\left(-0.5 x^T (AA^T + D)^{-1} x\right)$$

  - Where d is the dimensionality of the space
- Combined with the constraints on the number of columns in A (dimensions of principal subspace), and that $A^T D = 0$, this will give us the principal subspace

Missing information for PCA



- There is missing information about the observation X
  - Information about intermediate values drawn in generating X
  - We don't know z
- If we knew z for each X, estimating A (and D) would be simple

PCA with complete information

$$x = Az + E$$
$$P(x|z) = N(Az, D)$$

- Given complete information $(x_1, z_1), (x_2, z_2), \dots$
  - Representing $X = [x_1, x_2, \dots], \;\; Z = [z_1, z_2, \dots]$

$$\underset{A,D}{\operatorname{argmax}} \sum_{(x,z)} \log P(x, z) \; = \underset{A,D}{\operatorname{argmax}} \sum_{(x,z)} \log P(x|z)$$

$$= \underset{A,D}{\operatorname{argmax}} \sum_{(x,z)} \log \frac{1}{\sqrt{(2\pi)^d |D|}} \exp\left(-0.5(x - Az)^T D^{-1} (x - Az)\right)$$

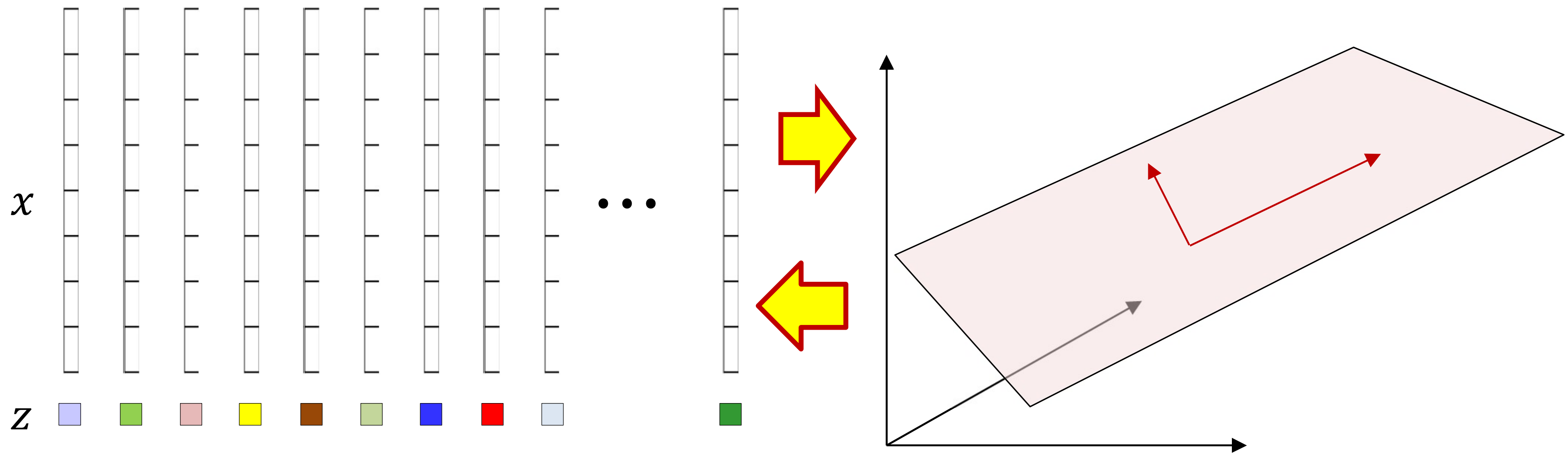
- Differentiating w.r.t A and equating to 0, we get the easy solution

$$A = XZ^+$$

  - (Some sloppy math (D is not invertible), but the solution is right)

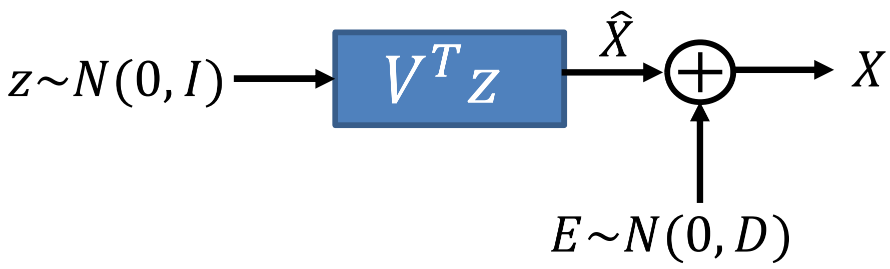
But we don't have z. It is missing

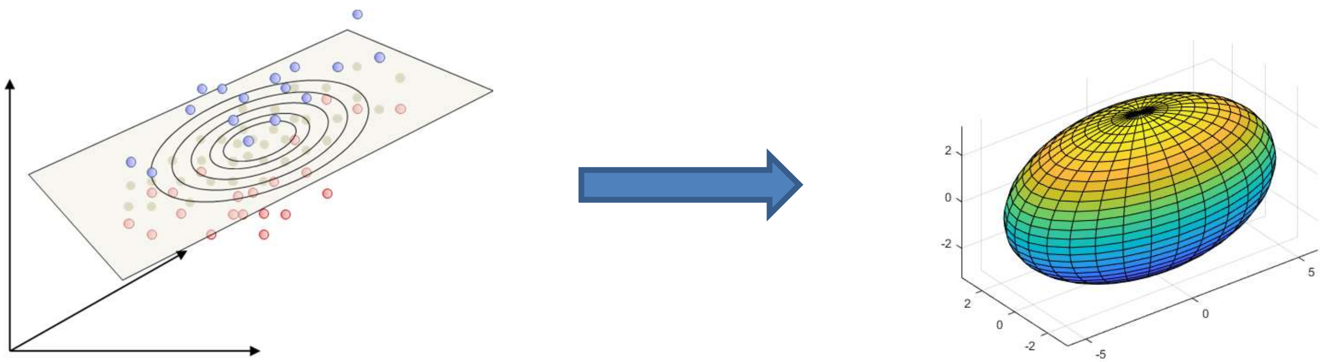
EM for PCA



- Initialize the plane
  - Or rather, the bases for the plane
- "Complete" the data by computing the appropriate zs for the plane
  - $P(z|X;\ A)$ is a delta, because E is orthogonal to A
- Reestimate the plane using the zs
- Iterate

The distribution modelled by PCA





- If z is Gaussian, $\hat{X}$ is Gaussian
- $\hat{X}$ and E are Gaussian => X is Gaussian
- PCA model: The observed data are Gaussian
  - Gaussian data lying very close to a principal subspace
  - Comprising "clean" Gaussian data on the subspace plus orthogonal noise

Poll 3

- PCA implicitly obtains maximum likelihood estimate of the transformation (the direction of the new bases) and the covariance of noise that embed its anisotropy.
  - True
  - False

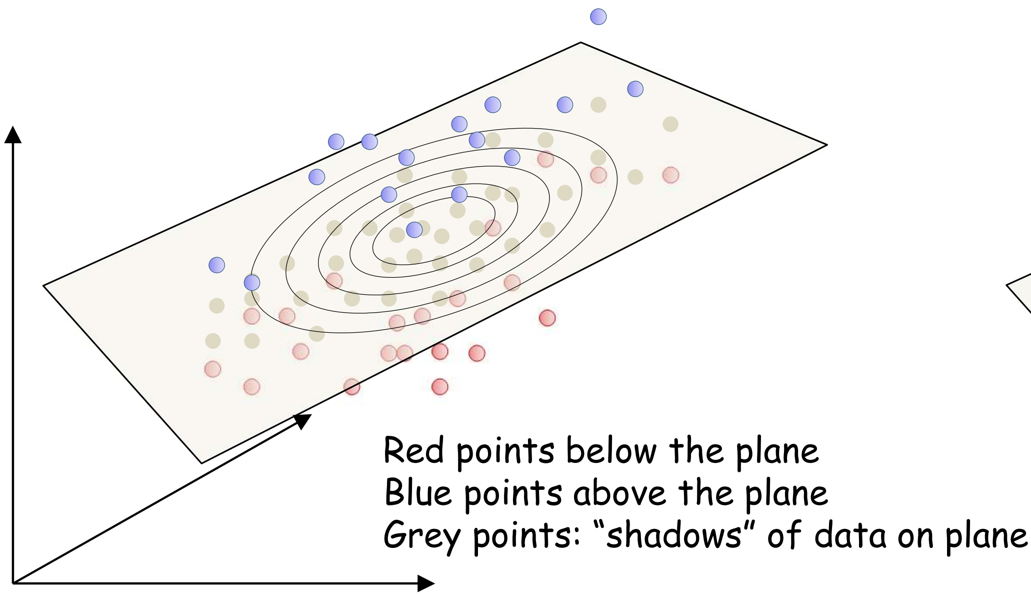
- When the observed data are (nearly) Gaussian, it can be decomposed to Gaussian data lying very close to a principal subspace plus parallel noise lying in the same plane
  - True
  - False

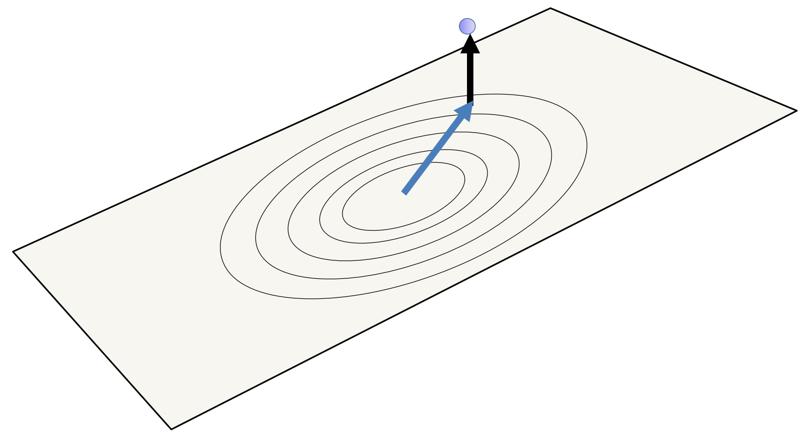
Poll 3

- PCA implicitly obtains maximum likelihood estimate of the transformation (the direction of the new bases) and the covariance of noise that embed its anisotropy.
  - **True**
  - False

- When the observed data are (nearly) Gaussian, it can be decomposed to Gaussian data lying very close to a principal subspace plus parallel noise lying in the same plane
  - **True**
  - False

Can we do better?





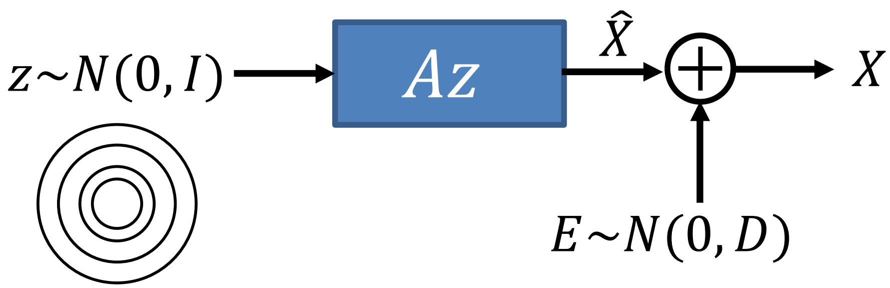
- PCA assumes the noise is always orthogonal to the data
  - Not always true
  - Noise in images can look like images, random noise can sound like speech, etc.
- Let’s us generalize the model to permit non-orthogonal noise

The Linear Gaussian Model

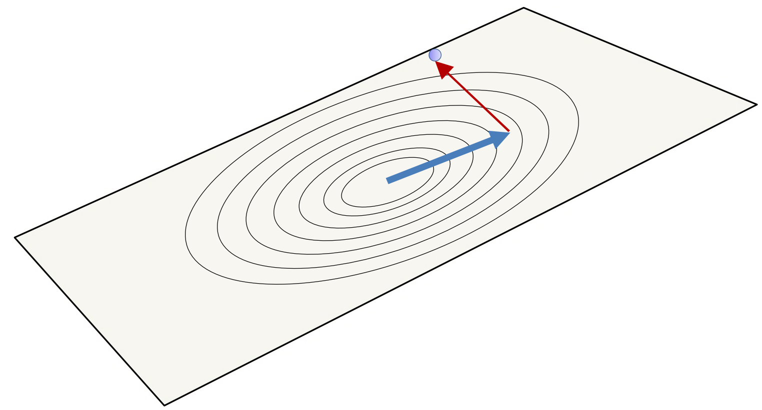
$z \sim N(0, I)$

$E \sim N(0, D)$

$X = Az + E$



D is full rank



- Update the model: The noise added to the output of the encoder can lie in *any* direction
  - Uncorrelated, but not just orthogonal to the principal subspace

- Generative model: to generate any point
  - Take a Gaussian step on the hyperplane
  - Add *full-rank* Gaussian uncorrelated noise that is independent of the position on the hyperplane
    - Uncorrelated: diagonal covariance matrix
    - Direction of noise is unconstrained
      - Need not be orthogonal to the plane

The linear Gaussian model

$z \sim N(0, I)$

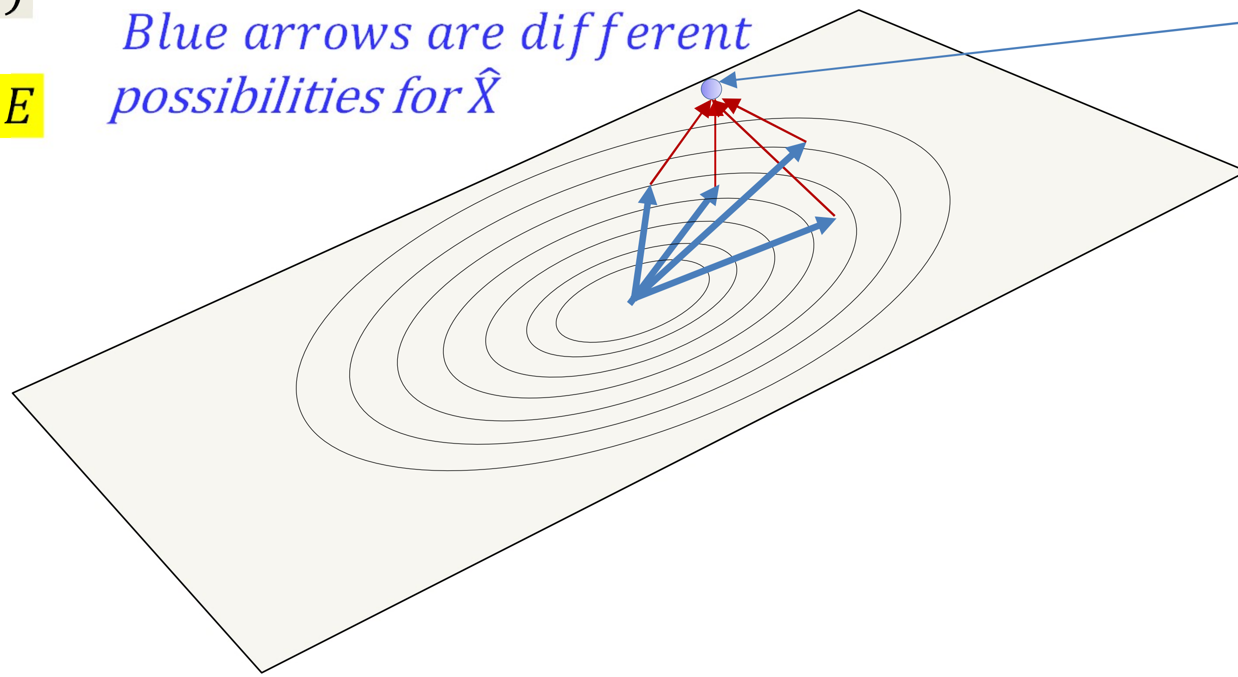
$E \sim N(0, D)$

$X = Az + E$

Red arrows are different possibities for E

Blue arrows are different possibilities for $\hat{X}$

$X = \hat{X} + E$



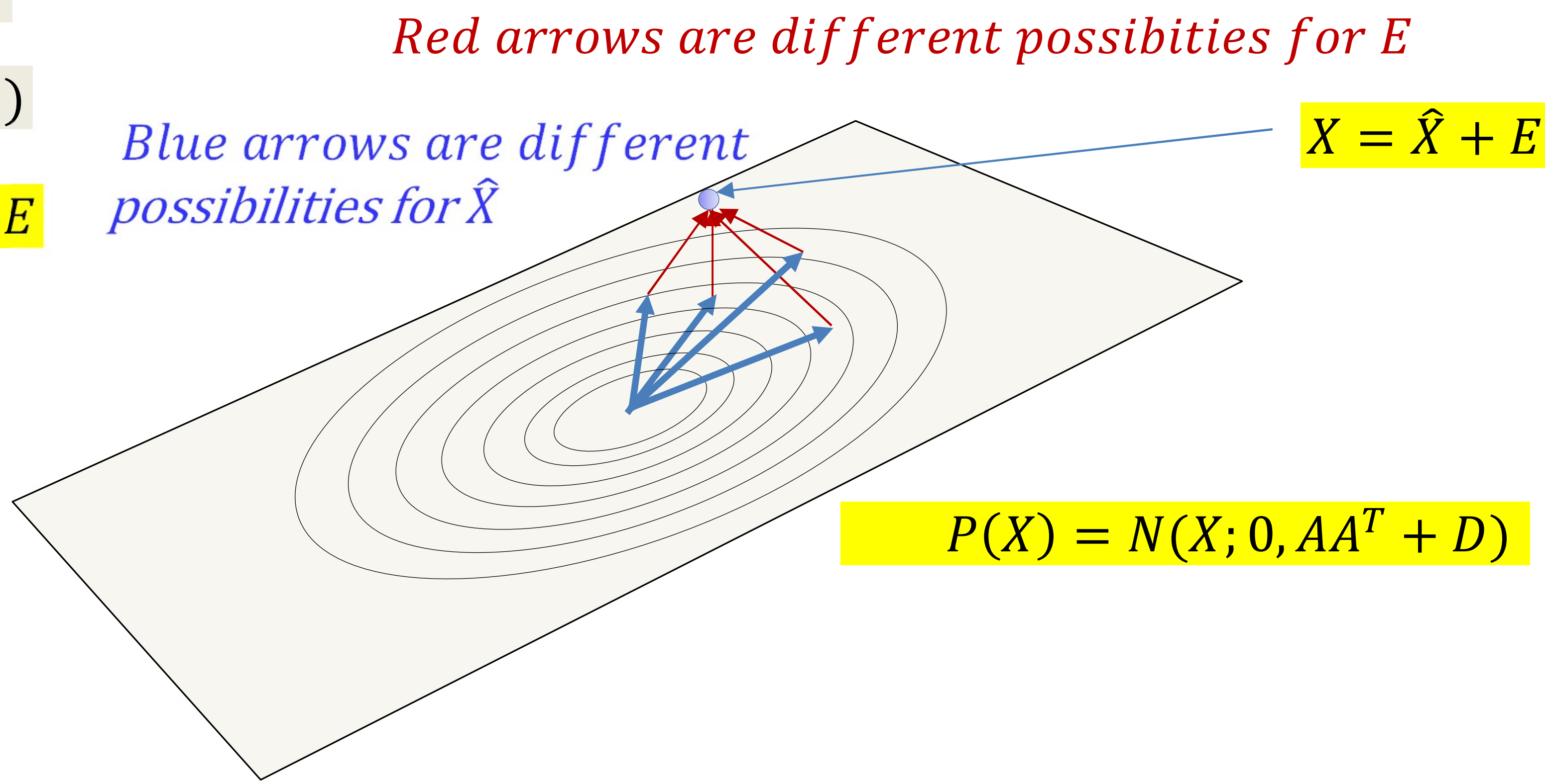
- The way to produce any data instance is no longer unique
  - though different corrections may have different probabilities

Revisiting the linear Gaussian model

$z \sim N(0, I)$

$E \sim N(0, D)$

$X = Az + E$



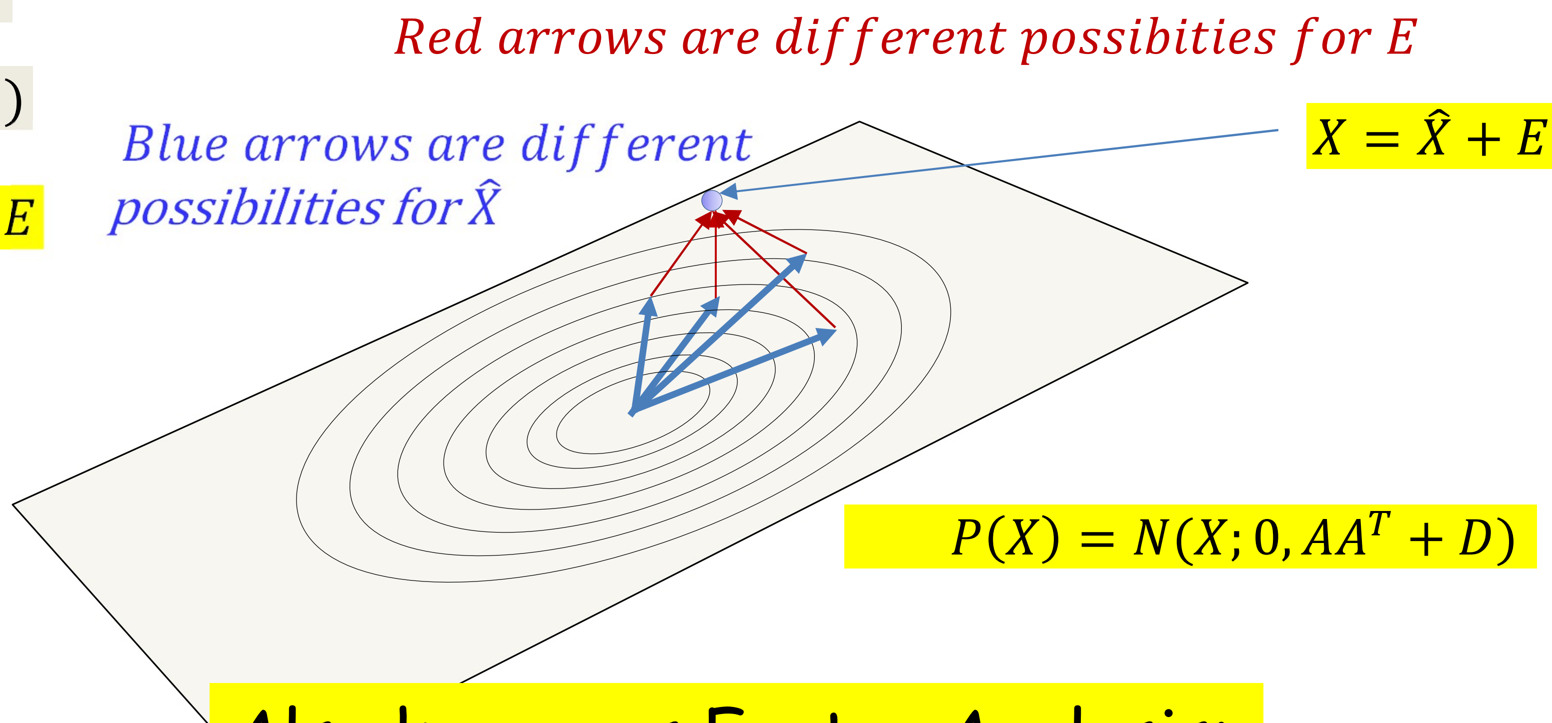
- The way to produce any data instance is no longer unique
  - though different corrections may have different probabilities
- This is still a parametric model for a Gaussian distribution
  - Parameters are A and D (assuming 0 mean)

Revisiting the linear Gaussian model

$z \sim N(0, I)$

$E \sim N(0, D)$

$X = Az + E$

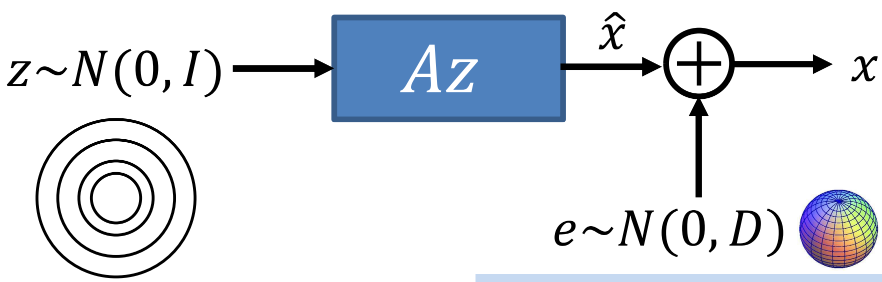


Also known as Factor Analysis:
A is the loading matrix
z are the factors
D is diagonal

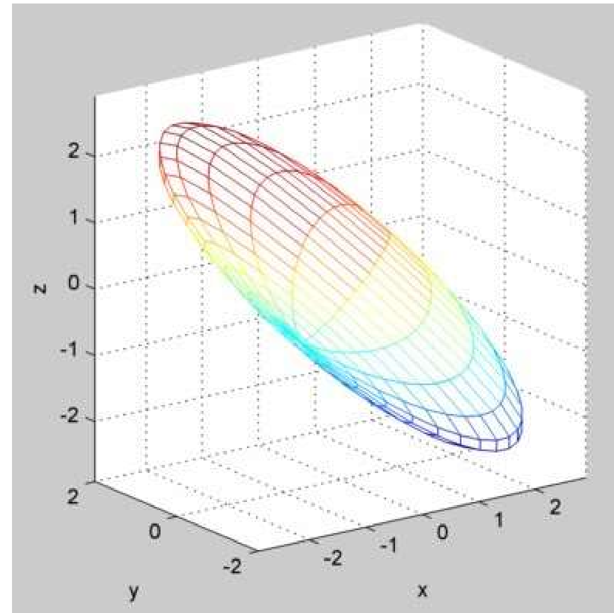
- The way to p … er unique
  - though diff … babilities
- This is in fact … distribution
  - Parameters are A and D (assuming 0 mean)

The probability distribution modelled by the LGM

$x = Az + e$



D is full rank diagonal



- The noise added to the output of the encoder can lie in *any* direction

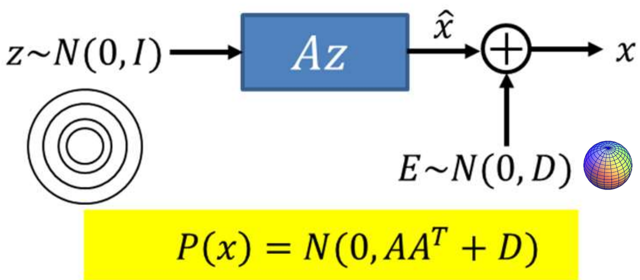
$$\hat{x} = Az \Rightarrow \qquad P(\hat{x}) = N(0, AA^T)$$

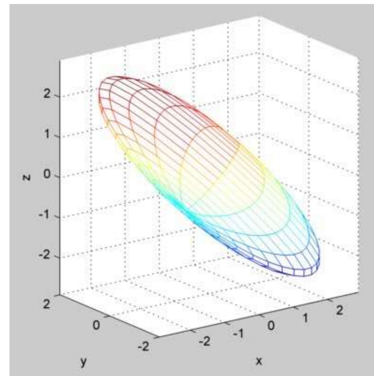
$$x = \hat{x} + E \Rightarrow \qquad P(x) = N(0, AA^T + D)$$

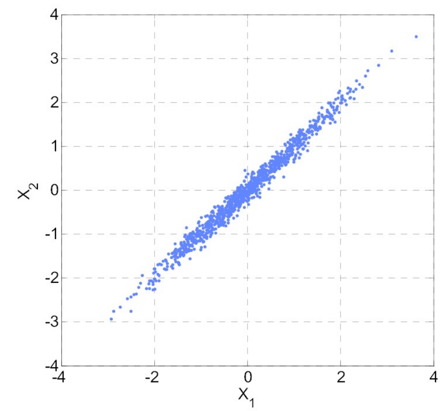
- The probability density of x is Gaussian lying mostly close to a hyperplane
  - With uncorrelated Gaussian
- Also

$$P(x|z) = N(Az, D)$$

The linear Gaussian model



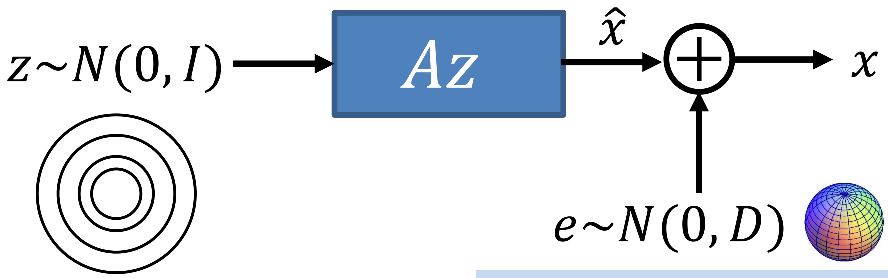




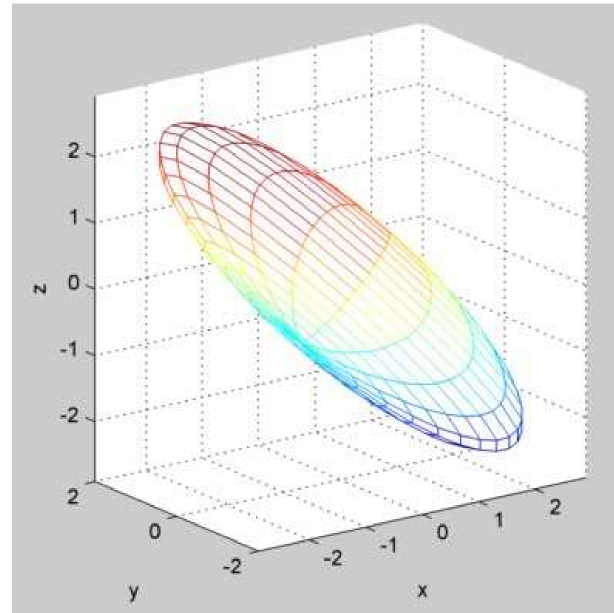
- Is a generative model for Gaussians
- Data distribution are Gaussian lying largely on a hyperplane with some Gaussian “fuzz”
  - Only components on the plane are correlated with one another
    - No correlations off the plane
  - Which allows us to model *some* correlations between components
    - Halfway between a Gaussian with a diagonal covariance, and one with a full covariance

ML estimation of LGM parameters

$$x = Az + e$$



D is full rank diagonal



$$P(x) = N(0, AA^T + D)$$

- The parameters of the LGM generative model are A and D
- The ML estimator is

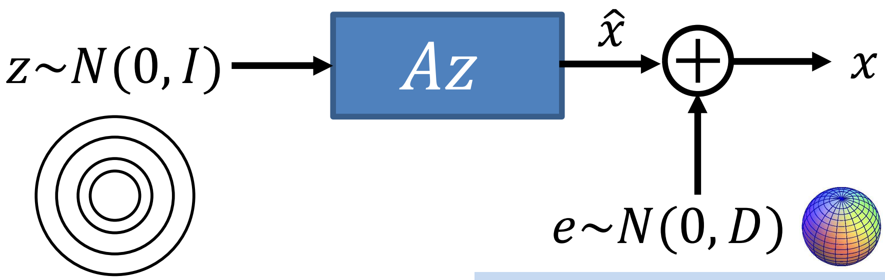
$$\underset{A,D}{\operatorname{argmax}} \sum_x \log \frac{1}{\sqrt{(2\pi)^d |AA^T + D|}} \exp\left(-0.5 x^T (AA^T + D)^{-1} x\right)$$

  - Where d is the dimensionality of the space

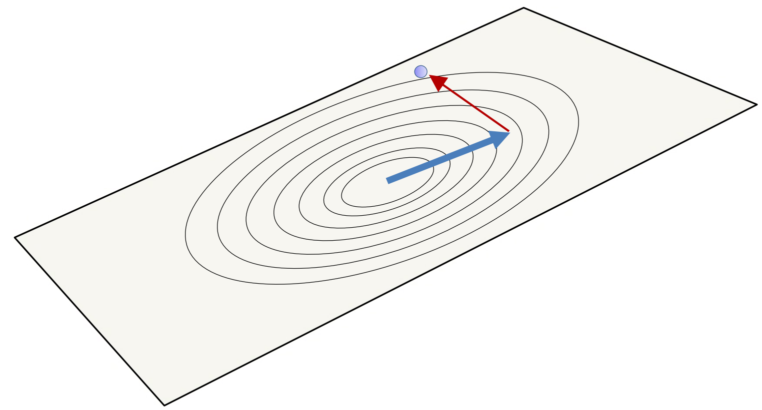
- As it turns out, this does *not* have a nice closed form solution
  - Because D is full rank

Missing information for LGMs

$x = Az + e$



D is full rank diagonal



- There is missing information about the observation X
  - Information about intermediate values drawn in generating X
  - We don't know z
- If we knew the z for each X, estimating A (and D) would be very simple

LGM with complete information

$$x = Az + e$$
$$P(x|z) = N(Az, D)$$

- Given complete information $X = [x_1, x_2, \dots],\ \ Z = [z_1, z_2, \dots]$

$$\underset{A,D}{\operatorname{argmax}} \sum_{(x,z)} \log P(x,z) = \underset{A,D}{\operatorname{argmax}} \sum_{(x,z)} \log P(x|z)$$

$$= \underset{A,D}{\operatorname{argmax}} \sum_{(x,z)} \log \frac{1}{\sqrt{(2\pi)^d |D|}} \exp\left(-0.5(x - Az)^T D^{-1}(x - Az)\right)$$

$$= \underset{A,D}{\operatorname{argmax}} \sum_{(x,z)} -\frac{1}{2}\log|D| - 0.5(x - Az)^T D^{-1}(x - Az)$$

- Differentiating w.r.t A and D equating to 0, we get an easy solution

LGM with complete information

$$\underset{A,D}{\operatorname{argmax}} \sum_{(x,z)} -\frac{1}{2}\log|D| - 0.5(x - Az)^T D^{-1}(x - Az)$$

- Differentiating w.r.t A and $D$and equating to 0, we get an easy solution
- Solution for A

$$\nabla_A \sum_{(x,z)} 0.5(x - Az)^T D^{-1}(x - Az) = 0 \quad \Rightarrow$$

$$\sum_{(x,z)} (x - Az)z^T = 0 \quad \Rightarrow \quad A = \left(\sum_{(x,z)} xz^T\right)\left(\sum_{z} zz^T\right)^{-1}$$

Pinv()

- Solution for D

$$\nabla_D \sum_{(x,z)} \frac{1}{2}\log|D| + 0.5(x - Az)^T D^{-1}(x - Az) = 0 \quad \Rightarrow$$

$$D = diag\left(\frac{1}{N}\left(\sum_{x} xx^T - A\sum_{(x,z)} xz^T\right)\right)$$

LGM with complete information

$$\underset{A,D}{\operatorname{argmax}} \sum_{(x,z)} -\frac{1}{2}\log|D| - 0.5(x - Az)^T D^{-1}(x - Az)$$

- Differentiating w.r.t A and $D$and equating to 0, we get an easy solution
- Solution for A

$$\nabla_A \sum_{(x,z)} 0.5(x - Az)^T D^{-1}(x - Az) = 0 \quad \Rightarrow$$

$$\sum_{(x,z)} (x - Az)z^T = 0 \quad \Rightarrow \quad A = \left(\sum_{(x,z)} xz^T\right)\left(\sum_{z} zz^T\right)^{-1}$$

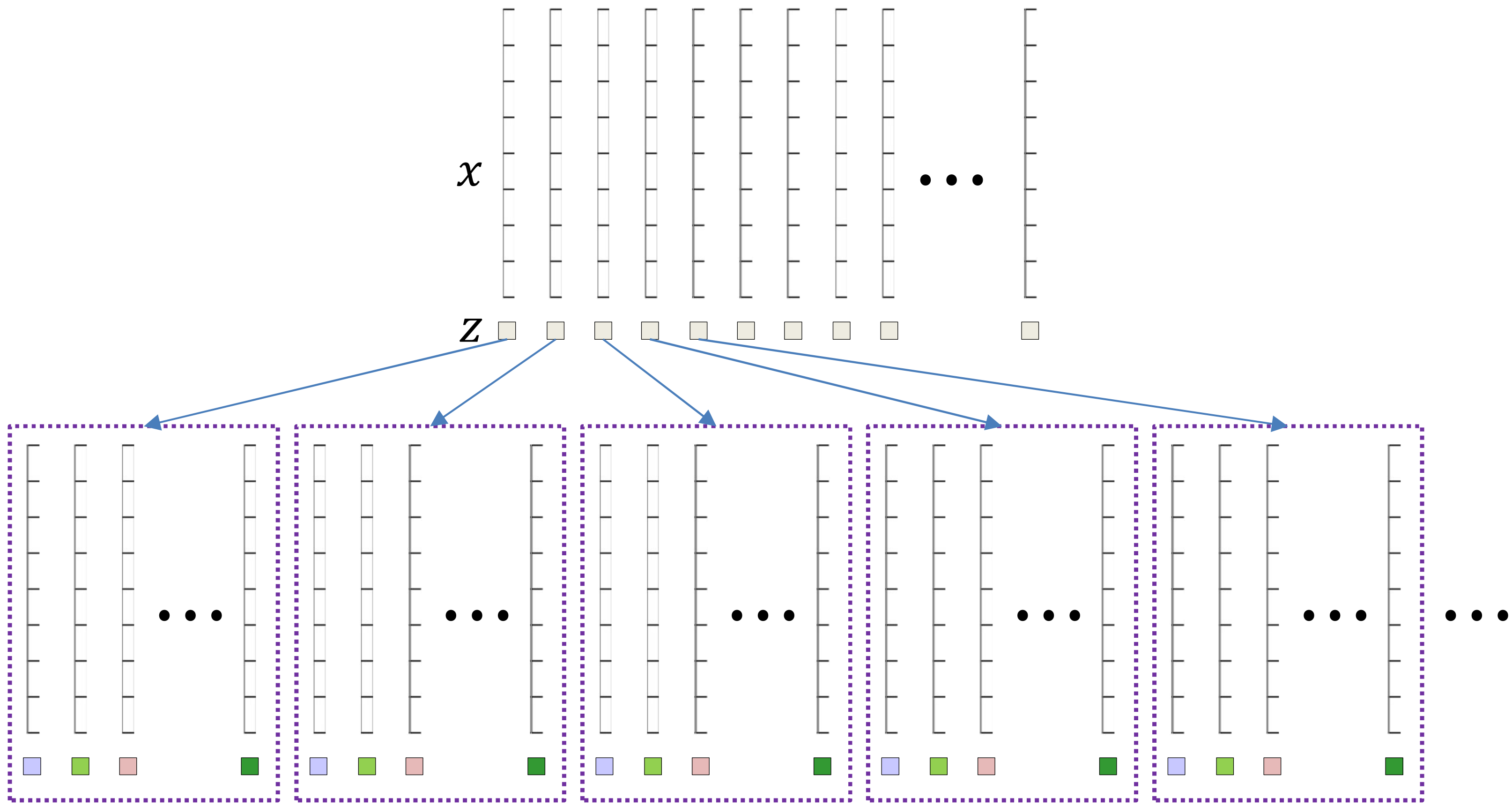
- Solution for D

$$\nabla_D \sum_{(x,z)} \frac{1}{2}\log|D| + 0.5(x - Az)^T D^{-1}(x - Az) = 0 \quad \Rightarrow$$

$$D = diag\left(\frac{1}{N}\left(\sum_{x} xx^T - A\sum_{(x,z)} xz^T\right)\right)$$

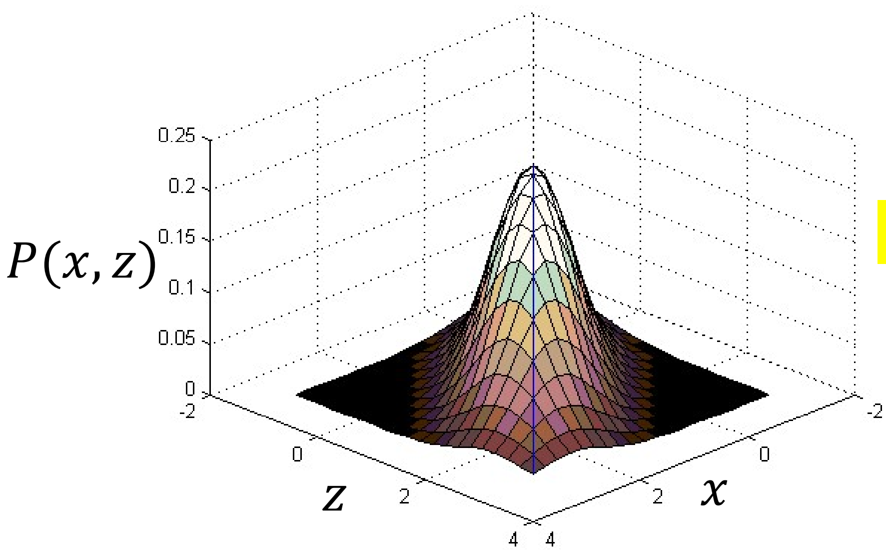
Unfortunately we do not observe z.
It is missing; the observations are incomplete

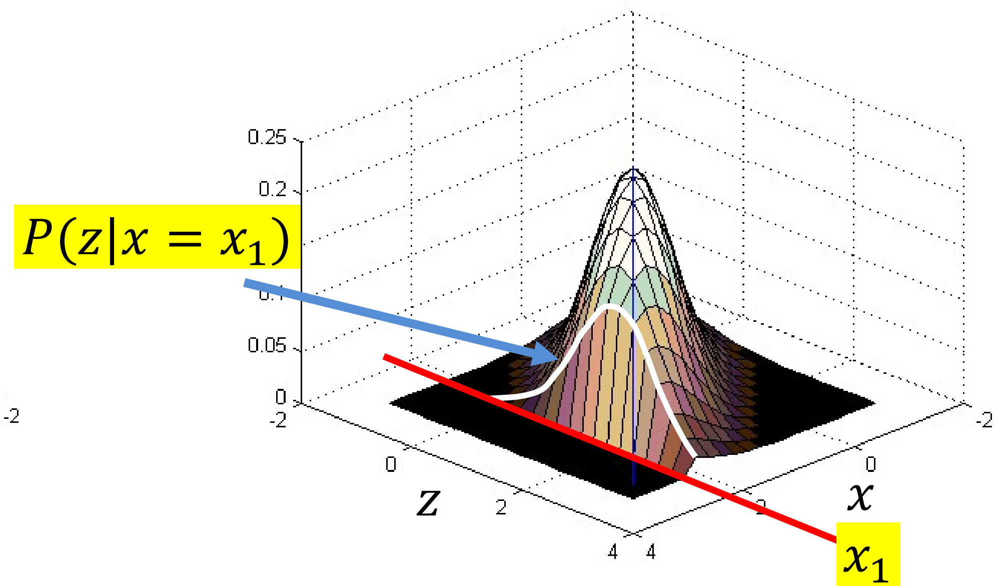
Expectation Maximization for LGM



- *Complete* the data
- Option 1:
  - In *every possible way* proportional to $P(z|x)$
  - Compute the solution from the completed data

The posterior $P(z|x)$



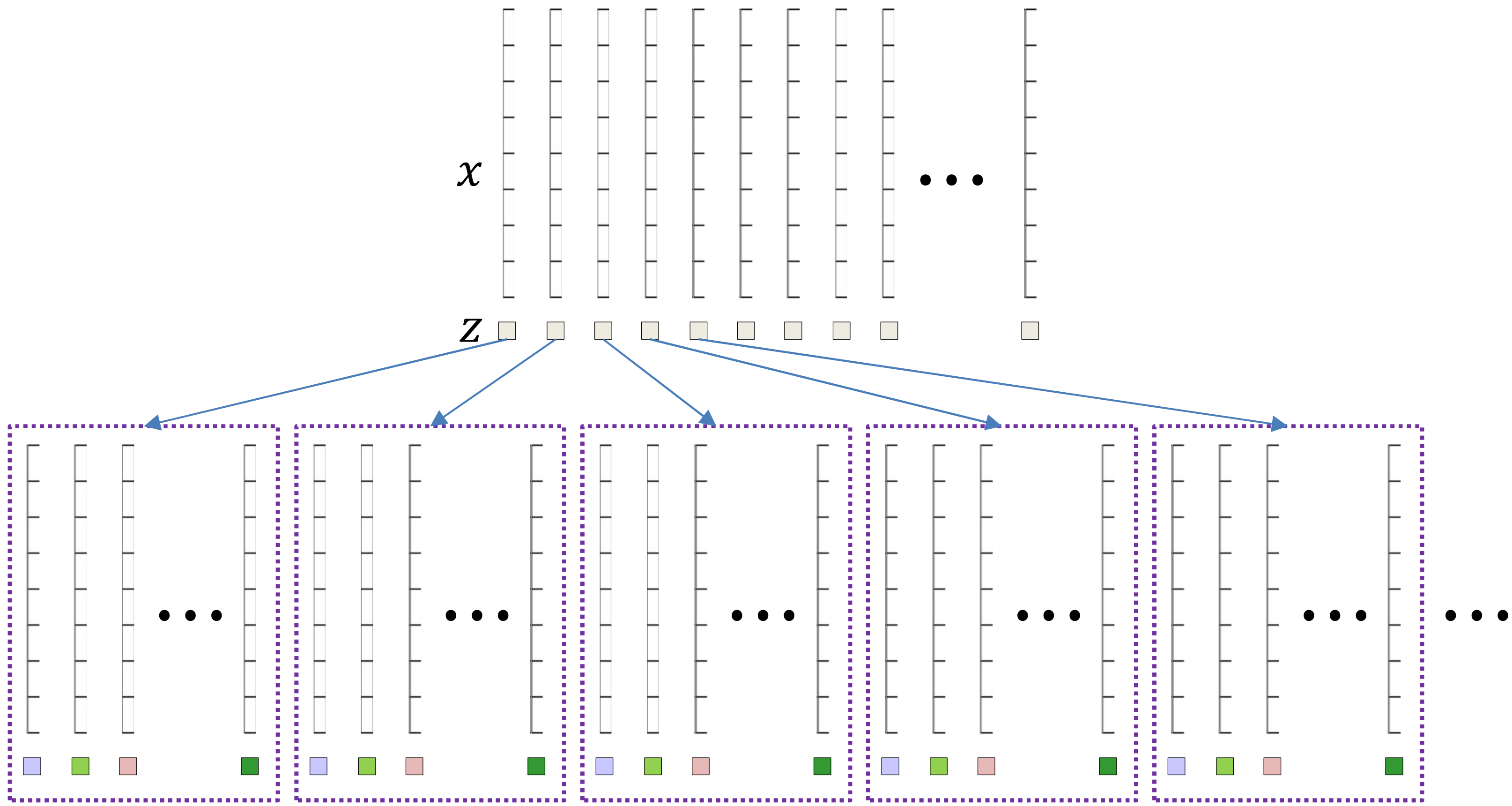


- $P(x)$ is Gaussian
  - We saw this
- The *joint* distribution of x and z is also Gaussian
  - Trust me
- The *conditional* distribution of z given x is also Gaussian

$$P(z|x) = N(z; A^T(AA^T + D)^{-1}x, I - A^T(AA^T + D)^{-1}A)$$

  - Trust me

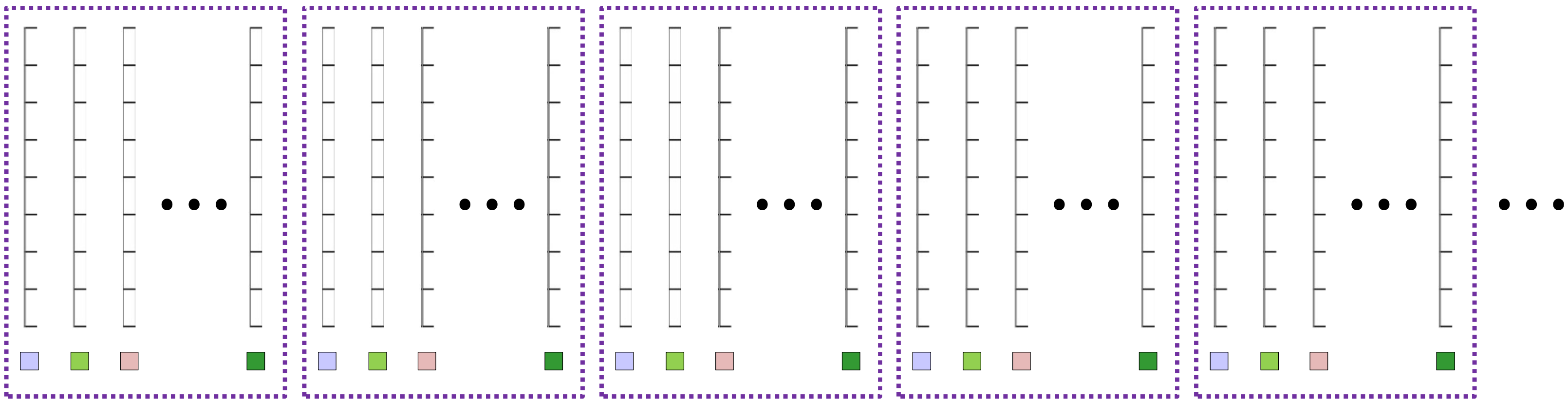
Expectation Maximization for LGM



- *Complete* the data
- Option 1:
  - In *every possible way* proportional to $P(z|x)$
  - Compute the solution from the completed data

$$P(z|x) = N(z; A^T(AA^T + D)^{-1}x, I - A^T(AA^T + D)^{-1}A)$$

Expectation Maximization for LGM



- *Complete* the data in *every possible way* proportional to $P(z|x)$
  - Compute the solution from the completed data
  - $\underset{A,D}{\operatorname{argmax}} \sum_{(x,z)} -\frac{1}{2}\log|D| - 0.5(x - Az)^T D^{-1}(x - Az)$
- The z values for each x are distributed according to $P(z|x)$. Segregating the summation by x

$$\underset{A,D}{\operatorname{argmax}} \sum_{x} \int_{-\infty}^{\infty} p(z|x) \left(-\frac{1}{2}\log|D| - 0.5(x - Az)^T D^{-1}(x - Az) \right) dz$$

LGM with incomplete information

$$\underset{A,D}{\operatorname{argmax}} \sum_x \int_{-\infty}^{\infty} p(z|x) \left(-\frac{1}{2}\log|D| - 0.5(x - Az)^T D^{-1}(x - Az) \right) dz$$

- Differentiating w.r.t A and $D$and equating to 0, we get an easy solution
- Solution for A

$$\nabla_A \sum_x \int_{-\infty}^{\infty} p(z|x)(x - Az)^T D^{-1}(x - Az)dz = 0 \quad \Rightarrow$$

$$\sum_x \int_{-\infty}^{\infty} p(z|x)(x - Az)z^T dz = 0 \quad \Rightarrow \quad A = \left(\sum_x \int_{-\infty}^{\infty} p(z|x)xz^T dz \right) \left(\sum_x \int_{-\infty}^{\infty} p(z|x)zz^T dz \right)^{-1}$$

- Solution for D

$$\nabla_D \left(N\log|D| + \sum_x \int_{-\infty}^{\infty} p(z|x)(x - Az)^T D^{-1}(x - Az)dz \right) = 0 \quad \Rightarrow$$

$$D = diag\left(\frac{1}{N} \left(\sum_x xx^T - A \sum_x \int_{-\infty}^{\infty} p(z|x)xz^T dz \right) \right)$$

These are closed form solutions,
Key: All terms integrate over *all possible completion of incomplete observations*, where the proportionality attached to any completion of x is P(z|x)

LGM with incomplete information

- It is actually an iterative algorithm (EM):
- Solution for A

$$A^{k+1} = \left(\sum_x \int_{-\infty}^{\infty} p(z|x; A^k, D^k) x z^T dz\right)\left(\sum_x \int_{-\infty}^{\infty} p(z|x; A^k, D^k) z z^T dz\right)^{-1}$$

- Solution for D

$$D = diag\left(\frac{1}{N}\left(\sum_x x x^T - A \sum_x \int_{-\infty}^{\infty} p(z|x; A^k, D^k) x z^T dz\right)\right)$$

These are closed form solutions,
Key: All terms integrate over *all possible completion of incomplete observations*, where the proportionality attached to any completion of x is P(z|x)

LGM with incomplete information

- It is actually an iterative algorithm (EM):
- Solution for A

$$A^{k+1} = \left(\sum_x \int_{-\infty}^{\infty} p(z|x;A^k,D^k)xz^T dz\right)\left(\sum_x \int_{-\infty}^{\infty} p(z|x;A^k,D^k)zz^T dz\right)^{-1}$$

- Solution for D

$$D = diag\left(\frac{1}{N}\left(\sum_x xx^T - A\sum_x \int_{-\infty}^{\infty} p(z|x;A^k,D^k)xz^T dz\right)\right)$$

These are closed form solutions,
Key: All terms integrate over *all possible completion of incomplete observations*, where the proportionality attached to any completion of x is P(z|x)

LGM with incomplete information

- It is actually an iterative algorithm (EM):
- Solution for A

$$A^{k+1} = \left(\sum_x xE[z|x]^T\right)\left(\sum_x \int_{-\infty}^{\infty} p(z|x; A^k, D^k) zz^T dz\right)^{-1}$$

- Solution for D

$$D = diag\left(\frac{1}{N}\left(\sum_x xx^T - A\sum_x xE[z|x]^T\right)\right)$$

These are closed form solutions,
Key: All terms integrate over *all possible completion of incomplete observations*, where the proportionality attached to any completion of x is P(z|x)

LGM with incomplete information

- It is actually an iterative algorithm (EM):
- Solution for A

$$A^{k+1} = \left(\sum_x xE[z|x]^T\right)\left(\sum_x E[zz^T|x]\right)^{-1}$$

- Solution for D

$$D = diag\left(\frac{1}{N}\left(\sum_x xx^T - A\sum_x xE[z|x]^T\right)\right)$$

These are closed form solutions,
Key: All terms integrate over *all possible completion of incomplete observations*, where the proportionality attached to any completion of x is P(z|x)

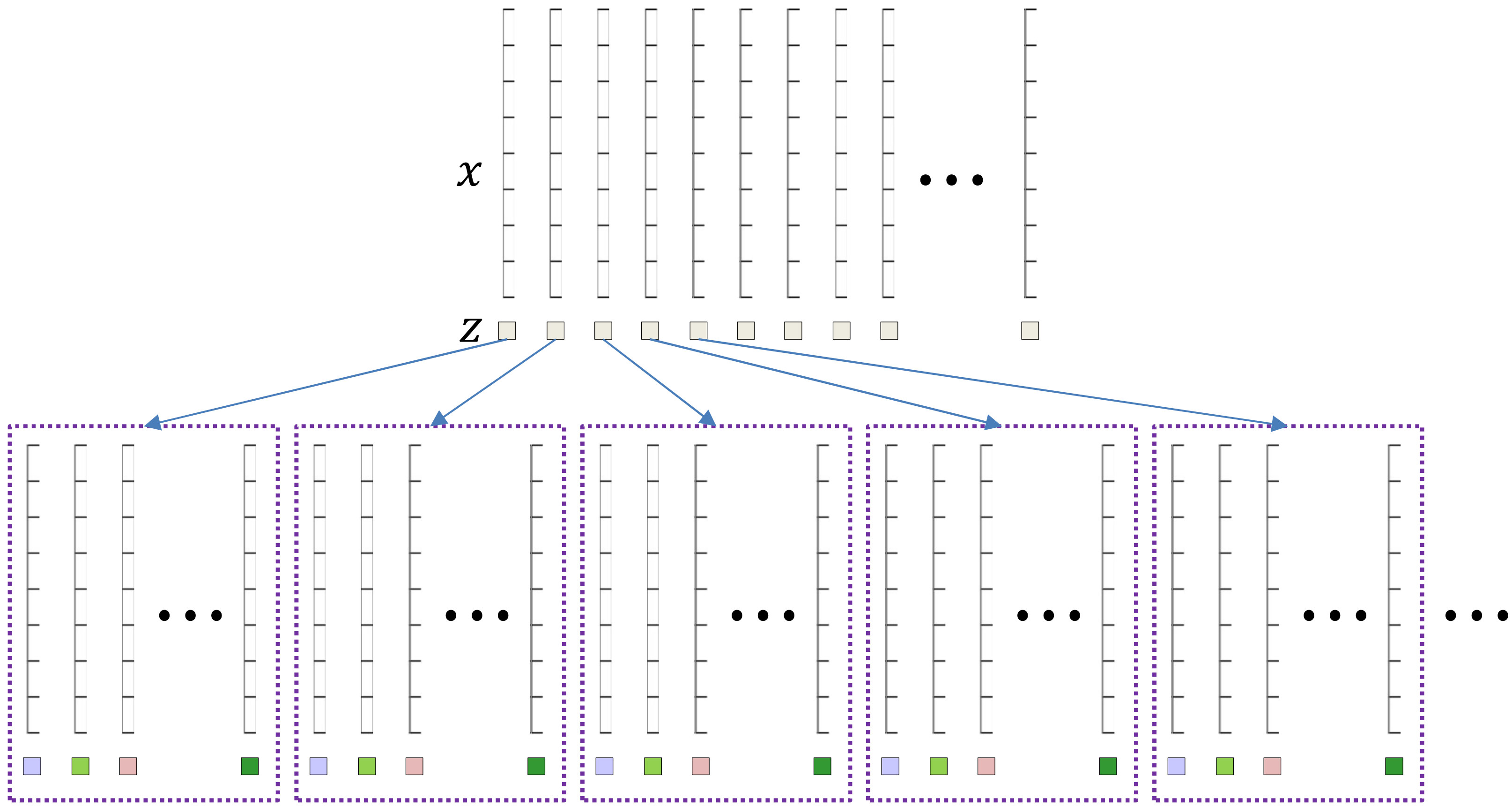
LGM with incomplete information

$$P(z|x) = N(z; A^T\left(AA^T + D\right)^{-1}x, I - A^T\left(AA^T + D\right)^{-1}A)$$

$$E[z|x] = A^T\left(AA^T + D\right)^{-1}x$$

$$E[zz^T|x] = I - A^T\left(AA^T + D\right)^{-1}A + E[z|x]E[z|x]^T$$

Expectation Maximization for LGM



- *Complete* the data

$$P(z|x) = N(z; A^T\left(AA^T + D\right)^{-1}x, I - A^T\left(AA^T + D\right)^{-1}A)$$

- Option 2:
  - ***By drawing samples from*** $P(z|x)$
  - Compute the solution from the completed data

LGM from drawn samples

- Since we now have a collection of *complete vectors,* we can use the usual complete-data formulae
- Solution for A

$$A^{k+1} = \left(\sum_{(x,z)} xz^T\right)\left(\sum_{z} zz^T\right)^{-1}$$

- Solution for D

$$D^{k+1} = diag\left(\frac{1}{N}\left(\sum_{x} xx^T - A^k \sum_{(x,z)} xz^T\right)\right)$$

These are closed form solutions

Draw missing components from P(z|x; A^k, D^k) to *complete the data*

Estimate parameters from completed data

Poll 4

- Select all that are true
  - PCA is a specific instance of Linear Gaussian Models
  - LGM need all dimensions of the observation to be observed and cannot handle the case of missing information.
  - We can get global optimal of LGM easily through the EM algorithm.

Poll 4

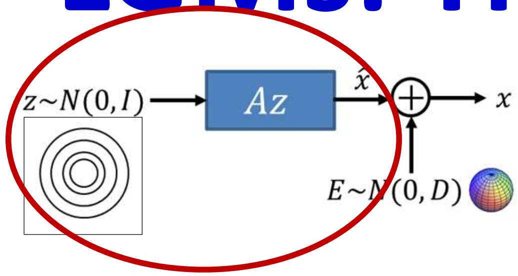
- Select all that are true
  - **PCA is a specific instance of Linear Gaussian Models**
  - LGM need all dimensions of the observation to be observed and cannot handle the case of missing information.
  - We can get global optimal of LGM easily through the EM algorithm.

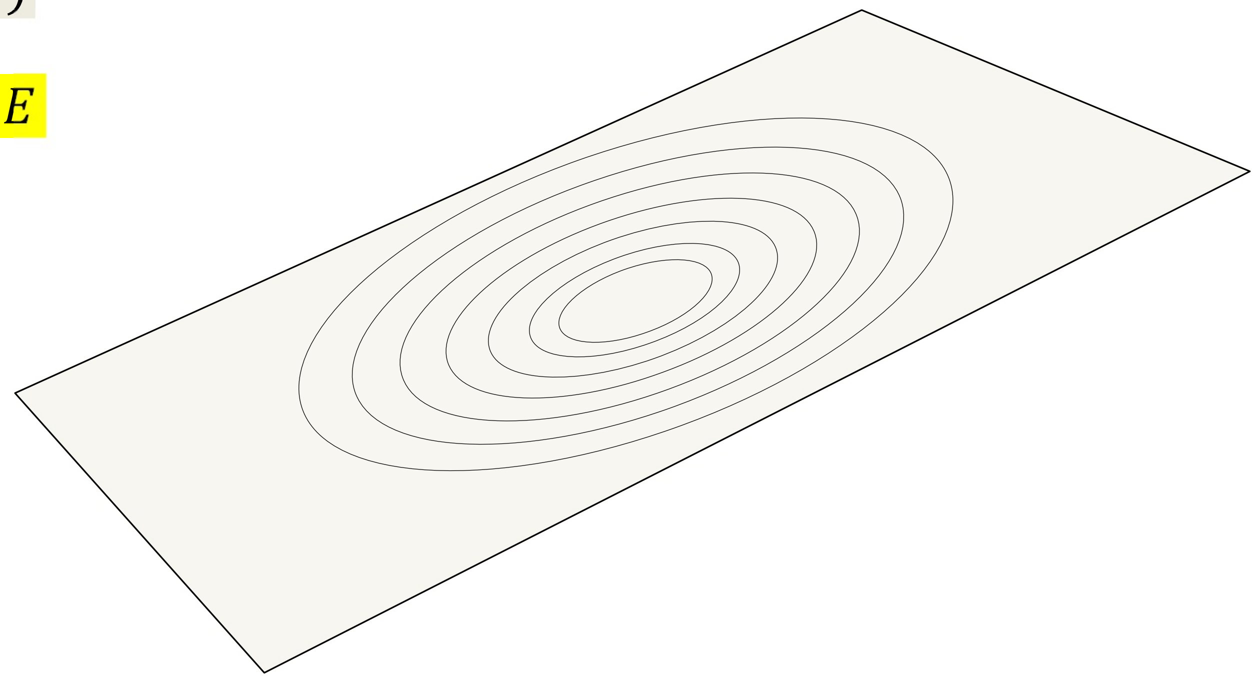
LGMs: The intuition

$z \sim N(0, I)$

$E \sim N(0, D)$

$x = Az + E$





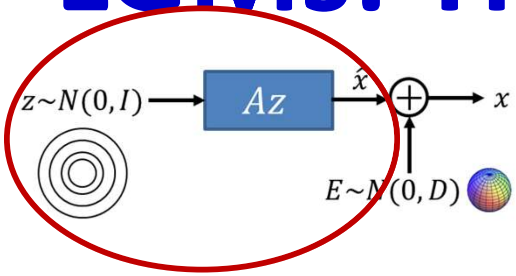
- The linear transform stretches and rotates the K-dimensional input space onto a K-dimensional hyperplane in the data space
- The isotropic Gaussian in the input space becomes a stretched and rotated Gaussian on the hyperplane

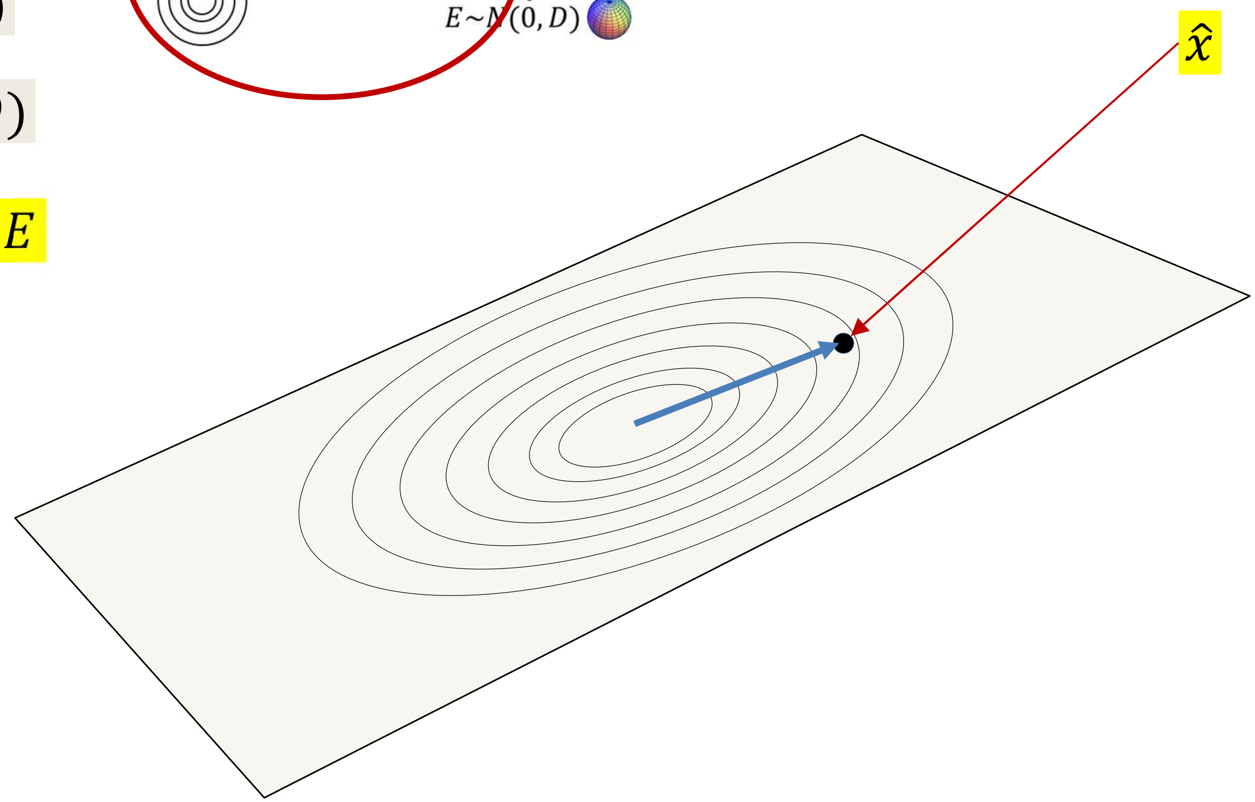
LGMs: The intuition

$z \sim N(0, I)$

$E \sim N(0, D)$

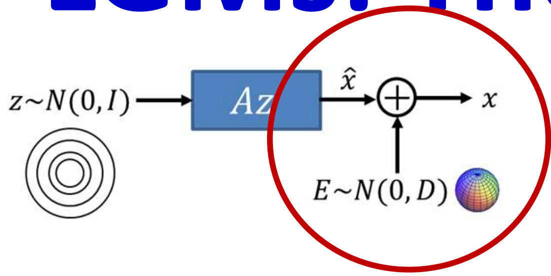
$x = Az + E$





- Drawing samples: The first step places the z somewhere on the plane described by A
  - The distribution of points on the plane is also Gaussian

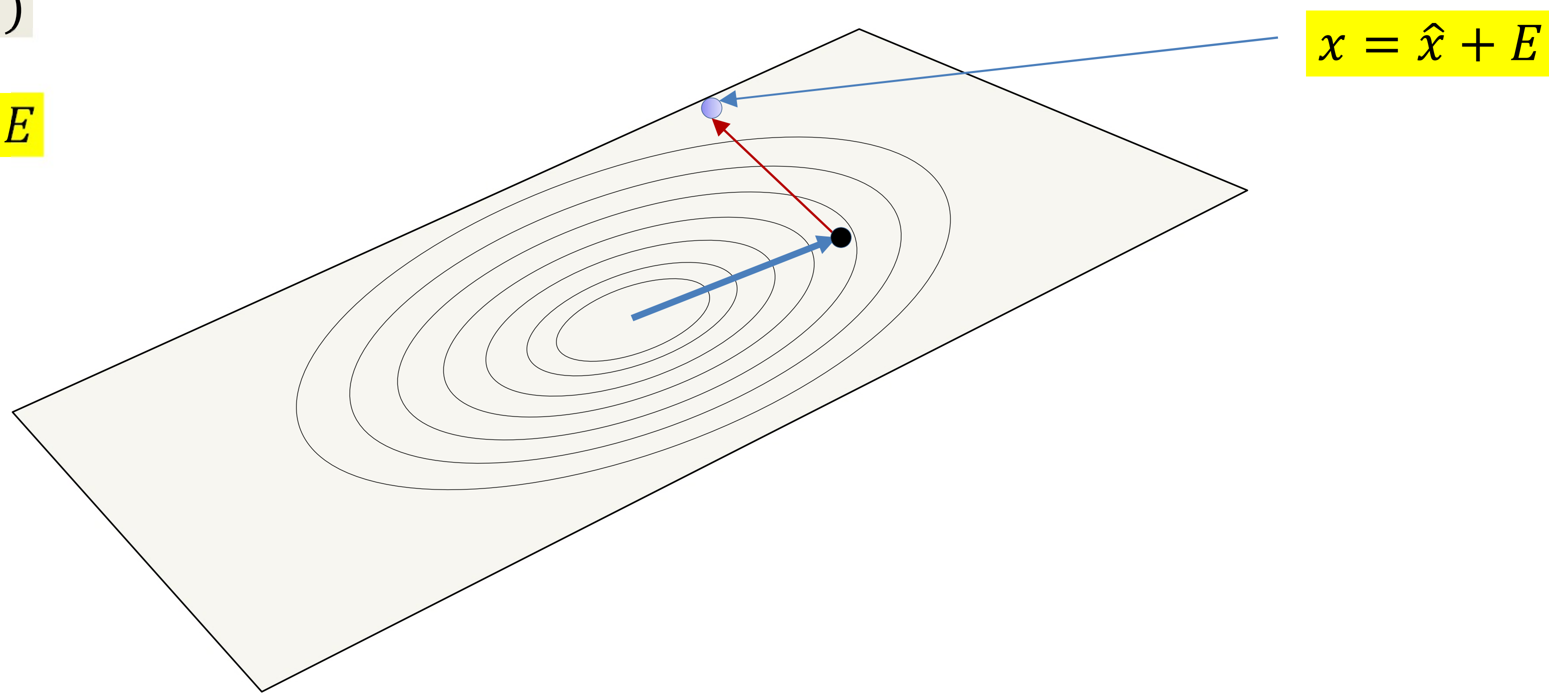
LGMs: The intuition



$z \sim N(0, I)$

$E \sim N(0, D)$

$X = Az + E$



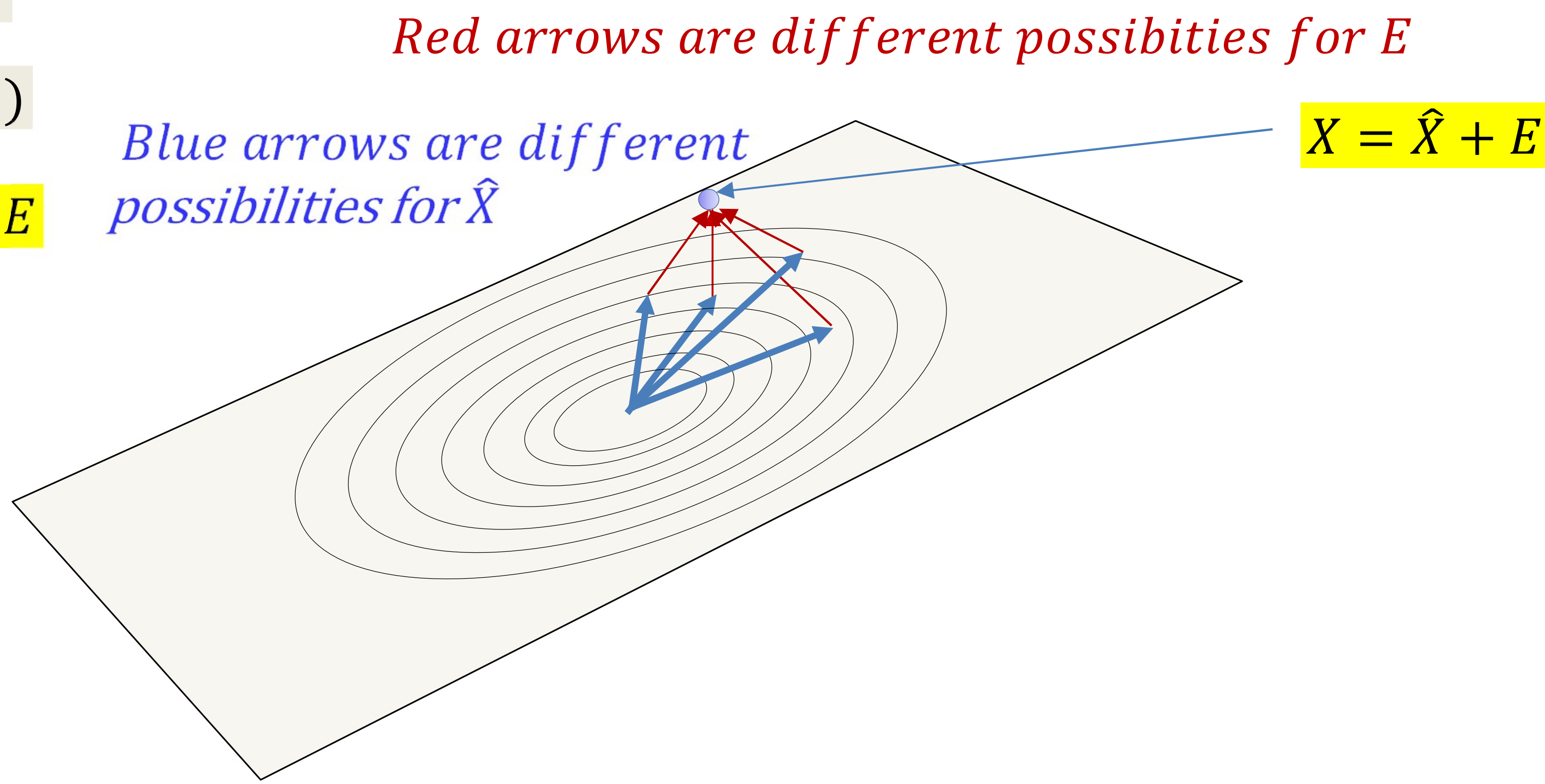
- LGM model: The first step places the z somewhere on the plane described by A
  - The distribution of points on the plane is also Gaussian
- Second step: Add Gaussian noise to produce points that aren't necessarily on the plane
  - Noise added is not revealed

EM for LGMs: The intuition

$z \sim N(0, I)$

$E \sim N(0, D)$

$X = Az + E$



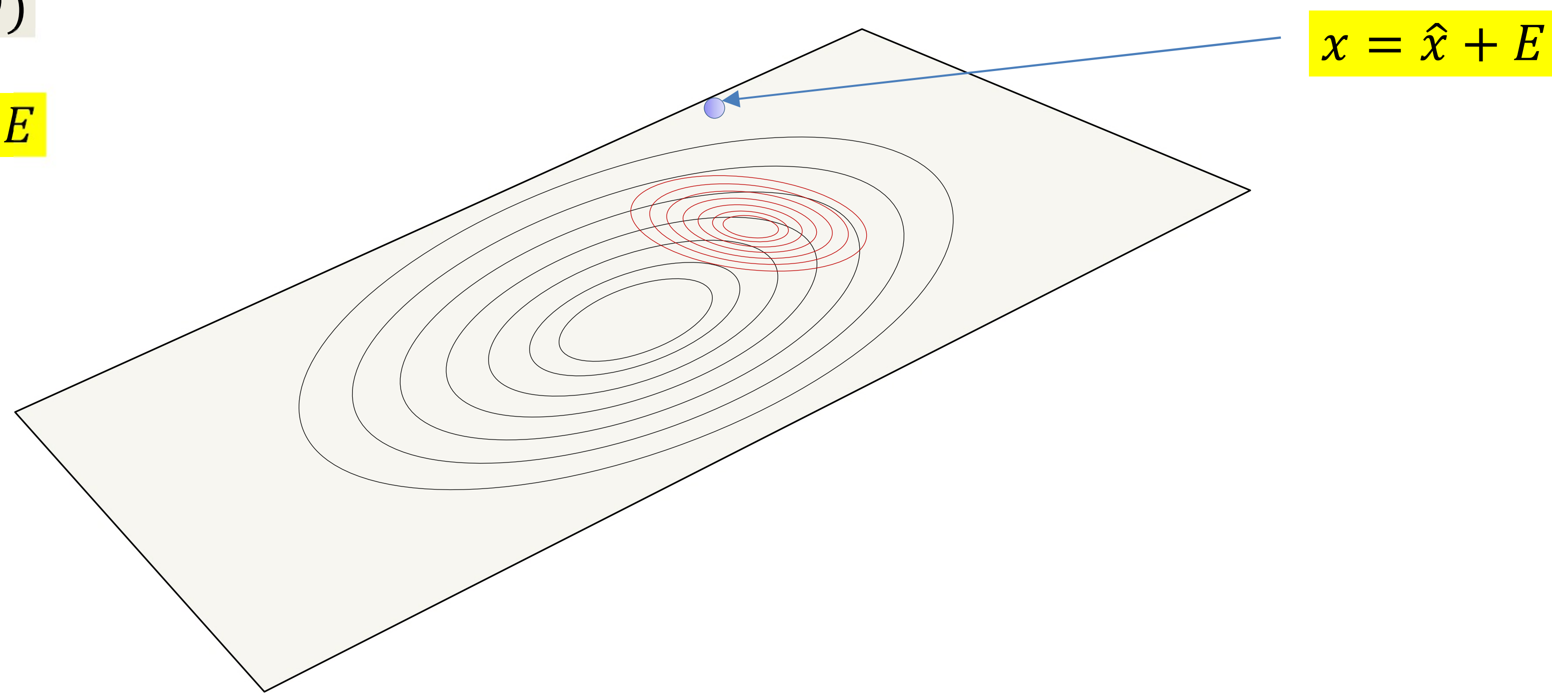
- In an LGM the way to produce any data instance is not unique
- Conversely, given only the data point, the “shadow” on the principal plane cannot be uniquely known

EM Solution

$z \sim N(0, I)$

$E \sim N(0, D)$

$x = Az + E$



- The posterior probability $P(z|x)$ gives you the location of all the points on the plane that *could* have generated x and their probabilities

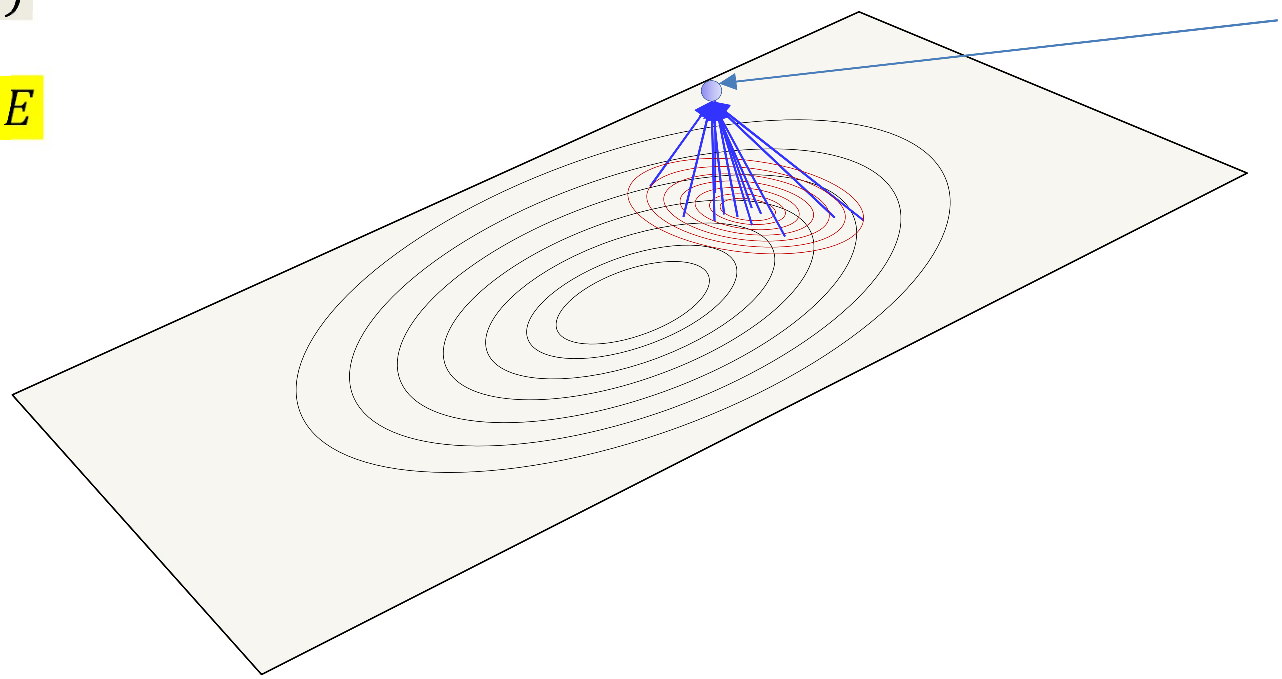
EM Solution

$z \sim N(0, I)$

$E \sim N(0, D)$

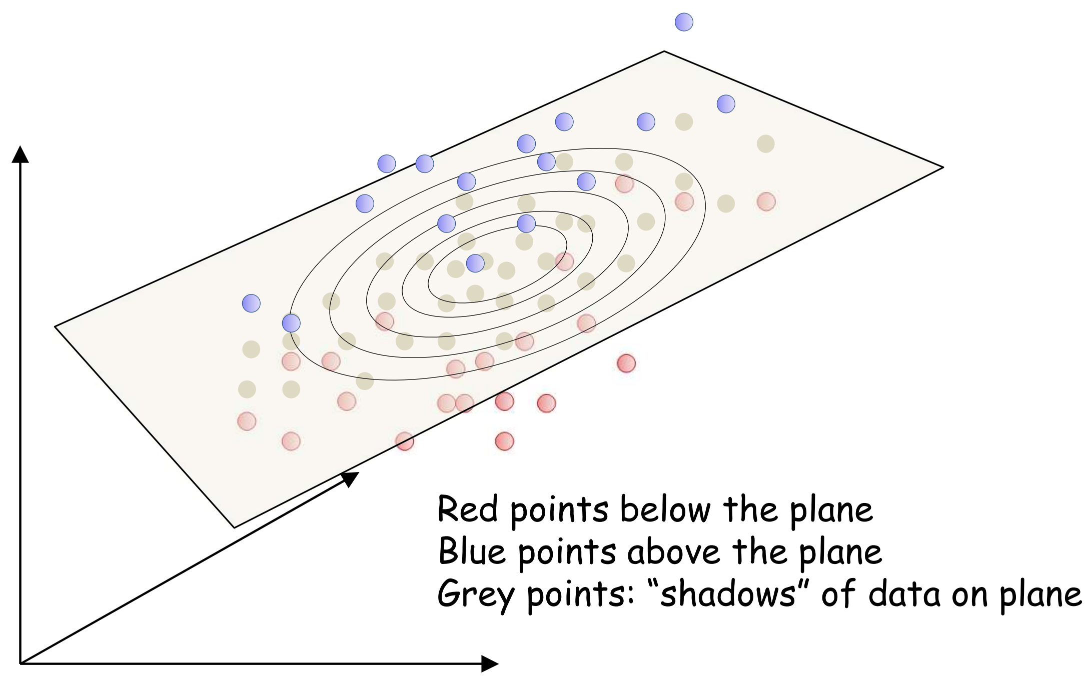
$X = Az + E$

$X = \hat{X} + E$



- Attach the point to *every* location on the plane, according to $P(z|x)$
  - Or to a sample of points on the plane drawn from $P(z|x)$
- There will be more attachments where $P(z|x)$ is higher, and fewer where it is lower

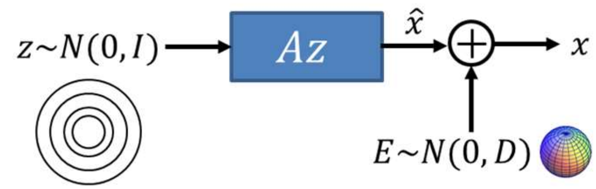
EM Solution

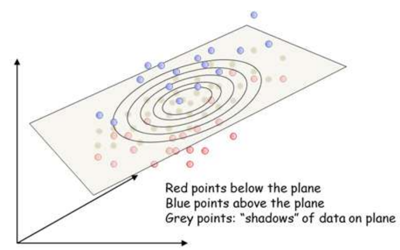


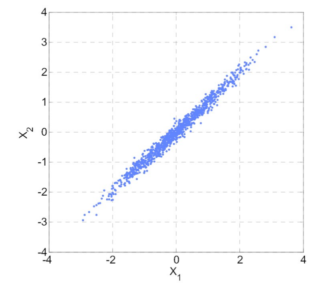
- Attach *every* training point in this manner
- Let the plane rotate and stretch until the total tension (sum squared length) of all the attachments is minimize
- Repeat attachment and rotation until convergence…

Summarizing LGMs

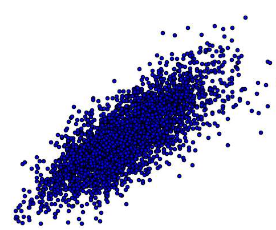
- LGMs are models for *Gaussian* distributions
- Specifically, they model the distribution of data as Gaussian, where most of the variation is along a *linear* manifold
  - They do this by transforming a Gaussian RV z through a linear transform $f(z) = Az$ that transforms the K-dim input space of z into a K-dimensional hyperplane (linear manifold) in the data space

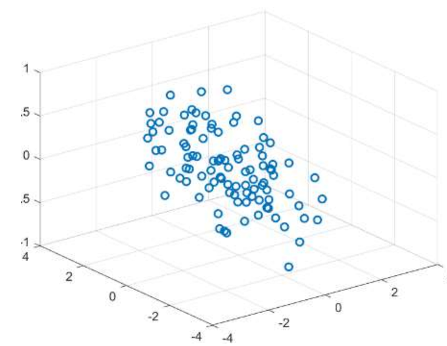






- They are excellent models for data that actually fit these assumptions
  - Often, we can simply assume that data lie near linear manifolds and model them with LGMs
  - PCA, an instance of LGMs, is very popular





Story for the day

- EM: An iterative technique to estimate probability models for data with missing components or information
  - By iteratively “completing” the data and reestimating parameters

- PCA: Is actually a generative model for Gaussian data
  - Data lie close to a linear manifold, with orthogonal noise

- Factor Analysis: Also a generative model for Gaussian data
  - Data lie close to a linear manifold
  - Like PCA, but without directional constraints on the noise