Principal Component Analysis III

CS771: Introduction to Machine Learning

Purushottam Kar

Announcements

No classes next week on account of mid-semester recess

Classes resume on Monday, October 14

Quiz 3 to be held on October 16

Recap of Last Lecture

Applications of PCA

- **Space savings**: $\mathcal{O}\big((n+d)\cdot k\big)$ vs $\mathcal{O}(nd)$
- **Time savings** (w.r.t linear models): $\mathcal{O}\big((n+d)\cdot k\big)$ vs $\mathcal{O}(nd)$
- **Noise removal**: dims with small(er) singular values may be noise
  - *Left singular vectors give us (new) low-dimensional representation of data*
  - *Example application: foreground-background separation in videos*
- **Learning prototypes**: right singular vecs can be treated as prototypes
  - *Data can be linearly approx. in terms of these prototypes (not so in GMM etc)*
  - *Example applications: eigenfaces, LSA, collaborative filtering*

Many (equivalent) interpretations of PCA

- *Gives smallest reconstruction error, largest variance preservation*

Probabilistic PCA

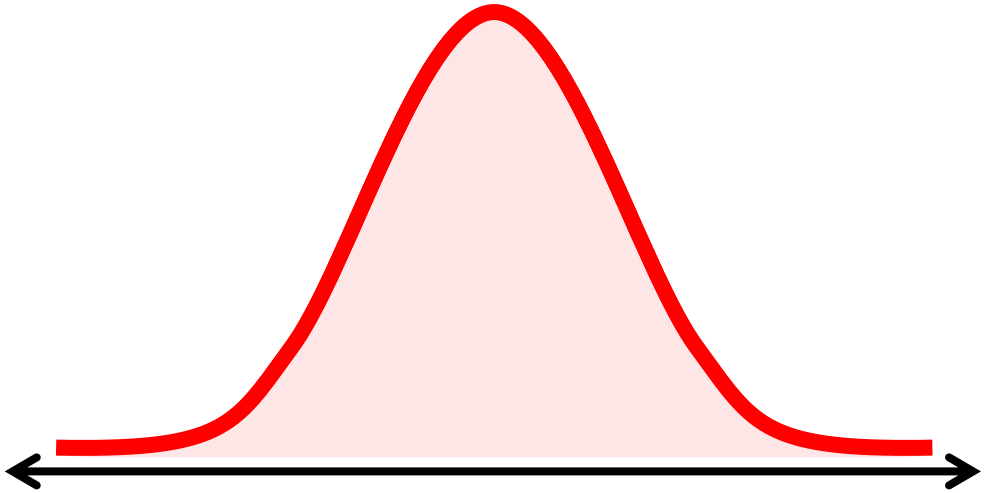
The real data was actually sampled from a k-dim standard Gaussian, but got linearly mapped to a d-dim space and some noise got added

Probabilistic PCA

The real data was actually sampled from a k-dim standard Gaussian, but got linearly mapped to a d-dim space and some noise got added

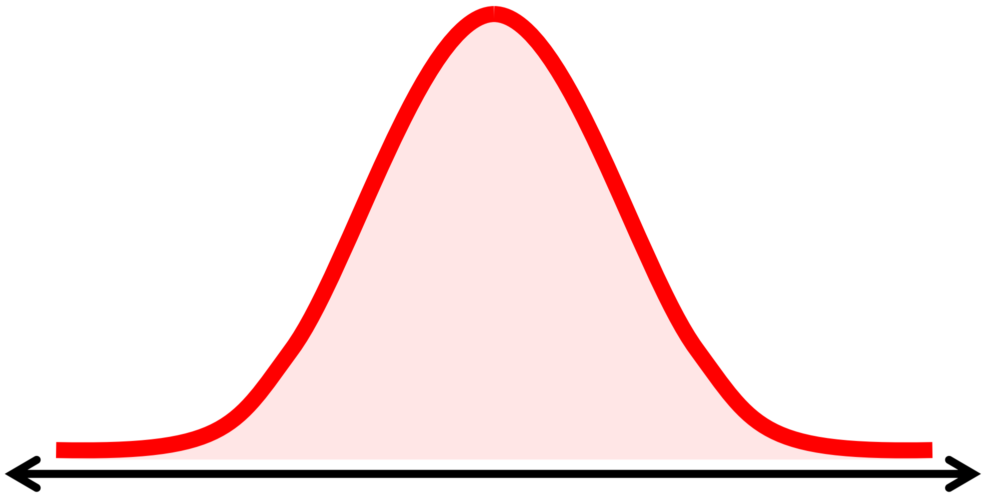
Probabilistic PCA

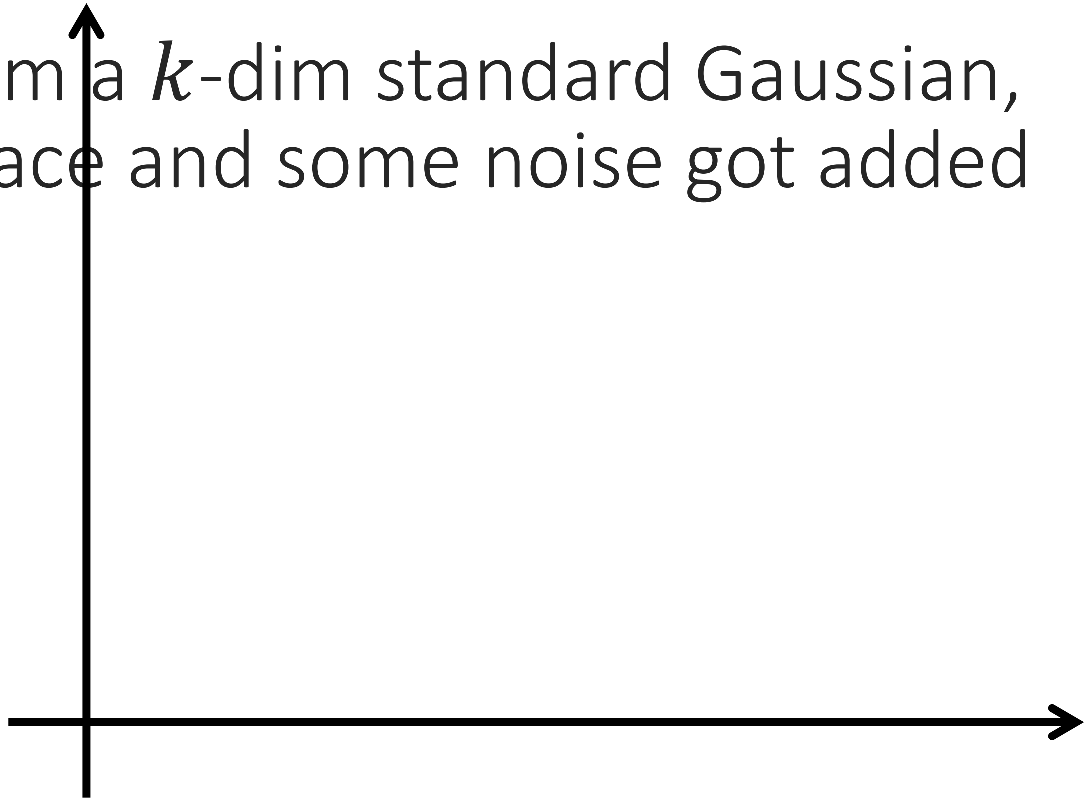
The real data was actually sampled from a k-dim standard Gaussian, but got linearly mapped to a d-dim space and some noise got added



Probabilistic PCA

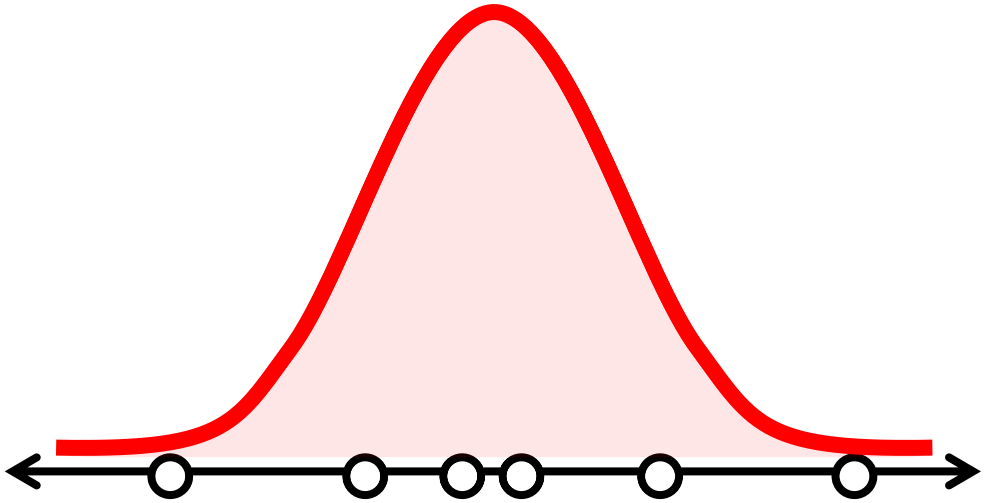
The real data was actually sampled from a k-dim standard Gaussian, but got linearly mapped to a d-dim space and some noise got added

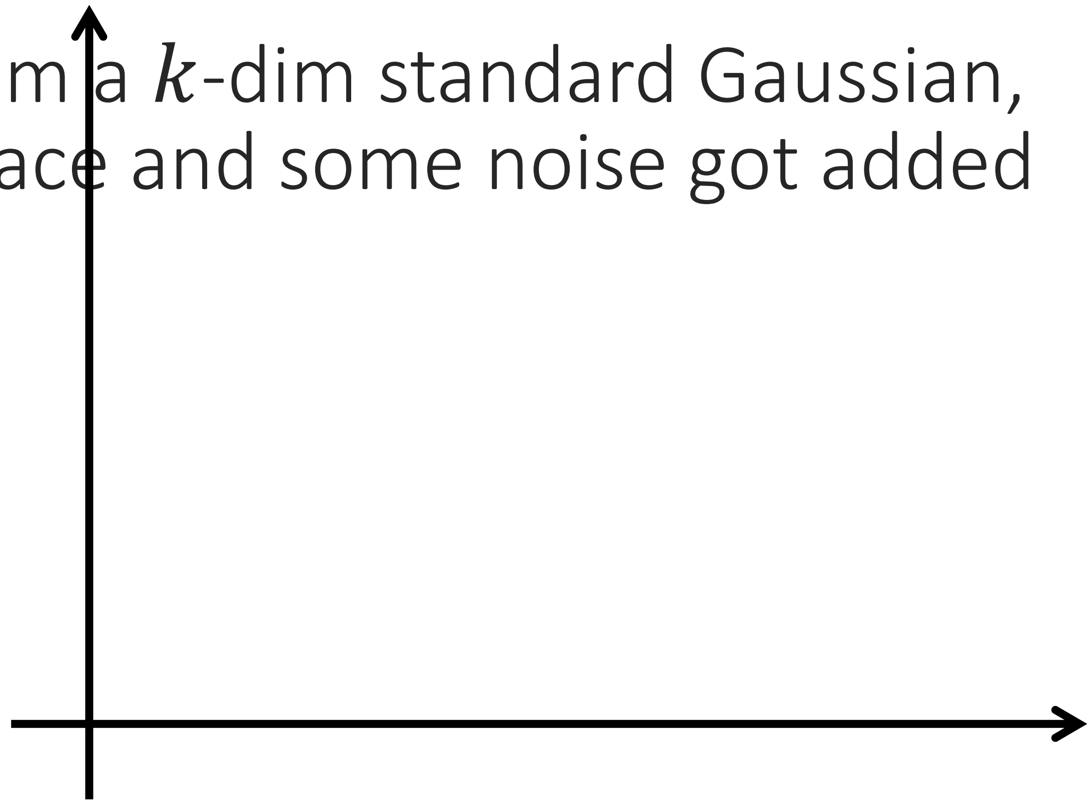




Probabilistic PCA

The real data was actually sampled from a k-dim standard Gaussian, but got linearly mapped to a d-dim space and some noise got added

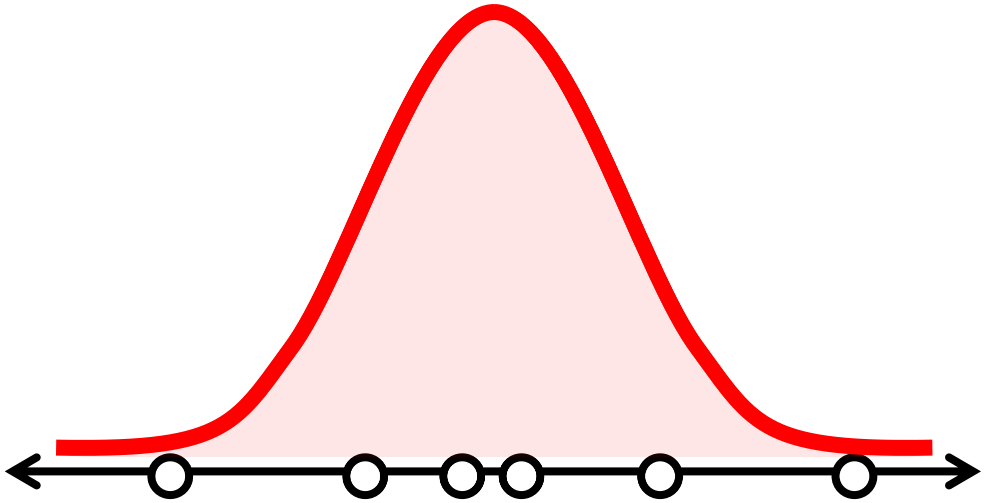


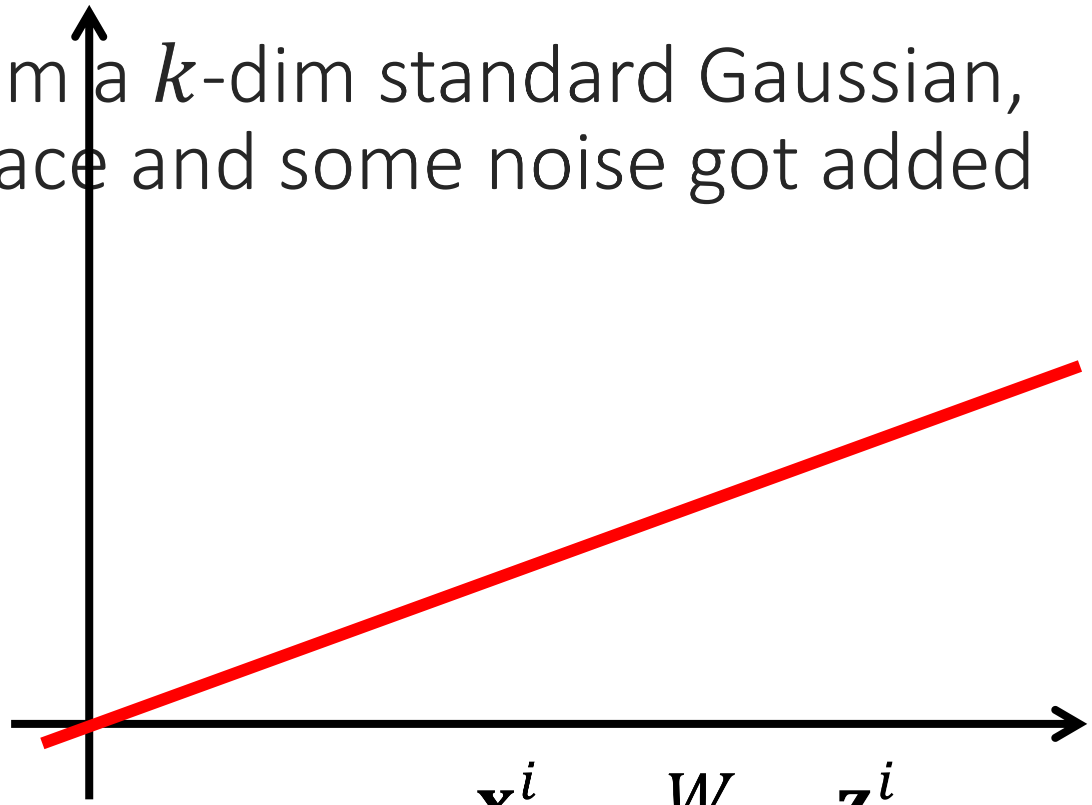


$$\mathbf{z}^i \sim \mathcal{N}(\mathbf{0}, I_k)$$

Probabilistic PCA

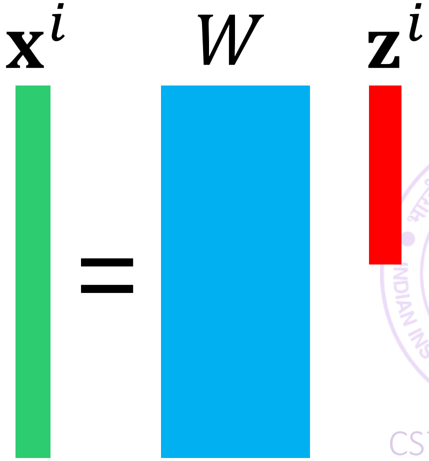
The real data was actually sampled from a k-dim standard Gaussian, but got linearly mapped to a d-dim space and some noise got added





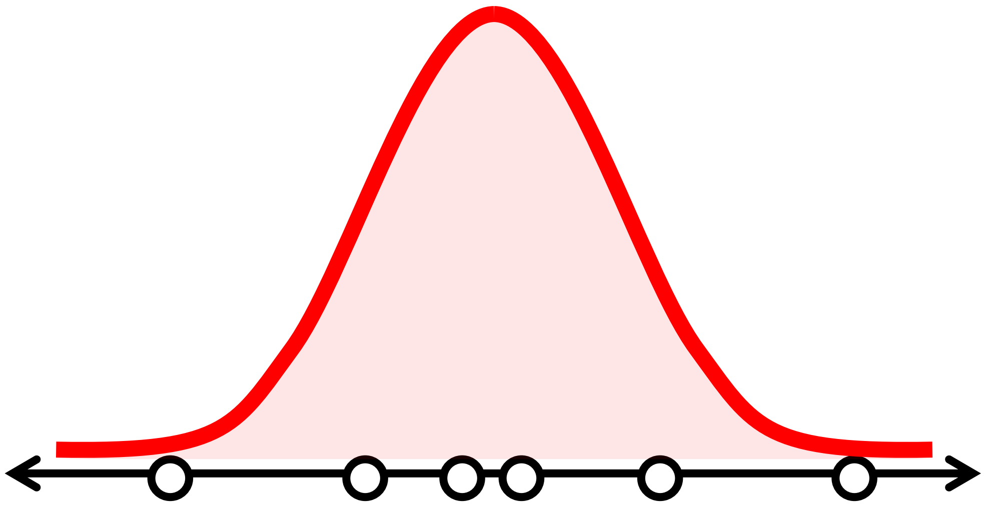
$$\mathbf{z}^i \sim \mathcal{N}(\mathbf{0}, I_k) \qquad \mathbf{x}^i = W\mathbf{z}^i \,.$$

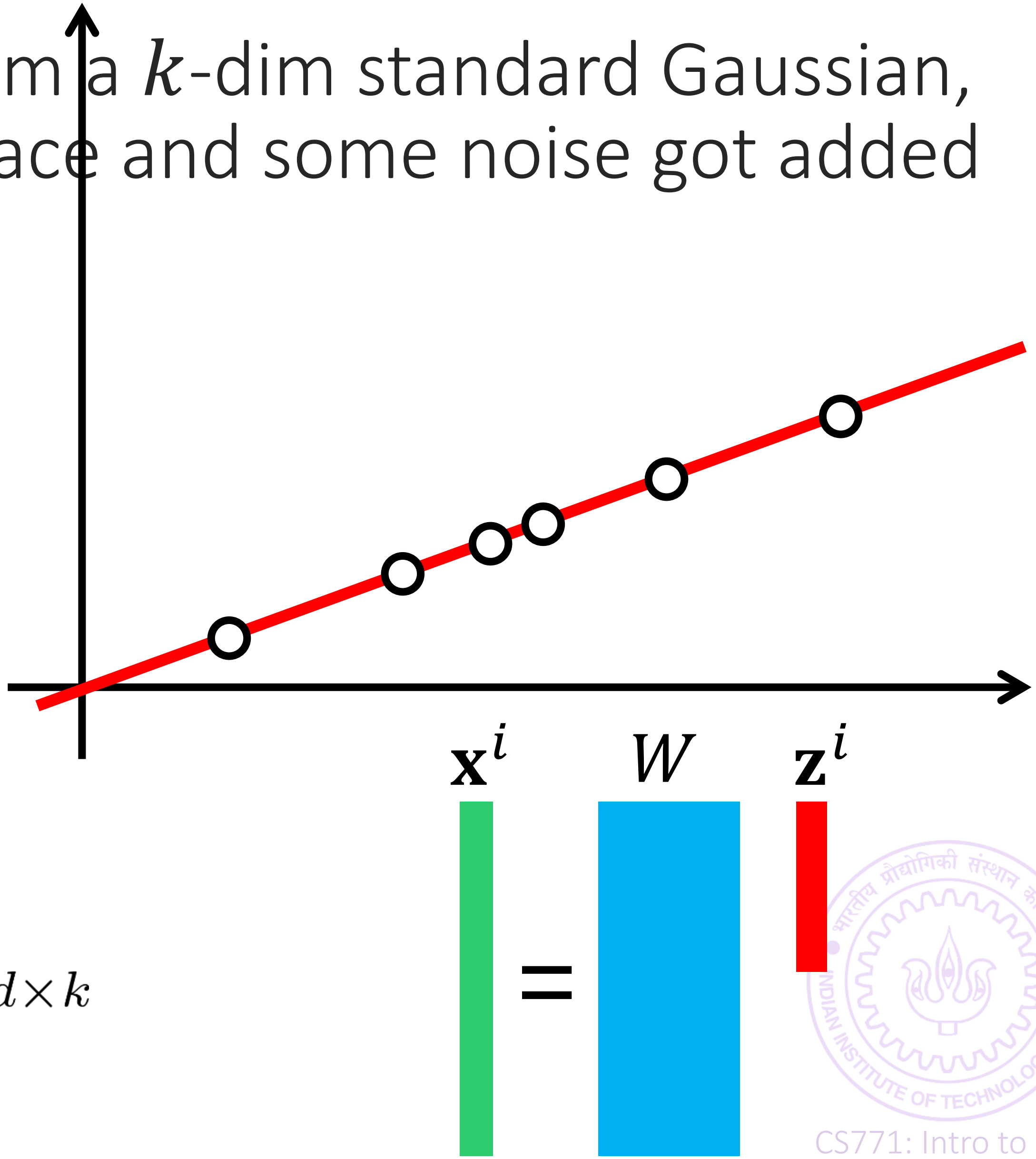
$$W \in \mathbb{R}^{d\times k}$$



Probabilistic PCA

The real data was actually sampled from a k-dim standard Gaussian, but got linearly mapped to a d-dim space and some noise got added



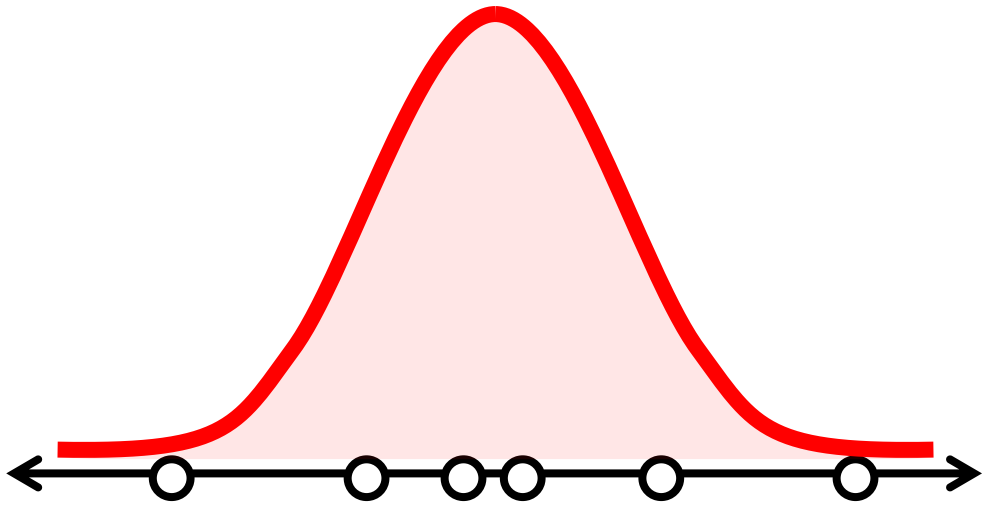


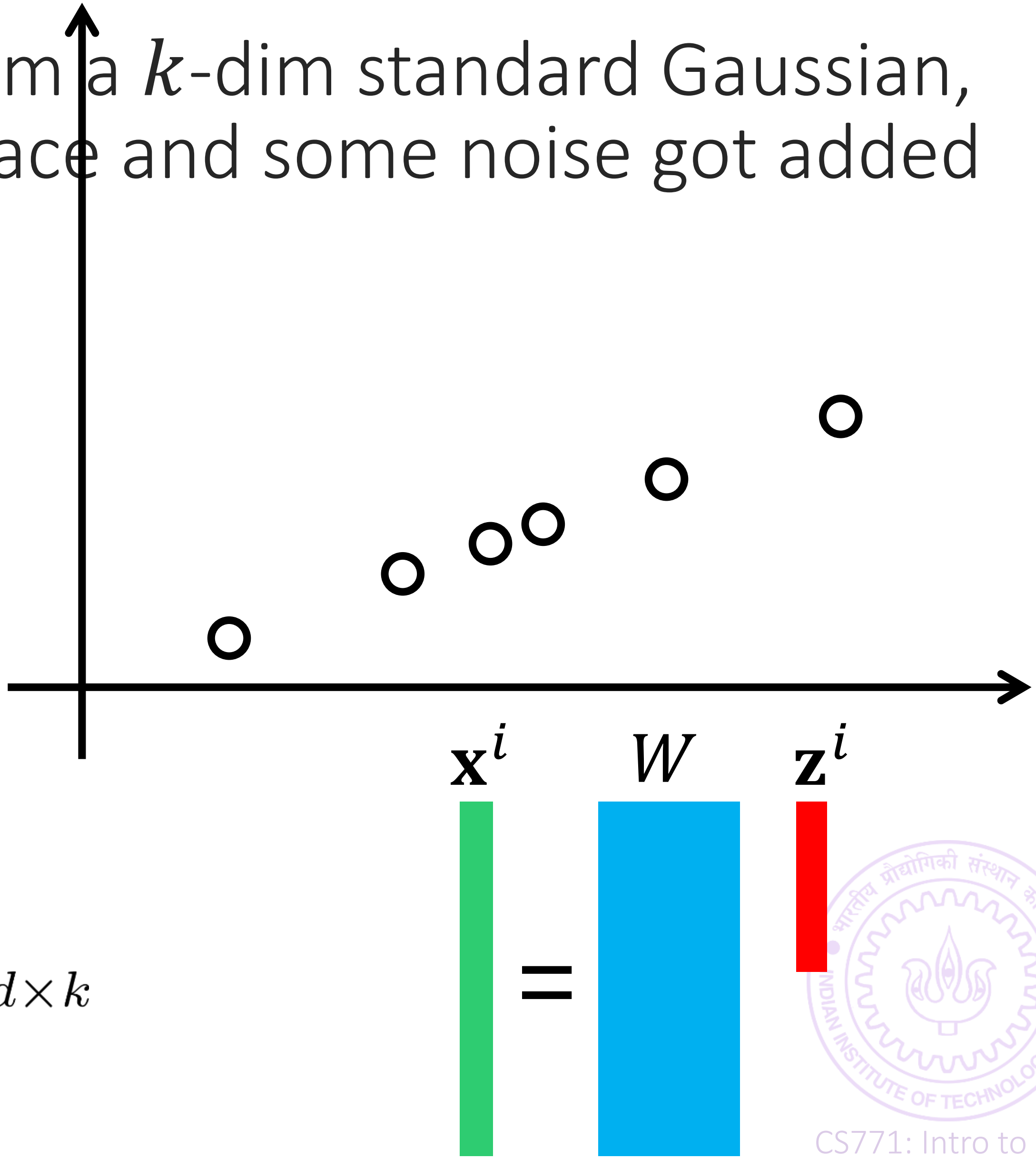
$\mathbf{z}^i \sim \mathcal{N}(\mathbf{0}, I_k)$ $\mathbf{x}^i = W\mathbf{z}^i$.

$W \in \mathbb{R}^{d \times k}$

Probabilistic PCA

The real data was actually sampled from a k-dim standard Gaussian, but got linearly mapped to a d-dim space and some noise got added



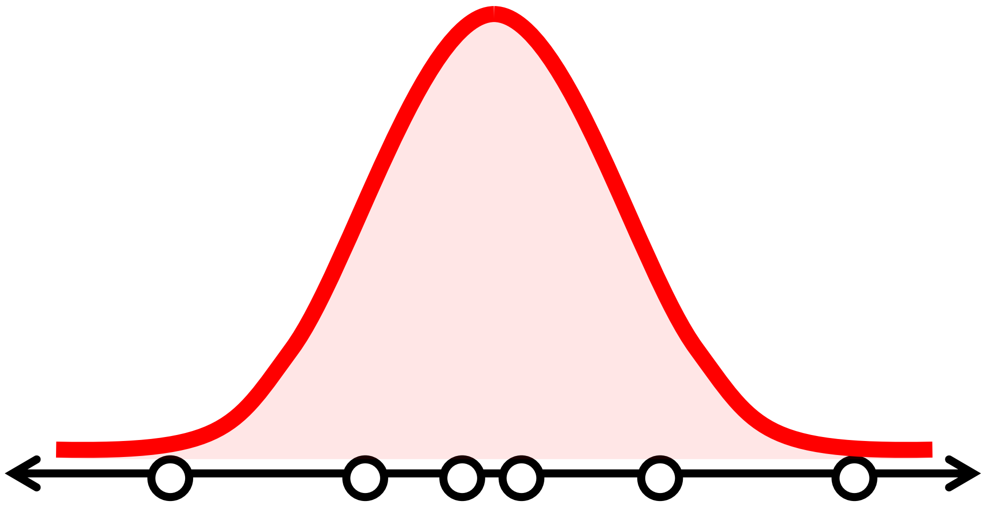


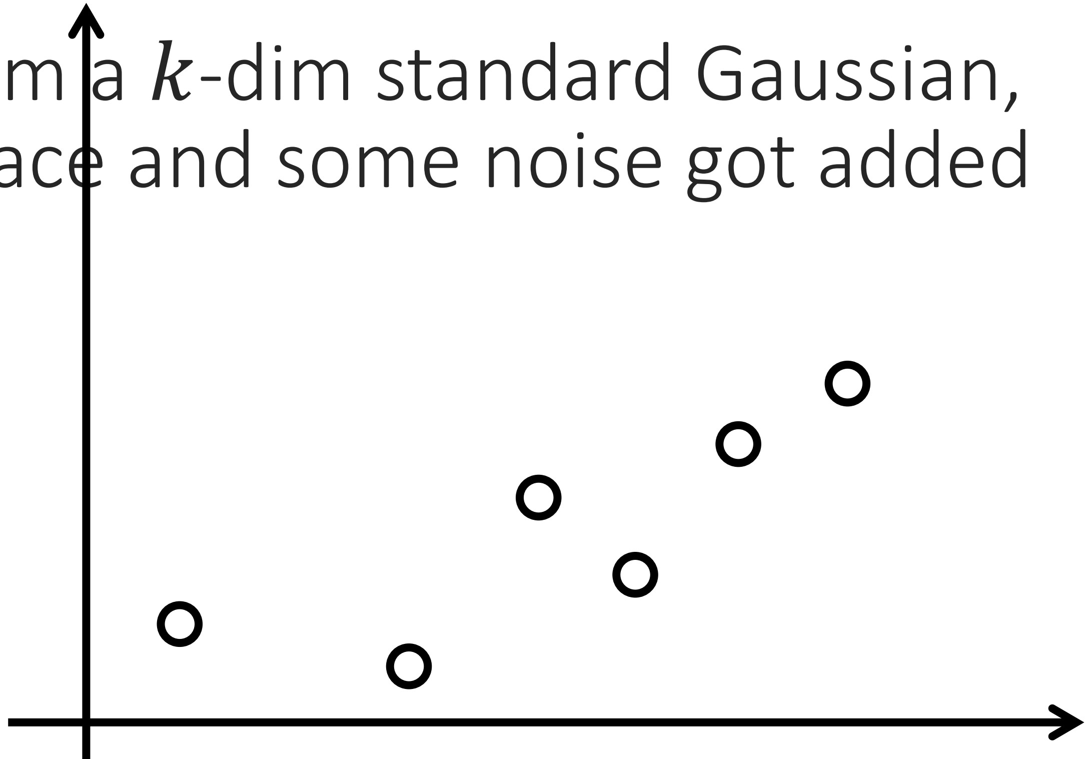
$$\mathbf{z}^i \sim \mathcal{N}(\mathbf{0}, I_k) \qquad \mathbf{x}^i = W\mathbf{z}^i$$

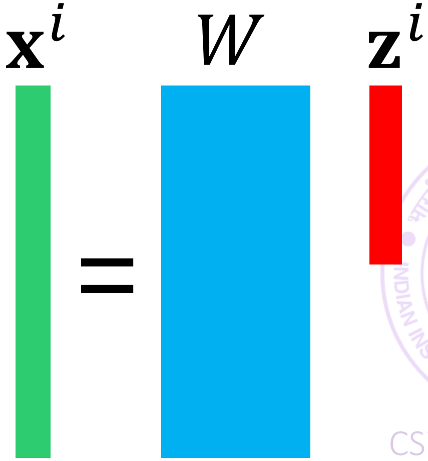
$$W \in \mathbb{R}^{d \times k}$$

Probabilistic PCA

The real data was actually sampled from a k-dim standard Gaussian, but got linearly mapped to a d-dim space and some noise got added







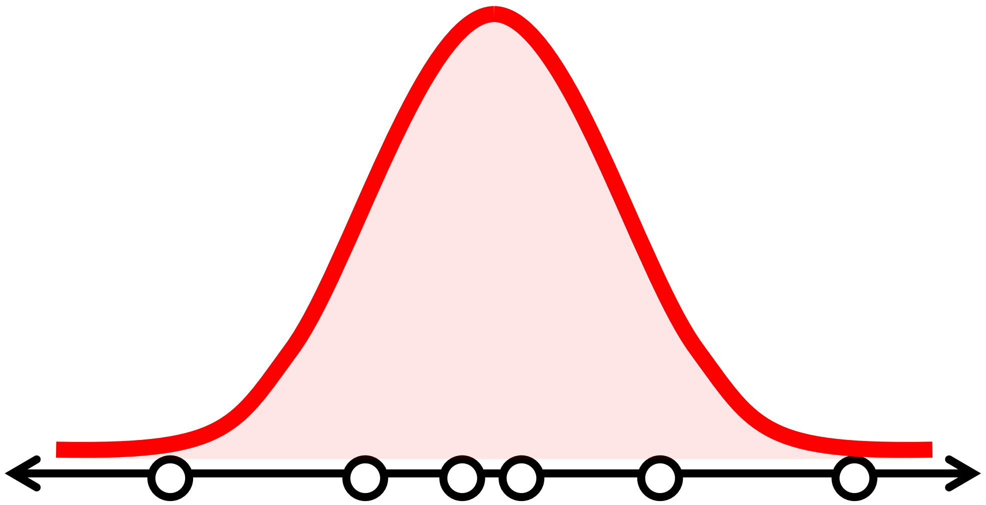
$$\mathbf{z}^i \sim \mathcal{N}(\mathbf{0}, I_k)$$

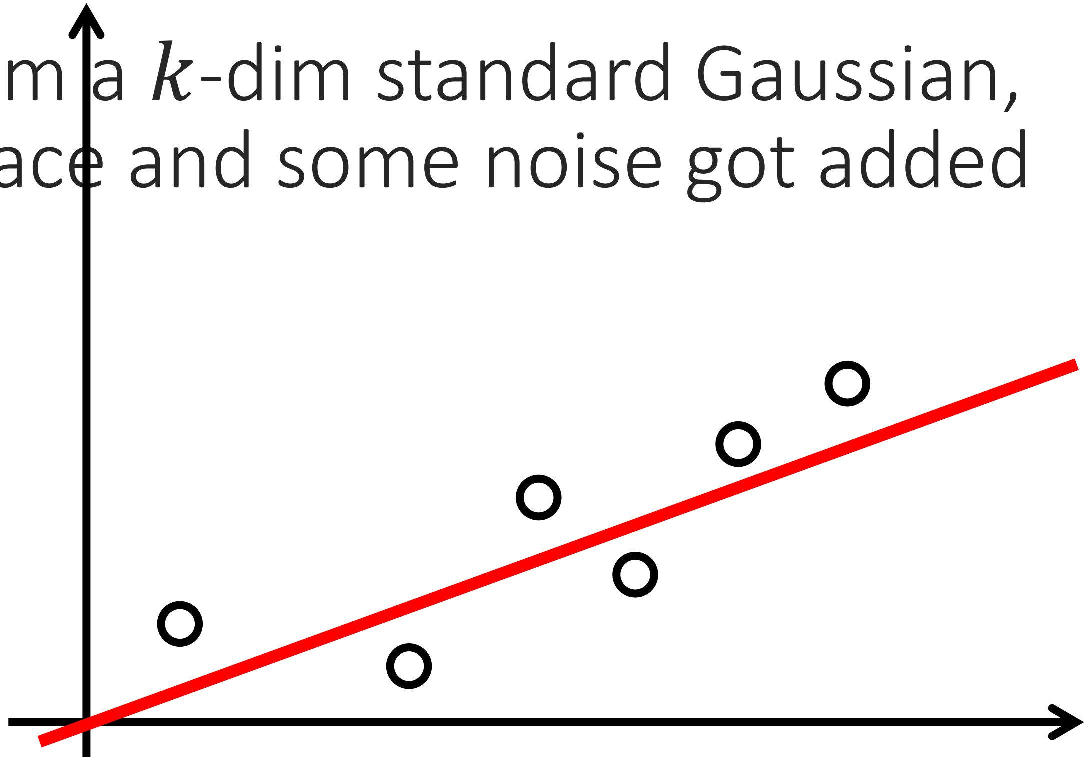
$$\mathbf{x}^i = W\mathbf{z}^i + \boldsymbol{\epsilon}^i, \;\; \boldsymbol{\epsilon}^i \sim \mathcal{N}(\mathbf{0}, \sigma^2 \cdot I_d)$$

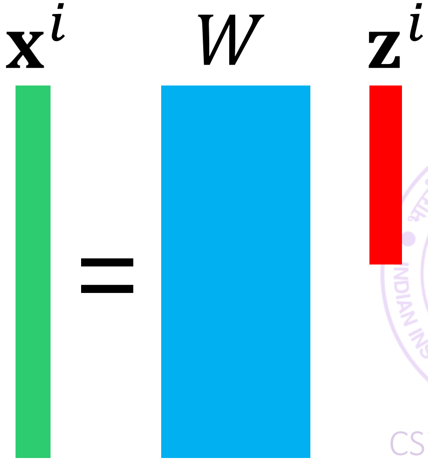
$$W \in \mathbb{R}^{d \times k}$$

Probabilistic PCA

The real data was actually sampled from a k-dim standard Gaussian, but got linearly mapped to a d-dim space and some noise got added







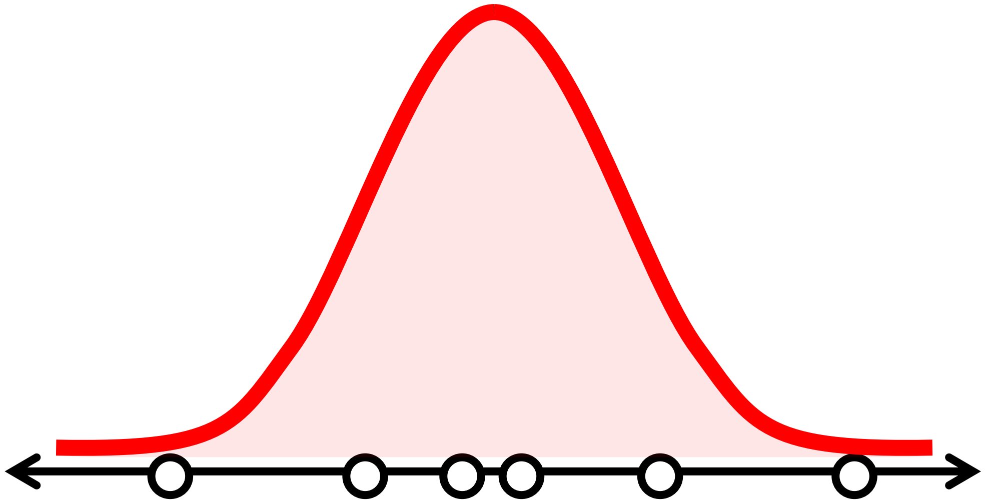
$$\mathbf{z}^i \sim \mathcal{N}(\mathbf{0}, I_k)$$

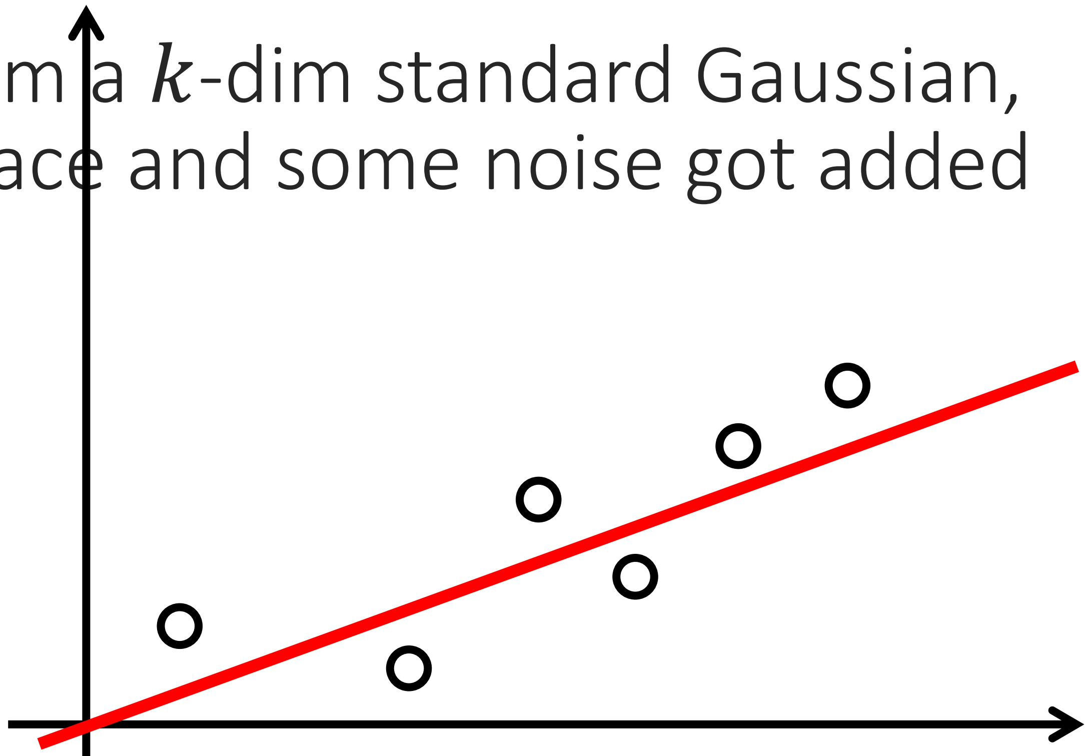
$$\mathbf{x}^i = W\mathbf{z}^i + \boldsymbol{\epsilon}^i, \;\; \boldsymbol{\epsilon}^i \sim \mathcal{N}(\mathbf{0}, \sigma^2 \cdot I_d)$$

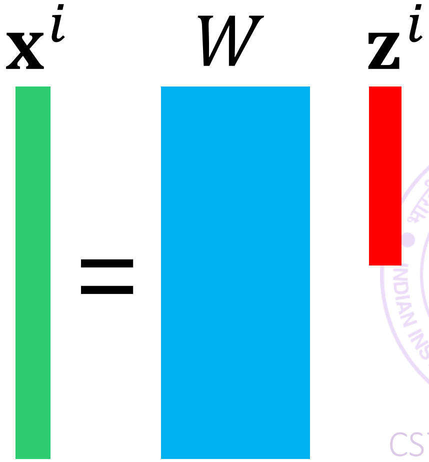
$$W \in \mathbb{R}^{d \times k}$$

Probabilistic PCA

The real data was actually sampled from a k-dim standard Gaussian, but got linearly mapped to a d-dim space and some noise got added







$$\mathbf{z}^i \sim \mathcal{N}(\mathbf{0}, I_k)$$

$$\mathbf{x}^i = W\mathbf{z}^i + \boldsymbol{\epsilon}^i,\ \boldsymbol{\epsilon}^i \sim \mathcal{N}(\mathbf{0}, \sigma^2 \cdot I_d)$$

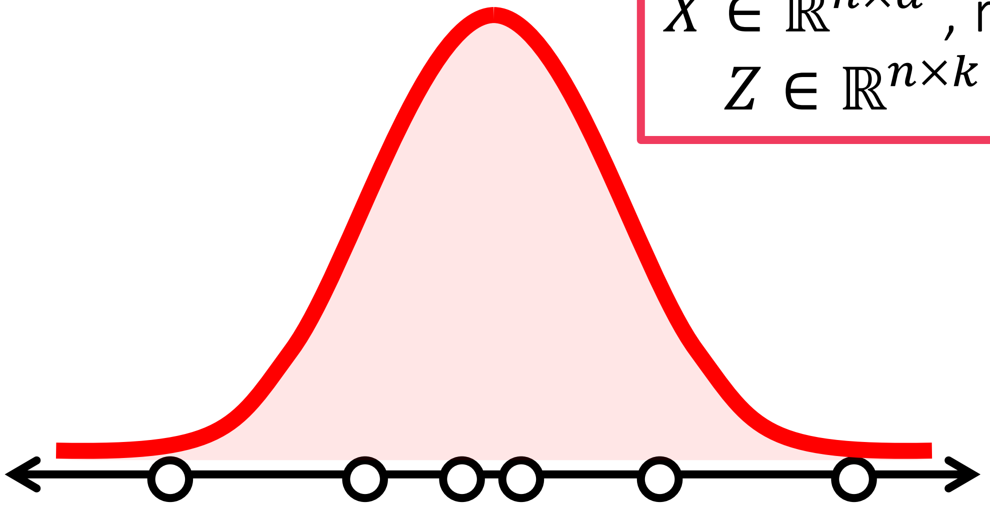
Dictionary/Factor Loading matrix

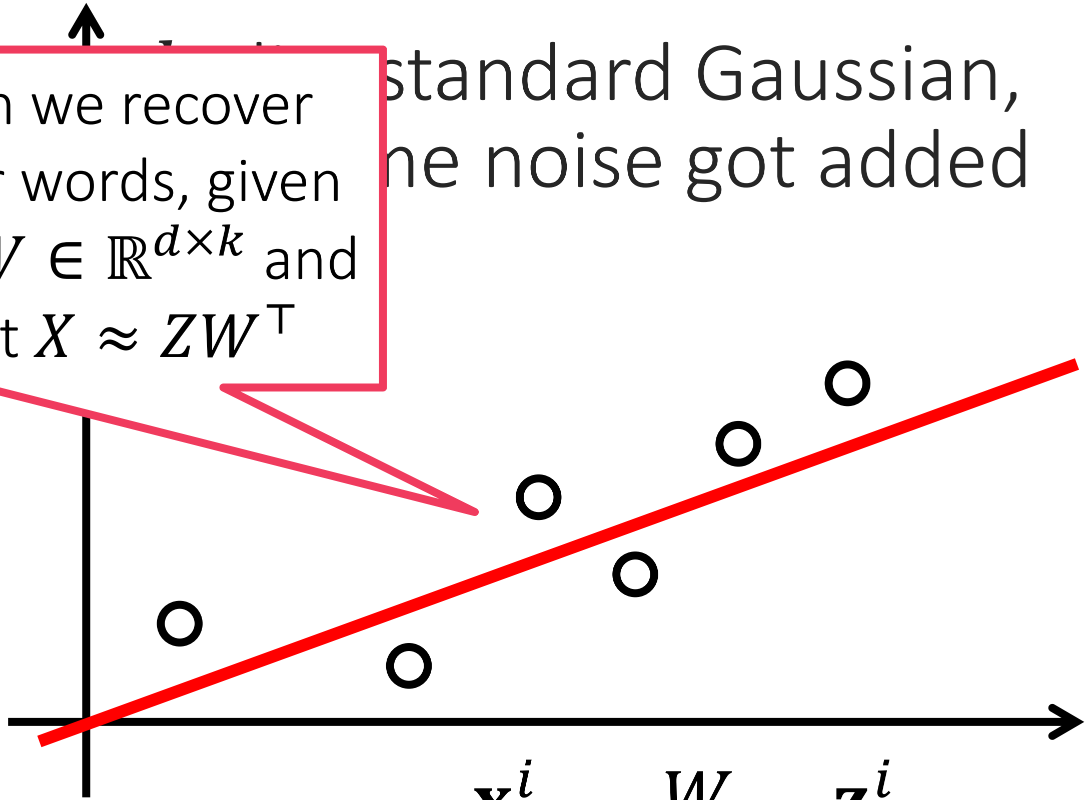
$$W \in \mathbb{R}^{d \times k}$$

Probabilistic PCA

The real data was act standard Gaussian, but got linearly mapp ne noise got added

Given $\mathbf{x}^1, \ldots, \mathbf{x}^n$, can we recover $W, \mathbf{z}^1, \ldots \mathbf{z}^n$? In other words, given $X \in \mathbb{R}^{n\times d}$, recover $W \in \mathbb{R}^{d\times k}$ and $Z \in \mathbb{R}^{n\times k}$ such that $X \approx ZW^\top$

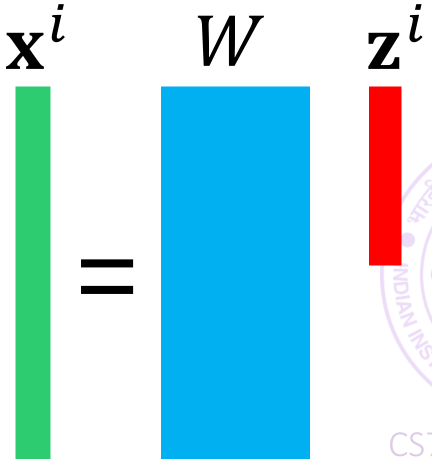




$$\mathbf{z}^i \sim \mathcal{N}(\mathbf{0}, I_k) \qquad \mathbf{x}^i = W\mathbf{z}^i + \boldsymbol{\epsilon}^i, \ \ \boldsymbol{\epsilon}^i \sim \mathcal{N}(\mathbf{0}, \sigma^2 \cdot I_d)$$

Dictionary/Factor Loading matrix

$$W \in \mathbb{R}^{d\times k}$$



Probabilistic PCA [Tipping and Bishop, 1999]

Given samples $X = [\mathbf{x}^1, \ldots, \mathbf{x}^n]^\top \in \mathbb{R}^{n \times d}$, we wish to recover $W, \sigma, \mathbf{z}^i$

__Note__: this is a generative problem, i.e. deals with generation of feature vectors

As discussed before, the original data $\mathbf{z}^i \sim \mathcal{N}(\mathbf{0}, I_k)$ are "latent" – not seen

Also clear from the noise model that $\mathbb{P}\left[\mathbf{x}^i \mid \mathbf{z}^i, \sigma, W\right] = \mathcal{N}\left(\mathbf{x}^i; W\mathbf{z}^i, \sigma^2 \cdot I_d\right)$

More flexible models possible e.g. Factor Analysis – will see later

Will first see how to recover W, σ and then head into recovering $\mathbf{z}^i$

Some mildly painful integrals later we can get

$$\mathbb{P}\left[\mathbf{x}^i \mid \sigma, W\right] = \int_{\mathbf{z} \in \mathbb{R}^k} \mathbb{P}\left[\mathbf{x}^i \mid \mathbf{z}^i, \sigma, W\right] \cdot \mathbb{P}\left[\mathbf{z}^i\right] d\mathbf{z} = \mathcal{N}(\mathbf{0}, \sigma^2 \cdot I + WW^\top)$$

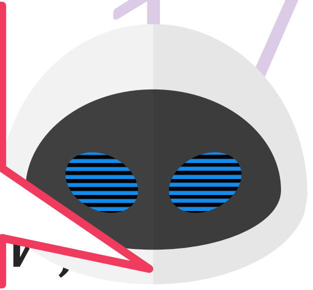
The above also assumes $\mathbb{P}\left[\mathbf{z}^i \mid \sigma, W\right] = \mathbb{P}\left[\mathbf{z}^i\right] = \mathcal{N}\left(\mathbf{z}^i; \mathbf{0}, I_k\right)$

Thus, to simplify life, we can pretend for a moment that our samples X were really generated from $\mathcal{N}(\mathbf{0}, \sigma^2 \cdot I + WW^\top)$ and there are no $\mathbf{z}^i$ in the picture.

Can we now do MLE to find σ, W?

Probabilist

It is very unusual for latent variables to simply integrate out like this leaving behind a nice Gaussian density. We got very lucky here ☺. Usually latent variables mean AltOpt/EM



Given samples X W,

- *Note: this is a generative problem, i.e. deals with generation of feature vectors*
- *As discussed before, the original data* $\mathbf{z}^i \sim \mathcal{N}(\mathbf{0}, I_k)$ *are "latent" – not seen*
- *Also clear from the noise model that* $\mathbb{P}\left[\mathbf{x}^i \mid \mathbf{z}^i, \sigma, W\right] = \mathcal{N}\left(\mathbf{x}^i; W\mathbf{z}^i, \sigma^2 \cdot I_d\right)$
- *More flexible models possible e.g. Factor Analysis – will see later*

Will first see how to recover W, σ and then head into recovering $\mathbf{z}^i$

Some mildly painful integrals later we can get

$$\mathbb{P}\left[\mathbf{x}^i \mid \sigma, W\right] = \int_{\mathbf{z} \in \mathbb{R}^k} \mathbb{P}\left[\mathbf{x}^i \mid \mathbf{z}^i, \sigma, W\right] \cdot \mathbb{P}\left[\mathbf{z}^i\right] d\mathbf{z} = \mathcal{N}\left(\mathbf{0}, \sigma^2 \cdot I + WW^\top\right)$$

- *The above also assumes* $\mathbb{P}\left[\mathbf{z}^i \mid \sigma, W\right] = \mathbb{P}\left[\mathbf{z}^i\right] = \mathcal{N}\left(\mathbf{z}^i; \mathbf{0}, I_k\right)$
- *Thus, to simplify life, we can pretend for a moment that our samples* X *were really generated from* $\mathcal{N}\left(\mathbf{0}, \sigma^2 \cdot I + WW^\top\right)$ *and there are no* $\mathbf{z}^i$ *in the picture.*
- *Can we now do MLE to find* σ, W*?*

An MLE For W, σ in PPCA

We get n samples X from $\mathcal{N}(0, \sigma^2 \cdot I + WW^\top)$. The log-likelihood is

$$\log \mathbb{P}[X \mid \sigma, W] = \frac{n}{2}\left(d \log 2\pi + \log|C| + \mathrm{tr}(C^{-1}S)\right)$$

where $C = WW^\top + \sigma^2 \cdot I_d$ *and* $S = \frac{1}{n} \cdot X^\top X$

Note*: C is always invertible because of* $\sigma^2 \cdot I_d$ *(imp. since* $|WW^\top| = 0$ ☹*)*

Can apply first order optimality to get MLE (painful derivatives though)

Thankfully, end result is very familiar. Let $S = Q\Lambda Q^\top$ *be eigendecomposition of* S *where* $Q = [\mathbf{q}^1, \dots, \mathbf{q}^d]$ *and* $\Lambda = \mathrm{diag}(\lambda_1, \dots, \lambda_d)$ *with* $\lambda_1 \geq \lambda_2 \geq \cdots$

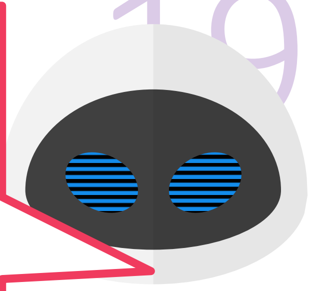
An ED always exists for S since it is square symmetric

$\widehat{W}_{\mathrm{MLE}} = \hat{Q}\sqrt{\widehat{\Lambda} - \hat{\sigma}^2_{\mathrm{MLE}} \cdot I_k}$ *where* $\hat{Q} = [\mathbf{q}^1, \dots, \mathbf{q}^k], \widehat{\Lambda} = \mathrm{diag}(\lambda_1, \dots, \lambda_k)$

$\hat{\sigma}_{\mathrm{MLE}} = \frac{1}{d-k}\sum_{j=k+1}^{d} \lambda_j$

A

If we decide to set $\sigma = 0$ (and not estimate it either) then we get $\widehat{W}_{\text{MLE}} = \hat{Q}\sqrt{\hat{\Lambda}}$ which, apart from the scaling with the eigenvalues, is just $\hat{V}$ in PCA! Thus, PCA ≡ PPCA (apart from a scaling factor) under the noiseless assumption ☺



W

$$\log \mathbb{P}[X \mid \sigma, W] = \frac{n}{2}\left(d \log 2\pi + \log|C| + \operatorname{tr}(C^{-1}S)\right)$$

where $C = WW^\top + \sigma^2 \cdot I_d$ *and* $S = \frac{1}{n} \cdot X^\top X$

Note*: C is always invertible because of* $\sigma^2 \cdot I_d$ *(imp. since* $|WW^\top| = 0$ ☹*)*

Can apply first order optimality to get MLE (painful derivatives though)

Thankfully, end result is very familiar. Let $S = Q\Lambda Q^\top$ *be eigendecomposition of* S *where* $Q = [\mathbf{q}^1, \dots, \mathbf{q}^d]$ *and* $\Lambda = \operatorname{diag}(\lambda_1, \dots, \lambda_d)$ *with* $\lambda_1 \geq \lambda_2 \geq \cdots$

An ED always exists for S since it is square symmetric

$\widehat{W}_{\text{MLE}} = \hat{Q}\sqrt{\hat{\Lambda} - \hat{\sigma}^2_{\text{MLE}} \cdot I_k}$ *where* $\hat{Q} = [\mathbf{q}^1, \dots, \mathbf{q}^k], \hat{\Lambda} = \operatorname{diag}(\lambda_1, \dots, \lambda_k)$

$\hat{\sigma}_{\text{MLE}} = \frac{1}{d-k}\sum_{j=k+1}^{d} \lambda_j$

PPCA Variants

- We can tweak several moving parts in the PPCA generative process
  - *Can instead assume that* $\mathbf{z}^i \sim \mathcal{N}(\boldsymbol{\mu}_z, \Sigma_z)$ *and estimate* $\boldsymbol{\mu}_z, \in \mathbb{R}^k, \Sigma_z \in \mathbb{R}^{k\times k}$
  - *Can assume non-spherical noise* $\boldsymbol{\epsilon}^i \sim \mathcal{N}(\mathbf{0}, \Sigma_\epsilon)$ *and estimate* $\Sigma_\epsilon \in \mathbb{R}^{d\times d}$
  - *A technique called* Factor Analysis *actually uses non-spherical noise model*
- Since PPCA is a generative model, it can model missing data too
  - *Suppose we have already found out* $\hat{\sigma}_{\text{MLE}}, \widehat{W}_{\text{MLE}}$ *using clean training data*
  - *If test data has missing features* $\mathbf{x}^t = [\mathbf{x}_o^t, \mathbf{x}_m^t] \in \mathbb{R}^d$*, use fact that marginals of Gaussians are Gaussian. Since we know* $\mathbf{x}^t \sim \mathcal{N}\left(\mathbf{0}, \hat{\sigma}^2 \cdot I + \widehat{W}\widehat{W}^\top\right)$*, we can see* $\mathbb{P}[\mathbf{x}_o^t] = \mathcal{N}\left(\mathbf{0}, \hat{\sigma}^2 \cdot I_o + \widehat{W}_o\widehat{W}_o^\top\right)$ *(*$\widehat{W}_o \in \mathbb{R}^{|o|\times k}$ *has only observed rows)*
  - *Missing test data is easier to handle, missing training data more challenging*
    - Need to apply the above trick when defining the likelihood of training data points
    - Each training data point may have different coordinates missing. If we do not even know when a coordinate has gone missing then this becomes more challenging ☹

Dimensionality Reduction using PPCA

- With PCA $X \approx \hat{X} = \hat{U}\hat{\Sigma}\hat{V}^\top$ we got low-dim feat. easily $\tilde{X} = \hat{U}\hat{\Sigma} \in \mathbb{R}^{n \times k}$
  - *These made sense because these are k-dim features which are just a rotation away (using V) from features $\hat{X}$ which we know approximate X very well*
  - *With PPCA too we can we recover the original low-dim features $\mathbf{z}^i \in \mathbb{R}^k$ by treating them as latent variables and applying AltOpt or EM*
  - *Need to be careful since these latent variables are (continuous) vectors now!*
- Earlier, we used a shortcut $\mathbb{P}\left[\mathbf{x}^i \mid \sigma, W\right] = \mathcal{N}(\mathbf{0}, \sigma^2 \cdot I + WW^\top)$ to get MLE for W, σ. To get hold of $\mathbf{z}^i$ we need proper AltOpt/EM

$$\mathbb{P}\left[\mathbf{x}^i \mid \sigma, W\right] = \int_{\mathbb{R}^k} \mathbb{P}\left[\mathbf{x}^i, \mathbf{z}^i \mid \sigma, W\right] \, d\mathbf{z}^i$$

  - *AltOpt will approximate integral using a single term (a single value for $\mathbf{z}^i$)*
  - *EM will lower bound the integral using another (easier to compute) integral*
  - *Need to replace "sum" over possible values of $\mathbf{z}^i$ with "integral" since $\mathbf{z}^i$ cont.*

PPCA – Alternating Optimization

We wish to get $\arg\max_{W,\sigma} \sum_{i=1}^{n} \ln\left(\int_{\mathbb{R}^k} \mathbb{P}\left[\mathbf{x}^i, \mathbf{z}^i \mid \sigma, W\right] \; d\mathbf{z}^i\right)$

Same old pesky sum-log-sum (actually sum-log-integral) form – difficult ☹

Approximate integral by its most dominant term $\int_{\mathbb{R}^k} \mathbb{P}\left[\mathbf{x}^i, \mathbf{z}^i \mid \sigma, W\right] d\mathbf{z}^i \approx \mathbb{P}\left[\mathbf{x}^i, \tilde{\mathbf{z}}^i \mid \sigma, W\right]$ *with* $\tilde{\mathbf{z}}^i = \arg\max_{\mathbf{z}} \mathbb{P}\left[\mathbf{x}^i, \mathbf{z} \mid \sigma, W\right] = \arg\max_{\mathbf{z}} \mathbb{P}\left[\mathbf{z} \mid \mathbf{x}^i, \sigma, W\right]$

Approx. integral by a single term may not be bad if the dist. $\mathbb{P}\left[\mathbf{z} \mid \mathbf{x}^i, \sigma, W\right]$ *has small variance – advantage of being cheaper than EM in computation time*

Thus, we wish to solve $\arg\max_{W,\sigma,\mathbf{z}^i} \sum_{i=1}^{n} \ln \mathbb{P}\left[\mathbf{x}^i, \mathbf{z}^i \mid \sigma, W\right]$

$\mathbb{P}\left[\mathbf{z}^i \mid \mathbf{x}^i, \sigma, W\right] = \mathcal{N}\left(\mathbf{z}^i; \boldsymbol{\mu}_{\mathbf{z}}^i, \Sigma_{\mathbf{z}}\right)$ *where* $\boldsymbol{\mu}_{\mathbf{z}}^i = (W^\top W + \sigma^2 \cdot I_k)^{-1} W^\top \mathbf{x}^i \in \mathbb{R}^k$ *and* $\Sigma_z = \sigma^2 \cdot (W^\top W + \sigma^2 \cdot I_k)^{-1} \in \mathbb{R}^{k \times k}$

Since for Gaussians, mode is mean, we have $\arg\max_{\mathbf{z}} \mathbb{P}\left[\mathbf{z} \mid \mathbf{x}^i, \sigma, W\right] = \boldsymbol{\mu}_{\mathbf{z}}^i$

This gives us one of the alternating updates, lets derive the other

PPCA – A

These derivations are routine but tedious. You can check [BIS] Chapter 12 for details. **Warning**: equation 12.42 in that book has an error. The correct expression is given in this slide

We wish to get arg max $\sum_{i=1}$ ln(

…e o… …fficult ☹

…ox… …$W]$

…, $\tilde{\mathbf{z}}^i$ … $| \mathbf{x}^i$

Because this the mean of $\mathbf{z}^i$ conditioned on $\mathbf{x}^i$. It is the marginal (unconditional) mean of $\mathbf{z}^i$ that is $\mathbf{0}$. Suppose we know that $\mathbf{x}^i = W\mathbf{z}^i + \boldsymbol{\epsilon}^i$ is a vector far off from the origin. The it is likely that $W\mathbf{z}^i \neq \mathbf{0}$ as well which immediately tells that $\mathbf{z}^i \neq \mathbf{0}$

We know that $\mathbf{z}^i \sim \mathcal{N}(\mathbf{z}^i; \mathbf{0}, I_k)$. Then how come $\boldsymbol{\mu}_{\mathbf{z}}^i \neq \mathbf{0}$?

Approx. integ… …*s*

small variance – advantage of being cheaper than EM in computation time

…we

…$\mathbb{R}^k$

$\Sigma_{\mathbf{z}}$

Because of some beautiful coincidences: 1) it turns out that $\mathbb{P}[\mathbf{z}^i \mid \mathbf{x}^i, \sigma, W]$ is a Gaussian, 2) for Gaussians, mean is mode which means that 3) $\boldsymbol{\mu}_{\mathbf{z}}^i$ is the MLE solution to the problem $\mathbf{x}^i = W\mathbf{z}^i + \boldsymbol{\epsilon}^i$ which is indeed a vector least squares regression problem

Why does the expression $\boldsymbol{\mu}_{\mathbf{z}}^i = (W^\top W + \sigma^2 \cdot I_k)^{-1} W^\top \mathbf{x}^i$ look like least squares?

This gives us one of the alternating updates, lets derive the other

PPCA – Alternating Optimization

$$\sum_{i=1}^n \ln \mathbb{P}\left[\mathbf{x}^i, \mathbf{z}^i \mid \sigma, W\right] = \frac{1}{2\sigma^2}\sum_{i=1}^n \left\|\mathbf{x}^i - W\mathbf{z}^i\right\|_2^2 + dn \ln \sigma$$

Thus, if $\mathbf{z}^i$ are fixed, we can obtain W, σ using first order optimality

$$\arg\min_W \sum_{i=1}^n \left\|\mathbf{x}^i - W\mathbf{z}^i\right\|_2^2 = \arg\min_W \operatorname{tr}(W^\top W A) - 2\operatorname{tr}(W^\top B)$$

where $A = \sum_{i=1}^n \mathbf{z}^i\left(\mathbf{z}^i\right)^\top$ and $B = \sum_{i=1}^n \mathbf{x}^i\left(\mathbf{z}^i\right)^\top$

*See [**BIS**] Chap 12 for detailed derivations - check that dimensionalities match*

Apply first order optimality to get $\widehat{W}_{\mathrm{MLE}} = BA^{-1}$

Once W is known, σ can also be found using first order optimality

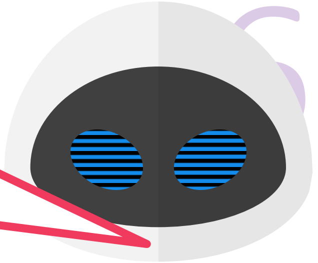
$$\hat{\sigma}_{\mathrm{MLE}} = \sqrt{\frac{1}{dn}\sum_{i=1}^n \left\|\mathbf{x}^i - \widehat{W}_{\mathrm{MLE}}\mathbf{z}^i\right\|_2^2}$$

PPCA – Alternating Optimization

ALTERNATING OPTIMIZATION

```
1. Initialize $W$
2. For $t = 0, 1, 2, \ldots$
   1. Update $\mathbf{z}^i$ fixing $W, \sigma$
      1. Let $\mathbf{z}^i = (W^\top W + \sigma^2 \cdot I_k)^{-1} W^\top \mathbf{x}^i$, for $i \in [n]$
   2. Update $W, \sigma$ fixing $\mathbf{z}^i$
      1. Calculate $A = \sum_{i=1}^n \mathbf{z}^i (\mathbf{z}^i)^\top$ and $B = \sum_{i=1}^n \mathbf{x}^i (\mathbf{z}^i)^\top$
      2. Update $W = BA^{-1}$
      3. Calculate $V = \sum_{i=1}^n \left\| \mathbf{x}^i - W\mathbf{z}^i \right\|_2^2$
      4. Update $\sigma = \sqrt{V/dn}$
```

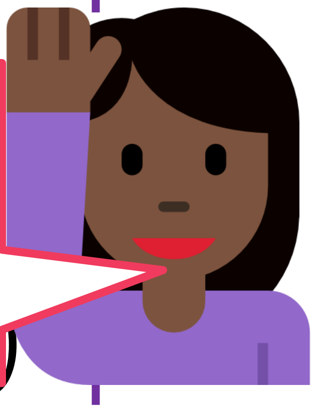
PPCA



Total of $\mathcal{O}(k^2(d+n)+kdn)$ time taken per iteration. In contrast, PCA using power + peeling method takes only $\mathcal{O}(kdn)$ time if we do power updates as $A\mathbf{v} = X^\top(X\mathbf{v})$ which takes $\mathcal{O}(dn)$ time per power step

```
1. Initialize $W$
2. For $t = 0, 1, 2, \ldots$
    1. Update $\mathbf{z}^i$ fixing $W, \sigma$
        1. Let $\mathbf{z}^i = (W^\top W + \sigma^2$
    2. Update $W, \sigma$ fixing $\mathbf{z}^i$
        1. Calculate $A = \sum_{i=1}^n \mathbf{z}^i($
        2. Update $W = BA^{-1}$
        3. Calculate $V = \sum_{i=1}^n \|\mathbf{x}^i - W$
        4. Update $\sigma = \sqrt{V/dn}$
```

$\mathcal{O}(k^3)$ time to calculate the inverse term and $\mathcal{O}(kdn + k^2d)$ time to calculate all $\mathbf{z}^i$ terms afterward



In practice, PCA is usually faster than AltOpt PPCA – no inverses required to solve PCA, just simple mat-vec multiplication steps ☺

$\mathcal{O}(k^2n + k^3)$ time for A^{-1}, $\mathcal{O}(dkn)$ time for B and finally $\mathcal{O}(k^2d)$ time for W

PPCA – Expectation Maximization

We wish to get $\arg\max_{W,\sigma} \sum_{i=1}^{n} \ln\left(\int_{\mathbb{R}^k} \mathbb{P}\left[\mathbf{x}^i, \mathbf{z}^i \mid \sigma, W\right]\, d\mathbf{z}^i\right)$

As before, given a model $\widehat{W}, \hat{\sigma}$, let $q\left(\mathbf{z}^i\right) = \mathbb{P}\left[\mathbf{z}^i \mid \mathbf{x}^i, \hat{\sigma}, \widehat{W}\right]$ and lower bound the integral using Jensen's inequality (the term C does not depend on W, σ)

$$\ln\left(\int_{\mathbb{R}^k} \mathbb{P}\left[\mathbf{x}^i, \mathbf{z}^i \mid \sigma, W\right]\, d\mathbf{z}^i\right) \geq \int_{\mathbb{R}^k} q\left(\mathbf{z}^i\right) \cdot \ln \mathbb{P}\left[\mathbf{x}^i, \mathbf{z}^i \mid \sigma, W\right]\, d\mathbf{z}^i + C$$

Some simple (but non-trivial) calculations show that the resulting EM algorithm looks very similar to the AltOpt with a few simple changes

Replace $\mathbf{z}^i$ *with* $\boldsymbol{\zeta}^i \triangleq \mathbb{E}\left[\mathbf{z}^i\right] = \int_{\mathbb{R}^k} q\left(\mathbf{z}^i\right) \cdot \mathbf{z}^i\, d\mathbf{z}^i = \left(W^\top W + \sigma^2 \cdot I_k\right)^{-1} W^\top \mathbf{x}^i$

This is the same as we used in AltOpt since for Gaussian, mode is the same as mean

Replace $\mathbf{z}^i\left(\mathbf{z}^i\right)^\top$ *with* $Z^i \triangleq \mathbb{E}\left[\mathbf{z}^i\left(\mathbf{z}^i\right)^\top\right] = \int_{\mathbb{R}^k} q\left(\mathbf{z}^i\right) \cdot \mathbf{z}^i\left(\mathbf{z}^i\right)^\top\, d\mathbf{z}^i$

$$= \boldsymbol{\zeta}^i\left(\boldsymbol{\zeta}^i\right)^\top + \sigma^2 \cdot \left(W^\top W + \sigma^2 \cdot I_k\right)^{-1}$$

*Rest of the algorithm (can be shown to) remain the same (see [**BIS**] Chap 12)*

PPCA – Expectation Maximization

EXPECTATION MAXIMIZATION

```
1. Initialize $W$
2. For $t = 0, 1, 2, \ldots$
    1. Update $\boldsymbol{\zeta}^i, Z^i$ fixing $W, \sigma$
        1. Let $\boldsymbol{\zeta}^i = (W^\top W + \sigma^2 \cdot I_k)^{-1} W^\top \mathbf{x}^i$, for $i \in [n]$
        2. Let $Z^i = \boldsymbol{\zeta}^i \left(\boldsymbol{\zeta}^i\right)^\top + \sigma^2 \cdot (W^\top W + \sigma^2 \cdot I_k)^{-1}$
    2. Update $W, \sigma$ fixing $\boldsymbol{\zeta}^i, Z^i$
        1. Update $W = \left(\sum_{i=1}^n \mathbf{x}^i \left(\boldsymbol{\zeta}^i\right)^\top\right) \left(\sum_{i=1}^n Z^i\right)^{-1}$
        2. Calculate $V = \sum_{i=1}^n \left\|\mathbf{x}^i - W\boldsymbol{\zeta}^i\right\|_2^2$
        3. Update $\sigma = \sqrt{V/dn}$
```

PPCA – Expectation Maxi

Time complexity of PPCA using EM roughly same as that of PPCA using AltOpt.

EXPECTATION MAXIMIZATION

```
1. Initialize $W$
2. For $t = 0, 1, 2, \ldots$
    1. Update $\boldsymbol{\zeta}^i, Z^i$ fixing $W, \sigma$
        1. Let $\boldsymbol{\zeta}^i = (W^\top W + \sigma^2 \cdot I_k)^{-1} W^\top \mathbf{x}^i$, for $i \in [n]$
        2. Let $Z^i = \boldsymbol{\zeta}^i (\boldsymbol{\zeta}^i)^\top + \sigma^2 \cdot (W^\top W + \sigma^2 \cdot I_k)^{-1}$
    2. Update $W, \sigma$ fixing $\boldsymbol{\zeta}^i, Z^i$
        1. Update $W = \left( \sum_{i=1}^n \mathbf{x}^i (\boldsymbol{\zeta}^i)^\top \right) \left( \sum_{i=1}^n Z^i \right)^{-1}$
        2. Calculate $V = \sum_{i=1}^n \left\| \mathbf{x}^i - W \boldsymbol{\zeta}^i \right\|_2^2$
        3. Update $\sigma = \sqrt{V / dn}$
```

Wrapping Up Linear Models

Have studied how linear models can be used to do classification (binary/multi), regression, clustering, dimensionality reduction

We looked at several techniques to do so

- ***Function Approximation Method****: define a loss function over the output (and optionally a regularizer over model parameters) and minimize (regularized) loss to learn a good model*
- ***Probabilistic Method****: define a likelihood over the output (and optionally a prior over model parameters) and use MLE/MAP to learn a good model*
- ***Bayesian Method****: use likelihood and prior to learn a posterior distribution over all (possibly infinite) models. At test time, aggregate responses from each one of these models, weighted by their posterior prob. to make final prediction*
- *Probabilistic and Bayesian methods usually come with a "generative story" specified by likelihood+prior dists. of how we assume data was generated*

Non Linear Models

Agenda for the next few weeks (after mid-sem recess)

- Kernel Learning
- Deep Learning

Let us take half-a-step towards learning non-linear models today

Generalized Linear Models (GLM/GLiM)

- Linear regression approximates label y^i as a linear func of features $\mathbf{x}^i$
  - *… and we try to learn a model* $\mathbf{w}$ *such that* $y^i \approx \mathbf{w}^\top \mathbf{x}^i$
  - *Thus, output is modelled as a linear function of features – not very flexible or powerful. Will learn techniques that model output as non-linear func of input*
- Generalized linear models are “somewhat” linear models
  - *In fact, we have already seen GLMs in action – logistic reg, linear reg*
  - *In logistic regression, the label* $y^i \in \{0,1\}$ *(or* $y^i \in \{-1,1\}$*). However,* $\mathbf{w}^\top \mathbf{x}^i$ *doesn’t give us* y^i *directly, rather it gives us* $\mathbb{P}\left[y^i = 1\right]$*. We pass* $\mathbf{w}^\top \mathbf{x}^i$ *through a (non-linear) thresholding function to obtain the final (binary) label*
  - *GLMs generalize this behaviour very nicely to allow us to make predictions on discrete/continuous labels even when* $\mathbf{w}^\top \mathbf{x}^i$ *does not directly give us* y^i
  - *Thus, GLMs are only “just” non-linear* ☺ – they learn linear scores but then threshold (or apply other non-linear wrappers) to predict interesting labels

Generalized Linear Models

GLMs assume likelihood models of a special kind (usually $\eta = \mathbf{w}^\top \mathbf{x}$)

$$\mathbb{P}[y \mid \eta] \propto \exp(y \cdot \eta - A(\eta) - h(y))$$

Called **canonical exponential family distributions** *(details in CS772 etc)*

The normalization "constant" is sometimes absorbed into $A(\cdot)$

Sometimes, we include a "dispersion" parameter to control variance i.e.

$$\mathbb{P}[y \mid \eta, \phi] \propto \exp\big((y \cdot \eta - A(\eta) - h(y))/\phi\big)$$

Example: Gaussian ($A(\eta) = \eta^2/2, h(y) = y^2/2, \eta = \mathbf{w}^\top \mathbf{x}$)

$$\mathbb{P}[y \mid \mathbf{x}, \mathbf{w}, \sigma^2] \propto \exp\left(\left(y \cdot \mathbf{w}^\top \mathbf{x} - \frac{(\mathbf{w}^\top \mathbf{x})^2}{2} - \frac{y^2}{2}\right)/\sigma^2\right) \propto \exp\left(-\frac{(y - \mathbf{w}^\top \mathbf{x})^2}{2\sigma^2}\right)$$

Example: Bernoulli ($A(\eta) = \ln(1 + \exp(\eta))\,, h(y) = 0, \eta = \mathbf{w}^\top \mathbf{x}$)

$$\mathbb{P}[y = 1 \mid \mathbf{x}, \mathbf{w}] = \exp(\mathbf{w}^\top \mathbf{x} - \ln(1 + \exp(\mathbf{w}^\top \mathbf{x}))) = 1/(1 + \exp(-\mathbf{w}^\top \mathbf{x}))$$

$$\mathbb{P}[y = 0 \mid \mathbf{x}, \mathbf{w}] = \exp(-\ln(1 + \exp(\mathbf{w}^\top \mathbf{x}))) = 1/(1 + \exp(\mathbf{w}^\top \mathbf{x}))$$

Generalized Linear Models

Example: Poisson ($A(\eta) = \exp(\eta)\,, h(y) = \ln(y!)\,, \eta = \mathbf{w}^\top\mathbf{x}$)

The Poisson distribution can model labels that take values in $\mathbb{N}$. Typically used if we want to predict counts e.g. number of students graduating etc

$\mathbb{P}[y = k \mid \mathbf{x}, \mathbf{w}] = \frac{1}{k!}\exp(k \cdot \mathbf{w}^\top\mathbf{x} - \exp(\mathbf{w}^\top\mathbf{x})) = \frac{\lambda^k e^{-\lambda}}{k!}$ *with* $\lambda = \exp(\mathbf{w}^\top\mathbf{x})$

Example: Gamma ($A(\eta) = \ln(-\eta)\,, h(y) = (1 - \phi^{-1})\ln(y)\,, \eta = -\exp(\mathbf{w}^\top\mathbf{x})$)

Almost like a continuous Poisson – can model labels that take only non-negative values e.g. time before patient visits again

Note: $\eta = -\exp(\mathbf{w}^\top\mathbf{x})$ *ensures* $\eta < 0$ *o/w gamma dist doesn't make sense*

$\mathbb{P}[y \mid \mathbf{x}, \mathbf{w}, \phi] \propto \exp\big((-y \cdot \exp(\mathbf{w}^\top\mathbf{x}) + \mathbf{w}^\top\mathbf{x} + (1 - \phi)\ln y)/\phi\big)$

Example: von-Mises likelihood $\mathbb{P}[y \mid \mathbf{x}, \mathbf{w}, \kappa] \propto \exp(\kappa\cos(y - \mathbf{w}^\top\mathbf{x}))$

A Gaussian on a circle/finite interval – usually used to predict angles $\in [0,2\pi]$

A "non-canonical" exponential family member – form of expression different

MLE with GLMs

Mostly, we have $\mathbb{P}[y \mid \mathbf{x}, \mathbf{w}] \propto \exp(y \cdot \mathbf{w}^\top \mathbf{x} - A(\mathbf{w}^\top \mathbf{x}) - h(y))$

Interesting fact: *always have the mean predicted label* $\mathbb{E}[y \mid \mathbf{x}, \mathbf{w}] = A'(\eta)$

Gaussian: $A'(\eta) = \eta = \mathbf{w}^\top \mathbf{x}$

Bernoulli: $A'(\eta) = 1/(1 + \exp(-\eta)) = \sigma(-\mathbf{w}^\top \mathbf{x})$ *($\sigma \equiv$ sigmoid function)*

Poisson: $A'(\eta) = \exp(\eta) = \exp(\mathbf{w}^\top \mathbf{x})$

Gamma: $A'(\eta) = -1/\eta = \exp(-\mathbf{w}^\top \mathbf{x})$ *(recall in this case $\eta = -\exp(\mathbf{w}^\top \mathbf{x})$)*

Solving for MLE becomes simple (C does not depend on $\mathbf{w}$) via (S)GD

$\mathcal{L}(\mathbf{w}) = -\ln \mathbb{P}[\mathbf{y} \mid X, \mathbf{w}] = \sum_{i=1}^n A\left(\mathbf{w}^\top \mathbf{x}^i\right) - y^i \cdot \mathbf{w}^\top \mathbf{x}^i + C$

Thus, we have $\nabla \mathcal{L}(\mathbf{w}) = \sum_{i=1}^n \left(A'\left(\mathbf{w}^\top \mathbf{x}^i\right) - y^i\right) \cdot \mathbf{x}^i$

Note that if for some point $A'\left(\mathbf{w}^\top \mathbf{x}^i\right) = y^i$ then that point does not contribute to the gradient. One way to get zero gradient (which will be a global optimum since $\mathcal{L}(\mathbf{w})$ is mostly convex) is to ensure $A'\left(\mathbf{w}^\top \mathbf{x}^i\right) = y^i$ for all points ☺

MLE with GLMs

Mostly, we have $\mathbb{P}[y \mid \mathbf{x}, \mathbf{w}] \propto \exp(y \cdot \mathbf{w}^\top \mathbf{x} - A(\mathbf{w}^\top \mathbf{x}) - h(y))$

Interesting fact: *always have the mean predicted label* $\mathbb{E}[y \mid \mathbf{x}, \mathbf{w}] = A'(\eta)$

Gaussian: $A'(\eta) = \eta = \mathbf{w}^\top \mathbf{x}$

Bernoulli: $A'(\eta) = 1/(1 + \exp(-\eta$

Poisson: $A'(\eta) = \exp(\eta) = \exp(\mathbf{w}$

Gamma: $A'(\eta) = -1/\eta = \exp($

Slightly different (but similar) updates required for gamma since we do not have $\eta = \mathbf{w}^\top \mathbf{x}$ but rather $\eta = -\exp(\mathbf{w}^\top \mathbf{x})$ in that case

Solving for MLE becomes simple (C does not depend on $\mathbf{w}$) via (S)GD

$\mathcal{L}(\mathbf{w}) = -\ln \mathbb{P}[\mathbf{y} \mid X, \mathbf{w}] = \sum_{i=1}^n A(\mathbf{w}^\top \mathbf{x}^i) - y^i \cdot \mathbf{w}^\top \mathbf{x}^i + C$

Thus, we have $\nabla \mathcal{L}(\mathbf{w}) = \sum_{i=1}^n (A'(\mathbf{w}^\top \mathbf{x}^i) - y^i) \cdot \mathbf{x}^i$

Note that if for some point $A'(\mathbf{w}^\top \mathbf{x}^i) = y^i$ *then that point does not contribute to the gradient. One way to get zero gradient (which will be a global optimum since* $\mathcal{L}(\mathbf{w})$ *is mostly convex) is to ensure* $A'(\mathbf{w}^\top \mathbf{x}^i) = y^i$ *for all points* ☺