CPSC 540 Assignment 5

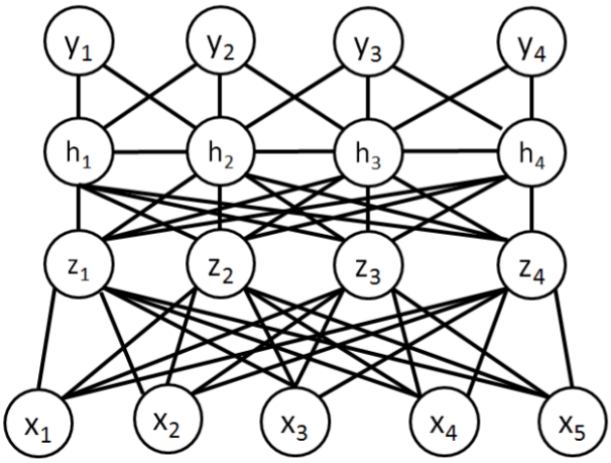
The assignment instructions are the same as for the previous assignment.

1. Name(s): Cody Griffith, Ziming Yin, Tim Jaschek
2. Student ID(s): Cody: 88416169 , Ziming: 88489166 , Tim: 91220160.

1 Undirected Graphical Models

1.1 Conditional UGM

Consider modeling the dependencies between sets of binary variables x_j and y_j with the following UGM which is a variation on a stacked RBM:



Computing univariate marginals in this model will be NP-hard in general, but the graph structure allows efficient block updates by conditioning on suitable subsets of the variables (this could be useful for designing approximate inference methods). For each of the conditioning scenarios below, draw the conditional UGM and informally comment on how expensive it would be to compute univariate marginals (for all variables) in the conditional UGM.

1. Conditioning on all the x and h values.

Answer: x and h block path, now everything is disconnected. Therefore computing univariate marginals is P-hard and trivial.

y_1 y_2 y_3 y_4

z_1 z_2 z_3 z_4

2. Conditioning on all the z and y values.

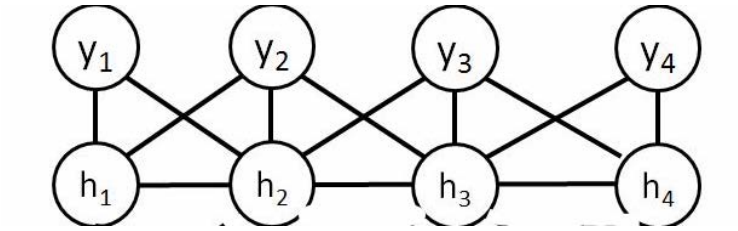
   Answer: Even though x and h are seperated, but h itself forms a chain. Thus computing univariate marginalss is cheaper than before

   h_1—h_2—h_3—h_4

   x_1 x_2 x_3 x_4

3. Conditioning on all the x and z values.

   Answer: Variables have two parents so it makes computation cheap.



1.2 Fitting a UGM to PINs

The function *example_UGM.jl* loads a dataset X containing samples of PIN numbers, based on the probabilities from the article at this URL: `http://www.datagenetics.com/blog/september32012`.[1]

This function fits a UGM model to the dataset, where all node/edge parameters are untied and the graph is empty. It then performs decoding/inference/sampling in the fitted model. The decoding is reasonable (it's $x = \begin{bmatrix}1 & 2 & 3 & 4\end{bmatrix}$) and the univariate marginals are reasonable (it says the first number is 1 approximately 40% of the time and the last number is 4 approximately 20% of the time), but because it assumes the variables are independent we can see that this is not a very good model:

1. The sampler doesn't tend to generate the decoding ($x = \begin{bmatrix}1 & 2 & 3 & 4\end{bmatrix}$) as often as we would expect. Since it happens in more than 1/10 of the training examples, we should be seeing it in more than 1/10 of the samples.

2. Conditioned on the first three numbers being 1 2 3, the probability that the last number is 4 is only around 20%, whereas in the data it's more than 90% in this scenario.

In this question, you'll explore using (non-degenerate UGMs) to try to fix the above issues:

[1] I got the probabilities from the reverse-engineered heatmap here: `http://jemore.free.fr/wordpress/?p=73`.

1. Write an equation for $p(x_1, x_2, x_3, x_4)$ in terms of the parameters w being used by the code.

   Answer: Here we would use

$$p(x_1, x_2, x_3, x_4) = \frac{1}{Z} \exp\left(\sum_{i=1}^{4} w_{i,x_i}\right).$$

2. How would the answer to the previous question change (in terms of w and v) if we use `E = [1 2]`?

   Answer: Now we have

$$p(x_1, x_2, x_3, x_4) = \frac{1}{Z} \exp\left(\sum_{i=1}^{4} w_{i,x_i} + v_{x_1,x_2,1}\right).$$

3. Modify the demo to use chain-structured dependency. Comment on whether this fixes each of the above 2 issues.

   Answer: About the first problem, it is better. Now we can get about 0—$\frac{1}{10}$ of `[1 2 3 4]`.
   About the second problem, it is getting better, but still less than 90%, now it is about 53%.

4. Modify the demo to use a completely-connected graph. Comment on whether this fixes each of the above 2 issues.

   Answer: About the first problem, it is better than chain-structured. Now we can get about $\frac{1}{20}$—$\frac{1}{10}$ of `[1 2 3 4]`.
   About the second problem, it is also getting better, now it is over 70%.

5. What would the effect of higher-order potentials be? What would the disdavantages of higher-order potentials be?

Answer: We will get closer to the true value, but the model easily overfit.

If you want to further explore UGMs, there are quite a few Matlab demos on the UGM webpage (`https://www.cs.ubc.ca/~schmidtm/Software/UGM.html`) that you can go through which cover all sorts of things like approximate inference and CRFs.

2 Bayesian Inference

2.1 Conjugate Priors

Consider a $y \in \{1, 2, 3\}$ following a multinoulli distribution with parameters $\theta = \{\theta_1, \theta_2, \theta_3\}$,

$$y \mid \theta \sim \text{Mult}(\theta_1, \theta_2, \theta_3).$$

We'll assume that θ follows a Dirichlet distribution (the conjugate prior to the multinoulli) with parameters $\alpha = \{\alpha_1, \alpha_2, \alpha_3\}$,

$$\theta \sim \mathcal{D}(\alpha_1, \alpha_2, \alpha_3).$$

Thus we have

$$p(y \mid \theta, \alpha) = p(y \mid \theta) = \theta_1^{I(y=1)}\theta_2^{I(y=2)}\theta_3^{I(y=3)}, \quad p(\theta \mid \alpha) = \frac{\Gamma(\alpha_1 + \alpha_2 + \alpha_3)}{\Gamma(\alpha_1)\Gamma(\alpha_2)\Gamma(\alpha_3)}\theta_1^{\alpha_1 - 1}\theta_2^{\alpha_2 - 1}\theta_3^{\alpha_3 - 1}.$$

Compute the following quantites:

1. The posterior distribution,

$$p(\theta \mid y, \alpha).$$

Answer: The Dirichlet distribution is a conjugate prior to the multinoulli. This means that if the prior distribution of the multinoulli parameters is Dirichlet then the posterior distribution is also a Dirichlet distribution.

$$\begin{aligned} p(\theta \mid y = i, \alpha) &\propto p(y = i \mid \theta, \alpha)p(\theta \mid \alpha) \\ &= \theta_1^{I(y=1)}\theta_2^{I(y=2)}\theta_3^{I(y=3)} \frac{\Gamma(\alpha_1 + \alpha_2 + \alpha_3)}{\Gamma(\alpha_1)\Gamma(\alpha_2)\Gamma(\alpha_3)}\theta_1^{\alpha_1 - 1}\theta_2^{\alpha_2 - 1}\theta_3^{\alpha_3 - 1} \\ &= \frac{1}{D(\alpha)}\theta_1^{I(y=1)}\theta_2^{I(y=2)}\theta_3^{I(y=3)}\theta_1^{\alpha_1 - 1}\theta_2^{\alpha_2 - 1}\theta_3^{\alpha_3 - 1} \\ &= \frac{1}{D(\alpha)}\theta_1^{\beta_1 - 1}\theta_2^{\beta_2 - 1}\theta_3^{\beta_3 - 1} \\ &\propto \frac{1}{D(\alpha^+)}\theta_1^{\beta_1 - 1}\theta_2^{\beta_2 - 1}\theta_3^{\beta_3 - 1} \end{aligned}$$

where $\beta_j = I(y = j) + \alpha_j$ and $\theta \mid y, \alpha \sim \mathcal{D}(\beta_1, \beta_2, \beta_3)$.

It is worthwhile to note that there is only one parameter that is ever different from previous parameters, $\beta_i = \alpha_i + 1$ if $y = i$, else $\beta_i = \alpha_i$, and that

$$D(\alpha^+) = \frac{\alpha_i}{\alpha_1 + \alpha_2 + \alpha_3} D(\alpha).$$

It will also help in the future to denote this pdf as

$$p(\theta \mid y = i, \alpha) = \frac{1}{D(\alpha^+)}\theta_i^{\alpha_i}\theta_{-i}^{\alpha_{-i} - 1}$$

which says $\theta \mid y = i, \alpha \sim \mathcal{D}(\alpha_i + 1, \alpha_{-i})$.

2. The marginal likelihood of y given the hyper-parameters α,

$$p(y \mid \alpha) = \int p(y, \theta \mid \alpha) d\theta,$$

Answer: To start, we notice that given both α and θ, we have redundant information, i.e $p(y \mid \theta, \alpha) = p(y \mid \theta)$. Using this, we have

$$\begin{aligned} p(y = i \mid \alpha) &= \int p(y = i, \theta \mid \alpha)\, d\theta \\ &= \int p(y = i \mid \theta) p(\theta \mid \alpha)\, d\theta \\ &= \int \theta_i \frac{1}{D(\alpha)} \theta_i^{\alpha_i - 1} \theta_{-i}^{\alpha_{-i} - 1}\, d\theta \\ &= \mathbb{E}_{\theta \mid \alpha}[\theta_i] \\ &= \frac{\alpha_i}{\alpha_i + \alpha_{-i}} = \frac{\alpha_i}{\alpha_1 + \alpha_2 + \alpha_3} \end{aligned}$$

Where $-i$ denotes all components that are not i. So we have arrived at the expectation of a Dirichlet distribution. (I hope we can just assume the mean of a Dirichlet rather than integrate over the T^2 simplex.)

3. The posterior mean estimate for θ,

$$\mathbb{E}_{\theta \mid y, \alpha}[\theta_i] = \int \theta_i p(\theta \mid y, \alpha) d\theta,$$

which (after some manipulation) should not involve any Γ functions.

Answer: We'll use a similar trick to before, assuming we have access to the moments of a regular Dirichlet distribution (integrate over the T^{n-1} simplex for an n parameter distribution.) We use the covariance of a regular Dirichlet distribution. Here we make note of this:

$$\mathrm{Cov}_{\theta \mid \alpha}(\theta_i, \theta_j) = \frac{-\alpha_i \alpha_j}{\alpha_0^2(\alpha_0 + 1)}$$

where we denote $\alpha_0 = \sum \alpha_i$. With the covariance, we can find the mixed second moment

$$\mathbb{E}_{\theta \mid \alpha}[\theta_i \theta_j] = \mathrm{Cov}_{\theta \mid \alpha}(\theta_i, \theta_j) - \mathbb{E}_{\theta \mid \alpha}[\theta_i] \mathbb{E}_{\theta \mid \alpha}[\theta_j] = \frac{\alpha_i \alpha_j}{\alpha_0(\alpha_0 + 1)}.$$

Notice we have

$$\mathbb{E}_{\theta \mid y, \alpha}[\theta_i] = \mathbb{E}_{\theta \mid y = i, \alpha}[\theta_i] \delta_{ij} + \mathbb{E}_{\theta \mid y = j, \alpha}[\theta_i](1 - \delta_{ij})$$

where either we have seen the same θ_i already or not. So we deal with these separately.

$$\mathbb{E}_{\theta \mid y = i, \alpha}[\theta_i] = \int \theta_i \frac{1}{D(\alpha^+)} \theta_i^{\alpha_i} \theta_{-i}^{\alpha_{-i} - 1}\, d\theta$$

$$
\begin{aligned}
&= \frac{\alpha_0}{\alpha_i} \int \theta_i^2 \frac{1}{D(\alpha)} \theta_i^{\alpha_i - 1} \theta_{-i}^{\alpha_{-i} - 1} \, d\theta \\
&= \frac{\alpha_0}{\alpha_i} \mathbb{E}_{\theta \mid \alpha}[\theta_i^2] \\
&= \frac{\alpha_0}{\alpha_i} \left(\frac{(\alpha_i + 1)\alpha_i}{(\alpha_0 + 1)(\alpha_0)} \right) \\
&= \frac{\alpha_i + 1}{\alpha_0 + 1} = \frac{\alpha_i + 1}{\alpha_1 + \alpha_2 + \alpha_3 + 1}
\end{aligned}
$$

$$
\begin{aligned}
\mathbb{E}_{\theta \mid y=j,\alpha}[\theta_i] &= \int \theta_i \frac{1}{D(\alpha^+)} \theta_i^{\alpha_i - 1} \theta_j^{\alpha_j} \theta_{-ij}^{\alpha_{-ij} - 1} \, d\theta \\
&= \frac{\alpha_0}{\alpha_i} \int \theta_i \theta_j \frac{1}{D(\alpha)} \theta_i^{\alpha_i - 1} \theta_{-i}^{\alpha_{-i} - 1} \, d\theta \\
&= \frac{\alpha_0}{\alpha_i} \mathbb{E}_{\theta \mid \alpha}[\theta_i \theta_j] \\
&= \frac{\alpha_0}{\alpha_i} \left(\frac{\alpha_i \alpha_j}{\alpha_0(\alpha_0 + 1)} \right) \\
&= \frac{\alpha_j}{\alpha_0 + 1} = \frac{\alpha_j}{\alpha_1 + \alpha_2 + \alpha_3 + 1}
\end{aligned}
$$

Where we finally have for any $y = j$,

$$
\mathbb{E}_{\theta \mid y=j,\alpha}[\theta_i] = \frac{(\alpha_i + 1)\delta_{ij} + \alpha_i(1 - \delta_{ij})}{\alpha_1 + \alpha_2 + \alpha_3 + 1}
$$

4. The posterior predictive distribution for a new independent observation $\tilde{y}$ given y,

$$
p(\tilde{y} \mid y, \alpha) = \int p(\tilde{y}, \theta \mid y, \alpha) d\theta.
$$

Answer: We once more will assume that we have access to the covariance of a regular Dirichlet distribution. With this, we have everything we need to find the predictive posterior probability.

$$
\begin{aligned}
p(\tilde{y} = j \mid y = i, \alpha) &= \int p(\tilde{y} = j, \theta \mid y = i, \alpha) \, d\theta \\
&= \int p(\tilde{y} = j \mid \theta) p(\theta \mid y = i, \alpha) \, d\theta \\
&= \int p(\tilde{y} = i|\theta)\delta_{ij} + p(\tilde{y} = j|\theta)(1 - \delta_{ij}) dp_{\theta \mid y=i,\alpha} \\
&= \mathbb{E}_{\theta \mid y=i,\alpha}[\theta_i]\delta_{ij} + \mathbb{E}_{\theta \mid y=i,\alpha}[\theta_j](1 - \delta_{ij})
\end{aligned}
$$

From what we learned in the previous problem, the relationship between conditional expectation allows us to shift back to the base Dirichlet measure with the appropriate scaling

$$\begin{aligned} p(\tilde{y} = j \mid y = i, \alpha) &= \mathbb{E}_{\theta \mid y=i,\alpha}[\theta_i]\delta_{ij} + \mathbb{E}_{\theta \mid y=i,\alpha}[\theta_j](1 - \delta_{ij}) \\ &= \frac{\alpha_0}{\alpha_i}\mathbb{E}_{\theta \mid \alpha}[\theta_i^2]\delta_{ij} + \frac{\alpha_0}{\alpha_i}\mathbb{E}_{\theta \mid \alpha}[\theta_i\theta_j](1 - \delta_{ij}) \\ &= \frac{(\alpha_i + 1)\delta_{ij}}{\alpha_0 + 1} + \frac{\alpha_j(1 - \delta_{ij})}{\alpha_0 + 1} \\ &= \frac{(\alpha_i + 1)\delta_{ij} + \alpha_j(1 - \delta_{ij})}{\alpha_1 + \alpha_2 + \alpha_3 + 1} \end{aligned}$$

Heuristically, this says that the probability of seeing another sample of label i given a sample of label i increases while other label probabilities decrease.

Hint: You can use $D(\alpha) = \frac{\Gamma(\alpha_1)\Gamma(\alpha_2)\Gamma(\alpha_3)}{\Gamma(\alpha_1+\alpha_2+\alpha_3)}$ to represent the normalizing constant of the prior and $D(\alpha^+)$ to give the normalizing constant of the posterior. You will also need to use that $\Gamma(\alpha + 1) = \alpha\Gamma(\alpha)$. For some calculations you may find it a bit cleaner to parameterize the posterior in terms of $\beta_j = I(y = j) + \alpha_j$, and convert back once you have the final result.

2.2 Empirical Bayes

Consider the model

$$y_i \sim \mathcal{N}(w^T\phi(x^i), \sigma^2), \quad w_j \sim \mathcal{N}(0, \lambda^{-1}),$$

where ϕ is a non-linear transformation of the features x^i (like a polynomial basis or RBFs). By using properties of Gaussians the marginal likelihood (marginalizing over all w_j) has the form

$$p(y \mid X, \sigma, \lambda) = (2\pi)^{-d/2}|C|^{-1/2}\exp\left(-\frac{y^TC^{-1}y}{2}\right),$$

which gives a negative log-marginal likelihood of

$$-\log p(y \mid X, \sigma, \lambda) = \log|C| + y^TC^{-1}y + \text{const.}$$

where

$$C = \frac{1}{\sigma^2}I + \frac{1}{\lambda}\Phi(X)\Phi(X)^T,$$

As discussed in class, the marginal likelihood can be used to optimize hyper-parameters like σ, λ, and even the basis ϕ.

The demo *example_basis* loads a dataset and fits a degree-2 polynomial to it. Normally we would use a test set to choose the degree of the polynomial but here we'll use the marginal likelihood of the training set. Write a function, *leastSquaresEmpiricalBaysis*, that uses the marginal likelihood to choose the degree of the polynomial as well as the parameters λ and

σ (you can assume that all λ_j are equal, and you can restrict your search for λ and σ to powers of 10). Hand in your code and report the marginally most likely values of the degree, σ, and λ.

Hint: the matrix C can be highly ill-conditioned and thus can cause Julia to do weird things, like saying $y^T C^{-1} y$ is negative even though C is positive-definite by construction. To compute the log-marginal likelihood in a more stable way, you can use the *chol* function (and may need to catch positive-definite errors when the code fails). You can then use the Cholesky factorization to compute $y^T C^{-1} y$ (by solving two linear systems) and $\log |C|$ (as twice the sum of the log of the diagonals).

Answer: The most likely values are from minimizing the negative log-marginal probability are:

$$p = 3,\ \lambda = 0.0001,\ \sigma = 0.1$$

The code for *leastSquaresEmpericalBayes.jl*:

```
function leastSquaresEmpericalBayes(X,y)
    (x1,x2) = size(X)
    pvec = 0:10
    range = -7:7
    lambdavec = 10.0.^range
    sigmavec = 10.0.^range
    k = length(pvec)
    n = length(lambdavec)
    m = length(sigmavec)
    NLMmatrix = zeros(n,m)
    NLM = 1E10
    bestlambda = 0
    bestsigma = 0
    bestp = 0
    for dim in 1:k
        p=pvec[dim]
        count=0
        for i in 1:n
            lambda = lambdavec[i]
            for j in 1:m
                sigma = sigmavec[j]
                Phi = polyBasis(X,p)
                C = eye(x1)*1/sigma^2+Phi*Phi'*1/lambda
                (NLMmatrix[i,j],infstage) = neglogmarginal(y,C,p)
                count = count+infstage
            end
        end
    (optimali,optimalj) = ind2sub(size(NLMmatrix), indmin(NLMmatrix))
```

```
        @printf("For degree p = %d, the minimum negative log-marginal prob
        is %f\n",p,NLMmatrix[optimali,optimalj])
        @printf("There were %d ill-posed C matrices out of a %d by %d
        parameter grid.\n",count,length(range),length(range))
        if NLMmatrix[optimali,optimalj]<NLM
            bestp = p
            bestlambda = lambdavec[optimali]
            bestsigma = sigmavec[optimalj]
            NLM = NLMmatrix[optimali,optimalj]
        end
        println("")
    end
    return (bestp,bestlambda,bestsigma,NLM)
end


function neglogmarginal(y,C,p)
    n = size(C)[1]
    if isposdef(C)
        U =chol(C)
        logdetU=0
        count =0
        # Compute the log determinant stably
        for i in 1:n
            logdetU +=2*log(U[i,i])
        end
        # Compute the y'inv(C)y stably
        b=U'\y
        x=U\b
        quad = y'*x
        # Put it all together
        NLM = (1/2)*logdetU+(1/2)*quad+(p/2)*log(2*pi)
    else
        NLM =Inf
        count = 1
    end
    NLM = NLM[1]
    return (NLM,count)
end

function polyBasis(x,p)
        n = length(x)
```

```
        Z = zeros(n,p+1)
        for i in 0:p
                Z[:,i+1] = x.^i
        end
        return Z
end
```

3 Very-Short Answer Questions

Give a short and concise 1-sentence answer to the below questions.

1. In UGMs, why is it easy to evaluate $\tilde{p}(x)$ but not $p(x)$?

   Answer: Recall that $\tilde{p}(x)$ is the unnormalized version of $p(x)$, i.e. $p(x) = \tilde{p}(x)/Z$. Evaluating $p(x)$ is NP-hard as evaluating Z is NP-hard. The computation of $\tilde{p}(x)$ is easy as it is just the product of the (given) potentials, where calculating the normalization constant requires an n dimensional integral.

2. Why we need to have a "burn in" phase when using a Markov chain Monte Carlo approximation?

   Answer: Due to the convergence of the MC, after a long time, the distribution will be close to the stationary distribution π. The "burn in" phase is necessary because the inital samples might be very far avaw from the stationary distribution of the Markov Chain, so we throw them away.

3. Why do use the parameterization $\phi_j(s) = \exp(w_{j,s})$ in UGMs?

   Answer: We choose to exponentiate the $\omega_{j,s}$ as this will give a log-linear model and leads to a convex negative log-likelihood. If we would not exponentiate and treat the potentials directly as parameters we would obtain a non-convex optimization problem.

4. Suppose you are given the initial probabilities and transition probabilities in a Markov chain. Describe how you could set the parameter of a hidden Markov model so that the x_j follow the given (non-hidden) Markov chain.

   Answer: If we set the parameters of the hidden chain Z to be exactly the same as the ones from our Markov chain X and then let z_j and x_j depend deterministicly in the way $p(x_i = c_1 | z_i = c_2) = \delta_{c_1,c_2}$.

5. What is the key advantage of the graph structure in restricted Boltzmann machines?

   Answer: The biepartie structure alows block Gibbs sampling given one type of variable (because the conditional UGM is disconnected).

6. What is the difference between a generative model and a discriminative model?

Answer: Wheras generative models use the naive Bayes approach $p(y|x) \propto p(y, x)$ a discriminative model directly fits $p(y|x)$ as in logistic regression. That means that the key difference is density estimation vs. conditional density estimation.

7. Why can fully-convolutional networks segment images of different sizes?

   Answer: The Parameter tieing makes it possible to segment different image sizes.

8. What is the key feature of a "sequence to sequence" RNN?

   Answer: Between the input sequence and the output sequence we have a hidden markov chain.

9. What are two advantages of the Bayesian approach to learning?

   Answer: Bayesian methods combine model averaging and regularization (as they are weighted by posteriori).

10. What is the difference between the posterior distribution and the posterior predictive distribution?

    Answer: While the posterior distribution $p(\omega|X, y, \lambda)$ is the probability that the parameters are correct after we have seen the data, the posterior predictive distribution $p(\tilde{y}|\tilde{x}, X, y, \lambda)$ models the probability of new data given the old data. So one of them models probabilities of parameters of a model wheras the other one tells you how probable a new outcome is.

11. What is the key property of a conjugate prior?

    Answer: For conjugate priors the prior distribution and the posterior distribution are from the same family, e.g. both beta distributed or both normal distributed.

4 Literature Survey

Reading academic papers is a skill that takes practice. When you first start out reading papers, you may find that you need to re-read things several times before you understand them, or that details will still be very fuzzy even after you've put a great amount of effort into trying to understand a paper. Don't panic, this is normal.

Even if you are used to reading papers from your particular sub-area, it can be challenging to read papers about a completely different topic. Usually, people in different areas use different language/notation and focus on very different issues. Nevertheless, many of the most-successful people in academia and industry are those that are able to understand/adapt ideas from different areas. (There are a ton of smart people in the world working on all sorts of amazing things, it's good to know how to communicate with as many of them as possible.)

A common technique when trying to understand a new topic (or reading scientific papers for the first time) is to read and write notes on 10 papers on the topic. When you read the first paper, you'll often find that it's hard to follow. This can make reading take a long time and might still leave you feeling that many things don't make sense; keep reading and trying to take notes. When you get to the second paper, it might still be very hard to follow. But when you start getting to the 8th or 9th paper, things often start making more sense. You'll start to form an impression of what the influential works in the area are, you'll start getting to used to the language and jargon, you'll start to understand what the main issues that people who work on the topic care about, and you'll probably notice some important references that weren't on your initial list of 10 papers. Ideally, you'll also start to notice how the topic has changed over time and you may get ideas of future work that you could do on the topic.

To help you make progress on your project or to give you an excuse to learn about a new topic, for this part you should write a literature survey of at least 10 academic papers on a particular topic. While your personal notes on the papers may be longer, the survey should be at most 4 pages of text (excluding references/tables/figures) in a format similar to the one for this document. Some logical components of a literature survey might be:

- A description of the overall topic, and the key themes/trends across the papers.
- A short high-level description of what was explored in each paper. For example, describe the problem being addressed, the key components of the proposed solution, and how it was evaluated. In addition, it is important to comment on the *why* questions: why is this problem important and why would this particular solution method make progress on it? It's also useful to comment on the strengths and weaknesses of the various works, and it's particularly nice if you can show how some works address the weaknesses of prior works (or introduce new weaknesses).
- One or more logical "groupings" of the papers. This could be in terms of the variant of the topic that they address, in terms of the solution techniques used, or in chronological terms.

Some advice on choosing the topic:

- The most logical/easy topic for your literature survey is a topic related to your course project, given that your final report will need a (shorter) literature survey included.
- If you are an undergrad, or a masters student without a research project yet, you may alternately want to choose a general area (like variance-reduced stochastic gradient, non-Gaussian graphical models, recurrent neural networks, matrix factorization, neural artistic style transfer, Bayesian optimization, etc.) as your topic.
- If you are a masters student that already has a thesis project, it could make sense to do a survey on a topic where ML intersects with your thesis (or where ML *could* intersect your thesis).

- If you are a PhD student, I would recommend using this an excuse to learn about a *completely different* topic than what you normally work on. Choose something hard that you would like to learn about, but previously haven't been to justify exploring carefully. This can be invaluable to your future research, because during your PhD it's often hard to allocate time to learn completely new topics.

HW 5 Literature Review - Principal Component Analysis in Predictive Policing

by Cody Griffith, Ziming Yin and Tim Jaschek

Abstract

In our final project we would like to apply unsupervised machine learning techniques for predictive policing. We decided to focus on unsupervised clustering. A powerful and frequently used tool for such tasks is principal component analysis.

Introduction

Before we begin with the literature review let us recap the basic theory of Principal Component Analysis (PCA). Let $X \in \mathbb{R}^{n\times d}$ denote the training data and $\tilde{X} \in \mathbb{R}^{t\times d}$ denote the test data. PCA can be seen as a change of basis in a way that fundamental pattern in the data can be identified more clearly, which allows an easier and more effective classifying. It can be used as a machine learning technique in the following sense:

i) *learning phase:* Find the principal components of the training data.
ii) *preparation phase:* Expand the test data in terms of the principal components of the training data. Here we can also use an approximation by just considering the most dominant principal components.
ii) *prediction phase:* Assign the test data to clusters according the their representation in terms of the principal components. For the assignment we can use other unsupervised learning methods.

However, as [2] shows, PCA can also used directly to classify the test data. More details follow in the summary section below.

The principal components can be identified by solving a Fredholm integral equation of 2^{nd} kind. In detail, let $Y = (y^1, \dots, y^d)$ represent the centered test data matrix, *i.e.* the matrix defined by $y^i = x^i - \overline{x}$, where $\overline{x} = \sum_{i=1}^d x^i/d$. The covariance matrix has a nice form in terms of Y

$$(\mathrm{Cov}(x^j, x^k))_{j,k} \propto \sum_{i=1}^{d} (x^i - \overline{x})(x^i - \overline{x})^T = YY^T,$$

where the constant is given by $1/d$. Principal components $(u_k)_{k=1,\dots,d}$ and principal directions $(v_k)_{k=1,\dots,d}$ are eigenvectors of the covariance and it's transpose respectively

$$YY^T u_k = \lambda_k u_k, \quad Y^T Y v_k = \lambda_k v_k, \qquad \text{where } \lambda_k \in \mathbb{R}^+ \text{ for } k = 1, \dots, d.$$

This is a discrete version of the Fredholm integral inequality. By the single value decomposition, the covariance matrix Y is given by $\sum_{k=1}^d \lambda_k u_k v_k^T$ and can be approximated by considering partial sums of this expansion.

Instead of solving for the principal components analytically, we ca consider the following optimization problem described in [1]: By considering just the most dominant principal components of X, we can project the data matrix to a lower dimension with $T = XW$. The columns of the weight matrix, w_i, represent the principal components. After normalizing and standardizing we can implement an algorithm to determine the principal components by

$$w_1 = \underset{\|w\|_2=1}{\arg\max} \{\|Xw\|_2^2\}, \quad w_i = \underset{\|w\|_2=1}{\arg\max} \left\{ \left\| Xw - \left(\sum_{s=1}^{i-1} X w_s w_s^T \right) w \right\|_2^2 \right\}.$$

Summaries

In this section we will give very brief summaries of the papers that we read for a literature review. We classified the papers in three categories, which are represented by the following subsections. Our original summaries where a lot longer but due to the page restriction we have to be very brief in the sequel.

General theory of PCA & connection to machine learning

Papers in this section provide a helpful introduction to the theory of PCA. We learned from them what PCA is, how we could implement it and that it can be very useful to cluster data.

Article [1]: Principal Component Analysis

This paper offers a intuitive understanding of PCA by building up each piece of the underlying analysis individually. According to the authors PCA has two major advantages

i) Allow for a high-dimensional matrix X to be projected into a lower-dimensional shadow.

ii) Preserve as much of the variation of the data post-projection.

The paper provides the algorithm to compute the principal components that we mentioned above. The authors point out that the principal components that we obtained by solving the optimization problem, can be used to define a typical multiple linear regression problem

$$X = TW^T + E = \hat{X} + E$$

where we call $\hat{X}$ our model for X and E our residuals of X. We then find the proportion of variance explained by the components in this model with

$$R^2 = 1 - \frac{\|E\|^2}{\|\hat{X}\|^2 + \|E\|^2}.$$

Lastly, a big claim made by this paper is as follows: "*If two components explain most of the variation in the data, then a scatter plot of their scores will reflect distances.*"

Article [2]: K-means Clustering via Principal Component Analysis

The paper *K-means Clustering via Principal Component Analysis* written by Chris Ding (Berkeley) and Xiaofeng He (Berkeley) appeared in the *Proceedings of the* 21^{st} *International Conference on Machine Learning*. The main result is that PCA can be viewed as a tool to perform K-means clustering.

It begins with a brief introduction to the K-means algorithm and the PCA algorithm. In the following the authors give a discussion of a two-class model. In the third section the main result for a general K-class model is stated and followed by a rigorous proof. A final section that shows results of numerical experiments completes the paper.

The K-means method assigns the data to K disjoint clusters $C_1, \dots, C_K$. The clusters are determined by their centroids, which are determined by minimizing the ℓ^2-error

$$J_K = \sum_{k=1}^{K} \sum_{i \in C_k} (x_i - m_k)^2,$$

where $(x_1, \dots, x_d) = X$ represent the feature vectors and $m_k = \sum_{i \in C_k} x_i / n_k$ is the centroid of cluster C_k and n_k is the number of points in C_k.

Consider Figure 1 for a visualization that Ding and He provide which shows how clustering via PCA is happening.

Kernel Principal Component Analysis

As pointed out before PCA is a linear transformation of the basis of the vector space of the test data. For more complex data spaces where nonlinear relations come into play there is a more general form of PCA, the so called Kernel Principal Component Analysis (KPCA). This method was introduced in [9] and applied in [12] and [4].

Article [9]: Kernel Principal Component Analysis

The paper *Kernel Principal Component Analysis* by Bernhard Schölkopf, Alexander Smola and Klaus-Robert Müller was the first to suggest KPCA and appeared in 1997. The summary for this paper is relatively long but we will summarize more papers on KPCA and need to introduce the notation.

Instead of performing a PCA in $\mathbb{R}$, the idea is to map the data in a higher dimensional space (the results work even for infinite dimensional Hilbert spaces). The map to this space is given by a nonlinearity $\Phi : \mathbb{R}^d \to F$, where for simplicity we will assume that $F = \mathbb{R}^D$ for some $d << D$. The authors show that for certain choices of the map Φ, we can still perform a PCA in F.

Assume for now that the vectors $\Phi(x^i)$ are normalized. We compute the transformed covariance matrix by $\overline{C} = \frac{1}{d} \sum_{i=1}^{d} \Phi(x^i)\Phi(x^i)^T \in \mathbb{R}^{D \times D}$. Note that the eigenvectors are in $\text{span}(\Phi(x^i))_i$. Indeed,

$$Cv_k = \lambda_k v_k \quad \Leftrightarrow \quad \frac{1}{d} \sum_{i=1}^{d} \Phi(x^i)\Phi(x^i)^T v_k = \lambda_k v_k \quad \Rightarrow \quad v_k = \sum_{i=1}^{d} \alpha_{ki} \Phi(x^i) \quad \text{for some } \alpha_{ki} \in \mathbb{R}.$$

If we define the kernel function $\kappa(x^i, x^j) = \Phi(x^i)^T \Phi(x^j)$ and multiply both sides of the eigenvalue problem for $\overline{C}$ by $\Phi(x^l)$ we end up with

$$\frac{1}{d}\sum_{i=1}^{d}\kappa(x^l, x^i)\sum_{j=1}^{d}\alpha_{kj}\kappa(x^i, x^j) = \lambda_k \sum_{i=1}^{d}\alpha_{ki}\kappa(x^l, x^i),$$

which can be expressed in matrix form by $K^2\alpha_k = \lambda_k dK\alpha_k$, where $(K_{ij})_{ij} = (\kappa(x^i, x^j))_{ij}, \alpha_k = (\alpha_{k,i})_i$. Now α_k can be computed by solving $K\alpha_k = \lambda_k d\alpha_k$ subject to the normalization condition $\lambda_k(\alpha_k, \alpha_k) = 1$ and then the kernel principle components are obtained by calculating

$$y_k(x) = \Phi(x)^T v_k = \sum_{i=1}^{d}\alpha_{ki}\kappa(x, x^i).$$

Note that we don't actually need to compute the values of $\Phi(x^i)$, all we need is $\kappa(x^i, x^j)$. Kernels which have been successfully used in Support vector machines are for example polynomial kernels, radial basis functions and sigmoid kernels. The authors provide with Figure 2 a helpful visualization for what is happening with the data.

Article [12]: Kernel Principal Component Anlaysis and its Applications in Face Recognition and Active Shape Models

This work of Quan Wang appeared in 2012. It focuses on reconstruction of pre-images from kernel PCA. As an example KPCA is applied to facial recognition and active shape models. Furthermore the author compares the performance of PCA and KPCA for classification problems.

Wang points out that pre-image reconstruction of a KPCA is possible via the approximation

$$\Phi(x) \approx P_m\Phi(x) = \sum_{k=1}^{m} y_k(x)v_k = \sum_{k=1}^{m}(\Phi(x), v_k)v_k.$$

Finding the exact pre-image x is computationally difficult but the problem can be simplified by another approximation, namely

$$\arg\min_z \rho(z) = \|\Phi(z) - P_m\Phi(x)\|^2.$$

Wang found that KPCA succeeded in revealing more complicated structures of data than standard PCA and that Gaussian kernel PCA seems to be promising in active shape modeling in providing more deformation patterns than traditional algorithms.

Article [4]: Face Recognition Using Kernel Principal Component Analysis

The paper *Face Recognition Using Kernel Principal Component Analysis* by Kwang In Kim, Keechul Jung and Hang Joon Kim appeared in the *IEEE Signal processing letters* in 2002. The recognition is performed by extracting a facial feature vector using a kernel PCA and classifying it using a SVM. The authors claim that this combination of KPCA and SVMs will improve the performance. Experimental results with a the ORL data base (face data bace) confirm the effectiveness of the proposed method. The authors sumarized their results in Figure 3.

Predictive Policing and possible applications for KPCA

We now arrived at the section that considers the area where we would like to apply PCA or even KPCA: Predictive Policing.

Article [7]: Using Machine Learning Algorithms to Analyze Crime Data

Predict crime patterns and compare accuracy among linear regression, additive regression, and decision stump. Linear regression have best correlation coefficient and error. They use five evaluation metrics.

i) Correlation Coefficients: covariance of algorithm output and real value.
ii) Mean Absolute Value: measures absolute difference between the predicted and observed values.
iii) Root Mean Squared Error: the square root of average of the total squared error.
iv) Relative Absolute Error: defined by $RAE = \frac{\sum_{j=1}^{N}|P_j - T_j|}{\sum_{j=1}^{N}|T_j - \bar{T}|}$,
v) Root Relative Squared Error denoted by $RRSE = \sqrt{\frac{\sum_{j=1}^{N}(P_j - T_j)^2}{\sum_{j=1}^{N}(T_j - \bar{T})^2}}$

Article [13]: Learning to Detect Patterns of Crime

For a typical predictive policing approach, a crime analyst is needed to manually determine patterns, which is quite time consuming and also could have behavior too high-dimensional. If we consider a crime pattern P, *Series Finder* is a solution to automating this process of building a general framework of crime patterns $\hat{P}$ and clusters similar crimes using similarity criteria

$$\gamma_{\hat{P}}(C_i, C_k) = \frac{1}{\Gamma_{\hat{P}}} \sum_j \lambda_j \eta_{\hat{P},j} s_j(C_i, C_k)$$

where λ_j are global weights for attribute j across all crime patterns P and use prior data, $\eta_{\hat{P},j}$ are pattern $\hat{P}$ specific weights to attribute j that change as $\hat{P}$ grows with $\eta_{\hat{P},j} = \sum_{i,k} s_j(C_i, C_k)$ and $\Gamma_{\hat{P}}$ is just a normalizing constant. The similarity s_j between crimes can be either categorically defined or numerical, each requiring a separate measure and a lengthy discussion, the paper will provide a better explanation. It is possible to define a similarity measure for a crime and a pattern using some d-norm, where d is chosen as some form of a hyper-parameter. Once the similarity is defined, choose the crime that has the maximum similarity to $\hat{P}$, then a cohesion metric is defined to determine if the crime should be added to $\hat{P}$ or not.
With the cohesion metric and the largest $\hat{P}$ grown, a ML algorithm can be used to form a gain function and then maximize to find the weights λ. Define the gain function g as

$$g(\hat{P}_l, P_l, \lambda) = \frac{1}{|P|} \sum_{i \in P} \mathbb{1}_{\{C \in \hat{P}\}} + \beta \frac{1}{|\hat{P}|} \sum_{i \in \hat{P}} \mathbb{1}_{\{C \in P\}},$$

where β allows for some tradeoff, also a hyper-parameter. The optimization problem is established as

$$\underset{\lambda}{\text{maximize}} \sum_l g(\hat{P}_l, P_l, \lambda) \quad \text{subject to} \quad \lambda_j \geq 0 \quad \text{and} \quad \sum_j \lambda_j = 1.$$

Where the problem with this set up is it is non-convex and non-linear. The assumption made in this paper is that it is reasonably smooth and hence a coordinate ascent algorithm should converge to a better choice in λ and the process can be iterated. Once a better choice in λ is found, the process of classifying $\hat{P}$ can be redone and this allows the entire pipeline to be iterated until a global optimal is found.
A case study showed improved performance over analyst classification with very few mistakes and outperformed other algorithms (HAC and NN) in the two important grouping metrics, precision and reciprocal rank. The clear advantage to Series Finder is that the weight selection is entirely learned, no human input required, where the competing algorithms require the global parameters to be set (very subjective). Consider figure 5 for results.

Article [3]: The effect of principal component analysis on machine learning accuracy with high-dimensional spectral data

This paper develops further into using PCA as a preprocessing technique (performed with the NIPALS algorithm) to determine if PCA is helpful for machine learning algorithms.
The NIPALS algorithm calculates each principal component one at a time (so you only calculate what you need) and can be found in the Appendix. After you have the principal component, you compute the residual, $E_n = X - t_n p_n'$, and then replace X with E_n and repeat to get the next principal component.
The results of the paper can be seen in the following tables. The columns are the preprocessing used:
RD - Raw Data, ND - Normalized Data, FD - Savitsky-Gorlay First Derivative (with a 7 point averaging), and FND - Normalized FD. The values C, σ and k are hyper-parameters in the linear SVM, RBF SVM, and KNN algorithms respectively. In table 2, P represents the number of principal components used. The bold values indicate statistical significance from pairwise t-tests performed, in table 1 this is done between each result and in table 2 this was done between the values in with and without PCA.
The conclusions that can be made here are:

- PCA is effective as a preprocessing technique, but further work showed that for rule learning algorithms (C4.5 decision trees and RIPPER) PCA typically performed worse.
- For most high dimensional data, 20 principal components account for about 99% of the data but in this dataset only a maximum of 6 were needed.
- PCA with linear and non-linear SVM along with KNN performed the best for this dataset.

Article [8]: Optimal Mean Robust Principal Component Analysis

Currently in robust PCA models, the ℓ_1-norm is used in the objective function while the ℓ_2-norm is used to calculate the mean. This paper proposes an objective function with removing the optimal mean automatically and suggest two new algorithms: RPCA-OM and CRPCA-OM. Let Z be an approximation matrix and b be an approximation of the mean. The mathematical form of the problem is: $\min_{b,\text{rank}(Z)=k} \sum_{i=1}^{n} \| x_i - b - Z_i \|_2^2$. In

order to make PCA robust to outliers, previous robust PCA uses a non-squared loss function, but center data via ℓ_2-norm distance based mean. Algorithm 1 proposes a way to remove mean automatically. The new robust PCA is to solve the same problem but with each term of the sum squared. Denote by $\mathbf{1}$ the column vector with all the elements being one, and $\|M\|_{2,1} = \sum_i \|m_i\|_2$, rewrite it in a matrix form: [1] $\min_{b,\mathrm{rank}(Z)=k} \| X - b\mathbf{1}^T - Z \|_{2,1}$ After setting derivative to zero, above problem can be transfered into $\min_{b,U\in R^{d\times k}, U^TU=I} \sum_{i=1}^{n} \| (I-UU^T)(x_i - b) \|_2$. Algorithm RPCA-OM can be found in the appendix.

The second algorithm CRPCA-OM is proposed to solve the following problem, where denote $\| M \|_*= Tr((M^TM)^{\frac{1}{2}})$ as the trace norm (nuclear norm): [2] $\min_{b,Z} \| X - b1^T - Z \|_{2,1} + \gamma \| Z \|_*$ Problem[2] can be seen as a convex relaxation of problem[1]

Article [5]: Sequential Karhunen-Loeve Basis Extraction And Its Application to Images

Traditional batch KL algorithm has high computational demand and makes KL not practicable. In this paper, the new sequential KL basis algorithm is faster and meanwhile guarantees accuracy. It uses a sequential iterative algorithm which avoids both the storage of all the image data and its simultaneous processing. This algorithm is based on partitioning the data into blocks of column vectors. It starts by calculating the SVD of the first block. Then, at every step, another block of columns is added and the updated SVD is calculated using the "partitioned R-SVD" algorithm. Being interested only in a low dimensional approximation to the KL basis, the algorithm both deletes basis vectors, corresponding to very small singular values and sets a limit on their number.

Appendix: Algorithms, Figures and Tabulars

Algorithm 1 The NIPALS algorithm

1: Take a vector x_j from X and call it t_n: $t_n = x_j$
2: Calculate p'_n: $p'_n = t'_n X / t'_n t_n$
3: Normalize p'_n
4: Calculate t_n : $t_n = X p_n / p'_n p_n$
5: Compare t_n in step 2 with that of step 4. If they are the same, stop. If they differ then repeat 2-4 until convergence.

Algorithm 2 The RPCA-OM algorithm

1: Initialize D as an identify matrix.
2: Update the columns of U by the k right singular vectors of $X(D^{\frac{1}{2}} - \frac{D11^T D^{\frac{1}{2}}}{1^T D1})$ vectors of corresponding to the k largest singular values.
3: Update b by $b = \frac{XD1}{1^T D1}$
4: Update the diagonal matrix D, where the $i-th$ diagonal element of D is updated by $d_{ii} = \frac{1}{2\|(I-UU^T)(x_i-b)\|_2}$
5: Stop when it converges.

Algorithm 3 The CRPCA-OM algorithm

1: Let $1 < \rho < 2$. Initialize $\mu = 0.1$, $E = 0$, $\Lambda = 0$
2: Update b and Z by solving $\min_{b,Z} \frac{1}{2} \parallel \tilde{X} - b1^T - Z \parallel_F^2 + \tilde{\gamma} \parallel Z \parallel_*$ where $\tilde{X} = X - E + \frac{1}{\mu}\Lambda$ and $\tilde{\gamma} = \frac{\gamma}{\mu}$
3: Update E by solving $\min_{E} \frac{1}{2} \parallel E - \tilde{X} \parallel_F^2 + \tilde{\gamma} \parallel E \parallel_{2,1}$ where $\tilde{X} = X - b1^T - Z + \frac{1}{\mu}\Lambda$ and $\tilde{\gamma} = \frac{1}{\mu}$
4: Update Λ by $\Lambda = \Lambda + \mu(X - b1^T - Z - E)$
5: Let $\mu = \min(\rho\mu, 10^8)$
6: Stop when it converges

Algorithm 4 The sequential K-L basis algorithm

1: Initialize $0th$ stage: form a matrix A_0 from the first $K_0 = K$ columns of the matrix A. Compute the R-SVD $A_0 = U_0 D_0 V_0^T$ and keep U_0 and D_0. A is a $M \times N$ input data matrix. $A = QR$ and $R = U' D V^T$ from SVD algorithm. K is the maximal number of columns for the U_i matrices.
2: While some data of A have not been processed yet, do the following.
- Read the matrix A_i containing the next P (number of columns in each block) columns of A, and append it to the matrix $ff \cdot U_{i-1} D_{i-1}$, where ff is the optional "forgetting factor" coefficient, which can reduce the contribution of previous blocks. Calculate the QR decomposition $ff \cdot U_{i-1} D_{i-1} | A_i = U'_{i-1} D'_{i-1}$ of the combined matrix.
- Compute SVD of D'_{i-1} as the product $D'_{i-1} = \tilde{U}_{i-1} \tilde{D}_{i-1} \tilde{V}^T$
- Pick out the K largest elements of the $\tilde{D}_{i-1}$. Delete the rest of the singular values as well as the energetically negligible singular values $\sigma_i^j, j \leq K$, satisfying $(\sigma_i^j)^2 < \epsilon \sum_{k=1}^{K} (\sigma_i^k)^2$ where ϵ is lower bound for the singular values that are retained. Let D_i be the diagonal matrix whose elements are the remaining K_i singular values.
- Remove from $\tilde{U}_{i-1}$ all the columns that corresponds to singular values that were removed above, and let $\hat{U}_{i-1}$ be the resulting matrix. Calculate $U_i = \hat{U}_{i-1} U'_{i-1}$

3: After all columns of A has been processed, the matrix U_i at the final stage is the output,

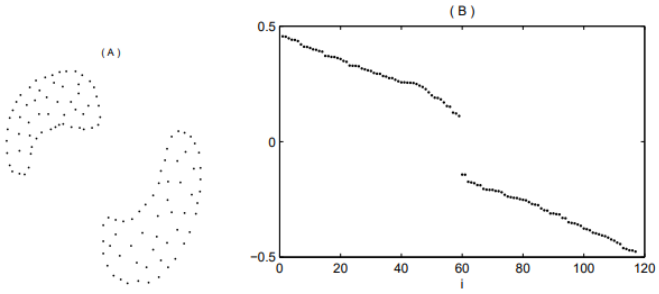


Figure 1. (A) Two clusters in 2D space. (B) Principal component $\mathbf{v}_1(i)$, showing the value of each element i.

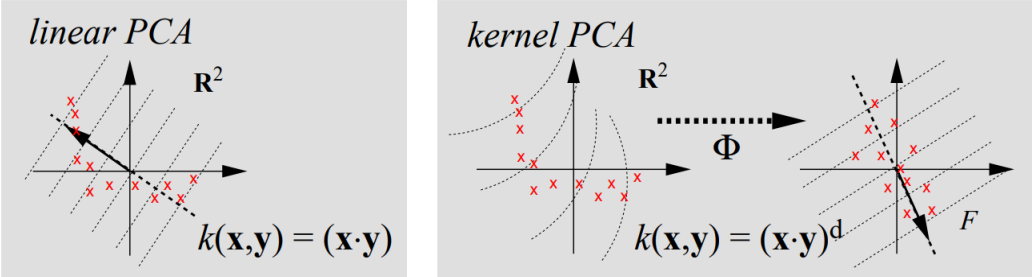


Figure 2 KPCA visualization

TABLE I
PERFORMANCES OF VARIOUS SYSTEMS

Systems	Error rates (%)
Eigenfaces [11]	10.0
Pseudo-2D HMM [11]	5.0
Probabilistic decision-based neural network [12]	4.0
Convolutional neural network [10]	3.8
Linear SVMs [6]	3.0
Kernel PCA	2.5

Figure 3 Face recognition via KPCA

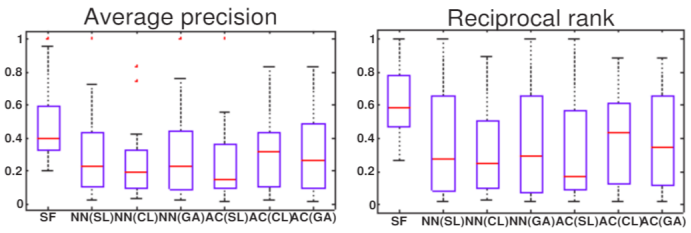


Figure 1: Average precisions and reciprocal ranks for all models. SF represents "Series Finder", NN represents "Incremental nearest neighbor classification" and AC represents "Agglomerative clustering".

Figure 4 Learning to detect patterns of crime

Table 1
Percentage classification error of different machine learning methods on acetaminophen dataset

Method	Pre-processing technique			
	RD	ND	FD	FND
Linear SVM	**6.45** ($C = 100$)	**2.76** ($C = 1$)	**3.23** ($C = 10000$)	**0.92*** ($C = 0.1$)
RBF SVM	**5.07** ($C = 1000$, $\sigma = 0.1$)	**2.76** ($C = 1000$, $\sigma = 0.001$)	**1.84** ($C = 1000$, $\sigma = 10$)	**0.92*** ($C = 10$, $\sigma = 0.01$)
k-NN	11.06 ($k = 1$)	7.83 ($k = 1$)	**4.61** ($k = 10$)	**4.15** ($k = 1$)
C4.5	10.14	7.83	**1.84**	**1.38**
RIPPER	15.67	11.06	**3.69**	**2.3**
Naive Bayes	25.35	13.82	25.81	**5.53**
Linear regression	27.65	16.13	25.35	20.28

Table 2
Percentage classification error of different machine learning methods with PCA on acetaminophen dataset

Method	Pre-processing technique			
	RD	ND	FD	FND
Linear SVM	5.07 ($P = 18$, $C = 0.1$)	**1.84** ($P = 13$, $C = 0.1$)	**3.23** ($P = 14$, $C = 0.01$)	**0.46** ($P = 4$ $C = 0.1$)
RBF SVM	6.91 ($P = 19$, $C = 100$, $\sigma = 0.001$)	**2.76** ($P = 16$, $C = 10$, $\sigma = 0.001$)	**2.23** ($P = 12$, $C = 10$, $\sigma = 0.001$)	**0.46** ($P = 5$ $C = 10$, $\sigma = 0.001$)
k-NN	11.06 ($P = 17$, $k = 3$)	5.99 ($P = 10$, $k = 1$)	**2.3** ($P = 14$, $k = 1$)	**0.0*** ($P = 4$, $k = 5$)
C4.5	7.83 ($P = 20$)	7.37 ($P = 19$)	7.37 ($P = 5$)	**1.38** ($P = 6$)
RIPPER	11.98 ($P = 20$)	8.29 ($P = 8$)	6.45 ($P = 5$)	**2.3** ($P = 3$)
Naive Bayes	38.71 ($P = 1$)	10.6 ($P = 8$)	11.52 ($P = 5$)	3.23 ($P = 2$)
PCR (PCA + linear regression)	9.22 ($P = 16$)	5.53 ($P = 20$)	8.29 ($P = 11$)	**1.38** ($P = 80$)

Figure 5 The effect of principal component analysis on machine learning accuracy with high-dimensional spectral data

References

[1] R. Bro and A. K. Smilde. Principal component analysis. *Analytical Methods*, 6(9):2812–2831, 2014.

[2] C. Ding and X. He. K-means Clustering via Principal Component Analysis. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 29–, New York, NY, USA, 2004. ACM.

[3] T. Howley, M. G. Madden, M.-L. O'Connell, and A. G. Ryder. The Effect of Principal Component Analysis on Machine Learning Accuracy with High Dimensional Spectral Data. In *Applications and Innovations in Intelligent Systems XIII*, pages 209–222. Springer, London, 2006.

[4] K. I. Kim, K. Jung, and H. J. Kim. Face recognition using kernel principal component analysis. *IEEE Signal Processing Letters*, 9(2):40–42, Feb. 2002.

[5] A. Levey and M. Lindenbaum. Sequential Karhunen-Loeve basis extraction and its application to images. *IEEE Transactions on Image Processing*, 9(8):1371–1374, Aug. 2000.

[6] Á. M. Márquez. A machine learning approach for studying linked residential burglaries, 2014.

[7] L. McClendon and N. Meghanathan. Using machine learning algorithms to analyze crime data. *Machine Learning and Applications: An International Journal (MLAIJ)*, 2(1), 2015.

[8] F. Nie, J. Yuan, and H. Huang. Optimal Mean Robust Principal Component Analysis. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1062–II–1070, Beijing, China, 2014. JMLR.org.

[9] B. Schölkopf, A. Smola, and K.-R. Müller. Kernel principal component analysis. In *International Conference on Artificial Neural Networks*, pages 583–588. Springer, 1997.

[10] M. Vaquero Barnadas. Machine learning applied to crime prediction. B.S. thesis, Universitat Politècnica de Catalunya, 2016.

[11] L. Wang. *Karhunen-Loeve expansions and their applications.* London School of Economics and Political Science (United Kingdom), 2008.

[12] Q. Wang. Kernel Principal Component Analysis and its Applications in Face Recognition and Active Shape Models. *arXiv:1207.3538 [cs]*, July 2012. arXiv: 1207.3538.

[13] T. Wang, C. Rudin, D. Wagner, and R. Sevieri. Learning to detect patterns of crime. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 515–530. Springer, 2013.