#h1 Chapter 18 - Dimensionality Reduction

#h2 Journeys into the subspace

#h3 18.1 - Introduction

#pg In this and the next chapter, we will continue thinking about about machine learning and data science problems in geometric terms. In the previous chapters, we framed our machine learning efforts as finding the right transformations to apply to our input data, as to produce output data with predictive power. We thought about our data as points in a some n-dimensional space, where each dimension corresponds to an input or intermediate feature of interest. For example, we can choose to represent the vital signs of a patient on our ICU as a point in 3-dimensional space, with dimensions body temperature, heart rate, and diastolic blood pressure. Take a large number of such points, each representing a patient, and we might find that the points cluster into two groups that represent two underlying medical states (e.g. hypothermia vs hyperthermia). But is there a more succinct way of representing a patient's vital signs? Do we really need blood pressure if we have heart rate? Does including blood pressure make us any better at detecting hypothermia or hyperthermia? More generally, since these measurements all correlate with each other to some degree, whether negatively or positively, do we even expect all areas within this 3-dimensional space to even have data points? If an increase heart rate is always associated with an increase in body temperature (we are not saying it is, it serves as a hypothetical) then there is an area in that 3-dimensional space that will never have any data points in it: the area where body temperature is high, but heart rate is low. In other words, whereas numerically all points could exist, is our actual data is confined to a specific region within the entire 3-dimensional space?

#im ../../../assets/figures/018/018-01.png 50 256 Figure 18.1 - In the movie '2001 - a space odyssey', a crew of astronauts travels through space on a space ship with a circular structure that serves at the crews quarters and command center. As it is spinning around its axis, it create a centrifugal force. This force pushes them toward the outer wall, is they experience as gravity, and the outer wall as essentially the floor on which they walk. In one classic scene, one of astronauts goes for a jog along this outer wall, with director Stanley Kubrick creating a powerful visual effect, without the help of any CGI. Relevant to our discussion here, is that the outerwall is essentially a real subspace of the circular structure the astronauts occupy. Yes, it is 3-dimensional, but eventhough it loops around on itself, it is serves and is experiences as a 2D plane.

#h3 18.2 - Subspaces

#pg The mathematics, such a confinement is an example of a subspace. It is natural to think of a subspace as values our data can never take on. If our 2D space of a patient's body temperature and heart rate ranges from -200 to 200 degrees Celcius and -100 to 1000 BPM respecitively, you can imagine that a whole section of that space will not have any data. There aren't really any reported cases where someone presented with both a body temperature lower than 0 degree Celsius and a heart rate of over 500 BPM. This is of course a very poorly chosen range for these particular vital signs, but you can probably think of more realistic scales of measurement where which some points simply will never happen. For example, in Figure 18-2 we plot the probability density function of a person's height and age, up until age 20.

#im ../../../assets/figures/018/018-02.png 50 256 Figure 18.2 - The probability density function of p(age|height). At birth, children vary in height around a mean of 50cm. Brighter colors indicate higher probabilities. As they grow up, the mean obviously starts to shift upwards before flattening out at the average height of the adult demographic they belong to. Peditracians use these graphs to check if your child is within the normal bounds of development in terms of height and other physical and behavioral factors and milstones the child should progress through on their way to adulthood., Notice the vast emptiness of most of the probability density function. Probably (and luckily for the mother) no children are ever born with a height beyond 120cm (about 4 feet). Similarly, virtually no adults of age 20 and up are ever under 80cm (2.5 feet). In very rare cases of genetic mutation, we do find adult inviduals smaller than 2.5 feet. The shortest person alive at the writing of this chapter is considered to be Jyote Amge, standing at a little over 60cm or 2 feet tall. Despite these rare and extreme outliers, we still generally consider the probability of a persons height and age to be non-zero in a particular region of all theorectical possible probabiltes: a subspace.

#br

#pg However, a subspace can also be confined to a lower set of dimensions. In our example, our space was 3-dimensional. But what if in that 3D space, in actuality, all points form a 2D dimensional plane, suspended in a particular orientation within the 3D space. Figure 18.3 visualizes this for a 2D plane in a 3D space. When looking at only 2 dimensions at a time, it looks like that the data points are not restricted to any kind of space. But once we visualize the same points for all 3 dimensions at once, it is clear that a lot of theoretically valid data points are missing in this data. But the data still varies along all of its 3 dimensions. Completely discarding one dimension can really only be done if there is absolutely no variation in the data within that dimensions: it is always constant. In all other cases, we rely on dimensionality reduction and basis functions to approximate these subspaces, removing dimensions by transforming data such that one dimension might still vary in the values it takes on, but that this variation can safely be considered noise, and therefore the dimension can be discarded all together (after the transformation, that is).

#h3 18.3 - Decreasing computational complexity

#pg Reducing the number of dimensions has both a practical and theoretical use case. Theoretically, it helps us understand if and how the different dimensions of our data can better explained by a lower number of independent underlying signals. Much more on that later. Let us start with the practical application of reducing and compressing our input data, and how that helps us build better models. Having a large number of features or dimensions in your data might seem like a good idea. After all, the more information we have to train our model on, the better. However, this is true only when those features are statistically independent from each other. If we start adding features that are highly correlated with other features, we are not only not adding any new information, but we might also even make training the model more difficult and prone to overfitting. The second problem of increasing the number of features is the so-called curse of dimensionality. As we add features, the computational complexity of fitting our model grows exponentially, which will cost us either in computing resources needed or time required to train the model, or both. It is not all bad, because even correlated features might contain signals that are not shared between features that are correlated. Therefore, discarding them might hurt us in the long run. Dimensionality reduction provides a way to disentangle the signals that were somehow mixed together into different features. This has the benefit of reducing the computational complexity and removing redudant features, both potentially improving the quality of our models. Another way to think about dimensionality reduction is data-compression. But like some compression methods, dimensionality reduction is not necessarily lossless. Depending on the algorithm, we make assumptions on how the signals could potentially be shared across features. The most well known method of Principal Component Analysis (PCA) is strictly linear, and assumes that any signal of interest is present in each feature to some degree specified by a weight. More recent methods like deep belief neural networks called autoencoders allow for more complex mixtures, but then also risk overfitting the data. In addition, it makes the interpretation of the underlying subspace much harder. Which brings us to the theoretical use of dimensionality reduction and basis functions.

#h3 18.4 - Known underlying signals

#pg Another way of thinking about dimensionality reduction is that we use the geometric and statistical properties of our data to find a small number of underlying signals that are distributed in some unequal way across the larger number of dimensions in our data. One feature dimension might be very tightly linked to one of those underlying signals, whereas other feature dimensions appear more as some kind of mixture of several underlying signals. Imagine having two features A and B in your data that correlate strongly. For high values of A, B is high, and vice versa. It is possible that A is causing B, or that B is causing A. If this is the case, you are probably building the wrong machine learning algorithm. But it is far more likely that A and B are both driven by a 3rd unseen variable. For example, your height is correlated with your body weight. Dimensionality reduction aims to take those two correlated features and extract from them a single signal. In this case, it is easy to intrepret that signal. It is a fundamental law of physics: a bigger container can hold more stuff. Being taller makes your 'container' (i.e. body) bigger, but the type of stuff remains the same. With all other things staying the same, the taller I get, the more I must weigh. This example is very much on the nose, so to speak. More often than not though, it is not immediately obvious how to interpret why, staitistically speaking, our dimensionality reduction algorithm is combining certain features in our data. And in these cases, it falls to us scientists and researchers to understand and define the derived signals. Therefore, before we start our exploration in how algorithms can use geometric and statistical properties of our data to achieve dimensionality reduction, we will first ocus on interpretation, rather than methodology.

#h3 18.5 - The allgory of Plato's cave

#pg In one version of the allegory of Plato's cave we are asked to imagine people confined to a cave, shackled and unable to look out into the open expanse of the world. Instead, the events that unfold in the world cast shadows onto a blank wall towards the back of the cave which they face (The allegory typically includes a fire within the cave to generate shadows, but I find that this simpler version is easier to convey without loosing its powerful idea). The movement of the shadows on the cave wall are correlated with the movement of the objects that create them. Animals and people that are moving outside the cave can get in the way of the rays cast by the sun towards the cave, creating the shadows. But they are mere shadows, without many of the details one would observe by looking at the object creating the shadows directly. I like to think of those shadows as our data. It contains many features which, when combined, yield predictive power. But in many cases, these features are the shadows cast by a lower number of underlying root constructs, processes or mechanisms: the features are merely proxies to the real thing. Plato's allegory is about several philospohical questions at once, but in the current context, it clearly addresses to how things appear to us, depending how they are measured. And how do we relate these observations to the underlying and not directly observable processes giving rise to them.

#im ../../../assets/figures/018/018-03.png 50 256 Figure 18.3 - Wayang is an Indonesian tradition that uses puppet-like figurines to re-enact religious Hindu stories, although they are also used to provide social commentary to current events. But unlike the western puppets like Punch and Judy, the Wayang puppets are flat but very intricate figurines controlled by human who places uses the shadows cast by the figurines on a wall, rather than observing the audience obseving the figurines directly. Moving figurines closer and further away from the light source creates a powerful illusion of depth. With the audience observing only the shadows cast by the figurines, it is a interesting and novel visual interprestation of the Plato's cave allegory.

#h3 18.6 - Diving into the ocean

#pg For those who feel the allegory of the cave is too esotheric, we can think of the meaningful insights dimensionality reduction (and manifold learning) have to offer in a more familar context. The term "construct" mentioned above is most often associated with psychology, and especially in research on personality types. Psychologists have long wondered how to define, describe and measure personality in a meaningful and quantifiable way. What is a personality made of? What are its parts? And how many different parts are there to a personality? Until recently, the accepted answer was the OCEAN personality test, where OCEAN is an acronym for openness, conscientiousness, extraversion, agreeableness, and neuroticism, sometimes referred to as the 'Big Five' descriptors of personality. These are the 5 semi-independent personality traits the researchers argued combine to create a specific personality. Their research indicated that you can create any personality by varying the strength of these 5 traits. But how did they get there? How did they know 5 was the right number, and how did they know that corresponded to these particular 5 traits? They used a method called factor analysis. Factor analysis is a dimensionality reduction method, specifically to find so-called latent variables. Think of a latent variable as the process that produces a signal of interest that is subsequently distributed across the features in our data. Factor analysis is often used interchanglable with PCA, since on the surface the methods have so much in common. But there are subtle differences though, not particularly relevant to our discussion, but something to keep in mind when you encounter one, the other, or both. In personality research,psychologists refer to these latent variables as traits. Traits are the qualities that make people different, like the stats of a character in an role-playing video game. Some people are introverts, some are extraverts.

#im ../../../assets/figures/018/018-04.png 50 256 Figure 18.4 - The five personality traits that make up OCEAN. I must be high on the neuroticism scale because the first thing I noticed is that the person who made this figure hadn't even ordered the traits by the order of the letters of the acrynomym.

#br

#pg Some people are high on openness, making them curious and not afraid of change. Score low on openness instead, and you are a more cautious person. But remember, the traits are latent, we can't observe or measure them directly. They are inferred. Experimentally, we will have to resort to studying the shadows cast by them: their proxies. And in psychology research, those shadows are very long and tidious questionaires about how you feel about and what you would do in certain situtations. You can imagine each question as a feature that reveals some mixture of the personality traits by the way you answer that question. And with enough questions, and enough people willing to answer them, you will see a pattern emerge. Certain pairs of questions are consistently answered in the same way. Some are consistently answered in oppositie ways. Factor analysis showed that all the answers to the questions could be predicted with reasonable accuracy by assuming that there were 5 traits underlying your proposenity to answer each question this way or another. A vector of 5 numbers that captures most of your personality. Factor analyisis had no issue identifying these 5 different traits statistically from the data, but obviously did know how to interpret them. That (currently) still requires humans. In any dimensionality reduction approach, your new lower dimensional space of constructs, latent variables, components, or signals is likely to be theorectically meaningful. For OCEAN, the psychologists associated the grouping of certain questions with their knowledge of psychological constructs, such as extraversion, neurotisicm and agreeability. Constructs that in isolated already seem to differ substiantially between individuals. The step forward was that a specific subset of 5 of these known traits were sufficient to explain most of the variability in invidiual differences in personality.

#im ../../../assets/figures/018/018-05.png 50 256 Figure 18.5 - The word cancer comes to us from Hippocratus, father of modern medicine. In astrology, and has as its representation a crab. In Greek mythology, this is a reference to the crab that pinched Heracles while he was fighting the Lernaean hydra. Hippocratus and his students adopted the word cancer, probably because of the finger-like spreading projections from a cancer remiinded them of shape of a crab.

#h3 18.7 - The epidemiology of cancer

#pg In this book on the topic, Author Siddharta Mukerjee calls cancer 'The emperor of all Maladies', for a good reason. In fact, for several very good reasons. The disease has made and continues to make a societal impact like no other disease. Feared because no one seems to be outside its reach, and feared for the devestation and suffering it can bring to our lives. But also feared because we really don't know what exactly we trying to cure, let alone how to do it (although of course incredible progress has been made). And feared because it is many diseases at once, with so many different causes. So many things appear to increase or descrease our risk in some measurable amount, but hardly ever in isolation. Whether or not you develop a specific cancer is up to a complex array of interacting factors, both genetic and environemental. Unlike other diseases like malaria (known enviromneetal cause) and Hunginton's disease (known genetic cause). the epidemiology of cancer is extremely complicated. This is why this chapter is perhaps a bit longer then you might expect from a technique that is'nt essential to machine learning, especially for the newer generation of neural networks that are designed to naturally take on this process themselves by their very architecture. We spend time on it because it will hopefully prime you the (geometric) patterns that underlie our experimental observations or even our every day experiences. When it comes to cancer, think about applying dimensionality reduction on a data set containing demographics of people at risk. When we build models capable of accurately predicting who is at greater risk than average,we can take necesssary steps to improve treatment and preventation, and that is a win of clinical relevance. But there is also a possible insinght our algorithms have found for us in their attempt to become as precise and accurate as possible. Dimensionality reduction methods, as well as the manifolds we will discuss in the next chapter, can all help us form new hypotheses to test, leaning on the systematic patterns they have found that link inputs and outcomes. As our algorithms grow more complex and non-linear, they wil be better at predicting. The downside of this is that the increased levels of complexity become much harder to interpret. Conventional methods and algorihms might be limited in their abilities, but they might still yield some valuable insights. Algorithms like Principal Component Analysis.

#h3 18.8 – The heart of PCA

#pg Indeed, PCA is still the tool most data scientists and machine learning engineers will reach for first. Both the math and the algorithm applying the math are not necessarily trivial. Since we want to convey intuition, we will work through a (largely visual) description of the underlying math, asking for the reader's suspense of disbelief on some of the proof we will ommit. Again, get the geometry behind the method and you'll hopefully have that 'aha' moment where, when given a blackboard and piece of chalk you can draw out not the formulas, but some visual representation of the geometry underlying it. By understanding, at a conceptual level, what PCA is all about makes grasping other dimensionality reduction algorithms a lot easier. And not just dimensionality reduction algorithms. It will also set up our discussion on manifolds in the next chapter, which represents in some way the superclass of all algorithmic objectives we have already encountered, including regression, clustering, classification and dimensionality reduction itself. And in order to understand PCA, we actually need to understand the method it relies on: singular value decomposition (SVD). PCA adds a bit of mathemtical flourish and convinience, but SVD is where the magic happens. And this magic (which is of course just math, and not magic) is finding the so called eigenvectors and their associated eigenvalues in our data. What these are and what value they add is perhaps best an understanding (or at least intitution) build from the group up. So, let us start there.

#im ../../../assets/figures/018/018-06.png 50 256 Figure 18.6 - The vector v = [2,1] in panel A can be made twice as long by multiplying each of its elements by a scalar, 2 in this case, giving us v'= [4,2]. as shown in panel B. If we want to rotate the same vector by 30 degrees, we need to apply a transformation matrix, because the new x coordinate of v now depends on the orginal x and y of v, as does the new coordinate of y, see panel C.

#h3 18.9 – Points as vectors

#pg If you have already worked through Chapter 12 on transformations, you remember you can implement and combine different types of transformations you apply to a set of points. Translation, rotation, shearing and scaling could all be accomplished by finding the appropriate parameters to go into your transformation matrix T. As we discussed previously, a point is nothing more than a coordinate in some arbitrary space. In mathematics, a point is defined as relative to some other point, (usually the origin), which we call a vector. When we write down v = [2,1], we actually define a vector from [0,0] to [2,1]. A vector conveys a direction in data space, with its length representing the distance travelled in that direction. The length is not the number of elements of the vector, we refer to that as its size (or dimensionality), rather it is the euclidean distance between the origin [0,0] and [2,1]. Which, in our case, is √(2^2 〖+1〗^2 ), which is √5, or roughly 2.236. Imagine if we wanted to make this vector twice as long, but still point in the same direction. How would we do that? Well, the really nifty thing about vectors is that that only requires us to apply a scalar (a constant) to each element in the vector to make that happen. In our case that gives us: 2∙[2,1] = [4,2]. Did this do the trick? Computing the length of the new vector yields: √(16+4) = √20=2√5 ≈4.47. It did! In other words, the transformation we wanted to apply to our vector (making it twice as long while still pointing in exactly the same really only required a single scalar value. This might not seem like a big deal just yet, but keep that thought with you as we dive a little deeper.

#im ../../../assets/figures/018/018-07.png 50 256 Figure 18.7 - In Flatland, we are asked to imagine what a sphere looks like to the Flatland inhabitants, which live in a world that only has two dimensions. How would it appear if its 3 dimensions started to intersect with the two dimenions of Flatland?

#h3 18.10 - Flatland

#pg In the famous novella 'Flatland' author Abbott describes a world called Flatland consisting of only 2 dimensions with little shapes happily going about their business traversing the 2 dimensional plane of their existence. Our protaganist, a square is one day visited by a 3D dimensional being (a sphere) crossing paths with his 2D dimensional plane. This leaves the 2D dimensional beings completely flabbergasted. They can't perceive the sphere in its full 3 dimensions. Instead they only see a 2D slice of it. The square needs to come to terms that the behavior of the sphere from his 2D point of view is similar to the shadows cast on the cave wall in Plato's allogory. He does manage to grasp this idea of Spaceland, the world of the sphere and other 3D shapes, but is unable to convinve his fellow Flatland inhabitatnts. Even worse, he is imprisoned for his radical ideas. By the way, the novella is both an interesting exercise is imagining geometry beyond what you are used to, it is also a social commentary, with shape complexity and dimensionality seveing as analogies to societal status and class.

#h3 18.11 - IndependenceWorld

#pg I want you to do a similar mental exercise. Imagine a world where we can observe and measure exactly the same things as in our own reality. People have heights, weights, heart rates, blood pressures, education levels and zipcodes, flowers have colors, we can sequence genes into strings of ACTG basepairs, and we can keep track of the weather, the rise and fall of the stock markets, and animals migrating across the planes of the Serengeti. For the sake of argument, let us focus on measuring quantities that, if we have enough of them, clearly appear to be normally distributed, such a height, the average retirement age of NFL players and ACT scores. That is a feature this imaginery world has common with ours - the normal distribution underlies many observable quantities. But here is the twist. In this new world, none of these measurements correlate with any of the other measurements. Each of them is completely independent, and without any correlation there certainly cannot be any causation either. If I imagine this world, the word 'frightning' comes to mind. Because there is literally nothing you can use to predict anything. What will happen next, why things happen. And since your behavior itself is also completetly independent from everything else, you would'nt be able exert any type of influence on this world anyway. Although by the same token nothing in the world can exert an influence on you, so at least we got that going for us. I have dubbed this imaginary world 'IndependenceWorld'. In mathematical terms, we can imagine IndependenceWorld has n different processes that each produce data completely independent from all the other processes, but of course do show variation within themselves. Things still change in IndependenceWorld, they just don't depend on anything else changing.

#im ../../../assets/equations/018/png/018-01.png 50 16

#im ../../../assets/equations/018/png/018-02.png 50 16 Equations 18.1-2 - A matrix B of size m x n, multiplied with an identity matrix I of size n x n yields exactly the same B. A matrix B of size m x n, when multiplied with a transformation matrix M of size n x n yields a new matrix X with dimensions m x n, reflecting the transformation matrix M applied to our matrix B.

#br

#pg This is easily simulated by drawing, for each process n, values from a normal distribution of μ = 0 and σ2 = 1. Within each dimension (representing a process or something we can measure), everything changes from one data point to another, but never as the result of a change in any other dimension. By chance, perhaps it may occasionaly appear that they do, but paraphrasing the disclosures at the end of movies: "the dependencies observed in our data are fictitious. No resemblance to any dependence between any quantifable process captured in IndependenceWorld is intended or should be inferred".

#h3 18.12 - Escaping into the CovarianceUniverse

#pg I described IndependenceWorld earlier as a frightning state of affairs, and I for one, would like to leave. But are we mathematically stuck in IndependenceWorld forever? Fortunately, we are not because there is a way out. We can apply a transformation to IndependenceWorld that eleminates its total internal independence. But what kind of transformation will do the trick? As you might recount from Chapter 12, an identity matrix I is a matrix with 1s on its diagonal, and 0s elsewhere.

#im ../../../assets/figures/018/018-08.png 50 192 Figure 18.8. (A) Our source data B has been created as to have zero covariance. You can confirm this by observation - the points are not oriented in any way. Transforming the data using an identitty matrix (B) yields X (C), identical to our source data in (A). No covariance has been introduced - with the covariance matrix computed from (C) being close to the identity matrix (B) we used to transform the data. Since points were created randomly from a distribution, and only a limited number of points were used, the covariance matrix (D) isn't identical to M in (B), due to noise in the limited sampling of random values to generate the data. The green arrows represent the original (A) and the trasnform axes (C).

#br

#pg If we use an identifty matrix as a transformation matrix and apply it to some matrix B, the result is exactly that the matrix B, see Figure 18.8. But what about anything but an identity matrix? What if we translate the points?

#im ../../../assets/figures/018/018-09.png 50 192 Figure 18.9. When we translate the points, they will no be centered around new origin, buy we are not introducing any covariance. Panel contents identical to Figure 18-8.

#br

#pg As you can see in Figure 18.9, the points are indeed translated, but their relative position remains completely unchanged. No covariance was introduced by this type of transformation. Perhaps if we scale the points instead? This, unfortunately, will stretch and/or compress the data along each axis, but it will do so independently of the other axis, see Figure 18.10. So, again, we hit the walls surrounding IndependenceWorld. The only way out and into a new domain we call the CovarianceUniverse, seems to be introducing non-zero values to the off-diagonal

#im ../../../assets/figures/018/018-10.png 50 192 Figure 18.10. When we perform a pure scaling, points along each axes will get stretched out or compressed, but completely independently from eachother, introducing no covariance. Panel contents identical to Figure 18-8.

#br

#pg elements in our transformation matrix. ¬ In Chapter 12, we did just that, by constructing a transformation matrix M of size n x n, which, unlike our previous three transformation matrices we used so far, has non-zero values on the off-diagonal elements as well. Specifically, we create matrices that allow us to rotate and shear our original data. Rotation seems promising, but in isolation, it is not introducing any covariance either, see Fig. 18.12. Put differently, the relative position the vectors point to stays exactly the same. And this is a logical consequence of our source data, designed to be completely symmetrical around the axes. However, rotation does work when it is combined with a scaling.

#im ../../../assets/figures/018/018-11.png 50 192 Figure 18.11. A pure rotation rotates the axes of the space.But since the data comes from a distribution that is symmetrical along its axes, no covariance is created. Panel contents identical to Figure 18-8.

#br

#pg By simulatously rotating and stretching/compressing the data along each axis, we do end escaping IndependenceWorld, as you can see in Figure 18.12. And finally, what about shearing? Interestingly enough, that is the only type of transformation we already examined that can, by itself, introduce covariance into a matrix without it.

#im ../../../assets/figures/018/018-12.png 50 192 Figure 18.12. A rotation combined with a scaling can introduce coviarance in our data. After this joint transformation, the points are both oriented and scaled differently along each new axis. Now, we see clearly how x and y have started to correlate. Panel contents identical to Figure 18-8.

#br

#pg Geting an intuition of why these different transformations, can, cannot or can sometimes introduce covariance under the right circumstances, is understanding the geometry of your data. We urge you to play around with the notebook that comes with this paragraph to convince yourself of what we outlined here, and hopefully get a feel for the geometry of your data given a covariance matrix or the change in geometry introduced by a transformation matrix, should you encounter one.

#im ../../../assets/figures/018/018-13.png 50 192 Figure 18.13. A pure shearing can introduce covariance on its own. However, we should not equate it directly to the scaling and rotation operation we outline just before. Notice how in 18.13, the axes in the new space are no longer orthogonal, whereas in Figure 18-12 they are. This is an important observation, and will be crucial in understand what eigenvectors really are. Panel contents identical to Figure 18-8.

#h3 18.13 - Color as useful an analogy

#pg With this new understanding how we can introduce covariance to a dataset under our belt, we can start to think about reversing this process. After all, our data always comes from the CovarianceUniverse to begin with. IndependenceWorld is merely a mathetical construct that will help us understand what covariance we can remove because it adds no value, and how this covariance came about to begin with. In this context, dimensionality reduction is the process of reverse engineering, and determining what parts of the process we are reverse engineering weren't necessary part sto be included. The previous notebook presented transformations and their effect on the covariance structure of your data in 2D, without any reference to what the axis represent. In the this paragraph (and notebook accompanying it) we will look at the same concepts in using an intuitive set of dimensions you are (hopefuly) quite familiar with already: color. And throughout the next sections, we will use continue to use color as an example as a data space, to hopefully instill in you the intuition of PCA, SVD, eigenvectors and eigenvalues.

#ts

#th

#tc Red

#tc Green

#tc Blue

#tc Vector

#tc Color

#tl

#tr

#tc Off

#tc Off

#tc Off

#tc [0,0,0]

#tc ../assets/figures/018/018-14a.png

#tl

#tr

#tc Off

#tc Off

#tc Off

#tc [1,0,0]

#tc ../assets/figures/018/018-14b.png

#tl

#tr

#tc Off

#tc Off

#tc Off

#tc [0,1,0]

#tc ../assets/figures/018/018-14c.png

#tl

#tr

#tc Off

#tc Off

#tc On

#tc [0,0,1]

#tc ../assets/figures/018/018-14d.png

#tl

#tr

#tc On

#tc On

#tc Off

#tc [1,1,0]

#tc ../assets/figures/018/018-14e.png

#tl

#tr

#tc On

#tc Off

#tc On

#tc [1,0,1]

#tc ../assets/figures/018/018-14f.png

#tl

#tr

#tc Off

#tc On

#tc On

#tc [0,1,1]

#tc ../assets/figures/018/018-14g.png

#tl

#tr

#tc On

#tc On

#tc On

#tc [1,1,1]

#tc ../assets/figures/018/018-14h.png

#tl

#te

#ca Table 18.1 - Light emitting devices still use combinations of primary colors red, green and blue to create their full gamut of possible colors to display

#br

#pg With some odd exceptions, all light-emitting devices (screens) create their wide range of colors by mixing together three primary colors in different ratios: red, gree, and blue (abbreviated to RGB). Red plus green is yellow, green plus blue is cyan and all three combined in equal proportions gives you a neutral gray. RGB is thus a cubic data space with its 8 corners are defined as the 8 maximum states the monitor can be in, outlined in Table 18.1. And along each dimension, only one of the channels (red, green, or blue) varies. Our various escape attempts out of fictional IndependenceWorld can be understood more deeply using color, rather than arbitrary, hypthetical or real but less intuitive dimensions of data, at least in our opinion. For example, we can think of the effects of the transformation swe detailed in the previous sections would have in 3D on a color image, like a photograph. If you used any of the photosharing apps out there, and have used some of the filters to make your image look more vibrant or warm, or some other color enhancing or alterating effect, you will in effect have applied a transformation to all the pixels in your image. Translating will shift the white point, scaling can make your image more vibrant, and rotation can yield some psychedelic looking images indeed by circularly shifting all the hues. Case in point: in IndependenceWorld we had three axis that were perfectily orthogonal to eachother. Along one axis we only varied red, on another only green and on the last one blue. If we consider red, green and blue to be truly be the primary colors (more on that later), then at least the RGB space garantuees that they arae completely indpendent of eachother. After applying any transformation to such a space, we can easily recognize how the new space an its axis allows us to mix colors together.

#h3 18.14 - Reversing the Process

#pg Now that we have realized we can introduce covariance into a dataset by transforming a dataset without any coviarance we can start to entertain the idea of reversing that. Again, keep in mind that in reality, there are no real matrices B or M, as there is no IndependenceWorld. What we start with is matrix X, the matrix of actual measurements that serve as inputs to our models, brought to you curtousy of a world in which most if not all things eventually correlate with eachother. inputs . What we are not looking for and trying to avoid is heavy But we can theoretically construct B and M such that they yield our data, by means of equation 18.2. Ironically, covariance is something we are both actively looking for and trtying to avoid. We are looking for covariance between inputs and outputs. Without it, we would be able to ever predict anything from the covariance between our input variables. As we discussed above, this poses all kinds of problems, computationally and mathematically. Getting rid of (some) if it beforef we start training our models can be beneficial.

#h3 18.15 - Is there life on Mars?

#pg Again, specifically using color as our data space gives us quite a bit of insight without much needed complexity. Imagine we are processing pictures taken by Curiosity on Mars. Those pictures vary in blue and green, but they are all pretty much heavy and constant on that famous reddish haze of Mars. Any meaningful detail therefore is in the blue and green channel, obscured in large by all the red. But in this new space we created, we can't get rid of the red dimension, because, well, there is no red dimesnion. Each axis is a combination of all three colors. Now we have our mathematical objective. We want to find a matrix T such that its inverse T-1 gets us to a space where the new dimensions are 1) independent of eachother and that it 2) redistributes any meaningful variance so that the first dimension explains the most variance in the data, the second whatever is left over after that, etc. The minus -1 superscript here doesn't mean we are raising M to the power of -1, in the same way we write squaring a variable x as x2. Instead it is the inverse of a matrix T. Mathematically, the inverse of a matrix (X-1) is defined as the matrix that when multiplied with the original matrix X yields the identity matrix. It is more useful to think about it as another really useful geometric and algebric trick: if the transformation matrix T get your from B to X, the inverse of that matrix gets you back to B from X. Computing the inverse of a matrix is not trivial, for now take our word for it and use the reference material to learn more about how to compute inverses of matrices. It is safe to say that many mathematicans probably love computers for the fact that they can compute inverses so quickly. Doing such inverses by hand starts to get virtually impossible beyond a certain low number of dimensions.For our RGB mars pcitures that means making sure that the meaningful information in our data (blue and green) all end up in the first dimensions, and the meaningfless noise of the red haze ends up in the last dimension. In that way, we can choose to remove it, since it has little value (remember: we are attempting dimensionality reduction), leaving a space that which we only have variations in green and blue. There is a particular set of vectors that when combined into a transformation matrix (we have 3 vectors in our current example) that orthongalizes the axes of our space. And there is a particular value that tells you, for each of these vectors, how much of the variance in your data each of those 3 vectors explained. The former is a called an eigenvector, the latter an eigenvalue. In principle you can use an infinite number of transformation matrices to warp the space, but only the eigenvectors will do so in a way that makes the new space have orthogonal axis. How we actually compute these eigenvectors and eigenvalues not trivial and certainly beyond what I am comfortable explaining, since I am not a mathematician. In addition, it won't give you a deeper understnading of the key insight and intuition. You will never code it yourself, as many efficient implementations already exist and have existed for many years. But by all means, follow the references if do wish to know more.

#h3 18.16 - Eigenvalues and Eigenvectors

So, what we set out to find is a transformation matrix M such that we can transform B to X and X back to B.

(18.3)

(18.4)

Equations 18.3 and 18.4 - A matrix B of size m x n, when multiplied with a transformation matrix M of size n x n yields a new matrix X. This process can be reversed completely by subsequently multiplying X by the inverse of M. The -1 is mathematical notation for the inverse of a matrix.

#h3 18.17 – Singular Value Decomposition

#pg Singular value decomposition is an extension of the method used to find the eigenvectors and eigenvalues. Finding the eigenvectors and eigenvalue got us of to a great start, but we didn't have all the necessary pieces to create S. S stands for scores the matrix where the dimensions are orthogonal, and sorted based on how much variance in the data they explain. But it isn't quite the same as the matrix B we imagined earlier. In matrix B, all the vectors were of unit length (which is a fancy way of saying that they were all 1). The matrix we want is matrix S, because indivudal dimensions are not required to be scaled the same, because we acknowledge that all mixtures of the three orthongal axis can happen. You can see that in figure 121212. So, rather than finding kust the eigenvector and eigenvalues, SVD factorizes (math breaking down something into its pieces) our data X into three matrices U, Sigma and V. Now we have the remaining piece of the puzzle: matrix U. Sigma now represent singular values, are related to eigenvalues by means of squared: singular value = eigenvalue^2. And V contains our eigenvectors. The matrices are also sorted such that later dimensions explain less and less variance. Finally, there is U, which is a diagonal matrix of mxm. This matrix tells us how, for each axis indepedently, each input was scaled. If you measure the height of a large group of people, they will all be different, which some of differences between truly indepedently of other things we might measure for those same poeple (e.g. weight, age, shoesize). That was what was lacking in B: in that matrix height was isolated to a single dimension, but its value was normalized with the other values in the vector (weight, age, shoesize) so that the resulting vector of a single data point was always 1. Again, how we actually factorize our matrix X into threse three matrices is outside of scope of this course, but feel free to use the references to learn more.

#h3 18.18 – Combining all ideas into a single framework: using PCA in practice

#pg If you want to know what the difference really is between PCA and SVD, I suggest you do not Google it. You heard us correctly. Do NOT Google it. With all the best intentions in the world, the internet forums on this topic are all over the place, with wildly different answers and explanations, and references. For me PCA is to SVD what chemistry is to physics. For a long time science considered chemistry and physics to be separate fields, each governed by their own laws. With the atomic model, this all changed. Chemistry, it turned out, was just physics. It is the same with PCA and SVD. PCA specifically aims to reconstruct B, which in PCA parlor is referred to the matrix of principal compoments or PC. It reveals the underlying mechanism or process that generates data like X. It is not specific to our particular data set X, it is valid to any data set with the same dimensions as X. Think of it as a function that you can parametrize in a certain way to create your own data. Use different paramters and youll get a different data set, but one that still is generated by the functions expressed by the PCs. The scores matrix we obtain from runing SVD contains both: the function and the paramter specific to our data X and only our dataset X (curtosy of matrix U). In pricniple, PCA can use a variety of methods and assumptions to get to these PCs, but it turns out the most effective way is (drumfill).... SVD! Scikit-learn definitely uses SVD under the hood for its own implementation of PCA, and I assume other toolboxes do the same. Since you will never be coding any of these methods yourself (hopefully), it is therefore just solid advice just to use an accepted PCA method, which in all likelihood will return everything that you need, including the intermediate results obtain by running the SVD algorithm internally. To recap, Figure 12.121212 is one of the best visualizations I know off explaining PCA, SVD, eigenvectors and eigenvalues. Use this as a reference if you are stuck in your reasoning while learning or actually using these methods. I definitely do.

#h3 18.19 – PCA - eigenfaces

#pg Eigenfaces were introduced by a now classic paper by Turk and Pentland back in 1991, long before we had the powerful neural nets we have today to detect and recognize faces using AI. Their solution was clever. Not only did it find a way to represent the image of human face using only a small vector of about 20 elements, rather than a the nuber of pixels in a rtpical image (128 x128 = 65300 elements), it also, at least to me, introduced the idea of that macnine learning could do more than just predict and categorize. It could also create. The notebook will show you in detail how Turk and Pentland applied PCA to images of faces, but I will briefly outline it here. An image, in the end, is just a vector. Yes, it is 2D, or sometimes 3D if we have a dimension for our colour channels, but we can flatten the whole thing into a single vector. If you have enough of those vectors (meaning you have enough pictures of faces) you can stack them all up end into a matrix X where each row represents a particular pixel position (and color channel if you use color images) and each entire column represents one vector created from flattening a single image of a face. It is this matrix on which we perform PCA. As outlijned above, PCA return the principal compments, ordered by the variance they explain in the data. Before you read on, can you think of what is likely to be single largest pricinpcal compnent across a lot of images of faces? If you reasoned that it must be the average across all faces, you are correct. That is by far the biggest principal compenent. Yes, all faces are differnt, but the global structure is pretty much the same. And if the images are all taken from a reasonable similar distnace and angle and light condition, that first component when cvisualized looks like a very average face. Not paritcularly old, or young. Note particularly femine or masculine, not particularly anything. It is actually the next n components that allow for that particular flavoring. Although you will not get one component for each semantic difference we can think off faces differ in (age, gender, race, ethnicity, etc), the weights of each idnviiduals principal componetns will be slighty but consistently different between different indivdudals. The vector of these weights is like a low dmensional finger print (or face print if you will) of your entire face. The only thing the algorithm has to do when it see a new iamge, is compute the weights of that image on the pricinapcal components and determine which vector in its database of known indivudals it is most similar to. But we don't have to stop here. In principal, with a large enough number of images, we can build either a completely new face by setting our weights to some random combination of weights seen before in the actual faces we used. You can also find look a likes, simply by finding people that have vector of weights that are similar to yours. This is one way certain apps work that tell you which celebretiy you most cloely resemble. Modern neural networks can produce faces with a degree of reasism that at least when seen as iamge, humans can no longer distinguish from real. And deep fakes are showing we are getting there with video as well. But all of those ideas were first expressed in this seminal paper by Turk and Pentland, all the way back in 1991. Interestingly, a year later Neil Stephenson's snow crash was released. In it he not only described things like the metaverse and avatars and other what seem like new and exciting ideas of the future, he coined those terms in that book. Talk about being visonariy.

#h3 18.20 – ICA - Source Seperation

#pg Independent Component Analysis (ICA) shares the same goal as PCA. Both are trying to find a set of basis functions (preferably as little as possible) that are sufficient or sufficient enough to explain the observed variance in the original data. However, ICA differs in two important ways. First, it doesn’t require the components to be strictly orthogonal. In addition, it attempts to distribute the explain variance in the data equally, as opposed to PCA which will strive to put as much variance in a single component as possible before considering a second orthogonal component. Unlike PCA, ICA does require some parameters to be set that will need to be adjusted to be optimal for its use case and data. Perhaps more important, strictly speaking ICA is not a dimensionality reduction technique, or least it was not intended to be one. Rather, ICA's main application area is to seperate signals that have been convolved and mixed together. For example, if you had two or more audio sources in a room, recorded through two different microphones,, ICA will be able to separate each speaker (to some degree), while PCA would fail. We have created a little mini LEGO world to demonstrate this, as shown in Figure 18.x - 18.y. You are reading this on a device with audio, you can click the figures below to hear the diffferent sounds in the scene, and how ICA is able to isolate each on of them, letting the patient focus on what the doctor has to say.

#h3 18.22 – Basis Functions - Heart Rates

#pg Even if you have never been in a hospital and have seen or had an Electrocardiogram in real life, we are all familair with that green line moving across the screen occasionaly spiking upward as the person heart beats, accompanied by a brief beep. Any medical procedural show will feature a shot of one, flat lining or not in every single episode, in everyt single season. However, what the ECG measures is a little bit more complicated than just a beat. The heart is very complex system of contracting muscles, and more like an orchestra playing a complicated piece of music than a synthesizer producing a single bleep. There are different stages to a heart beat, with chamber filling up with newly oxygenated blood, valves closing and opening, msucles contracting, etc. An ECG measures the electrical impulses your nervous system produces to make all of that happen in the correct sequence and at the correct time. If you look at a stereotypical recording of just a single heartbeat, shown n figure 212, you can see that it is actually a bunch of different signals combined, with a specific sequence of peaks and throughs in the electrical current recorded representing different underlying processes. The five most dominant features have been assigned letters PQRS and T. The events Q, R and S happen very close together and are commonly referred to as the QRS complex. It is at this time, your heart rate monitor will beep if it detect it. Abnormalties in how the heart function can be detected by looking the indivudal events PQRST and their relative amplitude, the distances in time between them, their presence or absence, etc. If the pattern is very different than the norm, there is likely a cardiological problem. What kind depends on how the pattern deviates from the norm. We can use a basis function approach take these complex timeseries data and instead represent them as some weighted mixture of basis functions. That way we can represent some as complex as a heartbeat by a vector of n basis functions, which makes detecting and understanding abnormalities so much easier. The choice of the correct basis functions that models each bit of the PRQST process idendepently (one basis function for the P peak, one for the R peak, etc). is something that I imagine is studied extensively. In the notebook, I simply divide the recording of each heartbeat into segments of equal duration, and that already seems to do a reasonable job at detecting abnormal heartbeats. If this is of interest to you, we of cours encourage you to do some follow up research on what undoubtbly are better basis function models.

#im ../../../assets/figures/018/018-01.png 50 128 Figure 18.X - Jean-Baptiste Joseph Fourier with the equation named after him for his work on the theory of decomposing signals into combinations of sines and cosines.

#h3 18.23 – Basis Functions - Fast Fourier Transform

#pg Not a lot people will think of the fourier transform as a basis function method. However, on closer inspection there are many similarities, enough to discuss it the this particular type of transform in the current context. The fourier transform, named after influential mathematician Jean-Baptise Joseph Fourier, is a mathematical transformation that decomposes signals varying in time or space into the frequency domain.

#im ../../../assets/figures/018/018-01.png 50 128 Figure 18.X - Anatomy of a sine wave. A sinewave is characterized by an amplitude, frequency and phase. If the sinewave has a frequency within the human range of detectable frequency, this frequency correlates with pitch, and amplitude with volume. Phase is the relative shift of the peaks and troughs relative to some baseline. Humans can't hear absolute phase, but relative phase between two or more signals can be perceived and is used by the brain to localize sounds in space.

#br

#pg Fourier claimed that any signal could be decomposed into a set of cosines and sines, although that claim turned out too broad. It is only true for stationary signals, that is signals that vary of time, yet their statistical characteristics stay the same. Luckily, many signals fall into this category, so Fourier's big idea remained as relevant as it is useful. It is most often applied to timeseries data, but has many applications in the space domain as well, espeically when it comes images. The core idea remains the same. Any signal can be decomposed into a finite set of sines and cosines varying in frequency and phase, each at a different amplitude.

#im ../../../assets/figures/018/018-01.png 50 128 Figure 18.X - Moving into the frequency domain reveal interesting features about your data you can't necessarily observe in its original form in the time domain. The signal in panel A is the combination of three sinusdial signal, each at a different frequency. When we apply the fourier transform on this signal, we can visualize the same signal in the frequency domain as a power spectrum. The three peaks correspond to the three frequencies used to create our signal in panel A. We can corrupt the signal significantly by adding white noise. If you use the notebook accompanying this paragraph, you will have a hard time even hearing the three original frequencies, in as much as you can no longer really see them in the signal with noise added in panel C. However, in the frequency domain, this noise is equally distributed across all frequencies, and the three peaks remain clearly visible. Furthermore, we can now filter our signal by setting all frequencies with a power below some threshold to zero and move back into the time domain. The reconstructed signal is virtually identical to the original signal and the white noise is no longer audible after our filtering operation.

#pg A good practical example of how to interpret the result of a Fourier transfrom is by applying it to the waveform created by recording the perfomance of a musical piece. During the recording, the thin membrame within a microphone will be pushed and pulled by the pressure waves the musical instruments produce (the thing that humans call sound, provided the frequency is within the range of what humans can perceive). This pushing and pulling over time is converted into voltage and digitized into postive and negative values and stored as sequence called a waveform. Playing back the waveform uses the same sequence to push and pull a different, much larger membrame of a speaker, which in turn reproduces the pressure waves the musical instruments produced during the recording, which we perceive as the original performance itself.

#im ../../../assets/figures/018/018-01.png 50 128 Figure 18.X - Most western music uses a specific set of ratios between each of the 12 notes that make up an octave. At each higher octave, the ratios are the same, with the base frequency doubling for each octave we move up. Although the last century has seen experimentation with musical styles that incorporated all 12 tones into a single piece of music, usually a subset of the 12 notes form a scale used throughout a single musical arrangement. You are familiar with the major and minor scales, but other used scales are the pentatonic, dionic and the beautifully named myxolydian scale.

#pg Modern recordings use a sample rate of 44000 per second, meaining 44000 numbers are needed to record one second worth of sound. The frequency of a sound determines its perceived pitch. And western music has settled on a particular set of intervals between frequencies that produce the do-re-mi-fa-so-la-ti-do progression of full note found in western music. In reality, there are also 5 half tones interspersed in the seven full tones ratios, given us 12 notes in total: C, C#/Db, D, D#/Eb, E, F, F#/Gb, G, G#,Ab, A, A#/Bb, and B. The C see that comes next is exactly one octave higher than the previous C. I am stressing western music because else where in the world, those intervals can be slightly different. Throughout Asia, through the middle east to to South East Asia, there are musical scales that have different intervals that feel absolutely natural to the people who grew up with them. To us they are likely to sound out of tune. The origin of these particular intervals across all forms of music is fascinating (at least to me). There are many theories on why specifically these ratios seem to 'make sense' to us. Is it merely exposure? Or this there something biophysical that makes these ratios preferred? There are a great number of books on this topic, I've listed a few in the refernece section. The one thing these book do seem to agree on is that music appreiciation is one of those really uniquly human qualities. Back to the mathematics. Regardless of why particular frequencies and the ratios between them seem to work, we use them. Consider Mozart's Clarinet concerto in A major (K-622). The fact that it is in A major tells use that certain frequencies will dominate the waveform of a recording of this concerto. Most likely, the three frequencies at the 4th octave that form the A major chord: A (440Hz), C# (554.37Hz) and E (698.46Hz). Some instruments, like the bassoons, will play the same chords but at a lower octave., other instruments like the violins might play them at a higher octave. You can figure out the frequencies at lower (or higher) octaves for any chord and just halving (or doubling) the frequencies of your current octave. And this is where the fourier transfrom comes in. By applying it to the waveform, we know can see the power of each frequency in our waveform. Power is eseentially a measure of relative presence. The higher the power of a frequency, relative to others, the more that frequency dominates the sound you perceive. Representing a timeseries in the frequency domain gives you a very convenient way to study and change the waveform that gave rise to the it.

#im ../../../assets/figures/018/018-01.png 50 128 Figure 18.X - 3 different ways to look at the first 30 seconds of Mozart's clarinet concerto in A Major. The top panel represents the waveform, which only really tells you something about the increase and decrease in overall amplitude. The middle panel shows the most dominant. We definitely can see the typical intervals between notes we expect to find in western music. What is perhaps surprising, is that although the A of A major is very prominently features in the piece (440Hz is the A the clarinet opens with), the other 2 frequencies of the A Major are less obviously present. It seems other notes of the A major scale were favored in this piece. Perhaps this is why Mozart's music stands out relative to his contemporaries. Perhaps this was part of his secret or intuitive recipe for composing great music. The bottom panel combines the two other panels in some sense. It shows us the dominant frequency as a function of time. Here, we can clearly see the different notes played by the clarinet as little bright streaks across time at a particular frequency.

#pg Because the transformation is reversible. A typical operation that is applied in the frequency domain is filtering. The phrase 'turn up the bass!' is literally an operation one can implement in the frequency domain with great ease. It involves increasing the power of the lower frequencies realtive to the higher frequencies, and operation easily accomplished with some multiplications. Increasing lower frequencies is technically called a low-pass filter. There many different filters one can imagine, and all transform the original waveform slightly differently, allowing us to create unique sounds. Accidentically, that's how a synthetizer works. And if electronic music isn't really your thing and your prefer good old dashioned guitar rock, it is also the thing that those effect pedals guitarists have allow you to do. It is even used in karaoke machines, where a bandpass filters gets rid of the frequencies associated with a human singing, leaving the instruments mostly untouched (there is more to it, but it is a good starting point). So far we have focussed on music, but we can apply it to any time or space varying signal. We can look at whether some naturally occuring phenoma show seasonaility, meaning they occur to wax and wane at a particular frequency. And we can decompose images, since the transform isn't limited to signals that vary only in one dimension (such as time) but also 2,3 or N dimensions. Turning up the bass applied to an image actuallt smooth the image, as you are accuentated the lower frequencies and atteunating the higher frequencies that make up the sharp edges in your image. Fourier introduced the world to the idea and proof that one can decompose any signal into a set of sines and cosines, but he wasn't the person coming up with the algorithm to do it efficeitnely. The Fast Fourier Transform algorithm that allows you to actually do this decomposition, ultimately is credited to James Cooley and John Tukey in around 1965, although others have proposed similar methods, even prior to Fourier. You can find implementation of the FFT algorirthm in many toolboxes across many programming languages. In Python, both numpy and scipy have their own set of functions to implement it, and I am sure equivalent classes exist for all other major programming languages.

#h3 18.24 – Basis Functions - JPEG Compression using DCT

#pg The Discrete Cosine Tranform (DCT) has a lot of similarities to the more general idea of the fourier transform detailed above, without all the details the fourier transform needs to be a complete theorem. Neverheless, DCT has proven to be an extremely useful application of basis functions to image compression, especially during the early years of the internet, when the transfer of data was still relatively slow. It is still widely used in many different compression algorithms, both for image and video, but also in very different domains such as cryptography. If you ever downloaded a move of questionable content, origin and quality, you are probably familiar with some of the odd artificats that can sometimes occur when the compression on the original file has been a little too aggressive. The Discrete Cosine Transform is quite similar to the fourier transform. For an image of size n by n, it will use n cosines at n different frequencies to capture the variation in pixel intensity in the horizontal and vertical direction of the image. The transform yields a set of parameters for each horizontal and vertical combination of each cosine at each frequency. In other words, we replace the original image with a set of weights on different combinations of cosines that when applied to the cosine and combined reconstruct the original image. So, like the fourier transform, the operation is reversible. However, replacing one matrix of nxn numbers by another clearly doesn't reduce the number of elements we need to store to disk. However, the compression is achieved by two clever insights about the nature of natural insights. First, JPEG compression divides the image into a set of 8x8 blocks. At this resolution, so far zoomed in, natural images more often than not vary quite smoothly. Look up from whatever device you are reading this, look straight ahead and take in your surroundings. Across everything in your visual field, there is quite a bit of variation going on. I imagine there objects that differ in color, shape, and the scene is quite heterogenous. Maybe some sharp edge formed by walls and doors if you are inside, or perhaps a row of trees that all vary in shape, size and color. But try to imagine only picking a tiny region from all that you can see. Most of the time, that tiny region is pretty homogenous. It is a small portion of a wall, or a desk, or the bark on a tree, or a little patch of perfectly blue sky with no clouds. If there are any changes in these smaller subsections of the entire image, they are likely to be quite smooth. And to fit those, you only need a handful of cosines at low frequencies to encode them. The second clever bit that actually achieves the compression is the realization that most of the weights in your 8x8 DCT matrix are essentially so close to zero they might as well be zero. And the proves to be correct. You can set a substantial amount of weights to zero (meaning you don't have to store them anymore, and hence the compression). that if you were to reverse the process, moving from the DCT weight back to the original image, people can't even tell you've removed quite a bit of data. The truth is that in natural images, the pixels are so highly correlated with nearby pixels, that there is a lot of redundant information being stored. JPEG takes advantage of this and gets filesizes down substantially, without users necessarily even noticing the image has changed. If you have used photoshop to save images as jpeg, you may have noticed it sometimes asks you about the quality of the image to be stored. That quality corresponds to your willingness to sacrafice more and more weights by increasing the threshold at which weights are kept.

#h3 18.25 - Beyond linear constraints - Autoencoders

#pg Note that the first constraint, that is linearity, limits the type of amount of variance we will be able to redistribute. More complex (non-linear) variance between variables requires different types of algorithms. An artificial neural network called an autoencoder is a surprisingly effective one that we will see later in greater detail. But to briefly touch on it as a method of dimensionality reduction and compression: the best way the think of an autoencoder is of a neural network with a bottleneck. Input data of some high dimensionality (say: images) serves as input, but is also the expected output as the neural network. For images, think of this dimensionality as the indivudual pixels of the image. Each pixel is its own dimension. And each pixel requires one input node. Since the expected output is the same as the input, we thus have the same number of output nodes (one per pixel). However, the layers in between input and output have progressively lower number of nodes until a halfway point where that number starts to increase again, up to the final output layer. By training the model to reconstruct the input image as its output, it needs to figure how to compress the image somehow so that it can be represented with a lower number of nodes, relative to the number of pixels. In the case of images, it will need to find a set of visual building blocks that can be combined into a full image.

#im ../../../assets/figures/018/018-01.png 50 128 Figure 018.X - Autoencoders are a class of deep belief neural networks that attempt to learn a lower dimensional or latent representation of a higher dimensional input, by letting the inputs flow through a lower dimensional bottleneck while learning to reconstruct the original input as precisely as possible. This process will force the algorithm to adopt a representation in whatever number of latent dimensions are present in the most narrow part of the bottleneck: the layer with the fewest nodes available. We will talk much more about this class of algorithms in the chapters to come on neural networks.

#h3 18.26 - Discussion

#pg Full disclosure: the increasing reliance of machine learning on deep belief neural networks has made the practice of dimensionality reduction somewhat obsolete. Autoencoders are specifically designed to find a lower dimensional space that is sufficient to reproduce and recreate inputs of a (much) higher dimension. But most neural networks have, by virtue of their architecture and design, a build-in way to for this process to occur naturally. Typically, DBNNs encode information

#im ../../../assets/figures/018/018-01.png 50 128 Figure 018.X - To human observers, the top and bottom image do not appear not to differ. If you were to zoom in to the pixel level however, you would find that bottom image contains a bit of noise that has been artificially added. But this changes nothing about the overall content of the images: both contain the same toy Lego firetruck. However, under some circumstances a modern day image classification will have no issue identiftying the firetruck in the top panel, but the might be wildly wrong about its predicton for the bottom image: 'polar bear', 'light bulb', 'screwdriver'. In some cases, there is no systematic way the classification changes just by adding virtually imperceivable noise. This behavior is concerning.

#pg about inputs with fewer and fewer nodes in deeper and deeper layers of the network. However, while we know this is happening, and has to happen, all the non-linearities that DBNNs allow to be applied to inputs to achieve accurate outputs make it extremely hard to interpret these deeper layers. In fact, some people have expressed a deep concern with this lack of understanding when the neural net community was confronted with bizarly succesful attempts to fool certain machine learning algorithms by manipulating the input data. It has given rise to a new field called adversial machine learning (not to be confused with general adversial networks) which studies the way machine learning algorithms can be duped and how to prevent it from happening. It is not surprising people want to fool the algorithms for some reason or another. What was surprising was that how well some of these tactics worked while they very clearly shouldn't have. Take Figure 18.1. Do you see a difference between these two images? Probably not. But they are different. The lower has a minute amount of noise added to it. So small that for practical purposes the image is the same. It still is the exact same firetruck. The latest generation of image classification neural networks are incredibly powerful. No matter what the angle, light condition, size in the image, orientation, etc, etc. a firetruck appeared as in an image, the neural network could generalize them all to the concept of a firetruck. This is ability human vision has to offer. But by adding noise so small a human doesn't even notice, the network all of sudden started predicting polar bears, or coleslaw, or whatever completely different category you can think off. This is concerning, and it also demonstrates we haven't actually replicated the ability of biological vision. And this is why understanding dimensionality reduction, preferably in geometric terms, even without applying it, is a useful critical thinking tool. For example, you'll be able think through why a particular AI of some media giant turned out to be so extremely biased, and what can be done to prevent that from happening again.

#h3 18.27 - Notebooks and Playgrounds

#pg The notebooks follow the examples given in this chapter, including different ways of computing the eigenvalues and eigenvectors, applying SVD and PCA to synthetics data, eigenfaces, ICA source seperation, JPEG compression, heart rate basis function fitting and the fast fourier transform.

#h3 18.28 - References

#bs

#be

#bp Murphy, K. P. (2012). Machine Learning: A Probabilistic Perspective. United Kingdom: MIT Press.

#bp Gurnsey, R. (2017). Statistics for Research in Psychology: A Modern Approach Using Estimation. United States: SAGE Publications.

#bp O'Neil, C., Schutt, R. (2013). Doing Data Science: Straight Talk from the Frontline. United States: O'Reilly Media.

#bp Raad, B. d. (2000). The big five personality factors : the psycholexical approach to personality. Germany: Hogrefe & Huber.

#bp Aron, E., Aron, A., Coups, E. J. (2013). Statistics for Psychology. United Kingdom: Pearson.

#bp Warr, K. (2019). Strengthening Deep Neural Networks: Making AI Less Susceptible to Adversarial Trickery. United States: O'Reilly Media

#bp Pfordresher, P., Harré, R., Tan, S. (2017). Psychology of Music: From Sound to Significance. United Kingdom: Routledge.

#bp Sacks, O. (2011). Musicophilia: Tales of Music and the Brain. United Kingdom: Pan Macmillan.

#bp Fleming, J. (2012). Interpreting the Electrocardiogram. Netherlands: Springer Netherlands.

#bp Nature-inspired Computing: Concepts, Methodologies, Tools, and Applications. (2016). United States: IGI Global.

#bp Matthew Turk, Alex Pentland; Eigenfaces for Recognition. J Cogn Neurosci 1991; 3 (1): 71–86

#be