Machine Learning

Lecturers: Prof. Ben Recht, Prof. Jitendra Malik, Stephen Tu
Scribe: Jonathan N. Lee

Contents

1 Introduction

These are not official notes from the CS 189 instructors or course staff but they are based on CS 189 lectures and discussion sections as well as outside sources. Email Jonathan with questions and/or corrections.

Unless otherwise stated, usually vectors will be lowercase (e.g. v), matrices will be uppercase (e.g. A), scalar constants will be lowercase Greek letters (e.g. γ). Random vectors will sometimes be uppercase, but it should be clear from context. i, j, and k are usually reserved for indices.

2 Useful Linear Algebra

These notes come directly from Stephen Tu's discussions and notes.

2.1 Norms and Inner Products

The L^p-norm is defined as:

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}$$

Any norm denoted by $\|\cdot\|$ in a vector space V is a function mapping $V \to \mathbb{R}$ with the following properties:

1. $\|x\| \geq 0$ and $\|x\| = 0$ if and only if $x = 0$.
2. $\|ax\| = |a|\|x\|$ for scalar a.
3. Triangle inequality: $\|x + y\| \leq \|x\| + \|y\|$.

If we define the inner product in $\mathbb{C}^n$ as $\langle x, y\rangle = \sum_{i=1}^{n} x_i \overline{y}_i$, then we find that $\|x\|_2^2 = \langle x, x\rangle$. We also get Holder's inequality which states that for $\frac{1}{p} + \frac{1}{q} = 1$, $|\langle x, y\rangle| \leq \|x\|_p \|y\|_q$.

2.2 Orthogonal Decomposition

Let V be a subspace of $\mathbb{C}^n$, then we define the orthogonal subspace as:

$$V^\perp := \{v \in \mathbb{C}^n : \langle w, v\rangle = 0 \text{ for } w \in V\}$$

A fundamental idea in linear algebra that we will prove is the following:

$$\mathbb{C}^n = V \oplus V^\perp$$

which suggests that for some vector $w \in \mathbb{C}^n$ can be written as $w = w_1 + w_2$ where $w_1 \in V$ and $w_2 \in V^\perp$ and that this decomposition is unique.

It is easiest first to show that a decomposition exists. Let $\{v_1, \ldots, v_d\}$ be the orthonormal basis of V. So w can be rewritten as:

$$
\begin{aligned}
w &= \sum_{i=1}^{r} \langle w, v_i \rangle v_i + \left(w - \sum_{i=1}^{r} \langle w, v_i \rangle v_i \right) \\
&:= w_1 + w_2
\end{aligned}
$$

By construction, we have that $w_1 \in V$. We must now show that $w_2 \in V^{\perp}$:

$$
\begin{aligned}
\langle w_2, v_j \rangle &= \langle w - \sum_{i=1}^{r} \langle w, v_i \rangle v_i, v_j \rangle \\
&= \langle w - \langle w, v_i \rangle v_j, v_j \rangle \\
&= 0
\end{aligned}
$$

Is this a unique decomposition? Assume there exist two decompositions $w = w_1 + w_2 = u_1 + u_2$. Then $0 = (w_1 - u_1) + (w_2 - u_2)$.

Since $\|x + y\|_2^2 = \|x\|_2^2 + \|y\|_2^2$ for orthogonal vectors x and y, then $0 = \|w_1 - u_1\|_2^2 + \|w_2 - u_2\|_2^2$. This tells us that the decomposition is unique.

This fact is useful for a number of reasons. In this course, we will primarily rely on it to justify that $w = A\alpha + w_n$ where $A^{\top} w_n = 0$. This is just saying that w can be written as a vector in the subspace defined by the columns of A and $w_n \in A^{\perp}$ since $\langle w_n, a_i \rangle = 0$ for any column a_i of A.

2.3 Orthogonal Projectors

Following from the previous section, the orthogonal projection P_V of w onto the subspace $V = \text{span}\{v_1, \ldots, v_r\}$ where $\langle v_i, v_j \rangle = 0$ for $i \neq j$ is given by:

$$
P_V(w) = \sum_{i=1}^{r} \langle w, v_i \rangle \frac{v_i}{\|v_i\|^2}
$$

For a subspace U represented by orthonormal basis vectors, we can make a stronger statement:

$$
P_V(w) = UU^{\top} w
$$

3 Stochastic Gradient Descent

3.1 Motivation

Want to find a w^* such that $f(w^*)$ is minimized potentially under some constraints. Local minimizer is some w^* where $f(w^*) \leq f(w)\ \forall w$ s.t. $||w - w*|| < r$ for some $r \in \mathbb{R}$.

Global minimizer is some w^* such that $f(w^*) \leq f(w)\ \forall w$.

3.2 Convex Functions

Convex function $f(\cdot)$ is a function where, given any points w_1 and w_2, $f(w)$ is less than the line segment between $f(w_1)$ and $f(w_2)$ for $w \in [w_1, w_2]$. Formally:

$$f(w_1 + t(w_2 - w_1)) \leq f(w_1) + t(f(w_2) - f(w_1)) \qquad \text{where } t \in [0, 1]$$

3.3 Gradients

Gradient ∇f always points in the direction of steepest ascent, which means opposite direction is steepest descent. If we want to minimize a function with w, then we want to drive f to be as small as possible. Taylor's theorem will come in handy, which we can think of as the multivariate mean-value theorem for some local point x_0:

$$f(x) = f(x_0) + \nabla f(x_0 + t(x - x_0))^\top (x - x_0) \qquad \text{where } t \in [0, 1]$$

We can use Taylor's theorem to show that some vector v is in the descent direction of $f(\cdot)$ at x_0 if $v^\top \nabla f(x_0) < 0$.

Proof: We plug in $x = x_0 + \hat{t}v$ into Taylor's theorem for some $\hat{t} > 0$ and find that

$$\begin{aligned} f(x_0 + \hat{t}v) &= f(x_0) + \hat{t}\nabla f(x_0 + \hat{t}tv)^\top v \\ &= f(x_0) + \hat{t}\nabla f(x_0 + \tilde{t}v)^\top v \end{aligned}$$

where $\tilde{t} \in [0, \hat{t}]$. And we know that $\nabla f(x_0 + \tilde{t}v)^\top v < 0$ from our assumption only when $\hat{t}$ is small. Therefore we find that $f(x_0) > f(x_0 + \hat{t}v)$ and so v is in the descent direction.

3.4 Optimality for Convex Functions

Want to prove that if $\nabla f(w^*) = 0$ and $f(\cdot)$ is a convex function then $f(w^*) \leq f(w)$ for all w.

Proof: Using our definition of convexity, we get the following

$$f(w_* + t(w - w_*)) \leq f(w_*) + t(f(w) - f(w_*))$$

and the rearrange the terms to get

$$f(w) \geq f(w_*) + \frac{f(w_* + t(w - w_*)) - f(w_*)}{t}$$

Using Taylor's theorem around the point w_*, we find

$$\begin{aligned} f(w_* + t(w - w_*)) &= f(w_*) + t\nabla f(w_* + \hat{t}((w_* + t(w - w_*)) - w_*))^\top (w - w_*) \\ &= f(w_*) + t\nabla f(w_* + \hat{t}t(w - w_*))^\top (w - w_*) \end{aligned}$$

for some $\hat{t} \in [0, 1]$. So

$$f(w_* + t(w - w_*)) - f(w_*) = t\nabla f(w_* + \hat{t}t(w - w_*))^\top (w - w_*)$$

Then as t tends to zero in the limit, we find that

$$f(w) \geq f(w_*) + \nabla f(w_*)^\top (w - w_*)$$

We find that $f(w) \geq f(w_*)$ when $\nabla f(w_*) = 0$.

3.5 Properties of Convex Functions

If a function f is convex, then:

- αf is also convex if $\alpha \geq 0$.
- $f + g$ is convex if g is also convex.
- $\max\{f(x), g(x)\}$ is convex if g is also convex.
- $f(Ax + b)$ is convex if A is a matrix and b is a vector.

All norms are convex.

Proof: We know from the triangle inequality of L^p-norms that $\|x + y\|_p \leq \|x\|_p + \|y\|_p$. We can also rewrite the definition of convexity as

$$f((1-t)w_1 + tw_2) \leq (1-t)f(w_1) + tf(w_2) \qquad \text{where } t \in [0, 1]$$

Therefore, we can just take $f(x) = \|x\|_p$:

$$\|(1-t)w_1 + tw_2\|_p \leq \|(1-t)w_1\|_p + \|tw_2\|_p$$

which we know is true from the triangle inequality presented earlier.

3.6 The Gradient Descent Algorithm

We have shown the the negative gradient of a function will take us in the steepest descent direction, and by following that direction, we will end up at a minimum (and, necessarily, a global minimum for convex functions). Here is the descent algorithm:

Algorithm 1 Gradient Descent Algorithm

```
1: procedure GDA(f)
2:     Choose w_0 ∈ R^n
3:     k ← 0
4:     while f(w_k) is not converged do
5:         Choose α_k > 0
6:         w_{k+1} = w_k − α_k ∇_w f(w_k)
7:         k ← k + 1
8:     return w_k
```

3.7 Stochastic Gradient Descent Algorithm

There are several reasons why we might want to introduce stochasticity into the gradient descent algorithm. The primary reason is that often our function $f(\cdot)$ that we are trying to optimize is composed of several sub-functions:

$$f(w) = \frac{1}{n} \sum_i f_i(w)$$

And if n is very large, we may end up with an enormous amount of computation just for one step in traditional gradient descent. We would then have to repeat these steps to end up with any reasonable convergence. Instead, we could potentially get an unbiased estimate of the gradient that might be more feasible computationally. Then in the expectation, we'd have the gradient. Formally, we have a function $g(w)$ such that $\mathbb{E}[g(w)] = \nabla f(w)$.

Define $g(w) = \nabla f_i(w)$ where i is sampled uniformly at random from the set $\{1, \ldots, n\}$. Then we find

$$\begin{aligned}
\mathbb{E}[g(w)] &= \sum_i \Pr(i) \nabla f_i(w) \\
&= \sum_i \frac{1}{n} \nabla f_i(w) \\
&= \nabla \frac{1}{n} \sum_i f_i(w) \\
&= \nabla f(w)
\end{aligned}$$

So we can generate a new algorithm that takes advantage of these unbiased estimates. Instead of evaluating the gradient over the sum of functions, we will just choose one function at random, evaluate the gradient, update the parameter and repeat this process.

Algorithm 2 Stochastic Gradient Descent Algorithm

```
1: procedure SGD(f)
2:     Choose w_0 ∈ R^n
3:     k ← 0
4:     while f(w_k) is not converged do
5:         Choose α_k > 0
6:         Sample i ~ U(1, n)
7:         w_{k+1} = w_k − α_k ∇_w f_i(w_k)
8:         k ← k + 1
9:     return w_k
```

4 Support Vector Machines

4.1 Linear Separators and Margins

We can define a binary linear classifier as a function that classifies one side of a hyperplane as one class and the other side as the other class. More formally, consider a hyperplane defined by the set of points that satisfy the equation $w^\top x + \beta = 0$. So our classifier $h(x)$ can be defined as follows:

$$h(x) = \begin{cases} c_0 & w^\top x + \beta \geq 0 \\ c_1 & w^\top x + \beta < 0 \end{cases}$$

w is the normal vector to the hyperplane. β is considered a bias. We can show that $\frac{\beta}{\|w\|_2}$ is the perpendicular distance from the origin to the hyperplane.

Proof: Consider a point x_0 on the plane. The projection of x_0 onto the normal vector is $\frac{w^\top x_0}{\|w\|_2}$, which is the scalar component of x_0 orthogonal to the plane and therefore is the distance. From the definition of a hyperplane, we find that this distance is equivalent to $\frac{\beta}{\|w\|_2}$.

The geometric distance, γ from the hyperplane to any given point x_1 which might not be on the plane is $\gamma = \frac{w^\top x_1 + \beta}{\|w\|_2}$

Proof: If x_0 is the point on the plane that is closest to x_1, then $x_0 = x_1 + \tau \frac{w}{\|w\|_2}$.

$$\begin{aligned} w^\top x_0 + \beta &= 0 \\ w^\top (x_1 + \tau \frac{w}{\|w\|_2}) + \beta &= 0 \\ \frac{-w^\top x_1 - \beta}{\|w\|_2} &= \tau \end{aligned}$$

The *margin* in a linearly separable classification is the perpendicular distance from the closest point x to the hyperplane.

$$\text{Margin} = \min_i \frac{-w^\top x_i - \beta}{\|w\|_2}$$

Notice that if we decide to scale w or b by a constant factor, the distance τ to any point will not change. So our geometric distance is invariant to the scaling of w and β as long as we scale both of them by the same factor. With this in mind, we define the functional distance from a point x to the hyperplane as $f(x) = w^\top x + \beta$. This distance doesn't have a geometric interpretation, but we can think of it as the confidence of the classification of x. Then we can rewrite the geometric distance as:

$$\tau = \frac{f(x)}{\|w\|_2}$$

We'd also like to enforce the constraint that the data is classified correctly. Notably, $y_i f(x_i) \geq 0, \forall i$ where $y_i \in \{-1, 1\}$ is the corresponding label for x_i.

The question now remains: if we want a good classifier of linearly separable data, which hyperplane should we pick? A good choice could be the one that maximizes the margin. So we form the optimization problem

$$\max_{w,\beta}[\min_{i} \frac{y_i f(x_i)}{\|w\|_2}]$$

This turns out to be relatively hard to solve. So we simplify things by putting a stronger constraint on $f(x_i)$. Remember that we can scale $f(x)$ with invariance on the margin. Therefore we scale the functional distance such that $y_i f(x_i) \geq 1$. So the optimization problem is:

$$\max_{w,\beta} \frac{1}{\|w\|_2}$$

which is equivalent to

$$\min_{w,\beta} \|w\|_2$$

under the constraint that $w^\top x_i + \beta \geq 1$. Lagrangians are used to solve this constrained optimization problem.

The is the fundamental optimization problem for a support vector machine.

4.2 Slack Variables for Soft Margins

Not all data is linearly separable and many classifiers may perform better with less sensitivity to outliers. We introduce slack variables to this problem in order to amend this issues of hard-separating hyperplanes. Suppose that instead of the hard constraint from the last section, we let

$$y_i(w^\top x_i + \beta) \geq 1 - \xi_i$$

for $\xi_i \geq 0$. We find that if $0 < \xi_i < 1$ then x_i is allowed to fall into the margin. If $\xi > 1$ then the data point is allowed to fall on the wrong side of the hyperplane.

Of course, we must also penalize these cases in our optimization problem.

$$\min_{w,\beta} \|w\|_2 + C \sum_i \xi_i$$

where C is some scalar hyperparameter. We can rewrite this by substituting for the slack variable; however, we want to put a lower bound of 0.

$$\min_{w,\beta} \|w\|_2 + C \sum_i (1 - y_i(w^\top x + \beta))_+$$

where $(x)_+ = \max\{0, x\}$

We can solve this using the stochastic gradient technique developed in the first section.

5 Maximum Likelihood Estimation

5.1 General Estimation

In some cases, we'd like to the estimate the parameter of a density function based on the samples that we draw from it. For example consider the pdf $p(x|\theta)$ where we draw samples $x_1, \ldots, x_n$ i.i.d. Our estimate would look like this:

$$\hat{\theta} = \arg\max_{\theta} \mathcal{L}(\theta) = \arg\max_{\theta} p(x_1, \ldots, x_n|\theta)$$

Since are drawn samples are i.i.d., we can simply write

$$p(x_1, \ldots, x_n|\theta) = \prod_{i=1}^{n} p(x_i|\theta)$$

We could now solve for $\hat{\theta}$ by taking the derivative wrt θ and set the derivative equal to zero. Typically, we would want to take the log of the joint distribution since log is monotonically increasing and would split the product into sums, which makes the derivative easier.

$$\arg\max_{\theta} p(x_1, \ldots, x_n|\theta) = \arg\max_{\theta} \log p(x_1, \ldots, x_n|\theta)$$

5.2 Continuous Uniform Distribution

Consider the pdf $f(x|\theta) = \frac{1}{\theta}$ where $0 \leq x \leq \theta$. Assume that we have samples $x_1, \ldots, x_n$. So our likelihood function becomes

$$\begin{aligned}
\mathcal{L}(\theta) &= p(x_1, \ldots, x_n|\theta) \\
&= \prod_{i}^{n} \frac{1}{\theta} \\
&= \theta^{-n}
\end{aligned}$$

Take the log of the likelihood: $\log \mathcal{L}(\theta) = -n \log(\theta)$. Then:

$$\frac{d}{d\theta} \log \mathcal{L}(\theta) = \frac{-n}{\theta}$$

We assumed that $\theta \geq x_n$ where $x_n \geq x_i$, for $i \in \{1, \ldots, n\}$. Otherwise it wouldn't make sense to get a parameter that is less than a sample we received. We also notice that the derivative of the likelihood function is decreasing, so we hope to choose our least value which is $\hat{\theta} = x_n$.

5.3 Exponential Distribution

This example demonstrates a more formulaic approach. Consider $f(x|\theta) = \theta e^{-\theta x}$.

$$\begin{aligned}
\log \mathcal{L}(\theta) &= \log p(x_1, \ldots, x_n|\theta) \\
&= \log \prod_i^n p(x_i|\theta) \\
&= \log \prod_i^n \theta e^{-\theta x_i} \\
&= \sum_i^n \log \theta - \theta x_i \\
&= n \log \theta - \theta \sum_i^n x_i
\end{aligned}$$

We then set the derivative wrt θ equal to zero.

$$\frac{d}{d\theta} \log \mathcal{L}(\theta) = \frac{n}{\theta} - \sum_i^n x_i$$

Therefore

$$\hat{\theta} = \frac{n}{\sum_i^n x_i}$$

This happens to be the reciprocal of the sample mean.

5.4 Gaussian Distribution

We now look at the single variable Gaussian distribution. Again we apply the same techniques where

$$\begin{aligned}
p(x_1, \ldots, x_n|\mu, \sigma^2) &= \prod_i \frac{1}{\sqrt{2\sigma^2\pi}} \exp\left(-\frac{(x_i - \mu)^2}{2\sigma^2}\right) \\
\log\left[p(x_1, \ldots, x_n|\mu, \sigma^2)\right] &= \sum_i -\frac{(x_i - \mu)^2}{2\sigma^2} - \frac{1}{2}\log\left[2\sigma^2\pi\right]
\end{aligned}$$

To find the first parameter μ, we take the derivative w.r.t μ and then set it equal to zero (because we know the function will be concave).

$$\sum_i \frac{x_i - \mu}{\sigma^2} = 0$$

Therefore we find that $\mu = \frac{1}{n}\sum x_i$. For the variance:

$$\log\left[p(x_1,\ldots,x_n|\mu,\sigma^2)\right] = \sum_i -\frac{(x_i-\mu)^2}{2\sigma^2} - \frac{1}{2}\log\left[\sigma^2\right] - \frac{1}{2}\log\left[2\pi\right]$$

$$\begin{aligned}\frac{\partial}{\partial\sigma^2}\log p(\cdot) &= -\frac{n}{\sigma^2} - \left(\sum_i \frac{(x_i-\mu)^2}{2}\right)\cdot\left(\frac{\partial}{\partial\sigma^2}\frac{1}{\sigma^2}\right)\\ &= -\frac{n}{\sigma^2} + \left(\sum_i \frac{(x_i-\mu)^2}{2}\right)\cdot\left(\frac{1}{(\sigma^2)^2}\right)\\ &= \frac{1}{2\sigma^2}\left(\frac{1}{\sigma^2}\sum_i (x_i-\mu)^2 - n\right)\end{aligned}$$

So we then find that

$$\begin{aligned}0 &= -\frac{n}{2\sigma^2} + \frac{1}{2(\sigma^2)^2}\sum_i (x_i-\mu)^2\\ \sigma^2 &= \frac{1}{n}\sum_i (x_i-\mu)^2\end{aligned}$$

And we can just plugin the sample estimate mean $\hat{\mu}$.

6 Properties of Positive Definite Matrices

7 Multivariate Gaussians

7.1 Equation and Intuition

We define the probability density function of a Gaussian in the single variable case as the following.

$$f(x|\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

In the multivariate case we consider the covariance matrix and mean vector.

$$f(x|\mu,\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}}\exp\left(-\frac{1}{2}(x-\mu)^\top\Sigma^{-1}(x-\mu)\right)$$

We define the covariance matrix Σ as

$$\Sigma = \mathbb{E}\left[(x-\mu)(x-\mu)^\top\right]$$

Consider a change of variables where $z = Ax + b$. The covariance is not affected by translational shifts. Therefore:

$$\begin{aligned}\Sigma &= \mathbb{E}\left[(z-\mu)(z-\mu)^\top\right] \\ &= \mathbb{E}\left[(Az - A\mu)(Az - A\mu)^\top\right] \\ &= A\mathbb{E}\left[(z-\mu)(z-\mu)^\top\right]A^\top \\ &= A\Sigma A^\top\end{aligned}$$

Notice that we make the assumption that the covariance matrix is invertible. In fact it is assumed to be positive definite. There are several conditions when it would not be positive definite. The first is when one or more of the random variables are constant. This is because $\text{Cov}(X_i, c) = 0$ where c is a constant. Therefore we would end up with a row and column of zeros in the covariance matrix making it singular.

The second is when any RV can be written as a linear combination of the others. Consider an RV $X_c = \alpha X_a + \beta X_b$. For each component of some column of Σ, $\text{Cov}(X_i, X_c) = \alpha\text{Cov}(X_i, X_a) + \beta\text{Cov}(X_i, X_b)$. So this column is just a linear combination of the columns of X_a and X_b, so the matrix is singular.

More formally, we can see that if Σ is singular then $\Sigma v = v^\top \Sigma v = 0$. We saw earlier that $v^\top \Sigma v = v^\top \text{Cov}(x)v = \text{Cov}(v^\top x) = \text{Var}(v^\top x)$. So we see that Σ is singular if some linear combination of the x components leads to zero variance.

We will now show that $f(x|\mu, \Sigma)$ is indeed a valid probability density function. So we must show that

$$\int_{\mathbb{R}^n} \exp\left(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right) dx_1 \dots dx_n = (2\pi)^{n/2}|\Sigma|^{1/2}$$

First we will show that the determinant of a matrix A which has a full set of eigenvalues is the product of the eigenvalues. $|A - \lambda I|$ has characteristic polynomial $(\lambda_1 - \lambda)\dots(\lambda_n - \lambda)$ since any value for λ such that $(A - \lambda I)v = 0$ defines an eigenvalue. Now setting $\lambda = 0$, we find that $|A| = \prod_i \lambda_i$.

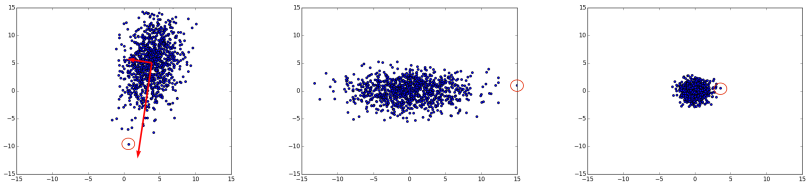
7.2 Geometric Interpretation of Covariance

Like in the univariate case, the concepts of variance and covariance play in a role in the shape of the Gaussian distribution. The isocontours of a multivariate Gaussian are in fact elliptical, which we will show later.

Consider the spectral decomposition of $\Sigma^{-1} = U\Lambda U^{-1}$. Because the covariance matrix is a symmetric matrix, we know it is orthogonally diagonalizable and so we can make a stronger statement: $\Sigma^{-1} = U\Lambda U^\top$. But what does this mean intuitively when we hit it with x? $x^\top \Sigma^{-1} x = x^\top U\Lambda U^\top x = x^\top U\sqrt{\Lambda}\sqrt{\Lambda}U^\top x = \|\sqrt{\Lambda}U^\top x\|_2^2$.

There are two interesting properties of $U^\top x$. The first is that since $U^\top$ is orthogonal, it preserves the dot product. Also $U^\top x$ is a transformation of x into the basis of the eigenvectors of Σ^{-1}. And $\sqrt{\Lambda}$ is a diagonal matrix so it essentially just stretches the transformed in x in the direction of the eigenvectors.

Therefore, if we sampled some data points from a Gaussian, we can transform a given point x into the coordinate system aligned with the eigenvectors by finding $U^\top(x-\mu)$. You should find that it rotates such that the eigenvectors are aligned with the axes. What happens when you plot $\sqrt{(\Lambda)^{-1}}U^\top(x-\mu)$? You should find that it resembles a Gaussian distribution with $\mu = 0$, $\Sigma = I_d$. Why is this the case? Consider the situation where $x \sim \mathcal{N}(0, I_d)$ and we want to choose a transformed $x' = Tx$ such that we get a distribution that looks like $x \sim \mathcal{N}(0, \Sigma)$. So to get x into the region such that it will have the correct density under $p(x|0, \Sigma)$, we should rotate it into the axis of the eigenvectors and scale it down, which is exactly what the inverse covariance matrix does. So $T = \sqrt{\Lambda}U^\top$. See the figure below.



8 Decision Theory

9 Linear Regression

9.1 Intro

In the following section, we often will use the fact that $X^\top X$ is invertible and so it will be helpful to establish under what conditions this is true. Assume that X is an $n \times d$ data matrix.

First let's show that $\ker(X) = \ker(X^\top X)$. This is trivially true in one direction because if $Xv = 0$ then surely $X^\top Xv = 0$. In the other direction, the proof can be trickier. We find that if $X^\top Xv = 0$ then $v^\top X^\top Xv = \|Xv\|_2^2 = 0$ and this is true only if $Xv = 0$.

Next we will show that if $\text{rank}(X) = d$ then $X^\top X$ is invertible. We saw that $\ker(X) = \ker(X^\top X)$ and we know that $\ker(X) = \{0\}$ by the rank nullity theorem. Therefore $\ker(X^\top X) = \{0\}$ and also it is square so it is invertible.

Because of these two facts, we can see that our features must be linearly independent in order for $X^\top X$ to be invertible.

Tangent: It might also be helpful to show that $\mathbb{R}^n = \ker(A) \oplus \text{im}(A^\top)$. Let some vector v be in the kernel of A. Therefore we know that $v \cdot a_i = 0$ where a_i is the i-th row vector in A. This also means that v is orthogonal to any linear combination of a_i vectors. So if there is a vector $a \in \text{im}(A^\top)$, then v must be orthogonal to it. Therefore $\ker(A) = \text{im}(A^\top)^\perp$.

This explains our orthogonal decomposition of vectors. If we have some vector $w \in \mathbb{R}^n$, then we can write $w = w_n + X^\top a$ where w_n is in the kernel of

X and so it is orthogonal to to any linear combination of the columns of $X^\top$, namely $X^\top a$.

9.2 Maximum Likelihood Model

There are several ways that we might want to motivate linear regression. The one that we will walk through first is the assumption that we are given data $x_1, \ldots, x_n$ and outputs $y_1, \ldots, y_n$ where $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$ and that there is a distribution $p(y|x)$ where $y \sim \mathcal{N}(w^\top x, \sigma^2)$. In other words, we assume that there is a true linear model weighted by some true w and the values generated are scattered around it with some error $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Then we just want to obtain the max likelihood estimation.

$$p(Y|X, w) = \prod_{i=1}^{n} p(y_i|x_i, w)$$
$$\log p(\cdot) = \sum_i -\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(y_i - w^\top x_i)^2$$

To make things simpler, we'd like to express $\sum_i (y_i - w^\top x_i)^2$ in matrix-vector form. We recognize that this is equal to

$$\|Y - Aw\|_2^2$$

where $A = \begin{bmatrix} x_1 & \ldots & x_n \end{bmatrix}^\top$, which is just the data matrix where each row is an entry. Therefore we just have to maximize $-\frac{1}{2}\|Y - Aw\|_2^2$.

$$\nabla_w \left(-\frac{1}{2}\|Y - Aw\|_2^2 \right) = A^\top (Y - Aw)$$
$$\hat{w} = (A^\top A)^{-1} A^\top Y$$

Another potentially more intuitive way to get this result would be to consider that we are trying to match $Y \approx A\hat{w}$. And so our error term will be the vector $Y - A\hat{w}$ and in the case where this error is minimized, we will have that the vector $Y - A\hat{w}$ is orthogonal to the columns of A and therefore in the kernel of $A^\top$. So $A^\top(Y - A\hat{w}) = 0$. Then we can solve for $\hat{w} = (A^\top A)^{-1} A^\top Y$.

In an attempt to prevent overfitting, we might want to add a regularization term. Instead of considering these problems as maximization of likelihood, we can instead think of them as minimization of a loss, specifically the loss that is $J(w) = \frac{1}{2}\|Y - Aw\|_2^2$. The solution is the same, but this gives us reason to add regularization in the form of $L2$ (ridge) or $L1$ (lasso) norms:

$$J_{L2}(w) = \frac{1}{2}\|Y - Aw\|_2^2 + \lambda\|w\|_2^2$$
$$J_{L1}(w) = \frac{1}{2}\|Y - Aw\|_2^2 + \lambda\|w\|_1$$

There is potentially more justification that we can use for this regularization. Consider the problem of maximum a posteriori (MAP), where we use Bayes' rule to account for a prior distribution over potential w, where $p(w|Y,X) \propto p(Y,X|w)p(w)$. For a simple and common case, consider $w \sim \mathcal{N}(0, \tau^2 I)$. We find that

$$\log p(Y,X|w) = \ldots - \frac{1}{2\sigma^2}(Y - Aw)^\top (Y - Aw)$$
$$\log p(w) = \ldots - \frac{1}{2\tau^2} w^\top w$$

Terms not relevant to the optimization problem are condensed to the "...". Adding these log terms and minimizing the negative just gives us the $L2$ regularized linear least squares loss that we saw earlier:

$$J(w) = -\log p(Y,X|w) - \log p(w) = \frac{1}{\sigma^2}(Y - Aw)^\top (Y - Aw) + \frac{1}{2\tau^2}\|w\|_2^2$$

10 Logistic Regression

See this documentfor detailed information.

10.1 Logistic Transformation

An important distinction must be made between two classes of models in machine learning. As we have seen before with Gaussian Discriminant Analysis, some models seek to recreate a distribution $p(x|y)$ and from that identify the appropriate class y. We have also seen models that find a decision boundary.

Suppose the goal is to model a probability $p(y|x)$. Vanilla linear regression would not result in an output bounded between 1 and 0. Linear regression of the log of p has a similar issue. $\log p(x)$ is unbounded only in one direction, so a linear function is again not suitable.

Finally $\log \frac{p(x)}{1-p(x)}$, known as the logistic transformation is unbounded and can be modeled linearly:

$$\log \frac{p(x)}{1 - p(x)} = w^\top x$$

Solving for $p(x)$ results in:

$$p(x) = \frac{1}{1 + \exp(-w^\top x)}$$

The decision rule is as follows: if $p(x) \geq 0.5$ classify as $y = 1$; if $p(x) < 0.5$ classify as $y = 0$. These two events occur when $w^\top x \geq 0$ and $w^\top x < 0$ respectively. Therefore, logistic regression yields linear classifier despite the seemingly non-linear form. The advantage of logistic regression is that it gives not only classification but also probabilities.

For convenience, it may be helpful to define the probabilities as

$$p(y = 1|x) = \mu(x)$$

and then, as a single function:

$$p(y|x) = \mu(x)^y(1 - \mu(x))^{1-y}$$

10.2 Likelihood

Given a set of examples, the likelihood is computed as:

$$\begin{aligned} p(y_1, \ldots, y_n|x_i, \ldots, x_n) &= \prod_{i=1}^{n} p(y_i|x_i) \\ &= \prod_{i=1}^{n} \mu(x_i)^{y_i}(1 - \mu(x_i))^{1-y_i} \end{aligned}$$

Taking the log results in:

$$\begin{aligned} p(\cdot|\cdot) &= \sum_{i=1}^{n} y_i \log \mu(x_i) + (1 - y_i) \log(1 - \mu(x_i)) \\ \nabla_w p(\cdot|\cdot) &= \nabla_w \left(\sum_{i=1}^{n} y_i \log \mu(x_i) + (1 - y_i) \log(1 - \mu(x_i)) \right) \\ &= \sum_{i=1}^{n} \frac{y_i}{\mu(x_i)} \frac{\partial \mu(x_i)}{\partial w} - \frac{1 - y_i}{1 - \mu(x_i)} \frac{\partial \mu(x_i)}{\partial w} \end{aligned}$$

Since $\partial \mu(x_i)/\partial w = \mu(x_i)(1 - \mu(x_i))x_i$:

$$\begin{aligned} \nabla_w p(\cdot|\cdot) &= \sum_{i=1}^{n} \left(\frac{y_i}{\mu(x_i)} - \frac{1 - y_i}{1 - \mu(x_i)} \right) \mu(x_i)(1 - \mu(x_i))x_i \\ &= \sum_{i=1}^{n} (y_i - \mu(x_i))x_i \end{aligned}$$

In vector notation, this is:

$$\nabla_w p(\cdot|\cdot) = X^\top (Y - \mu)$$

where the vector μ has components $\mu_i = \mu(x_i)$. Updates to the parameter w can be computed from this expression in a variety of ways.

11 Bias-Variance Trade-off

11.1 Mean Squared Error

Bias and variance are two sources of error that contribute to what we call the Mean Squared Error (MSE) which is defined as $\mathbb{E}\|\hat{\theta} - \theta\|_2^2$, where $\hat{\theta}$ is our

estimated parameter and θ is the true parameter. When we expand this term, we get

$$\begin{aligned}
\mathbb{E}\|\hat{\theta} - \theta\|_2^2 &= \mathbb{E}\|\hat{\theta} - \mathbb{E}[\hat{\theta}] + \mathbb{E}[\hat{\theta}] - \theta\|_2^2 \\
&= \mathbb{E}\left[(\hat{\theta} - \mathbb{E}[\hat{\theta}] + \mathbb{E}[\hat{\theta}] - \theta)^\top (\hat{\theta} - \mathbb{E}[\hat{\theta}] + \mathbb{E}[\hat{\theta}] - \theta)\right] \\
&= \mathbb{E}\left[\|\hat{\theta} - \mathbb{E}[\hat{\theta}]\|_2^2 + \|\mathbb{E}[\hat{\theta}] - \theta\|_2^2 + 2(\hat{\theta} - \mathbb{E}[\hat{\theta}])^\top (\mathbb{E}[\hat{\theta}] - \theta)\right]
\end{aligned}$$

We find that the last term is just zero if we expand it and then we end up with

$$\mathbb{E}\|\hat{\theta} - \theta\|_2^2 = \mathbb{E}\|\hat{\theta} - \mathbb{E}[\hat{\theta}]\|_2^2 + \|\mathbb{E}[\hat{\theta}] - \theta\|_2^2$$

We call the first term the variance and the second term the bias of the estimator. These terms will help us characterize the error of our models. We can think of variance as the susceptibility of our model to overfitting and bias tells us how far are estimate is from the actual in expectation.

Let's consider an interesting example of the bias-variance trade-off. Suppose $x \sim \mathcal{N}(\mu, \sigma^2 I)$. Using MLE, our estimate is $\hat{\mu} = x$ if we are only given one sample. Of course, we can see that $\mathbb{E}[\hat{\mu}] = \mu$. So our bias just goes to zero and our MSE is defined entirely by the variance:

$$\begin{aligned}
\mathbb{E}\|\hat{\mu} - \mu\|_2^2 &= \mathbb{E}\|\hat{\mu} - \mathbb{E}[\hat{\mu}]\|_2^2 \\
&= \mathbb{E}\mathrm{Tr}\|\hat{\mu} - \mathbb{E}[\hat{\mu}]\|_2^2 \\
&= \mathbb{E}\mathrm{Tr}\left[(\hat{\mu} - \mathbb{E}[\hat{\mu}])^\top (\hat{\mu} - \mathbb{E}[\hat{\mu}])\right] \\
&= \mathbb{E}\mathrm{Tr}\left[(\hat{\mu} - \mathbb{E}[\hat{\mu}])(\hat{\mu} - \mathbb{E}[\hat{\mu}])^\top\right] \\
&= \mathrm{Tr}\mathbb{E}\left[(\hat{\mu} - \mathbb{E}[\hat{\mu}])(\hat{\mu} - \mathbb{E}[\hat{\mu}])^\top\right] \\
&= \mathrm{Tr}(\mathrm{Cov}(x)) \\
&= n\sigma^2
\end{aligned}$$

where the last three equalities follow from the fact that $\hat{\mu} = x$ and the definition of covariance and trace.

Now let's do something counter-intuitive. Let's say that our estimator instead is $\hat{\mu} = \alpha x$ for $\alpha \in [0, 1]$, which gives us a nonzero bias since $\mathbb{E}[\hat{\mu}] = \alpha\mu$. More specifically, we will get

$$\begin{aligned}
\mathbb{E}\|\hat{\mu} - \mu\| &= \mathbb{E}\|\hat{\mu} - \mathbb{E}[\hat{\mu}]\|_2^2 + \|\mathbb{E}[\hat{\mu}] - \mu\|_2^2 \\
&= \mathbb{E}\|\alpha x - \alpha\mu\|_2^2 + \|\alpha\mu - \mu\|_2^2 \\
&= \alpha^2 n\sigma^2 + (1 - \alpha)^2\|\mu\|_2^2
\end{aligned}$$

Given this biased estimator, we find that for some values of α and $\|\mu\|$, we can actually get lower MSE than from using an unbiased estimator in this simple case. See James-Stein estimator for more details and conditions.

11.2 Another Perspective

Assume that we have some samples $(x_1, y_1), \ldots, (x_n, y_n)$ drawn from a random process constrained to $y_i = f(x_i) + \epsilon$ where ϵ is some random variable with zero-mean and $f(\cdot)$ is the true function, and we have an estimator $\hat{y} = h(x)$. The MSE with expectation taken w.r.t the same distribution is just

$$\begin{aligned}\mathbb{E}(h(x) - y)^2 &= \mathbb{E}(h(x) - f(x) - \epsilon)^2 \\ &= \mathbb{E}\left[((h(x) - f(x))^2\right] + \mathbb{E}[\epsilon^2]\end{aligned}$$

This last equation follows from the fact that the cross term goes to zero since ϵ is zero-mean. By applying the same add-subtract techniques we saw earlier, we can get

$$\mathbb{E}(h(x) - y)^2 = \mathbb{E}\left[(h(x) - \mathbb{E}[h(x)])^2\right] + \mathbb{E}\left[(f(x) - \mathbb{E}[h(x)])^2\right] + \mathbb{E}[\epsilon^2]$$

We recognize the first term as our variance, the second as bias, and the third as *irreducible error*.

From here, it might be useful to define categories of estimators in general:

$$\begin{aligned}f_s &= \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \text{loss}(f(x_i), y_i) \\ f_c &= \min_{f \in \mathcal{F}} \mathbb{E}\left[\text{loss}(f(x_i), y_i)\right] \\ f_* &= \min_{f} \mathbb{E}\left[\text{loss}(f(x_i), y_i)\right]\end{aligned}$$

11.3 Bias-Variance in Linear Regression

Let's continue with the example in the previous section but let $y_i = x_i^\top \theta_* + \epsilon_i$, where θ_* is the true parameter we want to estimate and $\mathbb{E}[\epsilon] = 0$ and $\text{Cov}(\epsilon) = \sigma^2 I$. Let $Y \in \mathbb{R}^n$, $X \in \mathbb{R}^{n \times d}$ and $\epsilon \in \mathbb{R}^n$ be the vectors and matrices containing the data. Alternatively, we could write

$$Y = X\theta_* + \epsilon$$

We will use the standard least squares estimator for this problem:

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{2} \|Y - X\theta\|_2^2$$

From earlier, we saw that the closed form solution is just

$$\hat{\theta} = (X^\top X)^{-1} X^\top Y$$

Now we want to find the bias and variance of estimator given the earlier conditions. Taking the expectation of the estimator gives us:

$$\begin{aligned}\mathbb{E}[\hat{\theta}] &= \mathbb{E}\left[(X^\top X)^{-1} X^\top (X\theta_* + \epsilon)\right] \\ &= \theta_*\end{aligned}$$

Since ϵ is zero-mean. On the other hand. To compute the variance, we will first show that the variance is the trace of the covariance of the estimator. By expanding the MSE, the term that we will get for the variance is just $\mathbb{E}\|\hat{\theta} - \mathbb{E}[\hat{\theta}]\|_2^2$. Using the properties of trace and covariance, we can get:

$$\begin{aligned}
\mathbb{E}\|\hat{\theta} - \mathbb{E}[\hat{\theta}]\|_2^2 &= \mathbb{E}\mathrm{Tr}\|\hat{\theta} - \mathbb{E}[\hat{\theta}]\|_2^2 \\
&= \mathbb{E}\mathrm{Tr}\left[(\hat{\theta} - \mathbb{E}[\hat{\theta}])^\top(\hat{\theta} - \mathbb{E}[\hat{\theta}])\right] \\
&= \mathbb{E}\mathrm{Tr}\left[(\hat{\theta} - \mathbb{E}[\hat{\theta}])(\hat{\theta} - \mathbb{E}[\hat{\theta}])^\top\right] \\
&= \mathrm{Tr}\mathbb{E}\left[(\hat{\theta} - \mathbb{E}[\hat{\theta}])(\hat{\theta} - \mathbb{E}[\hat{\theta}])^\top\right] \\
&= \mathrm{Tr}(\mathrm{Cov}(\hat{\theta}))
\end{aligned}$$

Now we must solve for the covariance of our estimator:

$$\begin{aligned}
\mathrm{Cov}(\hat{\theta}) &= \mathrm{Cov}((X^\top X)^{-1}X^\top Y) \\
&= (X^\top X)^{-1}X^\top \mathrm{Cov}(Y)(X^\top X)^{-1}X^\top)^\top \\
&= (X^\top X)^{-1}X^\top \mathrm{Cov}(\epsilon)X(X^\top X)^{-1}
\end{aligned}$$

Taking the trace of this gives us:

$$\begin{aligned}
\mathrm{Tr}[\mathrm{Cov}(\hat{\theta})] &= \mathrm{Tr}\left[(X^\top X)^{-1}X^\top \mathrm{Cov}(\epsilon)X(X^\top X)^{-1}\right] \\
&= \sigma^2\mathrm{Tr}\left[(X^\top X)^{-1}X^\top X(X^\top X)^{-1}\right] \\
&= \sigma^2\mathrm{Tr}\left[(X^\top X)^{-1}\right]
\end{aligned}$$

We know that the trace of a matrix is just the sum of the eigenvalues (this is simple to show for symmetric matrices via the Spectral Theorem) and that the eigenvalues of some matrix A^{-1} are the reciprocals of the eigenvalues of A. Let $\gamma_1, \ldots, \gamma_d$ be the eigenvalues of $X^\top X$. Then we have

$$\sigma^2\mathrm{Tr}\left[(X^\top X)^{-1}\right] = \sigma^2\sum_{i=0}^{d}\frac{1}{\gamma_i}$$

11.4 Regularization

Again, consider a single $x \sim \mathcal{N}(\mu, \sigma^2 I)$ or alternatively $x = \mu + \epsilon$, where $\epsilon \sim \mathcal{N}(\mu, \sigma^2 I)$. Our goal is to estimate μ, but if we take the mean squared error $\|\hat{x} - x\|_2^2$ we notice that we're fitting to the noise, which motivates the regularization. The optimum of $J(x) = \frac{1}{2}\|\hat{x} - x\|_2^2 + \frac{\lambda}{2}\|\hat{x}\|_2^2$ is just $\frac{1}{1+\lambda}x$ which we realize is similar to the form of the James-Stein estimator $\hat{x} = \alpha x$. This is considered noise rejection via regularization.

Another benefit of regularization is reducing number of possible solutions to optimization problem (overfitting). So, regularization solves two problems (reducing fit to noise and overfitting which are actually different ideas (*e.g.* you can have non-noisy data and still overfit because of complicated models).

12 Learning Theory and Generalization

Many times in lecture we have seen the following equation:

$$R[w] = R[w] - R_T[w] + R_T[w]$$

where $R[w] = \mathbb{E}\left[\text{loss}(w, (x, y))\right]$ and $R_T[w] = \frac{1}{n}\sum_{i=1}^{n} \text{loss}(w, (x_i, y_i))$. So we can think of our expected prediction error in a new way: it is the sum of the generalization error and the training error.

If we fix a w (i.e. don't look at the data), then

$$\begin{aligned}
\mathbb{E}\left[R_T[w]\right] &= \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} \text{loss}(w, (x_i, y_i))\right] \\
&= \frac{1}{n}\sum \mathbb{E}\left[\text{loss}(w, (x_i, y_i))\right] \\
&= \frac{1}{n}\sum R[w] \\
&= R[w]
\end{aligned}$$

So the empirical risk is an unbiased estimator of the risk as long as w is fixed. For the variance, let us first assume that the loss is bounded $\text{loss}(w, (x, y)) \leq B$. Then:

$$\begin{aligned}
\text{Var}(R_T[w]) &= \text{Var}\left(\frac{1}{n}\sum_i \text{loss}(w, (x_i, y_i))\right) \\
&= \frac{1}{n^2}\text{Var}\left(\sum_i \text{loss}(w, (x_i, y_i))\right) \\
&= \frac{1}{n^2}\sum_i \text{Var}\left(\text{loss}(w, (x_i, y_i))\right) \\
&\leq \frac{B^2}{n}
\end{aligned}$$

13 Kernels and Kernel Methods

This section will cover kernels for ridge regression and not support vector machines but the concept is the same. The point of kernel methods is to move past linear features in a more efficient way.

$$\min_w \frac{1}{2}\|Xw - Y\|^2 + \frac{\lambda}{2}\|w\|^2$$

Given the above minimization problem, it can be shown that to minimize over $w \in \mathbb{R}^d$ is to minimize over $\alpha \in \mathbb{R}^n$ where $w = X^\top \alpha$. Let $w = w_n + X^\top \alpha$ where

$Xw_n = 0$ via orthogonal decomposition mentioned earlier:

$$\begin{aligned}\min_w \frac{1}{2}\|Xw - Y\|^2 + \frac{\lambda}{2}\|w\|^2 &= \min_{\alpha, w_n} \frac{1}{2}\|X(X^\top\alpha + w_n) - Y\|^2 + \frac{\lambda}{2}\|X^\top\alpha + w_n\|^2 \\ &= \min_{\alpha, w_n} \frac{1}{2}\|XX^\top\alpha - Y\|^2 + \frac{\lambda}{2}\left(\alpha XX^\top\alpha + 2w_n^\top X^\top\alpha + w_n^\top w_n\right) \\ &= \min_{\alpha} \frac{1}{2}\|XX^\top\alpha - Y\|^2 + \frac{\lambda}{2}\alpha XX^\top\alpha\end{aligned}$$

The first term does not depend on w_n and the regularization term is minimized when $w_n = 0$.

$K = XX^\top$ is the *Gram Matrix*, a matrix of inner products. Then $K_{ij} = x_i^\top x_j$. The optimization problem that we're trying to solve depends only on n, not d.

α can be found by taking the gradient and setting it to zero resulting in $\alpha = (K + \lambda I)^{-1}Y$.

For any α, evaluating a test point x is just $w^\top x = (X^\top\alpha)^\top x = \sum_{i=1}^n \alpha_i x_i^\top x = \sum_{i=1}^n \alpha_i K(x_i, x)$. So evaluation depends only on the inner products.

A function $K(x, z)$ could act like an inner product of some feature lifting $\varphi(\cdot)$ of x and z, even if the feature lifting isn't explicitly computed. The only requirement is that K is positive semi-definite. If this is the case then using the Spectral Theorem with non-negative entries in Λ:

$$K = V^\top\Lambda V = (\Lambda^{1/2}V)^\top(\Lambda^{1/2}V)$$

Define $x_i = \lambda_i^{1/2}v_i$. Therefore K_{ij} can be written as $x_i^\top x_j$.

There are many functions that might allow for this:

- $K(x, z) = x^\top z$ (linear)
- $K(x, z) = (x^\top z + 1)^p$ for $p \in \mathbb{Z}$ (polynomial)
- $K(x, z) = \exp(-\gamma\|x - z\|^2)$ (Gaussian)

14 Unsupervised Learning

The general idea of unsupervised learning is to find some sort of structure in unlabeled data. Several ideas fall under this categorization. Most unsupervised learning techniques end up being matrix factorization (e.g. $X = AB$ where $A \in \mathbb{R}^{d\times r}$, $B \in \mathbb{R}^{r\times n}$ and $X \in \mathbb{R}^{d\times n}$.

14.1 Introduction

If we could identify some structure in the data given in an $n \times d$ matirx X, maybe we would not have to use all of it and can therefore solve the same problems with fewer features. Primary motivations for this are run-time, storage, generalization and interpretability. Generalization is affected because the number of samples you need in order to generalize often scales with dimension d.

The other goal we could aim for is sample reduction, or clustering.

14.2 Singular Value Decomposition

Every $X \in \mathbb{R}^{d \times n}$ admits the factorization $X = USV^\top$ for $U \in \mathbb{R}^{d \times d}$, $S \in R^{d \times d}$, and $V \in \mathbb{R}^{n \times d}$. S is diagonal. Let $n > d$. In some cases, people will take S to be rectangular and V to be square, but it does not matter. Note that usually matrices have been written as $X \in \mathbb{R}^{n \times d}$. Also $U^\top U = I_d$ and $V^\top V = I_d$ and $S = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_d)$ where $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_d$. X can also be written as:

$$X = \sum_{i=1}^{d} u_i \sigma_i v_i^\top$$

which can help us form a geometric interpretation of a linear transformation that is Xz for $z \in \mathbb{R}^n$:

$$Xz = \sum_{i=1}^{d} u_i \sigma_i (v_i^\top z)$$

When we take a linear transformation, we determine how much z lies in v_i then scale by σ_i and map according to u_i.

As an example, if we look at the unit circle, we transform via rotation into the coordinates of V, then scale components according to S and then rotate again to align with U vectors.

What is the relationship between eigenvalues and singular values? We know that $Xv_i = \sigma_i u_i$ and that $X^\top = VS^\top U^\top$. So then $X^\top X v_i = \sigma_i X^\top u_i = \sigma_i^2 v_i$.

So v_i is an eigenvector of $X^\top X$ with eigenvalue $\lambda_i = \sigma_i^2$. Same concept is applied for the u_i vectors: $XX^\top u_i = \sigma_i^2 u_i$.

Now, assume that A is positive semi-definite. Therefore we know that A has a representation $A = W\Lambda W^\top$ by the spectral theorem where $\Lambda = \text{diag}(\lambda_i)$ and $WW^\top = I_d$. In this case the eigenvalue decomposition is equal to the singular value decomposition just by definition.

Assume that B is symmetric but not necessarily psd. We can also use the spectral theorem to get $B = W\Lambda W^\top$. In this case, if we order the eigenvalues as $\lambda_1 > \ldots > \lambda_k > 0 > \lambda_{k+1} > \lambda_d$, then we define a diagonal matrix Γ where $\Gamma_{ii} = 1$ if $i \leq k$ and $\Gamma_{ii} = -1$ if $i > k$ and zero elsewhere. Then we can see that $B = W\Lambda\Gamma(W\Gamma)^\top$. Then we have the singular value decomposition where $\sigma_i = |\lambda_i|$. The eigenvalues of a matrix have no relation to the singular values (see example from lecture).

What if we want to find $\max_{\|z\|=1} \|Cz\|$ for some matrix C?

$$\begin{aligned} \|Cz\|^2 &= z^\top V S^2 V^\top z \\ &= \sum_i \sigma_i^2 (v_i^\top z)^2 \end{aligned}$$

which is maximized when $z = v_1$.

If we have a matrix A that is rank deficient such that $\sigma_{r+1} = 0$ and $\sigma_j = 0$ for all $j > r+1$, then we know that

$$A = \sum_{i=1}^{r} \sigma_i u_i v_i^\top$$

which means that our matrix can be written much more concisely (factored). In this case, we can write $X = U_r S_r V_r^\top$ where $U_r = \begin{bmatrix} u_1 & \dots & u_r \end{bmatrix}$, $S_r = \text{diag}(\sigma_1, \dots, \sigma_r)$ and $V_r = \begin{bmatrix} v_1 & \dots & v_r \end{bmatrix}$

Now consider the case where we let $\hat{X} = U_r^\top X = S_r V_r^\top$. So we've reduced $\hat{X}$ to a lower dimensional state. Given any classifier w, using the fundamental theorem of linear algebra, we can write $w = U_r \alpha + w_n$ where $U_r^\top w_n = 0$. So if we look at $w^\top X$ knowing that X has the form $X = U_r S_r V_r^\top$, then

$$\begin{aligned} w^\top X &= (U_r \alpha)^\top X + w_n^\top U_r S_r V_r^\top \\ &= (U_r \alpha)^\top X \end{aligned}$$

So to do any sort of classification, the only component we need is $U_r \alpha$. This tells us that if we train w in the high dimensional space, we can transform it down to the low dimensional space and have the same classifier (and vice-versa). Equivalently we could just train on the low dimensional space much more efficiently. This is because hitting X with w is the same has hitting $\hat{X}$ with a lower dimensional α.

14.3 Principal Component Analysis

What does it mean for data to have low rank? In the two dimensional case, this could be a line. Often times, data is not so simple, but we might find that certain singular values are much larger than others. In other words, the data is explainable along a subset of the singular vectors.

The process of removing these unimportant components is called *Principal Component Analysis*. The algorithm follows several steps:

1. Take as input data $X \in \mathbb{R}^{d \times n}$ and target dimension r.
2. Center X by computing $X_c = \begin{bmatrix} x_1 - \mu & \dots & x_n - \mu \end{bmatrix}$ where $\mu = \frac{1}{n} \sum_{i=1}^{n} x_i$.
3. Compute $\text{svd}(X_c) = USV^\top$
4. Return $\left[\hat{X} = S_r V_r^\top, U_r, \mu \right]$

U_r and μ will be used for transforming and centering new data. So consider a new data point x'. Our processed version will be $\hat{x}' = U_r^\top (x' - \mu)$.

One way that we can think of PCA is that we are identifying the directions of maximum variance. E.g. $r = 1$. Then we can setup the optimization problem

$\max_{\|u\|=1} \text{var}(u^\top x_i)$, where we're trying to find the direction. If we assume that the mean is zero:

$$\begin{aligned}
\text{var}(u^\top x_i) &= \frac{1}{n}\sum_{i=1}^{n}(u^\top x_i)^2 \\
&= \frac{1}{n}\sum_{i=1}^{n} u^\top x_i x_i^\top u \\
&= \frac{1}{n} u^\top \left(\sum_{i=1}^{n} x_i x_i^\top\right) u \\
&= \frac{1}{n} u^\top X X^\top u
\end{aligned}$$

So the maximum value of this is just the maximum eigenvalue of $XX^\top$ which is σ_1^2 and the maximizer is $u = u_1$.

We can now remove the influence of u_1 on each x_i by computing $\tilde{x}_i = x_i - (u_1^\top x_i)u_i$. So if we end up with $\tilde{X}$ which is orthogonal to u_1, then we can again try to find that direction of maximum variance, which will be u_2. This process can be continued.

The total variance in our data is just

$$\sum_{i=1}^{d} \sigma_i(X)^2$$

where $\sigma_i(A)$ indicates the i-th singular value of A. And the total variance in the reduced data set is

$$\sum_{i=1}^{r} \sigma_i(X)^2$$

So we can use these two to compare the amount of variance that we capture in the reduced data set.

15 Unsupervised Clustering

Motivations for clustering data points include segmentation, archetype identification, fast lookups, etc.

15.1 k-means Clustering

Partition points into k clusters where μ_j is the mean of cluster j and x_i is in cluster j if $\|x_i - \mu_j\|_2^2 \leq \|x_i - \mu_{j'}\|_2^2 \ \forall j'$. $C_j = \{i : x_i \text{ in cluster } j\}$. $\mu_j = \frac{1}{|C_j|}\sum_{i \in C_j} x_i$.

Cost of k-means algorithm is measured by

$$\min_{\mu_1,\ldots,\mu_k} \sum_{i=1}^{n} \min_{1 \leq j \leq k} \|x_i - \mu_j\|_2^2$$

This optimization problem is hard to solve. Algorithm:

1. Initialize $\mu_1, \ldots, \mu_k$
2. Assign $i \in C_j$ if $\|x_i - \mu_j\|^2 \leq \|x_i - \mu_{j'}\|^2$ for $j \neq j'$.
3. Set $\mu_j = \frac{1}{|C_j|} \sum_{i \in C_j} x_i$ for $1 \leq j \leq k$.
4. Unless assignments don't change, goto 2.

This algorithm is an example of alternating minimization where the overall problem is hard to solve but by alternating which components are fixed, the problem is divided into small, easier problems. Fix the means, then find the best assignment. Fix the assignments, then find the best means.

Initialization could be done randomly. Better algorithm is k-means++.

- Pick μ_1 at random from the data.
- For $c = 1, \ldots, k-1$.
  - Define for $i = 1, \ldots, n$, $d_i = \min_{1 \leq j \leq c} \|x_i - \mu_j\|^2$
  - Set $p_i = \frac{d_i}{\sum_i d_i}$
  - Set $\mu_{c+1} = x_i$ w.p. p_i.

This ensures that no mean will be picked twice since min distance is zero.

15.2 Hierarchical Clustering

The general idea here is to find a hierarchy among the data by iteratively fusing clusters. First define a method of measuring distance between clusters. There could be many ways but one is average linkage:

$$d(A, B) = \frac{1}{|A||B|} \sum_{a \in A} \sum_{b \in B} \text{dist}(a, b)$$

Another is centroid linkage:

$$d(A, B) = \text{dist}(\mu_A, \mu_B)$$

Greedy Algorithm: initialize with n clusters (one for each point). $C_i = x_i$. For all pairs of clusters, (A, B), compute $d(A, B)$. $C_{new} = A \cup B$ where $d(A, B)$ is minimized. Terminates when just one cluster remains.

15.3 Spectral Clustering

View the data as a graph with data points as notes and similarities as edges. So $sim(x_i, x_j) = \frac{x_i^\top x_j}{\|x_i\| \|x_j\|}$, or $sim(x_i, x_j) = k(x_i, x_j)$ for some kernel k, or $sim(x_i, x_j) = 1$ if $\|x_i - x_j\| \leq D_0$ and 0 otherwise.

Graph partition occurs when two subsets V_1 and V_2 of a graph V have the properties that $V_1 \cup V_2 = V$ and $V_1 \cap V_2 = 0$.

Define the $cut(V_1, V_2)$ as the weight of the edges that have one node in V_1 and one node in V_2. So

$$cut(V_1, V_2) = \sum_{i \in V_2} \sum_{j \in V_1} w_{ij}$$

Want to minimize the amount that gets cut, but also encourage not cutting nothing. Balanced cut: minimize $cut(V_1, V_2)$ subject to $|V_1| = |V_2| = n/2$.

Define a cut indicator $v \in \mathbb{R}^n$ where $v_i = 1$ if $i \in V_1$ and -1 if $i \in V_2$. So then the equation for the cut becomes:

$$\begin{aligned} cut(V_1, V_2) &= \frac{1}{4} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij}(v_i - v_j)^2 \\ &= \frac{1}{4} v^\top L v \end{aligned}$$

where $L_{ij} = -w_{ij}$ if $i \neq j$ and $L_{ij} = \sum_{k \neq i} w_{ik}$ if $i = j$. L is the graph Laplacian. L is symmetric and positive semi definite if all the weights are not negative. Also, $L1 = 0$ which is L times the all-ones vector.

Then the problem becomes minimize $v^\top L v$ subject to $v_i \in \{1, -1\}$ and $1^\top v = 0$. But we can do an approximation for the first condition and only constrain $v^\top v = n$. If the ones constraint did not exist, then the minimum is the smallest eigenvalue but v must be orthogonal to the ones vector and so the solution becomes the second smallest eigenvalue. This is because the smallest eigenvalue corresponds to the all-ones vector.

The components of v are no longer 1 and -1, so we determine which set they are part of by their signs.