Notes on Andrew Ng's CS 229 Machine Learning Course

Tyler Neylon

331.2016

These are notes I'm taking as I review material from Andrew Ng's CS 229 course on machine learning. Specifically, I'm watching these videos and looking at the written notes and assignments posted here. These notes are available in two formats: html and pdf.

I'll organize these notes to correspond with the written notes from the class.

As I write these notes, I'm also putting together some homework solutions.

1 On lecture notes 1

The notes in this section are based on lecture notes 1.

1.1 Gradient descent in general

Given a cost function $J(\theta)$, the general form of an update is

$$\theta_j := \theta_j - \alpha \frac{\partial J}{\partial \theta_j}.$$

It bothers me that α is an arbitrary parameter. What is the best way to choose this parameter? Intuitively, it could be chosen based on some estimate or actual value of the second derivative of J. What can be theoretically guaranteed about the rate of convergence under appropriate conditions?

Why not use Newton's method? A general guess: the second derivative of J becomes cumbersome to work with.

It seems worthwhile to keep my eye open for opportunities to apply improved optimization algorithms in specific cases.

1.2 Gradient descent on linear regression

I realize this is a toy problem because linear regression in practice is not solve iteratively, but it seems worth understanding well. The general update equation is, for a single example i,

$$\theta_j := \theta_j + \alpha(y^{(i)} - h_\theta(x^{(i)}))x_j^{(i)}.$$

The delta makes sense in that it is proportional to the error $y - h_\theta$, and in that the sign of the product $(y - h_\theta)x$ guarantees moving in the right direction. However, my first guess would be that the expression $(y - h_\theta)/x$ would provide a better update.

For example, suppose we have a single data point (x, y) where $x \neq 0$, and a random value of θ. Then a great update would be

$$\theta_1 := \theta_0 + (y - \theta_0 x)/x,$$

since the next hypothesis value h_θ would then be

$$h_\theta = \theta_1 x = \theta_0 x + y - \theta_0 x = y,$$

which is good. Another intuitive perspective is that we should be making *bigger* changes to θ_j when x_j is *small*, since it's harder to influence h_θ for such x values.

This is not yet a solidified intuition. I'd be interested in revisiting this question if I have time.

1.3 Properties of the trace operator

The trace of a square matrix obeys the nice property that

$$\text{tr } AB = \text{tr } BA. \tag{1}$$

One way to see this is to note that

$$\text{tr } AB = a_{ij}b_{ji} = \text{tr } BA,$$

where I'm using the informal shorthand notation that a variable repeated within a single product implies that the sum is taken over all relevant values of that variable. Specifically,

$$a_{ij}b_{ji} \text{ means } \sum_{i,j} a_{ij}b_{ji}.$$

I wonder if there's a more elegant way to verify (1).

Ng gives other interesting trace-based equations, examined next.

- Goal: $\nabla_A \text{tr } AB = B^T$.

Since

$$\text{tr } AB = a_{ij}b_{ji},$$

we have that

$$(\nabla_A \text{tr } AB)_{ij} = b_{ji},$$

verifying the goal.

- Goal: $\nabla_{A^T} f(A) = (\nabla_A f(A))^T$.

This can be viewed as

$$(\nabla_{A^T} f(A))_{ij} = \frac{\partial f}{\partial a_{ji}} = (\nabla_A f(A))_{ji}.$$

- Goal: $\nabla_A \text{tr}(ABA^TC) = CAB + C^TAB^T$.

I'll use some nonstandard index variable names below because I think it will help avoid possible confusion. Start with

$$(ABA^TC)_{xy} = a_{xz}b_{zw}a_{vw}c_{vy}.$$

Take the trace of that to arrive at

$$\alpha = \text{tr}(ABA^TC) = a_{xz}b_{zw}a_{vw}c_{vx}.$$

Use the product rule to find $\frac{\partial\alpha}{\partial a_{ij}}$. You can think of this as, in the equation above, first setting $xz = ij$ for one part of the product rule output, and then setting $vw = ij$ for the other part. The result is

$$(\nabla_A \alpha)_{ij} = b_{jw}a_{vw}c_{vi} + a_{xz}b_{zj}c_{ix} = c_{vi}a_{vw}b_{jw} + c_{ix}a_{xz}b_{zj}.$$

(The second equality above is based on the fact that we're free to rearrange terms within products in the repeated-index notation being used. Such rearrangement is commutativity of numbers, not of matrices.)

This last expression is exactly the ij^{th} entry of the matrix $CAB + C^TAB^T$, which was the goal.

1.4 Derivation of the normal equation

Ng starts with

$$\nabla_\theta J(\theta) = \nabla_\theta \frac{1}{2}(X\theta - y)^T(X\theta - y),$$

and uses some trace tricks to get to

$$X^TX\theta - X^Ty.$$

I thought that the trace tricks were not great in the sense that if I were faced with this problem it would feel like a clever trick out of thin air to use the trace (perhaps due to my own lack of experience with matrix derivatives?), and in the sense that the connection between the definition of $J(\theta)$ and the result is not clear.

Next is another approach.

Start with $\nabla_\theta J(\theta) = \nabla_\theta \frac{1}{2}(\theta^TZ\theta - 2y^TX\theta)$; where $Z = X^TX$, and the doubled term is a valid combination of the two similar terms since they're both real numbers, so we can safely take the transpose of one of them to add them together.

The left term, $\theta^TZ\theta$, can be expressed as $w = \theta_{i1}Z_{ij}\theta_{j1}$, treating θ as an $(n{+}1)\times 1$ matrix. Then $(\nabla_\theta w)_{i1} = Z_{ij}\theta_{j1} + \theta_{j1}Z_{ji}$, using the product rule; so $\nabla_\theta w = 2Z\theta$, using that Z is symmetric.

The right term, $v = y^TX\theta = y_{i1}X_{ij}\theta_{j1}$ with $(\nabla_\theta v)_{i1} = y_{j1}X_{ji}$ so that $\nabla_\theta v = X^Ty$.

These all lead to $\nabla_\theta J(\theta) = X^TX\theta - X^Ty$ as before. I think it's clearer, though, once you grok the sense that

$$\begin{aligned}\nabla_\theta \theta^TZ\theta &= Z\theta + (\theta^TZ)^T, \text{and}\\ \nabla_\theta u^T\theta &= u,\end{aligned}$$

both as intuitively straightforward matrix versions of derivative properties.

I suspect this can be made even cleaner since the general product rule

$$\nabla(f\cdot g) = (\nabla f)\cdot g + f\cdot(\nabla g)$$

holds, even in cases where the product fg is not a scalar; e.g., that it is vector- or matrix-valued. But I'm not going to dive into that right now.

Also note that $X^T X$ can easily be singular. A simple example is $X = 0$, the scalar value. However, if X is $m \times n$ with rank n, then $X^T X e_i = X^T x^{(i)} \neq 0$ since $\langle x^{(i)}, x^{(i)} \rangle \neq 0$. (If $\langle x^{(i)}, x^{(i)} \rangle = 0$ then X could not have rank n.) So $X^T X$ is nonsingular iff X has rank n.

Ng says something in a lecture (either 2 or 3) that implied that $(X^T X)^{-1} X^T$ is *not* the pseudoinverse of X, but for any real-valued full-rank matrix, it *is*.

1.5 A probabilistic motivation for least squares

This motivation for least squares makes sense when the error is given by i.i.d. normal curves, and often this may seem like a natural assumption based on the central limit theorem.

However, this same line of argument could be used to justify any cost function of the form

$$J(\theta) = \sum_i f(\theta, x^{(i)}, y^{(i)}),$$

where f is intuitively some measure of distance between $h_\theta(x^{(i)})$ and $y^{(i)}$. Specifically, model the error term $\varepsilon^{(i)}$ as having the probability density function $e^{-f(\theta, x^{(i)}, y^{(i)})}$. This is intuitively reasonable when f has the aforementioned distance-like property, since error values are likely near zero and unlikely far from zero. Then the log likelihood function is

$$\ell(\theta) = \log L(\theta) = \log \prod_i e^{-f(\theta, x^{(i)}, y^{(i)})} = \sum_i -f(\theta, x^{(i)}, y^{(i)}),$$

so that maximizing $L(\theta)$ is the same as minimizing the $J(\theta)$ defined in terms of this arbitrary function f. To be perfectly clear, the motivation Ng provides only works when you have good reason to believe the error terms are indeed normal. At the same time, using a nice simple algorithm like least squares is quite practical even if you don't have a great model for your error terms.

1.6 Locally weighted linear regression (LWR)

This idea is that, given a value x, we can choose θ to minimize the modified cost function

$$\sum_i w^{(i)} (y^{(i)} - \theta^T x^{(i)})^2,$$

where

$$w^{(i)} = \exp\left(-\frac{(x^{(i)} - x)^2}{2\tau^2}\right).$$

I wonder: Why not just compute the value

$$y = \frac{\sum_i w^{(i)} y^{(i)}}{\sum_i w^{(i)}}$$

instead?

I don't have a strong intuition for which approach would be better, although the linear interpolation is more work (unless I'm missing a clever way to implement LWR that wouldn't also work for the simpler equation immediately above). This also reminds me of the $k-$nearest neighbors algorithm, though I've seen that presented as a classification approach while LWR is regression. Nonetheless, perhaps one could apply a locality-sensitive hash to quickly approximately find the k nearest neighbors and then build a regression model using that.

1.7 Logistic regression

This approach uses a hypothesis function of the form

$$h_\theta(x) = g(\theta^T x),$$

where

$$g(z) = 1/(1 + e^{-z}).$$

The update equation from gradient descent turns out to be nice for this setup. However, besides "niceness," it's not obvious to me why the logistic function g is signifcantly better than any other sigmoid function.

In general, suppose

$$h_\theta(x) = \tau(\theta^T x),$$

where τ is any sigmoid function. Then

$$[\nabla_\theta \ell(\theta)]_j = \sum_i \left(\frac{y^{(i)}}{h^{(i)}} - \frac{1 - y^{(i)}}{1 - h^{(i)}}\right) \frac{\partial \tau^{(i)}}{\partial \theta_j}.$$

Here, $\tau^{(i)}$ indicates the function τ evaluated on θ and $x^{(i)}$. We can split up the term inside the sum like so:

$$a^{(i)} = \frac{y^{(i)}}{h^{(i)}} - \frac{1 - y^{(i)}}{1 - h^{(i)}} = \frac{y^{(i)} - h^{(i)}}{h^{(i)}(1 - h^{(i)})},$$

and

$$b^{(i)} = \frac{\partial \tau^{(i)}}{\partial \theta_j}.$$

Intuitively, the value of $a^{(i)}$ makes sense as it's the error of $h^{(i)}$ weighted more heavily when a wrong output value of h is given with higher confidence. The value $b^{(i)}$ makes sense as the direction in which we'd want to move θ_j in order to increase $\tau^{(i)}$. Multiplied by the sign of $a^{(i)}$, the end result is a movement of θ_j that decreases the error.

It's not obvious to me which choice of function τ is best. Consider, for example, the alternative function

$$\tau(z) = \frac{\arctan(z)}{\pi} + \frac{1}{2}.$$

In this case, when $z = \theta^T x$,

$$b^{(i)} = \frac{\partial \tau^{(i)}}{\partial \theta_j} = \frac{x_j^{(i)}}{1 + (\theta^T x)^2}.$$

In general, sigmoid functions will tend to give a value of $b^{(i)}$ that is small when $|\theta^T x|$ is large, symmetric around $\theta^T x = 0$, and has the same sign as x_j since $\tau - 1/2$ will be an increasing odd function; the derivative of an increasing odd function is even and positive; and this positive even function will be multiplied by x_j to derive the final value of $b^{(i)}$. The values of $b^{(i)}$ will be small for large $|\theta^T x|$ because $\tau(\theta^T x)$ necessarily flattens out for large input values (since it's monotonic and bounded in $[0, 1]$).

Ng states that the assumption of a Bernoulli distribution for $p(y|x;\theta)$ results in the traditional gradient update rule for logistic regression, but there is another choice available in the argument. The definition of a Bernoulli distribution means that we must have

$$p(y|x;\theta) = \begin{cases} h_\theta(x) & \text{if } y = 1, \text{ and} \\ 1 - h_\theta(x) & \text{otherwise.} \end{cases}$$

There are a number of expressions that also capture this same behavior for the two values $y = 0$ and $y = 1$. The one used in the course is

$$p(y|x;\theta) = h^y(1-h)^{1-y},$$

which is convenient since we're working with the product $p(y) = \prod_i p(y^{(i)})$. However, we could have also worked with

$$p(y|x;\theta) = yh + (1-y)(1-h),$$

which also gives exactly the needed values for the cases $y = 1$ and $y = 0$. Although the traditional expression is easier to work with, it seems worth nothing that there is a choice to be made.

1.8 Using Newton's method for logistic regression

Ng provides the alternative logistic regression update formula

$$\theta := \theta - H^{-1}\nabla_\theta \ell(\theta) \tag{2}$$

without justifying it beyond explaining that it's the vector version of what one would do in a non-vector setting for finding an optima of $\ell'(\theta)$, namely,

$$\theta := \theta - \frac{\ell'(\theta)}{\ell''(\theta)}.$$

The H in the above equation is the *Hessian* of ℓ, which I'll write as

$$H = (h_{ij}) = \left(\frac{\partial^2 \ell}{\partial\theta_i \partial\theta_j}\right).$$

Here is one way to arrive at (2):

$$\big[\nabla\ell(\theta+\alpha)\big]_i \approx \big[\nabla\ell(\theta)\big]_i + \sum_j \frac{\partial}{\partial\theta_j}\big[\nabla\ell(\theta)\big]_i \cdot \alpha_j. \tag{3}$$

This is the linearized approximation of $\nabla\ell$ around θ; another perspective is to see this as the first-order Taylor polynomial of $\big[\nabla\ell\big]_i$ around θ.

We can rewrite (3) more concisely as

$$\nabla\ell(\theta+\alpha) \approx \nabla\ell(\theta) + H\alpha. \tag{4}$$

Recall that our goal is to iterate on θ toward a solution to $\nabla \ell(\theta) = 0$. Use (4) toward this end by looking for α which solves $\nabla \ell(\theta + \alpha) \approx 0$. If H is invertible, then we have

$$\nabla \ell(\theta) + H\alpha = 0 \quad \Leftrightarrow \quad H\alpha = -\nabla \ell(\theta) \quad \Leftrightarrow \quad \alpha = -H^{-1}\nabla \ell(\theta).$$

In other words, the desired iteration is

$$\theta := \theta + \alpha = \theta - H^{-1}\nabla \ell(\theta),$$

which confirms the goal, equation (2).

1.9 The exponential family

A class of distributions fits within the exponential family when there are functions a, b, and T such that

$$p(y; \eta) = b(y) \exp(\eta^T T(y) - a(\eta)) \tag{5}$$

expresses a distribution in the class.

Ng uses this form to justify the particular use of $g(z) = 1/(1 + e^{-z})$ in logistic regression. I see how expressing a Bernoulli distribution in this form naturally result in the traditional form of logistic regression. However, as of the end of lecture 4, I don't think Ng has justified the power of the exponential family beyond explaining that it captures many popular distribution classes. (To be clear, I believe that this idea is very useful; I just am not convinced that *why* it's useful has been explicitly explained yet.)

I haven't fully grokked the exponential family yet, but I noticed that the main restriction in expression (5) seems to be that the only interaction between η and y is multiplicative in the sense that we could equally-well have written the density function as

$$p(y; \eta) = b(y)c(\eta) \prod_i d_i(y_i)^{\eta_i},$$

in the case that both η and y are vectors.

1.10 Generalized linear models

Ng doesn't spell this out, but I think the following line of argument is why generalized linear models are considered useful.

Suppose $\eta = \theta^T x$ and that we're working with a distribution parametrized by η that can be expressed as

$$p(y;\eta) = b(y) \exp\big(\eta^T T(y) - a(y)\big).$$

Then the log likelihood function is given by

$$\ell(\theta) = \sum_i \log\big(p(y^{(i)};\eta^{(i)})\big) = \sum_i \log(b(y^{(i)})) + \eta^{(i)T} T(y^{(i)}) - a(\eta^{(i)}).$$

We can express the gradient of this as

$$\nabla \ell(\theta) = \sum_i x^{(i)} \big(T(y^{(i)}) - a'(\eta^{(i)})\big),$$

which corresponds to the gradient ascent update rule

$$\theta := \theta + \alpha\big(T(y) - a'(\theta^T x)\big)x. \tag{6}$$

Let's see how this nice general update rule looks in a couple special cases. For simplicity, I'll stick to special cases where η is a scalar value.

1.10.1 Gaussian GLM

In this case, $a(\eta) = \eta^2/2$ so $a'(\eta) = \eta$; and $T(y) = y$. Then the update rule (6) becomes

$$\theta := \theta + \alpha(y - \theta^T x)x,$$

which is just gradient ascent applied to least squares.

1.10.2 Logistic regression GLM

In this case, $a(\eta) = \log(1 + e^\eta)$ and $T(y) = y$. So $a'(\eta) = 1/(1 + e^{-\eta})$, and we can write the update rule (6) as

$$\theta := \theta + \alpha(y - g(\theta^T x))x,$$

where g is the logistic function. This matches the earlier update equation we saw for logistic regression, suggesting that our general approach isn't completely insane. It may even be technically correct.

2 On lecture notes 2

The notes in this section are based on lecture notes 2.

2.1 Why Gaussian discriminant analysis is like logistic regression

Ng mentions this fact in the lecture and in the notes, but he doesn't go into the details of justifying it, so let's do that.

The goal is to show that

$$p(y = 1 \mid x) = \frac{1}{1 + e^{-\theta^T x}}, \tag{7}$$

where θ is some function of the learned parameters ϕ, Σ, μ_0, and μ_1; and we can consider x as being augmented by a new coordinate $x_0 = 1$ to effectively allow the addition of a constant in the expression $\theta^T x$. In Gaussian discriminant analysis, we learn a model for $p(x \mid y)$ and for $p(y)$. Specifically,

$$\begin{array}{rcl} p(y) & = & \phi^y (1-\phi)^{1-y}, \\ p(x \mid y = 0) & = & N(\mu_0, \Sigma), \text{ and} \\ p(x \mid y = 1) & = & N(\mu_1, \Sigma), \end{array}$$

where $N(\mu, \Sigma)$ indicates the multivariate normal distribution. From this we can derive

$$p(y = 1 \mid x) = p(x \mid y = 1) p(y = 1) / p(x). \tag{8}$$

For $j = 0, 1$, let $A_j = p(x \mid y = j)$. Then we can write

$$p(x) = p(x \mid y = 0)p(y = 0) + p(x \mid y = 1)p(y = 1) = A_0(1 - \phi) + A_1\phi.$$

Plug this expression for $p(x)$ back into (8) to get

$$p(y = 1 \mid x) = \frac{\phi A_1}{(1-\phi)A_0 + \phi A_1} = \frac{1}{1 + \frac{A_0}{A_1}\left(\frac{1-\phi}{\phi}\right)}. \tag{9}$$

At this point it will be useful to introduce the shorthand notation

$$\langle x, y \rangle := x^T \Sigma^{-1} y,$$

noting that this is linear in both terms just as the usual inner product is. It will also be convenient to define

$$B_j = \langle x - \mu_j, x - \mu_j \rangle = \langle x, x \rangle - 2\langle \mu_j, x \rangle + \langle \mu_j, \mu_j \rangle.$$

Our ultimate goal (7) can now be reduced to showing that the denominator of (9) is of the form $1 + \exp(a\langle b, x \rangle + c)$ for some constants a, b, and c.

Notice that

$$\frac{A_0}{A_1} = \frac{\exp(-1/2B_0)}{\exp(-1/2B_1)} = \exp\left(\frac{1}{2}(B_1 - B_0)\right) = \exp\left(\frac{1}{2}\big(\langle 2\mu_0 - 2\mu_1, x \rangle + C\big)\right),$$

where C is based on the terms of B_j that use μ_j but not x. The scalar factor of A_0/A_1 in (9) can also be absorbed into the constant c in the expression $\exp(a\langle b, x \rangle + c)$. This confirms that the denominator of (9) is indeed of the needed form, confirming the goal (7).

2.2 Naive Bayes

I've heard that, in practice, naive Bayes works well for spam classification. If its assumption is so blatantly false, then why does it work so well? I don't have an answer, but I wanted to mention that as an interesting question.

Also, independence of random variables is not just a pair-wise property. You could have three random variables x_1, x_2, and x_3 which are all pair-wise independent, but not independent as a group. As an example of this, here are three random variables defined as indicator functions over the domain $\{a, b, c, d\}$:

$$\begin{aligned} x_1(w) &= [w \in \{a, b\}] \\ x_2(w) &= [w \in \{b, d\}] \\ x_3(w) &= [w \in \{a, d\}] \end{aligned}$$

If we know the value of x_i for any i, then we have no extra information about the value of x_j for any $j \neq i$. However, if we know any two values x_i and x_j, then we've completely determined the value of x_k.

This is all leading up to the point that the naive Bayes assumption, stated carefully, is that the entire set of events $\{x_i \mid y\}$ is independent, as opposed to just pair-wise independent.

I'm curious about any theoretical justification for Laplace smoothing. As presented so far, it mainly seems like a hacky way to avoid a potential implementation issue.

3 On lecture notes 3

The notes in this section are based on lecture notes 3.

3.1 Support vector machines

At one point, Ng says that any constraint on w will work to avoid the problem that maximizing the functional margin simply requires making both w and b larger. However, I think many constraints make less sense than $||w|| = 1$. The only case that doesn't fit well with $||w|| = 1$ is the case that the best weights are actually $w = 0$, but this vector can only ever give a single output, so it's useless as long as we have more than one label, which is the only interesting case. On the other hand, a constraint like $|w_3| = 1$ would avoid the exploding (w, b) problem, but it may be the case that the best weight vector has $w_3 = 0$, which would fail to be found under such a constraint. In general, constraints tend to assume that some subset of values in (w, b) are nonzero, and anything more specific than $||w|| = 1$ would add an assumption that may end up missing the truly optimal weights.

3.1.1 max min $\le$ min max

Ng notes that for any function $f(x, y)$,

$$\max_x \min_y f(x,y) \;\le\; \min_y \max_x f(x,y).$$

Let's justify this. Let

$$\begin{aligned} x_0 &= \text{argmax}_x \min_y f(x,y), \quad \text{and} \\ y_0 &= \text{argmin}_y \max_x f(x,y). \end{aligned}$$

Then

$$\begin{aligned} \max_x \min_y f(x,y) = &\; \min_y f(x_0, y) \\ \le &\; f(x_0, y_0) \\ \le &\; \max_x f(x, y_0) = \min_y \max_x f(x,y), \end{aligned}$$

which completes the proof.

There are only two inequalities in that proof; if they were equalities, then the final result would itself be an equality as well. Specifically, if (x_0, y_0) is a saddle point in the sense that

$$f(x, y_0) \leq f(x_0, y_0) \leq f(x_0, y) \quad \forall x, y,$$

then

$$\max_x \min_y f(x, y) \;=\; \min_y \max_x f(x, y).$$

3.1.2 The name of kernels

Quick side question: Why are kernels called kernels?

I found this link which states that the word *kernel*, in mathematics, was first used in the context of a group homomorphism f to indicate the set $f^{-1}(0)$, the pre-image of the identity element. Interpreting the word *kernel* as meaning something like *core* or *seed*, this usage in group theory makes sense when thinking about the first isomorphism theorem.

In the context of SVMs, the name *kernel* comes from the idea of a positive-definite kernel, which originated mainly in work done by James Mercer, although as far as I can tell the actual name *positive-definite kernel* came from a 1904 paper of David Hilbert on Fredholm integral equations of the second kind.

Based on this math.stackexchange answer, it sounds like the group-theoretic and algebraic use cases — that is, the two listed above — of the word *kernel* are not formally linked. Rather, it seems that Ivar Fredholm used the French word *noyau* in a paper on integral equations that later became *kernel* in the aforementioned paper of Hilbert. *Noyau* roughly means *core*, and was perhaps used because it is inside an integral when a convolution is performed. The referenced answer states that the original Hilbert paper never justifies the term, so I suppose it's up to folks who understand that paper to make an educated guess as to why it was chosen; or by looking at the corresponding Fredholm paper.

3.1.3 The Gaussian kernel

Let's confirm that the Gaussian kernel is indeed a kernel. As a side note, this kernel is also referred to as the radial basis function kernel, or the RBF kernel.

I had some difficulty proving this myself, so I found this answer by Douglas Zare that I'll use as the basis for this proof.

The goal is to show that the function

$$k(x, y) = \exp\left(\frac{-||x-y||^2}{2\sigma^2}\right)$$

is a kernel.

The proof will proceed by providing a function ϕ_w such that the inner product $\langle \phi_x, \phi_y \rangle = k(x, y)$ for any vectors $x, y \in \mathbb{R}^n$. I'll be using an inner product defined on functions of $\mathbb{R}^n$ as in

$$\langle f, g \rangle = \int_{\mathbb{R}^n} f(z) g(z) dz.$$

Begin by letting $c_\sigma = \left(\frac{\sqrt{2}}{\sigma\sqrt{\pi}} \right)^{n/2}$. Define

$$\phi_w(z) = c_\sigma \exp\left(\frac{-||z - w||^2}{\sigma^2} \right).$$

Then

$$\langle \phi_x, \phi_y \rangle = \int_{\mathbb{R}^n} c_\sigma^2 \exp\left(\frac{-||z - x||^2}{\sigma^2} \right) \exp\left(\frac{-||z - y||^2}{\sigma^2} \right) dz.$$

Let

$$a = \frac{x + y}{2}, \quad b = \frac{x - y}{2}, \quad v = z - a,$$

so that

$$\begin{aligned} z - x &= z - a - b = v - b, \\ z - y &= z - a + b = v + b. \end{aligned}$$

Then

$$\begin{aligned} \langle \phi_x, \phi_y \rangle &= c_\sigma^2 \int \exp\left(\frac{-||v - b||^2 - ||v + b||^2}{\sigma^2} \right) dv \\ &= c_\sigma^2 \int \exp\left(\frac{-2||v||^2 - 2||b||^2}{\sigma^2} \right) dv. \end{aligned}$$

The last equation uses the facts that $||v - b||^2 = \langle v - b, v - b \rangle = \langle v, v \rangle - 2\langle v, b \rangle + \langle b, b \rangle$, and that, similarly, $||v + b||^2 = ||v||^2 + 2\langle v, b \rangle + ||b||^2$.

Continuing,

$$\langle \phi_x, \phi_y \rangle = c_\sigma^2 \exp\left(\frac{-2||b||^2}{\sigma^2} \right) \int \exp\left(\frac{-2||v||^2}{\sigma^2} \right) dv.$$

Make the substitution

$$\frac{\sqrt{2}v}{\sigma} = u, \quad dv = \left(\frac{\sigma}{\sqrt{2}}\right)^n du.$$

We get

$$\begin{aligned}
\langle \phi_x, \phi_y \rangle &= c_\sigma^2 \exp\left(\frac{-2||b||^2}{\sigma^2}\right)\left(\frac{\sigma}{\sqrt{2}}\right)^n \int \exp(-||u||^2) du \\
&= c_\sigma^2 \exp\left(\frac{-2||b||^2}{\sigma^2}\right)\left(\frac{\sigma\sqrt{\pi}}{\sqrt{2}}\right)^n \\
&= \exp\left(\frac{-2||x-y||^2}{4\sigma^2}\right) \\
&= \exp\left(\frac{-||x-y||^2}{2\sigma^2}\right) \\
&= k(x, y),
\end{aligned}$$

which completes the proof.

3.1.4 Sequential minimal optimization (SMO)

SMO can be used to solve the dual optimization problem for support vector machines. I wonder: Can an SMO approach be used to solve a general linear or quadratic programming problem?

The original SMO paper by John Platt specifically applied the algorithm to support vector machines. Based on a quick search, I don't see any applications of SMO to other fields or problems. I could be missing something, but my answer so far is that SMO either hasn't been tried or simply doesn't offer new value as a more general approach.

TODO any remaining notes from lecture notes 3

4 On lecture notes 4

The notes in this section are based on lecture notes 4.

4.1 The question of k in the case of finite $\mathcal{H}$

In the case of finite $\mathcal{H}$, I was confused for a moment about the need to include k in the following line of thinking:

$$\begin{aligned}\epsilon(\hat{h}) &\le \hat{\epsilon}(\hat{h}) + \gamma \\ &\le \hat{\epsilon}(h^*) + \gamma \\ &\le \epsilon(h^*) + 2\gamma.\end{aligned}$$

This argument is given in the context of

$$\gamma = \sqrt{\frac{1}{2m} \log \frac{2k}{\delta}},$$

where is where the value $k = |\mathcal{H}|$ comes in.

My confusion arose because the line of inequalities only involve two particular hypotheses, $\hat{h}$ and h^*, so that it seems unnecessary to build the value of γ on top of a limit for *all* possibly hypotheses in $\mathcal{H}$. However, I realized that the exact value of $\hat{h}$ is in fact dependent on the entire set $\mathcal{H}$, and so the same argument given with k replaced by 2 would no longer be sound.

4.2 The informal argument for infinite $\mathcal{H}$

In this part of the notes, Ng argues that we may informally consider a theoretically infinite hypothesis space with p parameters as something like a finite space with $\log(k) = O(p)$ since it can be represented with a single floating-point number per parameter. However, he also proposed the counterargument that we could have inefficiently used twice as many variables in a program to represent the same hypothesis space. I disagree with this counterargument on the basis that the actual number of hypothesis functions represented is identical, and we can count k based on this number. To put it slightly more formally, suppose we have two sets of functions,

$$H_1 = \{f_p : p \in P_1\} \quad \text{and} \quad H_2 = \{f_p : p \in P_2\},$$

where f_p represents a hypothesis function specified by parameters p. To fit Ng's example, P_1 could be a list of p 64-bit doubles while elements of P_2 could inefficiently hold $2p$ doubles. However, despite the sets P_1 and P_2 having different sizes, the sets H_1 and H_2 have the same size, thus helping to abstract away implementation details.