A quick(ish) reference for some concepts whose intuitions/simple methods of understanding sometimes escape me.

David Khachatrian

Ongoing.

Contents

1 Information Theory **3**
- 1.1 Entropy and Cross-Entropy . . . 3
- 1.2 KL divergence and mutual information. . . . 4
- 1.3 Entropy of variables vs. entropy of distributions . . . 5

2 Statistics **7**
- 2.1 Bayes' Theorem . . . 7
  - 2.1.1 Substitution to the rescue. . . . 8

3 Constrained Optimization Problems **10**
- 3.1 Equality constraints only: The Method of Lagrange Multipliers/The Lagrangian . . . 10
  - 3.1.1 Intuition with just one constraint . . . 10
  - 3.1.2 Doin' me some gradients . . . 11
  - 3.1.3 The Lagrangian: a packaged function . . . 12
  - 3.1.4 Extension to more than one constraint . . . 13
  - 3.1.5 Explaining the convenience, and a more generalizable intuition. . . . 14
- 3.2 Constraints with Inequalities: the Karush-Kuhn-Tucker (KKT) Conditions . 16
  - 3.2.1 Not nearly as scary as they make it out to be. . . . 16
  - 3.2.2 If it's not that bad, why did you talk so much? . . . 19

4 Machine learning methods. **20**
- 4.1 Boosting. . . . 20
  - 4.1.1 An analogy to Taylor expansions. . . . 20
  - 4.1.2 Meta-algorithm vs. algorithm . . . 21
  - 4.1.3 Where the analogy is left wanting. . . . 21
  - 4.1.4 Gradient boosting. . . . 22

5 Physics. **24**
- 5.1 The Boltzmann distribution . . . 24
  - 5.1.1 Stating our problem. . . . 24
  - 5.1.2 A lazy way out. . . . 25
  - 5.1.3 Counting (is hard). . . . 25
  - 5.1.4 Counting Is Hard II: Probability Strikes Back. . . . 26
  - 5.1.5 A fork in the road. . . . 29
  - 5.1.6 Planning our attack. . . . 29
  - 5.1.7 The continuous case: An easy battle. . . . 30
  - 5.1.8 The Boltzmann factor. . . . 31

5.1.9 Bolting beyond Boltzmann. 33

6 Drafts and WIPs **35**
6.1 Integral transforms. 35
6.1.1 Short refresher on vector spaces. 35
6.1.2 Functions, function spaces. 37
6.1.3 An inner product for our function space. 38
6.1.4 Beyond the countable, and picking a basis. 39
6.1.5 The integral transform: a concatenation of inner products. 40
6.1.6 The Laplace Transform: a differential equations-friendly integral transform. 41
6.2 The Fourier and Laplace Transforms. 41
6.2.1 Revisiting the Taylor series. 41
6.2.2 Discrete function spaces. 42
6.2.3 Equipping our vector space. 43
6.2.4 Re-revisiting the Taylor series. 43
6.2.5 But first, let's talk eigenfunctions. 44
6.2.6 The Fourier transform and wiggles over time. 44
6.2.7 The Laplace transform as a patch for the Fourier transform. 45
6.3 Generating functions: dealing with uneven probabilities. 45
6.4 Central Limit Theorem 45

Index **46**

1. Information Theory

1.1 Entropy and Cross-Entropy

There are a number of ways to think about/get to the Shannon entropy (H). One simple way is to say what we want it to mean qualitatively and impose some desired characteristics to determine its mathematical formulation. In this case, it may make more sense to start with cross-entropy first.

Say you're sending me messages from an alphabet of symbols A. Some are more likely than others, and we'll call the probability distribution associated with the actual generator of symbols p. I'm over here on the other side thinking like I'm a pair of smartypants that has figured out how likely you are to send each symbol to me. We'll call my expected distribution (which may or may not be the correct distribution) q.

Now we want to measure how surprised I am by any given message you send me -- let's call it τ.[1] We would imagine the following properties would be useful for τ to have:

1. The more surprised I am, the larger τ gets (otherwise, it wouldn't exactly be doing a good job measuring my surprise)
2. If symbols are independently generated, we can construct the total surprise of my message by adding the total surprise of each symbol in the message, i.e., $\tau_M = \sum_{i=1}^{m} \tau_{M[i]}$ where $M \in A^m$ is a string of symbols, m is the cardinality of M, and $M[i]$ denotes the i'th element of M.

Using these two criteria, a reasonable mapping is

$$\tau_{a\sim q} = \log\left(\frac{1}{q(a)}\right), a \in A$$

where $q(a)$ is the probability I think you'll send the symbol a.

I mean, that's great and all, but that works for individual instances of strings or symbols. How much should I *expect* to be surprised by any given symbol? Well, that'd depend on how often you *actually* send me a symbol, alongside how often I expect to receive that particular symbol.

Hmm, this smells of expectation! And indeed, that's all we do -- take the expectation over A (using the *actual* probability of each symbol occurring, i.e., using p) of my surprise per symbol:

$$E_{a\sim p}[\tau_{a\sim q}] = \sum_{a\in A} p_a \log\left(\frac{1}{q(a)}\right) = H(p,q)$$

[1] Rhymes with "wow". Alas, not standard notation for a measure of surprise.

We denote this metric $H(p, q)$ (where p and q are the actual and presumed distributions, respectively) and call it the **cross-entropy** (because that sounds pretty cyberpunk to me. I like to think they throw in "cross" because it measures the surprise caused by "crossing" the distributions p and q together.)

Now, we'd imagine that I'd be the least surprised if my presumed distribution of symbols were in fact the actual distribution, i.e. when $p = q$. In fact, this is true!

$$H(p, p) = H(p) = \sum_{a \in A} p_a \log\left(\frac{1}{p_a}\right)$$

is called the **entropy** (or **Shannon entropy**) of the probability distribution and written H.[2] Usually, the *log*s written above are base-2. This permits a way of thinking of the value of the Shannon entropy: if I'm only allowed to ask the same series of questions to you to figure out which symbol you want to send me and I know the actual probability distribution p of symbols, how many questions should I expect to ask (i.e. mean/average) before I figure out the answer? The cross-entropy is the same thing, expect I don't necessarily know the actual probability distribution p of symbols, I just think I do (and I think it's q) and base my series of questions based on q.

1.2 KL divergence and mutual information.

Now, hopefully that makes it clear that $p \neq q \implies H(p, q) > H(p)$ — if I don't know the actual distribution, I'm not going to be able to answer the most optimal series of questions. This suggestions to us a notion of "distance" (i.e. a metric) between the probability distributions p and q:

$$KL(p \mid\mid q) = H(p, q) - H(p)$$

This metric is called the **Kullback–Leibler divergence** (because names) or the **KL divergence** (because initialisms), or seemingly most rarely but probably most clearly the **relative entropy**. Out loud, you'd say $KL(p \mid\mid q)$ is "the [blah] of p with respect to q". The closer the KL divergence is to 0, the closer q is to being p, and $KL(p \mid\mid q) = 0 \implies p = q$ at every point in the domain of q (which is the same as the domain of p). (From the above formula, it should be clear that the KL divergence is not symmetric. The first argument is the "correct" distribution and we're measuring how suboptimal the second argument/distribution is at replicating the first one.)

This makes describing the **mutual information between two random variables X and Y** in terms of a KL divergence fairly intuitive. Just running off the name, if X and Y were independent, you'd expect no mutual information between them — observing one variable wouldn't tell you anything about the other variable, you don't gain a lot of information about one from the other. In such a case, we know something about their probability distributions, namely that they're independent, i.e., $P(X, Y) = P(X)P(Y)$, which implies

[2] You know, it'd arguably make more sense for the ***S**hannon* entropy to be denoted S, which would also be a happy notational coincidence with the symbol used for entropy in most other fields. Instead, we have a notational collision with *enthalpy*. Ah well, such is the arbitrariness of a symbol's meaning. (I suppose it's fitting.)

$KL\left(P(X,Y) \mid\mid P(X)P(Y)\right) = 0$. And in fact, one way of writing the mutual information between random variables X and Y is exactly

$$I(X;Y) = KL\left(P(X,Y) \mid\mid P(X)P(Y)\right)$$

which I think is pretty neat.

- DK (4/24/18)

1.3 Entropy of variables vs. entropy of distributions

Now, there is another way of approaching mutual information by defining a conditional entropy between two random variables (itself requiring a defintion of joint entropy in order to put the equation in a "plug n' chug" form). This raises some questions, the first and most pressing of which being "*Variable?* We've been talking about distributions this whole time!"[3] , and the answer to which is "Yes, variable." Since we've been focusing on distributions this whole time, the switch to a random variable X is actually fairly straightforward – we use the distribution from which X is drawn.

More explicitly, say X is drawn from a probability distribution p. Then the **entropy of** X is

$$H(X) = \sum_{x \in X} p(x) \log\left(\frac{1}{p(x)}\right)$$

This may be reminiscent of our $\tau_{a \sim q}$ from earlier, except that now we're using the actual distribution of X (which is p), so it'd be $\tau_{x \sim p}$.

We can sort of think of it like this: the information entropy only really makes sense for probability distributions. So, if we want to figure out how hard it is to encode a random variable X, we kind of "pull out" the probability distribution that X is drawn from and use that in our formula.

Now, unfortunately, the notation gets muddy when we allow ourselves to shove in these random variables as arguments. For example, what does $H(X,Y)$ (X and Y being random variables) mean? H is provided two arguments – is it the cross-entropy between the underlying distributions of X and Y? Nope, it's the **joint entropy** of X and Y, i.e.,

$$H(X,Y) = H(p)$$

where p is the joint distribution of the random variables X and Y. Our main defense against confusing the two formulae is that random variables are (normally – hopefully!) denoted by capital letters while distributions are usually denoted by lowercase letters.

You may wonder "what's the point?" Well, this begins to allow information theory analogues for intuitions on random variables gleaned from statistics. The main missing piece

[3] More an outcry than anything, but still appropriate.

at this point is the **conditional entropy** of a random variable Y with respect to X. We'd expect that we could relate the joint entropy and conditional entropy of random variables with one another, like how we can do so with the joint and conditional probability distributions of the variables, especially since we've defined the entropy of a random variable in terms of its probability distribution. And since:

1. we applied a logarithm to the probability distribution (and took the expected value) when defining the entropy of a random variable; and
2. the relationship between conditional and joint probability depends on multiplication:

$$P(Y|X) \times P(X) = P(X,Y)$$

we'd want the relationship between conditional and joint entropy to hold via addition:

$$H(Y|X) + H(X) = H(X,Y)$$

In fact, that's exactly the case! You can derive the formula for conditional entropy based on the above relationship. And just to circle back to information gain, we'd expect that we might gain some information about a random variable X when we observe the random variable Y, depending on how the joint distribution P(X,Y) compares with P(X). Earlier, we defined the mutual information in terms of a KL divergence:

$$I(X;Y) = KL\left(P(X,Y) \;||\; P(X)P(Y)\right)$$

but we can also capture the quantity of "how much does knowing Y (on average) help us figure out X (and vice-versa)?" using the language of entropies of random variables:

$$I(X;Y) = H(X) - H(X|Y)$$

This is the expected amount of information gained by knowing the state of the random variable Y. Specific values of Y may give more information about X – may lead to a much tighter conditional distribution – than others. Then, you can look at it for specific cases, i.e., the **information gain** about X from observing a specific state y for Y, IG(X,Y = y), which is related to the mutual information via $E_{y\sim Y}\left[IG(X,Y)\right] = I(X;Y)$). So if we observe that Y took on the value y, the information gain for X would be

$$IG(X, Y = y) = KL\left(P(X,Y) \;||\; P(X|Y = y)\right)$$

Writing the above in terms of information entropy would be

$$IG(X, Y = y) = H(X) - H(X|Y = y)$$

where $H(X|Y = y)$ is an expectation over the conditional distribution $P(X|Y = y)$:

$$H(X|Y = y) = \sum_{x \in X} p(x|y) \times \log\left(\frac{1}{p(x|y)}\right)$$

The main takeaway is that *defining the entropy of a random variable as it is above allows for the migrations of intuitions about random variables from statistics and probability.* A worthy cause![4]

- DK, 5/14/18

[4]The notational confusion is still unfortunate though.

2. Statistics

2.1 Bayes' Theorem

I can never seem to remember Bayes' Theorem directly, as they write it out in textbooks. It makes so much more sense to me to think about it from the relationships between conditional and joint probabilites/distributions, and one of the common tricks to make Bayes' Theorem useful in practice also comes to me far more easily when explicitly thinking about events as being sampled from a *sample space* of possibilities/outcomes.

Consider two possible events A and B. Let's keep in mind that A is just one possible outcome out of a set of possibilities, as is B; we'll say α is the set of possibilities from which A was drawn and β is the set of possibilities from which B was drawn, i.e. $A \sim \alpha$ and $B \sim \beta$ (this will be good to remember later). Now:

$$\text{Pr[A and B both occur]} := P(A, B)$$

Assuming individual events happen separately, there are two ways for both A and B to occur:

1. A happens first, then B happens.
2. B happens first, then A happens.

("Duh", I know.)
Keeping in mind that the first event might affect the probability of the second event occurring (i.e. remembering that conditional probabilities exist), we can write:

$$P(A, B) = P(A) \times P(B \mid A) = P(B) \times P(A \mid B)$$

And then it's simple to write out Bayes' Theorem as it's often written (we'll write it perhaps a bit more evocatively):

$$P(B) \times P(A \mid B) = P(A) \times P(B \mid A)$$

$$\begin{aligned} P(A \mid B) &= \frac{P(A) \times P(B \mid A)}{P(B)} \\ &= P(A) \times \frac{P(B \mid A)}{P(B)} \end{aligned}$$

Using the Bayesian interpretation: At first we thought the probability that A occurs is $P(A)$. After we saw that B happened, we re-evaluate the probability that A occurs with a scaling factor $\frac{P(B|A)}{P(B)}$, which answers the following question: considering I've seen B

occur, how much *more* likely did I observe B due to A also being the case (the numerator $P(B \mid A)$), versus my having observed B just because of how common/rare it is (the denominator, $P(B)$)? To see why this scaling factor makes sense, let's consider some edge cases:

- *A and B are uncorrelated*: Then observing B is irrelevant when it comes to predicting A. So our scaling factor should be 1. And in fact, the lack of correlation implies $P(B \mid A) = P(B) \implies \frac{P(B|A)}{P(B)} = 1$.
- *A precludes B*: Then via contrapositivity, if we saw B, A must not be the case. So our scaling factor should be 0. And since A precludes B, $P(B \mid A) = 0 \implies \frac{P(B|A)}{P(B)} = 0$.
- *B implies A (and no other outcome from α)*: Then the total probability must become 1, and the scaling factor must come out to be $\frac{1}{P(A)}$. We'll come back to this in a bit.

Also worth knowing the fancy terminology:

1. the ***a priori* probability** or just the **prior** is what we thought would be the probability that a random variable takes on a certain value before we observed anything. So the *a priori* probability (or just prior) for the event A would be $P(A)$. If we consider A to be a random variable instead of an event, we're guessing the distribution of A and so $P(A)$ would be an ***a priori* distribution** (or again, just the prior).
2. the ***a posteriori* probability** or just the **posterior** is what we think the probability that a random variable takes on a certain value is after observing something. In this case, the *a posteriori* probability (or just posterior) of the event A after observing B is $P(A \mid B)$. If we consider A to be a random variable instead of an event, we're guessing the distribution of A after observing a random variable/event B and so $P(A \mid B)$ would be an ***a posteriori* distribution**, (or again, just the posterior).

2.1.1 Substitution to the rescue.

Now, in cases of inference via supervised learning, we have some data on the probability of one of the observable variables — let's say we observe variables $A \sim \alpha$ — and the labels/targets for the observed variables (let's say B β). Then we can approximate $P(B \mid A)$ and $P(A)$ via empirical counts — and would want to fit a continuous function, e.g. a Gaussian, to $P(A)$ to handle out-of-sample feature combinations. So we've already guessed some prior $P(A)$, and we're trying to improve it by calculating the posterior $P(A \mid B)$. But what if we don't know $P(B)$? Do we need to also guess a function for $P(B)$? Well, that would involve another outside assumption which may or may not be true, and usually we want to make as few assumptions as possible and "let the data speak for itself". So then is our guessing and data collection all for naught!? Thankfully, not so! The answer lies right under our noses -- or in this case, in our previous calculations.

Consider $P(A, B)$ again. What would we get if we added $P(A, B)$ over all possible values of A? (Remember we said that $A \sim \alpha$, so A could have been some other event within the set α.) That's basically just saying that we don't care what value A is, so we end up with $P(B)$![1] And conveniently, we'd already have a way to estimate these values:

[1] '!' used to denote excitement, not factorialization.

$$\begin{aligned} P(B) &= \sum_{A \in \alpha} P(A, B) \\ &= \sum_{A \in \alpha} P(A) \times P(B \mid A) \end{aligned}$$

Our summand is the same as the values we've estimated either by guessing ($P(A)$) or from our data ($P(B \mid A)$)! With that, we can rewrite our earlier equation as

$$P(A \mid B) = \frac{P(A) \times P(B \mid A)}{\sum_{A' \in \alpha} P(A') \times P(B \mid A')}$$

With that, we can crunch the numbers and perform Bayesian inference like a champ or have a machine do it for us like a prudent delegator.

Returning to our discussion of our scaling factor, we needed to show that if we're trying to predict $P(A \mid B)$ via

$$P(A \mid B) = P(A) \times \frac{P(B \mid A)}{P(B)}$$

and B implies A (and nothing else from α, then the right-hand side should equal 1.

Expanding P(B), we get

$$P(A \mid B) = \frac{P(A) \times P(B \mid A)}{\sum_{A' \in \alpha} P(A') \times P(B \mid A')}$$

Via our implication, we have that $P(B \mid A') = 0 \forall A' \neq A$. So the denominator simplifies to just $P(A) \times P(B \mid A)$, which is exactly the numerator! So then $(B \implies A) \implies P(A \mid B) = 1$ (which implies $\frac{P(B|A)}{P(B)} = \frac{1}{P(A)}$), as expected.

- DK (4/24/18)

3. Constrained Optimization Problems

Motivated to do this when I was reading this paper and realized I forgot how we get to/use the KKT conditions (which is implied in Eq. 2 in the paper).

3.1 Equality constraints only: The Method of Lagrange Multipliers/The Langrangian

NB: Khan Academy's videos on the subject really make clear one potential intuition for one constraint.

3.1.1 Intuition with just one constraint

Say you want to optimize (maximize or minimize) a smooth function $f(\vec{x})$ subject to the constraint $g(\vec{x}) = c$ (let's say $\vec{x} \in R^n$ -- and we may just write x instead of $\vec{x}$). (It's worth remembering that the constraints themselves can also be expressed as functions -- they just happen to be set to specific constants.) So our goal is to find the best point(s), $\vec{x}^*$. For the purposes of explanation, we'll say our goal is maximization, but it all applies for minimization too.

A way to build up the intuition is to consider the contour lines of $f(\vec{x}) = m$ for particular values of m. Our goal then is to maximize m while having $\vec{x}$ satisfy $g(\vec{x}) = c$. If it didn't fulfill this second requirement, then we'd just be ignoring the constraint and solving an unconstrained optimization problem -- in which case, we'd just set the gradient equal to zero and solve (say, what a useful thought -- let's put that in our back pocket for later...).

Let's call the constraint contour line (which we aren't allowed to change and is set to some constant c) G and the function contour line (which we can change by varying m) $F(m)$. If we think about it for a bit, we'll see that *solving our problem is analogous to choosing the largest value of m so that $F(m)$ "touches" G in the fewest number of places while still actually "touching" G.* Consider the alternatives:

1. *$F(m)$ does not touch G at all:* Well, that means that when we look at the set of points that comprise the contour line $f(x, y) = m$ (i.e., the **level set** of f corresponding to a value of m), *none* of those points lie on $g(\vec{x}) = c$. So we ignored the constraint again -- oops.
2. *$F(m)$ touches G too many times:* This implies that $F(m)$ cuts across G (if it didn't, then where did the "extra" touches come from?) And in that case, why not increase m a bit more? We're assuming f is relatively well-behaved, so if you nudge m up a bit to m^+, the contour line $F(m^+)$ will be fairly close to $F(m)$ -- and so, still cross G somewhere.

(By the way, I've been using the tortured phrase "minimal number of times but not zero like c'mon don't be cheeky" because there can be more than one location on the constraint curve that have the same maximal value for f. Consider maximizing $f(x, y) = x^2y^2$ subject to $x^2 + y^2 = 1$. The symmetry leads to four points that satisfy the criteria -- bump up f's output higher and you're off the circle, bump it down and you're cutting across the circle (and also not maximizing f).)

Now, if $F(m)$ and G just barely touch but do not cross, then their instantaneous "slopes" at their touching point must be in the same orientation and the contour lines are moving in the exact same "direction" -- if they weren't, then the touching point is a crossing point. So how do we capture this notion?

3.1.2 Doin' me some gradients

The (nonzero) gradient of a function is always perpendicular to its contour lines. (Seems like a deep statement, but with a bit of thought, you can see that it just comes from the definition/intuition of contour lines (on which the value is constant) and gradients (which points toward the direction that increases the function output the largest -- with no portion of the "step" being wasted on movement that would keep the value constant).)

So, we can convert out "contour lines $F(m)$ and G are in the same direction" directly to "the gradients of f and g are in the same direction", i.e.

$$\nabla f = \lambda \nabla g$$

where ∇ is the usual gradient operator and $\lambda \neq 0$, the scalar proportionality constant (it's the *direction* that matters, not the magnitude), is called the **Lagrange multiplier** (dude got a lot of things named after him).

It's actually worth looking at this a bit more. We originally framed our goal by trying to get $F(m)$ and G to touch as little as possible. But we can also frame it in terms of ∇f and its relation to the constraint functions:

$$\nabla f = \lambda \nabla g \quad \Longleftarrow \quad \nabla f \text{ is perpendicular to the contour } G$$

It makes sense that the right-hand side would be the case -- if ∇f *did* have some part of it along G, then we're wasting that part! Why wouldn't we go along G a bit more? We'd still be meeting our constraint, and since we stepped partially along the direction of steepest ascent, we'd have increased f while we were at it! It'll be worth remembering this observation a little down the line, so keep it in your pocket for later (or some other handy container, if you are doomed to the fate of clothing without functional storage capabilities).

Anyway, that's great and all, but we only have n equations (each of the n elements of the vector formulation above) and now we have $(n + 1)$ unknowns (all the coordinates for x, plus λ). Well, there's a reason the above relation used a(n) $\Longleftarrow$. On the left-hand side, we've only captured that the gradients have to be in the same direction -- we haven't added our constraint! (The right-hand side encapsulates both, since we have G as the contour corresponding to the specific constraint $g(x) = c$.) So our full set of $(n + 1)$ equations with $(n + 1)$ unknowns is

$$\nabla f = \lambda \nabla g$$
$$g(x) = c$$

Now go to town! Worth remembering that all of this provides *necessary* but **not** *sufficient* conditions for optimality. Sufficient conditions would involve, for example, proving the that Hessian matrix of f is negative semidefinite when trying to maximize f (analogous to the second-derivative test in the single-dimensional case), and even if you manage that you're only guaranteed local maximality. Sounds like a lot of qualifications, but we've actually narrowed the search space a great deal with these conditions, so it's not as terrible as it sounds.

3.1.3 The Lagrangian: a packaged function

The above system of equations works great for people, but people have also spent so much time and energy to make computers solve math problems for us! Most of these programs are particular good and finding the zeros of a function (without any fancy constraints). So how could we repackage the (n+1) equations above into one function we can find into a zero-finder?

Well, let's rewrite the above equations first:

$$\nabla f - \lambda \nabla g = 0$$
$$g(x) - c = 0$$

Alright, now what? Well, if we were to write something like

$$L(x) = f(x) - g(x)$$

we'd be *almost* there, because if we took the gradient of L and set it equal to zero, we get the "direction" constraint back. But at the moment, we're making it so that the magnitudes of the two gradients have to be the same too (which doesn't have to be true) *and* we forgot to incorporate our constraint again!

Well, why don't we reintroduce λ as a variable in such a way that it handles the proportionality problem *and* have it so that $L_\lambda = g(x) - c$ (so that we reincorporate our constraint into the function)? Might sound tricky, but in fact we can modify L to satisfy these requirements fairly simply:

$$\mathcal{L}(x, \lambda) = f(x) - \lambda \left(g(x) - c\right)$$

Now that it's achieved its final form (thankfully didn't take ten episodes of powering up), we change L to $\mathcal{L}$ and call it the **Lagrangian** of $f(x)$ (because dude needs more things named after him -- and you know, he *did* revolutionize the study of classical mechanics with this formulation).

Worth noting is that if we define the Lagrangian as

$$\mathcal{L}^+(x, \lambda^+) = f(x) + \lambda^+ \left(g(x) - c\right)$$

we still get the same answer to our optimization problem -- the only difference is that compared with the λ we get from the $\mathcal{L}$ formulation, $\lambda^+ = -\lambda$.

A neat consequence of the formulation of $\mathcal{L}$ is that we can consider λ as a measure of how much we could improve $f(x)$ by incrementing the value of c (which we've been considering a constant) by a differential amount. While it may seem to be "clear" just by taking $\mathcal{L}_c = \lambda$, it's a bit more subtle than that, since $\mathcal{L}(x, \lambda; c)$ was formulated with c as a constant. The proof for this observation involves:

- forming a new function, $\mathcal{L}^*(x^*(c), \lambda^*(c), c) = \mathcal{L}^*(c)$, a single-variable function that parameterizes the input coordinates of the answer(s) to the optimization problem (and also the Lagrange multiplier) with respect to c;
- doing the multivariable chain rule;
- thinking a bit to notice that a lot of things equal zero to get that $\frac{d\mathcal{L}^*}{dc} = \lambda$;
- and, having remembered that we're interested specifically about the points on $\mathcal{L}$ that optimize f (which is exactly what $\mathcal{L}^*(c)$ captures), realizing that the above result implies that first statement of the paragraph before this bulleted list.

What a mouthful.

3.1.4 Extension to more than one constraint

Would be kind of a shame if we did all of this just to solve problems with just one constraint. But thankfully, the extension is fairly simple to describe!

Say we want to optimize f subject to k constraints $g_1 = c_1, g_2 = c_2, \cdots, g_k = c_k$. Now, in all but the most trivial of cases, it would be impossible to have the gradients of all of these different functions in the same direction. Intuitively (-ish, and assuming you feel comfortable-ish with concepts in linear algebra), if they can't all be in the same direction, you'd think that the "next best thing" would be that the gradient of f is in the same direction as some linear combination of the gradients of $g_1, g_2, \cdots, g_k$, i.e. that

$$\nabla f = \sum_{i=1}^{k} \lambda_i \nabla g_i$$

That statement above is in fact a condition that is met in the answer to our optimization problem! (How convenient.)

Now we have n equations but $n + k$ unknowns. We once again fix that by actually incorporating our constraints:

$$\nabla f = \sum_{i=1}^{k} \lambda_i \nabla g_i$$
$$g_1 = c_1$$
$$\cdots$$
$$g_k = c_k$$

We can once again package everything together as a Lagrangian by having that $\mathcal{L}_{\lambda_i} = g_i - c_i$:

$$\begin{aligned}\mathcal{L}(x, \lambda_1, \cdots, \lambda_k) &= f - (\lambda_1(g_1 - c_1) + \cdots + \lambda_k(g_k - c_k)) \\ &= f - \sum_{i=1}^{k} \lambda_i(g_i - c_i)\end{aligned}$$

And Bob's your uncle.

3.1.5 Explaining the convenience, and a more generalizable intuition.

Earlier we just kind of accepted the convenience of our guess, but it's worth figuring out why it works. Remember that observation you kept in your pocket (or other handy container)?

$$\nabla f = \lambda \nabla g \quad \Longleftarrow \quad \nabla f \text{ is perpendicular to the contour } G$$

If ∇f had any part of it along G, then we could step along G and further increase f. This sounds extensible to more than one constraint! And in fact, the "convenient" result captures this for the contour line created by the intersection of all the constraints:

$$\nabla f = \sum_{i=1}^{k} \lambda_i \nabla g_i \iff \nabla f \text{is perpendicular to the intersection of all constraint contours } \bigcap_i G_i$$

Here, $\bigcap_i G_i$ serves as the one contour line on which we can move along that satisfies all the constraints. (Presumably such continuous arcs exist -- otherwise, there are only discontinuities (i.e. discrete points), and so each point in the set would have to be checked individually.)

Now, since we're dealing with the same construct (a contour along which we can move that satisfies the constraints), the same reasoning applies pretty much verbatim – ***if*** ∇f ***weren't perfectly perpendicular to*** $\bigcap_i G_i$***, we'd be wasting the component of the gradient going along*** $\bigcap_i G_i$***,*** $\nabla f^{\parallel}$***. So we'd just step along the contour and improve our result.***[1] So that explains why the right-hand side makes sense. But how does that imply the left-hand side? Specifically, how do we get the linear combination part, $\sum_{i=1}^{k} \lambda_i \nabla g_i$?

Well, since we're all on contour lines at the same time, $\bigcap_i G_i$ is necessarily perpendicular to all the gradients ∇g_i. Then any linear combination $\sum_{i=1}^{k} \lambda_i \nabla g_i$ is perpendicular to $\bigcap_i G_i$. In fact, the gradients form a *basis* for the space perpendicular to $\bigcap_i G_i$, because of the symmetric property of the perpendicularity relation. More curtly, call the span of the gradients S. By construction, $\bigcap_i G_i \perp S$, and by symmetry, $S \perp \bigcap_i G_i$. We want ∇f to be perpendicular to $\bigcap_i G_i$ (as we've said before). Well then, that means $\nabla f \in S$, which implies that it ∇f can be written as a linear combination of S's basis vectors, i.e.,

[1] Excessive bolding, italicizing, and a gratuitous footnote used to emphasize the fact that this is the mathematically rigorous "intuition" to have/remember.

$\nabla f = \sum_{i=1}^{k} \lambda_i \nabla g_i$. Bam! Stick that beautiful $Q.E.D.$ square in the corner, we are done! ~~Would do it myself were I writing this in *LaTeX* and not Markdown.~~ Well, since we've migrated to LaTeX, I think we owe ourselves a box!

■

Totally worth it.

- DK, 4/28/18

3.2 Constraints with Inequalities: the Karush-Kuhn-Tucker (KKT) Conditions

3.2.1 Not nearly as scary as they make it out to be.

For all the pomp and circumstance around this, with the caravan of names in the name itself and the esoteric terms used in its description like "complementary slackness" and "dual feasibility", the Karush-Kuhn-Trucker (KKT) conditions aren't nearly as hard to follow as one would expect if the method of Lagrange multipliers for multiple constraints makes sense/is comfortable.

First, we pose the optimization problem in "standard form" (which mainly just saves us from lugging around extra constants like we did with c in the Lagrange multipliers explanation):

Optimize $f(x)$ subject to $g_i(x) \leq 0$, $h_j(x) = 0$, with $i \in \{1, \cdots, k\}$ and $j \in \{1, \cdots, l\}$ (so k inequality constraints and l equality constraints). Below, we'll assume "optimize" = "maximize". We'll point out where changes will occur if you're minimizing instead.

Primal feasibility

This time, before we do anything else, we're going to stick down the original constraints so we don't forget they exist:

$$g_i(x) \leq 0 \; \forall \; i, \; h_j(x) = 0 \; \forall \; j$$

This is called the **primal feasibility** condition because it's a condition for the feasibility of the original, main, "primal" problem.

The dual formulation

We refer to the original problem as the "primal" problem to contrast it with the "dual" problem. That sounds all fancy, but we've made a dual problem before when we formed the Lagrangian for our equality-constraints-only version. That is, the **dual problem** is simply a reframed but equivalent form of the primal problem. We did it before by solving a function that had all our constraints wrapped in one clean package (the Lagrangian form of the problem). And hey, that was both a neat *and* a useful idea, and those don't come around all that often, so let's use it until it goes out of style.

Worth noting is that we want to make our dual problem mirror the primal problem exactly (in *optimal value* as well as optimal location), i.e. form a strong duality, i.e. have no duality gap. The following provide *necessary* conditions, but not *sufficient* conditions for a strong duality.

So what would be the dual problem? We can do the same thing we did for the Lagrangian -- make a new function with some extra variables whose partial derivatives yield the constraints of our problem. Let's try it:
We define a function

$$L(x, \mu_1, \cdots, \mu_k, \lambda_1, \cdots, \lambda_l) = f(x) + \sum_i \mu_i g_i(x) + \sum_j \lambda_j h_j(x)$$

Maximize L (with no external constraints; i.e., find the locations where $\nabla L = \vec{0}$). (We'll come back to minimization later.)

Alright, well that's certainly something. Now we can recover the constraints by noting that $L_{\mu_i} = g_i$ and $L_{\lambda_j} = h_j$. But there are some things that are still funky with the inequality constraints here, so let's work on those.

(By the way, there are some signs we'd have to flip if we were doing a minimization instead. We'll discuss that later on.)

Dual feasibility

For one thing, our dual problem won't mimic our primal problem of optimizing f at all if we let any μ_i be less than zero. If we did, then we'd easily "win" the optimization game by choosing some x such that $g_i(x) < 0$ (which is still satisfies our primal feasibility conditions), then setting the corresponding μ_i to arbitrariliy large negative numbers -- who cares about $f(x)$ anyway!? Oh wait, we do. So we should probably make sure that we can't break our problem:

$$\mu_i \geq 0$$

This is called the **dual feasibility** condition because, well, otherwise our dual problem isn't all that useful.

Complementary slackness and stationarity

For one thing, we can notice that there are two possibilities for the value g_i takes at an optimal point:

1. $g_i(x^*) < 0$: That means the inequality constraint is not actually stopping the function from getting to a "better" location where $g_i > 0$ (our optimal-point finder didn't hit a wall – there's an open neighborhood around x^* that still satisfies the constraint), so the constraint isn't being restrictive at all. So we don't have to worry about it!

2. $g_i(x^*) = 0$: The inequality constraint *is* potentially stopping the function from reaching a more optimal point, so the constraint is actually affecting the outcome. So we should be sure to be othorgonal to the contour in our final answer.

These observations inform the following two constraints, **complementary slackness** and **stationarity**:

$$\begin{aligned} \mu_i g_i = 0 \; \forall \; i \qquad & \text{complementary slackness} \\ \nabla f = \sum_i \mu_i \nabla g_i + \sum_j \lambda_j \nabla h_j \qquad & \text{stationarity} \end{aligned}$$

Let's once again consider our two cases:

1. $g_i(x^*) < 0$: This constraint is inactive and so is not part of the contour lines we use to find the set of points on which f must lie. Recall that we form this set by evaluating the intersection of all the active constraints C; that the span of the gradients of the active constraint functions form a space S orthogonal to C; and that since ∇f must be orthogonal to C in order to be a potential extremum, $\nabla f \in S$ and can be written as a linear combination of the gradients of the active constraints. (Boy, another mouthful.) Well, g_i is not an active constraint, and so ∇g_i is not part of the basis for S. So we want its contribution in the stationary condition to be $\vec{0}$. We can do this by setting the corresponding $\mu_i := 0$. Not-so-coincidentally, this necessary consequence leads to compliance with the complementary slackness condition as well.

2. $g_i(x^*) = 0$: The complementary slackness condition is met no matter what value μ_i takes. And that's perfect: the constraint is active, so ∇g_i is part of the basis for S. So we *need* μ_i to be nonzero so that we can describe any vector in S (of which ∇f is an element, as we talked about when we "explained the convenience" for the multiple-equality-constraint formulation earlier).

So the two conditions are tied to one another in this interesting way that makes it a more-or-less direct extension of the multiple-equality-constraint formulation!

Finishing our duel with duals.

...And with that, we've formed our dual problem with only equalities! We'll copy them here to show we have enough equations for our unknowns:

$$\begin{aligned} h_j(x) &= 0 && \text{primal feasibility} \\ \mu_i g_i &= 0 \;\forall\; i && \text{complementary slackness} \\ \nabla f &= \sum_i \mu_i \nabla g_i + \sum_j \lambda_j \nabla h_j && \text{stationarity} \end{aligned}$$

That's $k + l + n$ equations for $k + l + n$ unknowns! Now stick the system of equations into a (probably numerical) zero-finder and you're done!

On minimizing f

Let's not forget that we had done all this assuming we were *maximizing* L (and thus f). If we were *minimizing* f, we would be trying to minimize L and we could change the above equations in one of the following ways:

1. Replace L with $L(x, \mu_1, \cdots, \mu_k, \lambda_1, \cdots, \lambda_l) = f(x) - \sum_i \mu_i g_i(x) - \sum_j \lambda_j h_j(x)$ (note the minus signs). Replace the stationarity condition with $-\nabla f = \sum_i \mu_i \nabla g_i + \sum_j \lambda_j \nabla h_j$ (again, note the minus sign).

2. Keep L the same. Replace the dual feasibility condition with $\mu_i \leq 0$

The sign changes just ensure that our dual problem is still well-formed for the minimization problem (if you don't flip the sign somewhere, then we can just make arbitrarily "good" values by playing with μ_i again; and if you flip the sign in the formulation of L, you have to make sure you update the gradient expression accordingly (Option 1)).

3.2.2 If it's not that bad, why did you talk so much?

...OK, so maybe this was a lot more to explain than I had given credit at first. But it's really not that bad! The main intuition is that either:

1. the inequality constraint is lax and doesn't actually do any "constraining"; or
2. the inequality constraint is actually stopping us, in which case it just becomes another equality constraint!

All the blah-blah-blah is to make sure we dotted our i's and crossed our t's. (And boy, did we...)

- DK, 4/30/18

4. Machine learning methods.

4.1 Boosting.

There's a lot of buzz about the winning models of many Kaggle competitions use gradient-boosted trees. Considering its apparent effectiveness, it'd be worth understanding what "gradient-boosted trees" are and how they work. But before we jump into *that*, we should probably figure out what "boosting" means in this context.

4.1.1 An analogy to Taylor expansions.

Boosting is a meta-algorithm involving the ensembling of many "weak" learners to form arbitrarily "strong" learners. A "weak" learner is a classifier/regressor (which we'll just call a *predictor*) that can only be weakly correlated with the true underlying classication/regression (*prediction*) model we are trying to learn. A "strong" learner should be able to approximate the true prediction model arbitrarily closely.

To make sense of the somewhat vague denotation above, I like to think of writing a Taylor expansion of some (presumably non-polynomial) function, say $f(x) = e^x$. Let's say we're expanding about $x_0 = c$.[1] Then the n'th-order Taylor expansion is

$$T_{n,c}(x) = \sum_{i=0}^{n} f^{(i)}(c) \times \frac{(x-c)^i}{i!}$$

In a sense, each term of the sum is weakly correlated with the target function f in that if we consider the i'th term $T_n[i]$, its i'th derivative is equal to that of f at $x = c$ and so can be called a "weak predictor" of f:

$$T_{n,c}[i]^i(c) = f^{(i)}(c)$$

Another important note is that none of our weak predictors are redundant; each of them contains at least *some* new information about f.[2] Alone, $T_n[i]$ is a pretty underwhelming approximator – it's only guaranteed to approximate f in a specific way at one

[1] Since the Taylor expansion is focused on being accurate *in the neighborhood of the point the expansion is centered around*, we can interpret this "Taylor model" as weighting inputs/examples near c as far more important than examples from other parts of the input space.

[2] If you fancy appropriating a linear algebra term, you can also consider each of the weak predictors "linearly independent" of each other.

This is actually more on-the-nose than you might think. Just as we can think of vector spaces of R^n as being spanned by n linearly independent basis vectors, we can think of a *function space* being spanned by an infinite number of linearly independent basis vectors, some of which are $(x-c)^0, (x-c)^1, (x-c)^2, \cdots$. So a Taylor expansion approximates a function f using a linear combination of only the basis vectors corresponding to polynomials, with each basis vectors scaled by a certain amount ($f^{(i)}(c) \times i!$).

specific point. That isn't necessarily useful for our goals — if we tried approximating e^x with the Taylor expansion's second term $T_{n,c}[i = 1](x) = (x - c)$, we'd be off by hundreds, thousands, millions almost everywhere!

The power of our "weak predictors" comes when we *combine* them in some way — in the case of the Taylor expansion, a uniformly weighted sum. As we use more terms (i.e., more "weak predictors"), the ensemble becomes arbitrarily accurate to the target function f near c and so is a "strong predictor" of f.

4.1.2 Meta-algorithm vs. algorithm

Importantly, we said that boosting is a *meta*-algorithm, a meta-strategy that we apply onto a strategy that *actually* performs the approximating. In our "Taylor boosting" method, the strategy we use to approximate f is to create an approximator $T_{n,c}[i]$ that has the same i'th derivative of f at c, and the meta-strategy was to combine all those approximators together via addition. We can perform boosting on other strategies. For example, consider a "Dirac boosting" method, where our weak approximators are functions D_i where

$$D_i(x) = \delta(x - i) \times f(x) = \mathbb{1}[x = i] \times f(i)$$

where δ is the Dirac delta function and $\mathbb{1}$ is the indicator function. So basically, our approximator memorizes the function at a single point perfectly and guesses that it equals zero everywhere else. Alone, this weak predictor is pretty terrible, but we can perform "Dirac boosting" by combining many such Dirac approximators D_i in the following way: given an input x, we find the Dirac approximator whose center is closest to x, and output that as the result.[3] The greater the number/density of Dirac approximators along the input space, the more accurate our boosted approximator becomes![4]

Even though we changed the algorithm by which we approximated our function (using our Dirac approximators instead of Taylor approximators), we still employed the *meta*-algorithm of boosting to combine our weak predictors into a strong predictor. This should hopefully make the distinction between the two clear.

4.1.3 Where the analogy is left wanting.

Now of course, the analogy isn't perfect. A Taylor expansion is performed on a known target function f, but in a data science context, we don't have access to the actual function

[3] (Basically a *k-nearest neighbors* regressor where $k = 1$.)

[4] Fun fact: Although there are cases where a Taylor expansion can perfectly recreate the target function (e.g., $f(x) = e^x$), our "Dirac boosting" method cannot *ever* perfectly recreate *almost any* function whose input space is over $\mathbb{R}$, even if we use an infinite number of Dirac approximators in our ensemble! More specifically, it can never perfectly recreate any function f if

$$\lambda(\{x \mid f(x) \neq 0\}) > 0$$

where λ is the *Lebesgue measure*. This is because our Dirac ensemble can only be made up of a *countable* number of approximators which each only memorize a single output, and it can be shown (via, e.g., Cantor's diagonlization argument) that the set of real numbers is a larger sort of infinity (whose size is *uncountably infinite*) than the set of natural numbers (whose size is *countably infinite*).

that is responsible for our data. We *do* have a dataset, a set of samples from the actual function, which can serve as an *approximation* of the true function. So the best we can do is to use the data to create such an approximation (our model).

Since the dataset is almost certain not to contain all of the intricacies of the underlying function, we normally don't want to make our model fit the data *too* well (good ol' *overfitting*). Instead, we tune our model so that it minimizes a *loss function (or objective function)*, which we hope we've set up cleverly enough so that the model is a really good approximation of the underlying function when it reaches a minimum of the loss function (for example, by introducing terms in the loss function that punish the model for overfitting the dataset). We didn't specify an explicit loss function when improving our "Taylor" and "Dirac" ensemble examples since we could explicitly observe the actual function we were trying to fit.

4.1.4 Gradient boosting.

Alright then, so what is gradient boosting?

To follow the usual algorithm on which gradient boosting is employed, we'll use decision trees. We have our model $M(x)$, training examples (x_a, y_a) and a loss function $L(y_a, y_p)$, where $y_p = M(x_a)$ is our current best prediction for y_a.[5] We'll be creating various weak learners $h_i(x)$, and we'll denote our ensembles of the first i weak learners (including $i = 0$) as $E_i(x)$. We decide that the way we're going to ensemble our weak learners is by a summation $E_i(x) = \sum_i h_i(x)$, and we enter the rodeo.

We start with a baseline prediction, the mean of y_a, i.e. $M(x) \leftarrow E_0(x) = h_0(x) = \bar{y_a}$. Now, we're going to create and fit a new weak learner $h_1(x)$. But since we're going to make our stronger learner via $E_1(x) = E_0(x) + h_1(x)$, we can just fit $h_1(x)$ to a sort of *pseudo-residual determined by the loss function*, $y_{pr,1} = -\frac{dL}{dE_0}\Big|_{(x_a, y_a)}$ for all (x_a, y_a) in our dataset. We then fit $h_1(x)$ to the dataset of pseudo-residuals $(x_a, y_{pr,1})$ as is appropriate for our weak learner — in the case of decision trees, we perform recursive partitioning until some user-specified condition is met (e.g. any further partitioning would yield leaves with fewer than c entries, or the information gain ratio of any such partition would be smaller than some threshold) and then taking the mean of each leaf as the predictor for any inputs that fall into that leaf at prediction time). Now that we've fit $h_1(x)$ to the pseudo-residuals as best as we can, adding h_1 to our previous best predictor will further decrease the loss function, so we update our model $M(x) \leftarrow E_1(x) = E_0(x) + h_1(x)$.

Now we're sort of at the same place we were at earlier, just with a slightly better model. So we can create a new weak learner $h_2(x)$ by fitting it to the "second-order" pseudo-residuals $y_{pr,2} = -\frac{dL}{dM}\Big|_{(x_a, y_a)}$ (where now $M = E_1(x)$), then find the constant multiplier that minimizes the main loss function, then we update our model —and on and on we go.

Here, we can view the $(i+1)$'th weak predictor as attempting to approximate the *gradient* of the loss function when the previous best predictor was used. We fit to the

[5] It's worth remembering that the loss function can be as fancy as we'd like as long as its gradient can be computed analytically from the actual and predicted values.

negative of the derivative since we're trying to *minimize* our loss function, so we want to correct toward the direction opposite the derivative (and since we're ensembling via addition, the "opposite" part comes into play at the pseudo-residual level). Since the gradient of the loss function determines how we create our weak learners/ensemble, we call this method **gradient boosting**. And it apparently works wonders when used with decision trees as the weak predictors.[6]

The expression for our pseudo-residuals may seem to come out of nowhere, but let's consider *actual* residuals for a moment: $y_r = y_p - y_a$. When we fit our weak learner to try and predict the negative of these residuals $\{-y_r\}$ and add this to our ensemble, we're taking a step in minimizing the mean-square-error of our predictor: $L = MSE = \frac{1}{2}\left(y_p - y_a\right)^2$. And you can see that $-\frac{d(MSE)}{dy_p} = -y_r$. So rather than stick to just this one type of residual, why not fit to the gradient of whatever loss function we'd like? Hence the term and our expression for "pseudo-residuals".

(Note: For further references and useful links, see this footnote.)[7]

- DK, 5/17/18 (shoot, it's late again...)

[6] For the right sorts of problems, when you tune it correctly!

[7] (The explanation provided by Abhishek Ghose in this Quora post is quite good and helped me properly grasp the concept of gradient boosting. Other main reference was this Kaggle blog post. This page allows for an interactive demo of gradient-boosted decision trees in action.)

5. Physics.

5.1 The Boltzmann distribution

I have an ultimate goal of having a good understanding as to why classical mechanics fails to predict blackbody radiation properly and leads to the so-called ultraviolet catastrophe. The situation involves analyzing the classical formulation of the Rayleigh-Jeans law, which relies on the classical equipartition theorem, which for specific cases can be derived from classical statistical mechanics.
Eventually, we'd like to also take a look at *why* particles absorb the frequencies they do, which involves understanding molecular orbital theory, and so the Schrodinger equation, and so the Hamiltonian, which implies Hamiltonian mechanics, which is apparently an improvement on Lagrangian mechanics, which is an application of the Lagrangian (which we've talked about before at 3.1.2 and so would be our jumping-off point).
But first, counting. (And by the way, a lot of the insight I gained came from this source (which isn't explicitly mentioned in this writeup itself).)

5.1.1 Stating our problem.

Let's say we have a system with N *indistinguishable* particles (i.e., the particles are all of the same "type") and a total energy E that can be swapped freely between the particles at increments of ΔE (and E must be an integer multiple of ΔE), say $E = k\Delta E$. We presume that our system is in thermal equilibrium and no energy or particles enter or exit the system (i.e. that the system is **closed**, and more specifically a *microcanonical ensemble*.). Given this situation, how can we figure out the probability that this system is in a particular state?

Before we continue, we need to determine exactly what we mean by "state". We consider two different levels of a state's description:

1. *macrostate*: This description only has a summarized view of the system. In this case, it describes how many particles at each energy (in terms of a multiple of ΔE). [1]
2. *microstate*: This description holds information not only about the macrostate, but also about other parameters that would predict how the macrostate would evolve (over time). In our case, in addition to the energies of each particle, we'd need to know each particle's position and momentum to know which particles collide with which other particles on the next "timestep" of the system's (*dynamic*) equilibrium.

[1] The macrostate focuses on energy because of energy's importance in physical systems. Minimizing some function of energy is often used to solve problems in Lagrangian and Hamiltonian mechanics, and more generally in the calculus of variations (which I hope to better understand and write about at some point).

In this case, since we know each particle's momentum, *we also know each particle's energy.*

We're interested in determining the probability that a system is in some particular *macrostate*.

5.1.2 A lazy way out.

One way we could potentially figure out our macrostate distribution is by starting at some given microstate, and observe the system evolve over a really long time, recording the macrostates at every differential step in time. The assumptions we'd be making were that:

1. the entire microstate space is connected, i.e., that there is a path from any one microstate configuration to any other microstate configuration (otherwise, we'd miss the macrostates associated with the disconnected microstates); and that
2. the proportion of times a macrostate appears in our record of the system's evolution corresponds to the probability that the macrostate would occur. [2]

The specifics of how the simulation ran would also imply other assumptions, which will be discussed below. But we can approach it through more direct means.

5.1.3 Counting (is hard).

Let's try to count how many microstates correspond to each macrostate. Before we can do that, we need to determine all the valid macrostates. We denote the number of particles with energy $i\Delta E$ as n_i (in terms of integer multiples of ΔE) that particle i has as n_i, then the valid macrostate configurations are vectors $\vec{n} = (n_0, n_1, n_2, \cdots, n_k)$ such that

$$\sum_{i=0}^{k} i \times n_i = k, \quad \sum_{i=0}^{k} n_i = N$$

Figuring out which vectors $\vec{n}$ for which this holds is a bit difficult to describe cleanly. The problem can be described as a form of the coin change problem.[3] In our case, the coins have value $1, 2, \cdots, k$, and we remove sets of cardinality greater than N.[4] The remaining sets are distinct valid macrostates. We can get each set into the form of $\vec{n}$ by setting n_i as the number of times i appears in the set, and setting $n_0 := N - \sum_{i=1}^{k} n_i$. We'll refer to the set of all valid macrostate configurations as V.

Now that we have valid macrostates, we can count how many *different* microstates yield a given macrostate. If all N particles were distinguishable, then every rearrangment

[2] This is actually an important assumption — that observing one system for a very long time tells us information about what a distribution of a large ensemble of systems would look like, and vice-versa. Such systems are called ergodic systems.

It's also important to remember that this is an *assumption*. There are cases where a system can be shown not to be ergodic.

[3] Which is itself a specific form of the knapsack problem.

[4] In this case, we're allowing sets to contain duplicates of the same element.

of them would result in a distinguishable microstate that would correspond to the same macrostate, and so the number of microstates that make up a macrostate would be $N!$ for any given macrostate.

However, we're dealing with *indistinguishable* particles, so we wouldn't notice if we swapped around particles at the same energy state i. We have to divide out these redundant occurrences, because otherwise we're overcounting the *number* of different microstates. There are n_i such particles at any given i, and so the total number of different microstates for a given macrostate $\vec{n}$ would be

$$w\left(\vec{n}\right) = \frac{N!}{\prod_{i=0}^{k} n_i!}$$

If we make the reasonable assumption that all distinguishable microstates are equally likely (after all, if none of them are more energetically favorable than any of the others, why would there be favorites?), $w(\vec{n})$ provides a way of *weighting* the different macrostates $\vec{n}$. [5] We can calculate the probability that the system is in a *macrostate* corresponding to $\vec{n}$ via a proportion:

$$P(\vec{n}) = \frac{w(\vec{n})}{\sum_{\vec{n}' \in V} w(\vec{n}')}$$

5.1.4 Counting Is Hard II: Probability Strikes Back.

6/6/2018 Update: It has come to my attention that my original main source could have been a bit clearer by stating that the assumption of a system with total fixed total energy (a *microcanonical ensemble*, which eventually reaches thermal equilibrium within itself) is appropriate for the discussion of the previous section, whereas a more "relaxed" assumption of a system with fixed *temperature* [6] (a *canonical ensemble*, which reaches thermal equilibrium with the system's surroundings, an entity which can supply energy into the system) is more appropriate in the following discussion wherein we end up assuming independence of energies between particles inside the system — they can independently "get" their energy from the system's surroundings and store it as potential energy, thereby "gaining" energy without violating the fixed temperature stipulation. (No wonder I needed to go through so much mental gymnastics to try and rationalize the independence step...)

Currently the below explanation remains unchanged since this discovery. Once the section is adjusted, this note will no longer precede it.

[5] If, for some reason, this assumption doesn't sit well, one need only tweak $w(\vec{n})$ to their liking. But considering experiments match the previously made assumption, Occam's Razor seems relevant.

[6] Remember that temperature ("intuitively", based on kinetic theory) is related to the average *kinetic* energy of particles in a system. Therefore, a fixed temperature only implies a fixed total *kinetic* energy within a system, not total energy (which could also include potential energies of various forms).

It's worth noting that in thermodynamics, temperature is actually used as an intensive variable of the system, but it would take quite a while to justify/explicate that. The above kinetic-theory way of thinking about temperature should be good enough to explain where the "extra" energy could come from in a system with fixed temperature.

So we've figured out the probability of the *system* being in some *macrostate*. But what if we're interested in individual particles? What would be the probability that, given some total amount of energy in the system $E_T = D\Delta E$, a particle would be at energy level $i\Delta E$?

Well, we know the probability of being in any particular macrostate, and within each macrostate, we know the fraction of particles at energy level $i\Delta E$ (that being n_i), so we do a weighted sum over the macrostate probabilities:

$$Pr[\text{particle in system has energy } i\Delta E] = P(i) = \sum_{\vec{n} \in V} n_i \times P(\vec{n})$$

and we can have the expected number of particles in a particular energy level (assuming N particles in the system):

$$\langle n_i \rangle = \langle n \rangle (i) = N \times P(i)$$

So what's the shape of $P(i)$ (or $\langle n \rangle (i)$) for large N and as $(D, \Delta E) \to (\infty, 0)$?[7] Well, let's consider how we expect the probability function to behave under a few actions.

First, let's figure out the probability that some particle is at energy i and some other particle is at energy j. The usual expression would be $Pr[i, j] = P(i) \times P(j \mid i)$, involving a conditional probability. But wait, we're assuming that we have an arbitarily large number of ΔE that we're able to partition. If that's the case, then no matter what i is, there's still (approximately) the *same* number of ΔE from which the second particle can take its j ΔE's. That is, the probability distribution (basically) *doesn't change* after taking away i of the ΔE's, so $P(j \mid i) \approx P(j)$ and $Pr[i, j] \approx P(i) \times P(j)$, implying that the energy levels of two individual particles are (approximately) *independent*. The approximation becomes more and more exact the more slices of energy there are, i.e., the closer ΔE is to zero, and when $\Delta E = dE$, independence holds exactly.[8] So, in the limiting case, $Pr[i, j] = P(i) \times P(j)$.

What about the probability that sum of two particle's energies equal $i + j$? Once again, as argued above, in our limiting case, the energy of a particle is independent of the erngy of any other particles. So the probability we're looking for is independent of all the other

[7] The two limiting cases serve different purposes. We're trying to fill in our graph of (y = number of particles) vs (x = energy level). Large N makes the ordinate take on more continuous values (making it be better approximated by a continuous function), while $\Delta E \to 0$ lets the abscissa take on more continuous values (which gives more points onto which one can fit a function).

[8] This is because there would always be an infinite number of ΔE's to choose from, no matter how many had been taken up via assignment to previous particles (since $D \to \infty$)— a weird thought, but would technically be true.

Worth noting that assuming that particle energy levels are completely independent is actually a bit crazy-making. If any particle can be at any energy level in $[0, E_T]$, then we can have, say, two particles with energy level $2E_T/3$ and exceed the total energy of the system! Indeed, according to the model, we could change the total energy of the system by a factor of N. Or indeed, have it be zero.

And in fact, having the step size be infinitesimal would mean that $idE = 0 \; \forall i \in \mathbb{N}$, so that every particle would have to have an *uncountably infinite* multiple of dE in order to have *any* energy. (Differentials are weird. As is approximating a discrete situation with a continuous probability function.)

All this to say, we should remember this an *approximation* and we should keep in mind how if we take the approximation to be true, we're bending our problem statement in quite an exotic way. Alas, this is the price we pay for clean derivation results.

particles in the system — it only depends on the two we're looking at. Then we could describe the probability as

$$Pr[i+j] = \sum_{k=0}^{\frac{i+j}{\Delta E}} \left(P(k) \times P\left(\frac{i+j}{\Delta E} - k \right) \right) = q(i+j)$$

which enumerates over energy pairs $(0, \frac{i+j}{\Delta E}), (\Delta E, \frac{i+j}{\Delta E} - \Delta E), \cdots, (i,j), \cdots, (\frac{i+j}{\Delta E}, 0)$.

We've previously assumed that all microstates have equal probability of occurring. Then that means *all* of the energy pairs above (which are essentially microstates of a two-particle ensemble) occur with equal probability. Then

$$Pr[i+j] = q(i+j) = \left(\frac{i+j}{\Delta E} + 1 \right) \times Pr[i,j] = B \times Pr[i,j]$$

But we saw earlier that $Pr[i,j] = P(i) \times P(j)$, so

$$q(i+j) = B \times P(i)P(j)$$

That means that P must be some function such that $P(i)P(j)$ becomes a function of $i+j$ *only* (because $q(i+j)$ is, as we described above, a function of only $i+j$). And the only type of function with such a property is an exponential function:[9]

$$Pr[\text{particle in system has energy } i\Delta E] = P_+(i) = C_+ \times e^{a_+ i}$$

where C_+ and a_+ have yet to be determined. We'll do that soon. But you know, I think it's a *tad* unlikely that a particle is more likely to be in high energy levels than low ones (especially considering our calculations have shown a decrease in probability as energy level increased), so let's first guess that the exponent is negative — if we happen to be wrong, the as-yet-undetermined parameter in the exponent will end up being negative:

$$Pr[\text{particle in system has energy } x\Delta E] = P(x;\Delta E) = C \times e^{-ax}$$

And finally, let's make an alternate expression in terms of energy directly, as opposed to in terms of discrete energy levels, as deriving the Boltzmann distribution will rely on an upcoming assumption on the energy specifically:

$$Pr[\text{particle in system has energy } E] = p(E) = Ae^{-bE}$$

It's worth noting that this is actually a pretty big shift in frame. $P(x)$ implies the energy is discretized by a step size ΔE and has arguments $x \in \mathbb{N}$ no matter how small ΔE becomes. The equivalent point for $P(x)$ in the function p is at $E = x\Delta E$, that is,

$$P(x;\Delta E) = C \times e^{-ax} \mapsto p(x\Delta E) = Ae^{-bx\Delta E}$$

[9] Note that the base could be any positive number that isn't 1 and the statement would still hold true. e is just the most convenient base to use.

That means unlike x, as $\Delta E \to 0$, the values E can take on becomes more and more granular, to the point where E can become arbitrarily close to any desired value.[10] Also, if an explicit upper bound of E_T isn't placed on the argument of p (which is not done in the original derivation of the Boltzmann distribution and so will not be done here), then we're now allowing particles to (very improbably, but technically) be able to take on arbitrarily large energies!

5.1.5 A fork in the road.

Now we can proceed with the derivation in two ways:

1. *Double-down on the fact that we're analyzing cases where $\Delta E \to 0$; that is to say, make $\Delta E = dE$ and E continuous*: This would arguably be the more logical of the choices given our preceding argument. In order for our exponential function to fit the probability distribution *exactly*, we needed the energies of two different particles be independent of each other, and in order for *that* to have been the case we'd need to take ΔE to 0.
2. *Continue our analysis under the assumption that ΔE is not a differential*: This technically aligns with how we understand physics today — i.e., that energy is quantized by the frequency of radiation, $\Delta E(\nu) = nh\nu$, with $n \in \mathbb{R}$ and h as **Planck's constant**, $h \approx 6.626\,\mathrm{J\,s}$. In this case, we'll have to accept that our argument, having previously assumed continuous E, doesn't take "full advantage" of that assumption.

We'll do the continuous case first, as it leads us to the form of the actual Boltzmann function. We'll do the discrete case afterward for a bit of fun, and because one of the big mathematical tricks needed in the derivation will be useful later on.[11]

5.1.6 Planning our attack.

But what's our plan of attack in finding out the correct parameters for our probability distribution(s)? It's the same plan that can be used when finding the correct parameters for *any* probability distribution — derive relationships via the distribution's moments. We can say some random variable ϵ takes on values E for the continuous case and some random variable X takes on values x in the discrete case. Then we can say that:

- the zeroth moment equals 1 (because they're probability distributions).
- The first moment (centered around 0) equals the mean, $E[X]$ (also written $\langle x \rangle$.
- The second moment (centered around $E[X]$) equals the variance, $E[X - E[X]]^2$.

...And so on. In this case, we'll only need to go as far as the first moment.[12]

[10] Though technically, it can never be *any* arbitrary value since the domain of E is always a subset of the rational numbers $\mathbb{Q}$.

[11] Specifically, it will be useful when trying to "patch" the classical equipartition theorem and Boltzmann function in order to describe *blackbody radiation* by having the average energy of an emitted electromagnetic wave depend on both temperature *and* frequency — but we're getting way ahead of ourselves!

[12] As it turns out, a different result (the equipartition theorem) will tell us the value of $\langle x \rangle$ for a system at thermal equilbrium at a given temperature T, so we'll eventually be able to pull the distribution out of the theoretical and into the practical — but again, we get ahead of ourselves.

5.1.7 The continuous case: An easy battle.

So right now we want to figure out A and b in

$$Pr[\text{particle in system has energy } E] = p(E) = Ae^{-bE}$$

using moments. Well, since we're treating energy as continuous, our moments are integrals (which are *much* nicer to deal with than series).

First, the zeroth moment of the probability distribution should equal 1:

$$\int_{E=0}^{\infty} Ae^{-bE}dE = 1$$
$$\cdots$$
$$A = b$$

OK, cool, we've stripped away a degree of freedom, but we need to remove one more before we have something usable. Note also that the infinite integral would not converge for exponentials with positive exponents. So common sense prevailed!

Anyway, onto the next moment. The first moment of the probability distribution should equal the mean of the random variable of our distribution:

$$\frac{\int_{E=0}^{\infty} AEe^{-bE}dE}{\int_{E=0}^{\infty} Ae^{-bE}dE} = \langle x \rangle$$
$$\cdots$$
$$b = \frac{1}{\langle E \rangle}$$

Awesome! We have the parameters for our probability distribution, and we have our final result:

$$Pr[\text{particle in system has energy } E] = p(E) = \frac{e^{-E/\langle E \rangle}}{\langle E \rangle}$$

If you use the result from the classical equipartition theorem[13] , you'd have that $\langle E \rangle = k_B T$ and so

$$p(E) = \frac{e^{-E/k_B T}}{k_B T}$$

And *that* is the **Boltzmann distribution** in all its glory!

[13] Which we'll hopefully go over later.

5.1.8 The Boltzmann factor.

A convenient thing about the Boltzmann distribution is how simple it is to calculate the relative probabilities of a particle being in one state over another. If we have two states with energies E_1 and E_2, we can calculate their relative probability by taking their ratio (we'll keep the denominator of the function as $\langle E \rangle$ for now):

$$\begin{aligned}
\frac{P(E_2)}{P(E_1)} &= \frac{\frac{e^{-E_2/\langle E \rangle}}{\langle E \rangle}}{\frac{e^{-E_1/\langle E \rangle}}{\langle E \rangle}} \\
&= \frac{e^{-E_2/\langle E \rangle}}{e^{-E_1/\langle E \rangle}} \\
&= e^{-(E_2 - E_1)/\langle E \rangle} \\
&= e^{-\Delta E/\langle E \rangle}, \ \Delta E = E_2 - E_1
\end{aligned}$$

The *Boltzmann factor* is simply the relative probability of a certain state compared to the state corresponding to $E = 0$ (however that may be described):

$$\begin{aligned}
\frac{P(E)}{P(0)} &:= F(E) \\
&= \frac{\frac{e^{-E/\langle E \rangle}}{\langle E \rangle}}{\frac{e^{-0/\langle E \rangle}}{\langle E \rangle}} \\
&= e^{-E/\langle E \rangle}
\end{aligned}$$

You'll see that the relative probability reflects the real-life observation that objects in the world are more likely to be found in low-energy states than high-energy ones (and so "prefer" low-energy states to high-energy ones). Note that a ratio of Boltzmann factors gives a ratio of relative probabilities of the two states:

$$\frac{F(E_2)}{F(E_1)} = \frac{\frac{P(E_2)}{P(0)}}{\frac{P(E_1)}{P(0)}} = \frac{P(E_2)}{P(E_1)}$$

The discrete case: A battle of wits.

Alright, now the discrete case. For ease of notation, we will work with $P(x)$ instead of $p(E)$ — again, the mapping between the two is $E \mapsto x\Delta E$, so it isn't too bad to interconvert:

$$Pr[\text{particle in system has energy } x\Delta E] = P(x; \Delta E) = C \times e^{-ax}$$

Again, we have the normalization condition:

$$\sum_{x=0}^{\infty} Ce^{-ax} = 1$$

This is a bit worrying at first. But note that

$$\begin{aligned}\sum_{x=0}^{\infty} e^{-ax} &= \sum_{x=0}^{\infty} \left(e^{-a}\right)^x \\ &= \frac{1}{1-e^{-a}} = (1-e^{-a})^{-1} \text{ if } \left|e^{-a}\right| < 1\end{aligned}$$

from the formula for the sum of an infinite converging geometric series. With that, we have that

$$C = 1 - e^{-a}$$

OK, let's move on to the first moment to figure out what a is in terms of $\langle x \rangle$:

$$\frac{\sum_{x=0}^{\infty} Cxe^{-ax}}{\sum_{x=0}^{\infty} Ce^{-ax}} = \frac{\sum_{x=0}^{\infty} xe^{-ax}}{\sum_{x=0}^{\infty} e^{-ax}} = \langle x \rangle$$

This looks absolutely terrifying at first. And for quite a while. ~~And possibly always.~~ But there's a clever trick.[14] Remember that in general,

$$\frac{d}{dx}\left(\ln f(x)\right) = \frac{\frac{df}{dx}}{f(x)}$$

In the same vein, we evaluate

$$\begin{aligned}\frac{d}{da} \ln \sum_{x=0}^{\infty} e^{-ax} &= \frac{\frac{d}{da}\sum_{x=0}^{\infty} e^{-ax}}{\sum_{x=0}^{\infty} e^{-ax}} \\ &= \frac{\sum_{x=0}^{\infty} \frac{d}{da} e^{-ax}}{\sum_{x=0}^{\infty} e^{-ax}} \\ &= \frac{-\sum_{x=0}^{\infty} xe^{-ax}}{\sum_{x=0}^{\infty} e^{-ax}} \\ &= -\langle x \rangle\end{aligned}$$

So then

$$\langle x \rangle = -\frac{d}{da} \ln \sum_{x=0}^{\infty} e^{-ax}$$

but we already saw that $\sum_{x=0}^{\infty} e^{-ax} = (1-e^{-a})^{-1}$ if $\left|e^{-a}\right| < 1$, so

[14] Which I take no credit for. I saw this trick in Chapter 1 of Eisberg and Resnick's *Quantum Physics of Atoms, 2nd ed.*

$$\begin{aligned}\langle x\rangle &= -\frac{d}{da}\ln\left(1-e^{-a}\right)^{-1}\\ &= -\left(1-e^{-a}\right)\times\frac{d}{da}\left(1-e^{-a}\right)^{-1}\\ &= -\left(1-e^{-a}\right)\times\left(-\left(1-e^{-a}\right)^{-2}\right)\times\left(e^{-a}\right)\\ \langle x\rangle &= -\frac{e^{-a}}{1-e^{-a}} = \frac{1}{e^{a}-1}\end{aligned}$$

Now let's solve for a:

$$a = \ln\left(1+\frac{1}{\langle x\rangle}\right)$$

Interestingly, this changes the base of our exponent:

$$e^{a} = \left(1+\frac{1}{\langle x\rangle}\right)$$

and so our probability distribution becomes:

$$P(x;\Delta E) = \frac{1}{1+\langle x\rangle}\times\left(1+\frac{1}{\langle x\rangle}\right)^{-x}$$

The form is reminscent of solutions to various simple steady-state problems — but alas, it's much less aesthetically pleasing and much less convenient to manipulate/work with than the Boltzmann distribution! ~~Yet another reason why quantum phenomena can be unpleasant to some.~~

5.1.9 Bolting beyond Boltzmann.

All of these calculations have been made with the assumptions that:

1. Particles within the system can be distinguished from one another by the energy state they are in (and particles within the same energy level are indistnguishable about each other); and
2. Any number of our identical particles can occupy any energy state.

These assumptions do not hold for particles in the regime of quantum mechanics. Specifically, in an ensemble of *bosons* (particles with integer spin), within a particular macrostate, individual bosons can *not* be distnguished from each other, even by energy level. That means there are *no* distinguishable microstates — the most granular information you can get is from what we've been calling the macrostate (the vectors $\vec{n}$ of dimension $\frac{E_T}{\Delta E}$ we would calculate via solving the equivalent coin-change problem as mentioned in 5.1.3). Moving forward from there would eventually lead to the **Bose-Einstein distribution**.

If instead of bosons we're dealing with *fermions* (particles with $\frac{1}{2}$-integral spin), then the *Pauli exclusion principle* also applies, meaning that our second assumption would *also* be incorrect and only two such fermions could be in the same energy state.[15] Moving forward from *there* would eventually lead to the **Fermi-Dirac distribution**.

...Phew, that was exhausting!

- DK, 5/29/18

[15] We could get the valid macrostates by solving the coin-change problem where the are only two coins per denomination (or solving it without the constraint then filtering out those sets with more than two coins of the same denomination).

6. Drafts and WIPs

(Avert your eyes! Unless you're fine with drafts and WIPs — in which case, enjoy!)

6.1 Integral transforms.

While this certainly won't answer all questions about integral transforms, it can hopefully shed some light on "where" integral transforms come from.

tl;dr: they can be thought of as changes of basis in a particular function space, with the basis vectors chosen based on what best fits the problem at hand. However, our transformation matrix (the kernel) is necessarily "infinite-dimensional" in order to span the function space, so finding the inverse transformation matrix (and therefore the inverse (integral) transform) is nontrivial.

6.1.1 Short refresher on vector spaces.

(In the following discussion, may be lazy and drop the $\vec{\cdot}$ arrow where we believe the fact that it's a vector is clear.)

We're already pretty comfortable with the idea of describing vectors as a linear combination of basis vectors. If we have some vector space $V \subseteq \mathbb{R}^n$, some $\vec{v} \in V$, and a basis $\{\vec{b}_1, \vec{b}_2, \cdots, \vec{b}_n\}$ for V, we can write v as a linear combination of basis vectors:

$$\vec{v} = \sum_{i=1}^{n} a_i \vec{b}_i$$

Let's emphasize the fact that we chose *a* basis for V and in most cases there is no such thing as *the* basis for V, since a **basis** (of a vector space V) is just a set of linearly independent vectors that span V. (We'll refer back to linear independence in a bit.)

Going back to $\vec{v}$, if we agree of the basis of V we're working with, we can fully "encode" v via an n-tuple where the i'th location holds a_i, the scalar coefficient by which we multiply the basis vector $\vec{b}_i$. In that sense, we can say that $\vec{v} = [a_1, a_2, \cdots, a_n]$. But we can always choose some *other* basis $\{\vec{\beta}_1, \vec{\beta}_2, \cdots, \vec{\beta}_n\}$ with which to express v, in which case we'd (almost certainly) need different coefficients $\alpha_1, \alpha_2, \cdots, \alpha_n$ in order construct the same vector. We haven't changed the *vector spaces*, we've only changed how we represent points *in* our vector space.

Tangent: The "meaning" of vectors.

So, what are we doing when we change bases? What does a vector of $[1, 0, 0, \cdots, 0]$ mean anyway? Arguably, the answer can be given in a "hippie"-sort of way: *"It means, like, whatever you want it to, man~"*

As an example, think of how one can solve a linear system of equations, say of three variables x, y, and z, using a matrix. Often, you'd represent an equation by a row vector comprised of the *coefficients* of the three variables, augmented by the scalar value they equal. That implies that a left-hand-side row vector $[1, 0, 0]$ "maps to" the expression x, $[1, 0, 0] \mapsto x$. That's exactly why you know you're done when you have a diagonal of ones on the left-hand-side of the augmented matrix, for example the row $\left[\begin{array}{ccc|c} 1 & 0 & 0 & a \end{array}\right]$ would mean that $x = a$. But if there's some better way to encode the equations, we could always choose one of those instead, e.g. $[1, 0, 0] \mapsto \frac{x+y}{2}$, $[0, 1, 0] \mapsto \frac{y+z}{2}$, $[0, 0, 1] \mapsto \frac{x+z}{2}$. It's all in *your* hands ~

Back to bases, and inner products.

All that said, when we change our basis from $\{\vec{b}_1, \vec{b}_2, \cdots, \vec{b}_n\}$ to $\{\vec{\beta}_1, \vec{\beta}_2, \cdots, \vec{\beta}_n\}$, we can think of it as changing our mapping of basis vectors from $e_i \mapsto b_i$ to $e_i \mapsto \beta_i$ where e_i is our standard basis vector for the i'th coordinate, $e_i \in V$, $e_i[j] = \delta(i, j)$.

But that's only part of the story. How do we describe $\vec{v}$ with our new basis? That is to say, how do we find the mapping for our coefficients $\{a_1, a_2, \cdots, a_n\} \mapsto \{\alpha_1, \alpha_2, \cdots, \alpha_n\}$ so that they "encode" the same point $\vec{v}$ in the vector space, i.e. that

$$\sum_{i=1}^{n} \alpha_i \vec{\beta}_i$$

and

$$\sum_{i=1}^{n} a_i \vec{b}_i$$

refer to the same point in the vector space?

Well, so far we have no way of measuring "how much" of one vector is in another. To make this clearer, consider two vectors $\vec{v}$ and $\vec{w}$, both in V. We want to create an operation that allows us to describe v as an addition of two vectors, one "parallel" to w and one "orthogonal" to w (both in the intuitive senses of the word):

$$\vec{v} = \vec{v}_{v \| w} + \vec{v}_{v \perp w}$$

If we have an operation that gets us one of the two component vectors, we can always define the other as a subtraction from the resultant vector. Turns out, we focus on similarity more (what a pleasant thought)! More specifically, we need to define an **inner product** $\langle \cdot, \cdot \rangle$ over the vector space V, which is just an operation that that takes in two elements from V and returns a scalar, and which also follows four key properties:

- Symmetry: $\langle u, v \rangle = \langle v, u \rangle$
- Linearity: $\alpha \in \mathbb{R} \to \langle \alpha u, v \rangle = \alpha \langle u, v \rangle$
- Distributivity: $\langle u + v, w \rangle = \langle u, w \rangle + \langle v, w \rangle$
- Positive Definiteness: $\langle v, v \rangle \geq 0$. $\langle v, v \rangle = 0 \to v = \vec{0}$

A vector space with an associated inner product is called an **inner product space**. We can choose *any* operation that fulfills these four requirements to be our definition of an

inner product! There are some standard ones though. In the case of vectors in $\mathbb{R}^n$, the standard is $\langle u, v \rangle = \sum_{i=1}^{n} u_i v_i$. With our definition of an inner product in hand, we can have that

$$\vec{v}_{v\|w} = \frac{\langle v, w \rangle}{\|w\|} \hat{w}$$

where $\hat{w}$ is the unit vector in the direction of w and the **norm** of the vector w is defined via the inner product $\|w\| := \sqrt{\langle w, w \rangle}$. The remaining component $\vec{v}_{v\perp w}$ is **orthogonal** to w, meaning $\langle \vec{v}_{v\perp w}, w \rangle = 0$.

Conveniently, if $\vec{w}$ is already of unit magnitude, then $\|w\| = 1, \hat{w} = \vec{w}$, and $\vec{v}_{v\|w} = \langle v, w \rangle \vec{w}$. That is, the "amount/number" of a unit vector w inside v is simply $\langle u, v \rangle$. And if the unit vector w is a basis vector b_i, then $\vec{v}_{v\|w}$ is that basis vector, scaled by $\langle v, b_i \rangle$![1]

Now we can finally answer our question! If we want to describe the same vector $\vec{v}$ in terms of a new basis $\{\vec{\beta}_i\}$ we can calculate the coefficients $\{\alpha_i\}$ simply by calculating $\frac{\langle v, \beta_i \rangle}{\|\beta_i\|}$ (with both v and $\{\beta_i\}$ represented in terms of a common basis — presumably the standard basis $\{e_i\}$ — as needed in order to calculate the inner product).

6.1.2 Functions, function spaces.

Now what if I told you that a function is just a vector? More accurately, a function *can be viewed* as an element of a **function space** (which needs qualification — e.g. "the space of all continuous functions" $\mathbb{C}^0$), with specific coefficients based on the choice of *basis functions* we use to span the function space in question.

Power series and discrete function spaces.

As a stepping stone, let's consider an n'th-degree power series:

$$p(x; n) = \sum_{i=0}^{n} a_n x^n$$

We're looking at a weighted sum of monomials. What does this remind us of?
Well, it kinda looks like a linear combination of vectors, doesn't it?
To show the mapping, we can map the natural basis vector:

$$\vec{e}_i \mapsto x^i$$

If we agree to this basis, our function $p(x; n)$ is fully described by the coefficients a_i:

$$p(x; n) = [a_1, a_2, \cdots, a_n]$$

So we have $p(x; n)$ as a vector in the vector space of all polynomials of degree no greater than n. Let's call this vector space of functions (or *function space*) P^n. Then $p(x; n) \in P^n$.

[1] "!" used to denote excitement, not factorialization.

But what's stopping us from letting $n \to \infty$? Let's let loose. If we do, we get the normal, "full" power series[2]:

$$p(x) = \sum_{i=0}^{\infty} a_n x^n$$

When we start playing with infinity, we have to start being a bit more clever. Our previously used "vector as an n-tuple" representation of a function f becomes a *bit* unreasonable. Even if we've agreed on our basis mapping $\{e_i \mapsto b_i(x)\}$ and we know exactly what the coefficient a_i for the i'th basis vector is for all $i \in \mathbb{N}$, we can't literally sit here and list them all out manually: $\vec{f} = [a_0, a_1, a_2, \cdots]$.

But how about this? Instead of encoding f into our vector space as a *tuple* $f \mapsto \vec{f} = [a_0, a_1, \cdots]$, how about we encode f as *another function $A_f(k)$ which outputs the appropriate coefficient for the k'th index of our (infinite-dimensional) vector*? Then if we ever need the value at the i'th index of our encoding, instead of indexing a tuple ($\vec{f}[i]$), we just call our coefficient-encoding function $A_f(i)$. As long as there *is* an underlying function that can give us $A_f(k)$, we're good. Let's come back to that in a moment.

6.1.3 An inner product for our function space.

Currently we have a vector space. But we *don't* yet have an inner product space. What would be a sensible inner product, a sensible measure of (norm-sensitive) "similarity" between functions?

Well, these are functions over a variable x. Let's hearken back to ye olde days of first-year calculus. Back then, we would construct Taylor-series approximations T of some original function f around a point $x = c$ and proclaim that T is "similar" to f at and in some neighborhood around that point c. Why? (Let's say T was a k'th-order approximation.) Well, because we specifically constructed T so that

$$\forall\, i \in \{0, 1, \cdots, k\}, T^{(i)}(c) = f^{(i)}(c)$$

— i.e., so that T matched f's slope, concavity, etc., around $x = c$ — we expect the *output* of T to be approximately the same as the *output* of f around $x = c$, even for wacky original

[2] A quick aside on **power series**: when dealing with the "full" (infinite-order) power series, we should consider radii of convergence if we want our power series to represent a desired fucntion. Many vectors $\{a_i\}$ map to functions that diverge pretty much anywhere besides $x = 0$. This may sound exotic, but it pops up in "mundane" functions. The seemingly innocuous vector $[1, 1, 1, \cdots]$ corresponds to $f(x) = \frac{1}{1-x}$ only within a radius of convergence of $|x| < 1$ — outside that radius, the function explodes. If we instead had a vector $[1, 2, 3, \cdots]$, and our function will blow up quite quickly for $x = 0$.

But we needn't always care whether or not the function it represents converges if all that interests us are the monomials' coefficients. When we're viewing a power series just as a sequence of coefficients (with operations that reflect polynomial arithmetic), we refer to it as a **formal power series** . The way polynomial multiplication works allows us to solve potentially really tricky problems by creating a (finite or infinite) power series with the appropriate coefficients, exponentiating the power series, and reading off the coefficient of a particular monomial. More on that in 6.3.

functions f.[3] All this to say, in choosing the inner product for functions in a particular space, we probably want to measure the "responses" of these functions to the input variable(s) they permit. Keeping this in mind and looking back at the requirements to be an inner product, we can see that a reasonable choice is an integral over the domain of x if x is continuous:

$$\langle f, g \rangle = \int_{Dom(x)} f(x)g(x)dx$$

or, if x is a discrete variable, then a summation instead:

$$\langle f, g \rangle = \sum_{x_i \in Dom(x)} f(x)g(x)$$

Note that an inner product space only has one inner product. Part of the description of a function space is the domain of the input variable, which would determine which form of the inner product would be appropriate.

Now this definition of an inner product would be great — we'd have a way of describing any function in terms of our desired basis by doing an integral! — *if* we can get it to converge. Hardly a guarantee: take $f(x) = x^0, g(x) = x^2$ (both natural unit vectors in our current basis of polynomial space!) and $x \in \mathbb{R}$ as just one example. But let's not give up on it just yet — we may just be able to make things work with another basis!

6.1.4 Beyond the countable, and picking a basis.

It would help if we expanded our horizons a bit. I mean, getting to have $x^i,\ i \in \mathbb{N}$ is great and all, but we're definitely not spanning nearly as much of our function space (say the space of smooth (i.e., infinitely differentiable) functions, $\mathbb{C}^\infty$) as we could. I mean, I can't even fully encode x^π or x^e with such a puny basis! And I like both π and e! ~~And especially pie! *Mmmm... Pie.*~~

Enough belly-aching. We're now going to expand our set of basis functions to be *continuous*:

$$B_{P+} = \{x^i \mid i \in [1, \infty)\}$$

Now we can handle x^π and x^e quite easily. But we still can't describe *every* function (namely, any functions which contain $x^i, i < 1$). And we're probably still in trouble with using our inner product — outside of $|x| < 1$, we're probably exploding.

Alright, let's not be so negative! Or wait... Actually, let's be negative!

$$B_P = \{x^{-i} \mid i \in [1, \infty)\}$$

---

[3] This breaks down with "pathological" functions such as $f(x) = \begin{cases} 0 & x = 0 \\ e^{1/x^2} & x \neq 0 \end{cases}$, which has $f^{(i)}(0) = 0\ \forall i \in \mathbb{N}$ and therefore has a Taylor series of exactly $T(x) = 0$ if centered around $c = 0$ — hardly how f acts outside the origin! But in this situation we can only grumble about these sorts of functions, called *non-analytic functions*, and explicitly exclude them from our analysis.

Alright, *this* looks promising! Now if we restrict the domain of our input variable to something like $x \in [1, \infty)$, our inner product won't explode just by taking the norm of a basis vector. This is promising, but there may be an even more useful set of basis functions to use — it's kind of lame to have to start at $x = 1$. What if we're modeling something over time? We don't want to just chop off the first horizontal unit's worth of data if we could help it!

Before we get carried away, we should realize what these changes of basis mean. You'll notice that in our switch from B_{P+} to B_P, we lost the ability to fully/easily encode x^π and x^e, and we had to change our allowed input domain from $(-1, 1)$ to $[1, \infty)$. So when we change our basis, it isn't without consequence. We end up *changing the part of function space we can describe*. Put in a way that may sound almost tautological, *different sets of basis functions describe different (potentially disjoint) sets of functions*. This is worth keeping in mind as we finally talk about integral transforms.

6.1.5 The integral transform: a concatenation of inner products.

So what is an integral transform? You could make your own if you wanted to! (Whether or not it would be useful is another question.) Essentially all it is is *a bunch of inner products over a set of previously selected basis functions*. As we talked about before, taking inner products of a vector $\vec{v}$ with basis vectors $\{\vec{b_i}\}$ and dividing by the norm of the basis vector provides us the coefficients needed to describe $\vec{v}$ in the vector space spanned by $\{\vec{b_i}\}$. Going back to functions again, we can say that *the integral transform produces from the input function $f(x)$ the vector representation of f, in the form of the coefficient-generating function $A_f(n)$, in the function space spanned by a set of basis functions $\{K(x;n)\}$*. In mathematical form,

$$A_f(n) = \langle K(x;n), f(x) \rangle = \int_{Dom(x)} K(x;n) f(x) dx$$

You're free to choose whatever set of basis functions $\{K(x;n)\}$ you'd like, and have the size of this set (measured by n) be whatever you'd like. Since the choice of $K(x;n)$ is at the core of the integral transform (it determines what functions you can represent and in what way they're represented), $K(x;n)$ is called the **kernel** of the transform.[4] With a properly restricted input domain and properly chosen kernel, the mapping between f and A_f is one-to-one, implying both the uniqueness of A_f *and* a possible to "invert" the transform back into the form of the original function!

You'll notice that in the above definition of the transform, I didn't divide by the norm of the basis vectors. Technically, if we want A_f to capture the *projections* of f onto the basis functions, we would indeed need to divide through by the norm:

[4] Which is unrelated to the kernel, i.e. nullspace, of a matrix in linear algebra

...Yeah, I know, it would have been nice if they were a bit more creative with the names. ~~I mean, "nullspace" was perfectly good for what it was describing — why'd they have to go and call it the "kernel" too? All it succeeded in doing was make me try to understand the connection between these two kernels in vain until I realized there was no connection and they're just named the same thing.~~ I found it confusing too.

$$\begin{aligned}\vec{f} &= \frac{\langle K(x;n), f(x)\rangle}{\|K(x;n)\|} \\ &= \frac{\int_{Dom(x)} K(x;n)f(x)dx}{\sqrt{\int_{Dom(x)} K(x;n)^2 dx}}\end{aligned}$$

Personally, I kind of like this form a little bit more, mainly because its seemingly out-of-nowhere form might prompt a student to ask "What's up with that denominator?". Which may lead to a discussion about this pretty interesting way of thinking of functions, which may help demystify the Transforms™ that otherwise appear to be delivered from on high. Alas, in this case, laziness prevails. See, because the denominator is invariant to the input function $f(x)$, all coefficient-generating functions A_f for *every* function is off by the same factor $1/\sqrt{\int_{Dom(x)} K(x;n)^2 dx}$ at any particular n. But if they're *all* off by the exact same factor at the exact same locations, the coefficient-generating functions will be unique whether you correct by that factor or not. And so often they don't until/unless they perform an inverse transform, at which point they make up for the difference.

6.1.6 The Laplace Transform: a differential equations-friendly integral transform.

6.2 The Fourier and Laplace Transforms.

General flow is:

1. Explore the Taylor series in terms of linear algebra (countably infinite basis vectors). Consider where and how it lacks power (discrete vs continuous). From this discrete case, extend to a continuous case (*un*countably infinite basis vectors!)

2. Define *linear independence* for vectors in function space, and use this to determine what our basis vectors can be. Massage out pseudo-Laplace transforms.

6.2.1 Revisiting the Taylor series.

Previously, we've discussed the Taylor series centered around a fixed point a:

$$\begin{aligned}f(x) &\approx \sum_{n=0}^{k} \frac{f^{(n)}(a)}{n!}(x-a)^n \\ &= \sum_{n=0}^{k} F_D(n;a)(x-a)^n \text{ where } F_D(n;a) = \frac{f^{(n)}(a)}{n!} \\ &= p(x;a,k)\end{aligned} \tag{6.1}$$

The rationale for the expression we choose for $F_D(n;a)$ is that $p(x;a,k)$ will have identical i'th order derivatives with $f(x)$ at a for $k \in \{0,1,\cdots,k\}$, i.e., that

$$p^{(i)}(x=a) = f^{(i)}(x=a) \; \forall \; i \in \{0,1,\cdots,k\}$$

So as k becomes larger and larger, $p(x;a,k)$ acts more and more like $f(x)$ at $x=a$, which we hope makes it "act" more and more like $f(x)$ in a neighborhood around a.[5] Letting $k \to \infty$ makes $p(x;a,k)$ "act" *exactly* like $f(x)$ at $x=a$. For some functions f, we can show that $p(x) = f(x) \; \forall x \in \mathbb{R}$ for *all* inputs of the function by taking the limit of Lagrange remainders. The most notable functions for this are $f(x) \in \{e^x, \cos(x), \sin(x)\}$.

6.2.2 Discrete function spaces.

Now let's go on a seemingly random ~~secant~~ tangent.

Consider some random polynomial $p(x;k)$ of order no greater than k. We could describe it as weighted sum of monomials, where the coefficients of the monomial x^n is given by the function $A(n)$:

$$p(x;k) = \sum_{n=0}^{k} A(n)x^n$$

What does this remind us of?

Well, it kinda looks like a linear combination of vectors, doesn't it?

Consider a k-dimensional vector space with the basis $B = \{e_n \mid n \in \{0,1,\cdots,k\}\}$. If we wanted to describe some vector in this space, we could do so as a linear combination of the basis vectors:

$$\vec{v}_A = \sum_{n=0}^{k} A(n)e_n = [A_0, A_1, \cdots, A_k]$$

That alone is kind of boring. What becomes more interesting is if we can create an *isomorphism* between this (to be honest, kinda bland) k-dimensional vector space and something more interesting. If we think of a neat, valid isomorphism for our basis vectors, some other neat consequences should follow.

Well, what if we make the mapping $e_n \leftrightarrow x^n$ (for $x \in \mathbb{R}$)? This is a valid mapping, because we can describe the set of $\{x^n \mid n \in \{0,1,\cdots,k\}\}$ as linearly independent, in that there's no way to make a weighted sum of other monomials $x^b, b \neq n$ identical to x^n for all x in our domain of interest.

Let's keep this going. What would $\vec{v}_A$ mean on the other side of our isomorphism?

$$\vec{v}_A = [A_0, A_1, \cdots, A_k] = \sum_{n=0}^{k} A(n)e_n \longleftrightarrow \sum_{n=0}^{k} A(n)x^n = p_A(x)$$

[5] This doesn't always work perfectly for all functions. For example, $f(x) = \begin{cases} 0 & x = 0 \\ e^{1/x^2} & x \neq 0 \end{cases}$ has $f^{(i)}(0) = 0 \; \forall i \in \mathbb{N}$, so its Taylor series centered at $x=0$ yields $p(x)=0$, which certainly doesn't capture how f acts outisde the origin!

where $p_A(x)$ is the (no greater than k'th-order) whose coefficients are enumerated by $A(n)$. This means that if we agree on a valid "encoding" process — in this case, that we'll describe $p_A(x)$ as a weighted sum of monomials x^n — then we can go from function to vector and vice-versa. When we let $k \to \infty$, we're allowing our vectors, described by $A(n)$, to describe the space of all possible polynomials P^∞.

Worth noting is that there's no reason why we *have* to choose the function x^n in our mapping $e_n \leftrightarrow x^n$. We can choose *any* function where it makes sense to say that the functions for different n are linearly independent of each other. For example, we could choose to have the mapping

$$e_n \leftrightarrow e^{nx}$$

since the set $\{e^n x \mid n \in \{0, 1, \cdots, k\}\}$ is linearly independent.[6] The more-or-less complete freedom in choosing our basis mapping will prove useful, as some "encodings" (i.e. some sets of basis functions) will be more useful in some contexts than others. The main caveat is that subset of function space implied by your basis mapping is always limited as to what functions it can "encode" fully and/or concisely — if you choose the mapping $e_n \leftrightarrow e^{nx}$, there is no $A(n)$ such that $\sum_{n=0}^{k} A(n)e^{nx} = x$, so we're definitely not spanning P^∞ with this new mapping and if you're interested in simple finite-degree polynomials, you're probably better off with our earlier encoding.[7]

6.2.3 Equipping our vector space.

Here's something trickier: how do we perform a projection on a function that's outside our subspace? That is to say, how would we actually *calculate* the function $A(n)$?

Well, so far we haven't touched on that. We've defined a vector space, but we haven't defined a *norm* $|\cdot|$ or an *inner product* $\langle \cdot, \cdot \rangle$ for our vector space meant to represent, in this case, polynomials. which we'd want to have capture "how much" of one vector is along in another vector would be used to calculate our coefficients $A(n)$ by supplying e_n as one of the inputs to our inner product. Well, that depends on what our metric is for considering two functions to be "close"

6.2.4 Re-revisiting the Taylor series.

Let's consider the Taylor series again, where we'll set $a := 0$ and let $k \to \infty$:

[6] In fact, since $e^n x = \left(\sum_{k=0}^{\infty} \frac{x^k}{k!}\right)^n$, we could view e^{nx} as a(n oddly) weighted combination of $x^0, x^n, x^{2n}, \cdots$, implicitly describing an infinite-degree polynomial with potentially only a finite number of coefficients.

[7] Worth noting is that trying to create a mapping that can recreate *all* functions is not the best idea. Recall that if we designate an input x in some domain D_x and some output y in some domain D_y, then a **function** $f \in \mathcal{P}(D_x \times D_y)$ is simply a set of ordered pairs (x, y) where each x appears only once in the set. (Here $\mathcal{P}$ denotes the *powerset* operator and $\times$ the *Cartesian product* operator.) So there are *many* wacky functions one could make and basically no hope in accounting for all of them in a simple-to-describe basis.

$$
\begin{aligned}
f(x) &\approx \sum_{n=0}^{\infty} F_D(n)x^n \text{ where } F_D(n) = \frac{f^{(n)}(0)}{n!} \\
&= p(x)
\end{aligned}
\tag{6.2}
$$

Relating the Taylor series to our vector-space isomorphism, setting $A(n) := F_D(n)$ picks a point $\vec{p}$ in our vector space P^k that corresponds to the polynomial p which is "closest" to f in the sense that p's value and first k'th derivatives agree with f's at $x = 0$. It's almost like p is f's "*projection*" onto P^k under our chosen definition of "closeness".

Our point p — and as such, our function that captures the coordinates of p, $A(n)$ — definitely has to be related to our definition of "closeness". In this case, the metric we're minimizing is $\sum_i |f^{(i)}(0) - p^{(i)}(0)|$. If our definition of closeness was instead, say, that they have identical definite integrals $\int_{x=0}^{1} f(x)dx = \int_{x=0}^{1} p(x)dx$, we'd have a different "closest" point in our vector space. This also shows that the metric of "closeness" we use determines whether there is a *unique* $A(n)$ for $f(x)$ in a given function space. If we're considering the space of polynomials, there is only one point that satisfies the "equal derivatives at a" constraint, but many that satisfy the "equal definite integral over $[0, 1]$" constraint.

6.2.5 But first, let's talk eigenfunctions.

In order to "trust" that the Fourier and Laplace transforms are capturing all of the "content" of our input function

6.2.6 The Fourier transform and wiggles over time.

The Fourier transform is the way one maps a function $f : t \mapsto y$, $t \in \mathbb{R}$, $y \in \mathbb{R}$ to its uniquely equivalent function $F : \omega \mapsto z$, $z \in \mathbb{R}^2$. Looking at it more explicitly through the lens of linear algebra, if we say f came from a space $\mathcal{F}_t$, i.e., $f \in \mathcal{F}_t$, and likewise that $F \in \mathcal{F}_\omega$, then the Fourier transform FT is a mapping $FT : \mathcal{F}_t \mapsto \mathcal{F}_\omega$.

The actual definition for the **Fourier transform** is

$$
F(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-i\omega t} f(t) dt \tag{6.3}
$$

and can be derived from taking the limit of the *discrete-time Fourier series* of $f(t)$ as the function's period $T \to \infty$.[8] The inverse map, aptly named the **inverse Fourier transform**, is

$$
f(t) = 2\pi \int_{-\infty}^{\infty} e^{-it\omega} F(\omega) d\omega \tag{6.4}
$$

[8] Which we hope to discuss further at some point.

Here we use $i = \sqrt{-1}$. (We may flip-flop between i and j to denote $\sqrt{-1}$. It should be clear from context.)

OK, cool, but what does it mean? Well, first let's remember one of the neatest things you can prove using Taylor series, i.e. *Euler's formula*:

$$e^{ix} = \cos(x) + i\sin(x)$$

Also, note that $\sin(x)$ and $\cos(x)$ are mutually orthogonal[9], and in fact all sinusoidal functions $\{\cos(ax) \mid a \in \mathbb{R}\}$ and $\{\sin(bx) \mid b \in \mathbb{R} - \{0\}\}$ are mutually orthogonal.[10] That means that if we were to describe functions in a vector space, then each sinusoidal function $\cos(ax), a \geq 0$ and $\sin(bx), b > 0$ would be linearly independent from one another.[11]

All this points to a *different basis* on which we can describe functions. The origina

6.2.7 The Laplace transform as a patch for the Fourier transform.

What if the Fourier transform explodes? For example, consider the function

6.3 Generating functions: dealing with uneven probabilities.

6.4 Central Limit Theorem

[9] That is to say, $\int_T \cos(x)\sin(x)dx = 0$ over their aggregate period T. To be proven later. See in the meantime this resource.

[10] When $b = 0$, $\sin(bx) = \sin(0) = 0 \forall x$ and so doesn't fit the definition of orthogonality since $\int_{-\infty}^{\infty} sin(0x)sin(0x)dx = \int_T 0dx = 0 \not> 0$.

[11] Only one "half" of $\mathbb{R}$ is needed to capture all non-redundant information as $\cos(-x) = \cos(x)$ and $\sin(-x) = -\sin(x)$.

Index

basis
    in linear algebra, 35
Boltzmann distribution, 30
Boltzmann factor, 31
boosting (machine learning), 20
Bose-Einstein distribution, 33
boson, 33

coin change problem, 25
cross-entropy, 4

dual problem, 16

entropy (information theory)
    conditional, 6
    for distributions, 4
    for random variables, 5
Euler's formula, 45

Fermi-Dirac distribution, 34
fermion, 34
Fourier transform, 44
function, 43
function space, 20

information gain, 6
inverse Fourier transform, 44

joint entropy, 5

kernel
    in integral transforms, 40
KL divergence, 4

Lagrange multiplier, 11
Lagrangian, 12
Lebesgue measure, 21
level set, 10
loss function, 22

mutual information (information theory), 4

norm, 37

overfitting, 22

Planck's constant, 29
posterior (Bayesian statistics), 8
power series, 38
    formal power series, 38
prior (Bayesian statistics), 8

relative entropy (information theory), 4

sizes of infinity
    countably infinite, 21
    uncountably infinite, 21