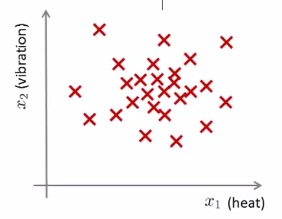
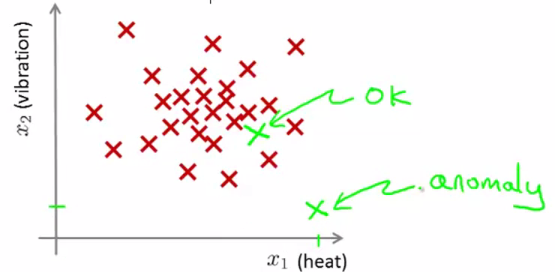
***Density Estimation***

**I. PROBLEM MOTIVATION**

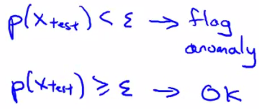
* **Anomaly Detection** = a reasonably commonly used type of ML.
* **1** interesting aspect: it's mainly for unsupervised problems, but w/ some aspects that are very similar supervised learning problems.
* Ex: Manufacturer of aircraft engines + as engines roll off the assembly line, you're doing QA + are testing/measuring features of the aircraft engine (heat generated, vibrations, etc.)
* End up w/ a data set of X1-X(m) feature vectors for m manufactured aircraft engines + plot the data



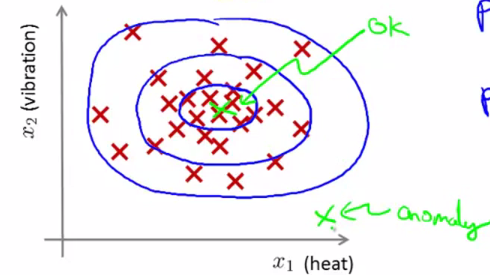
* The next day, a new aircraft engine rolls off the assembly line w/ some set of features X(test)
* Anomaly detection wants to know if this aircraft engine is **anomalous** in any way 🡪 should it undergo further testing or is it okay to ship it to a customer w/out further testing.

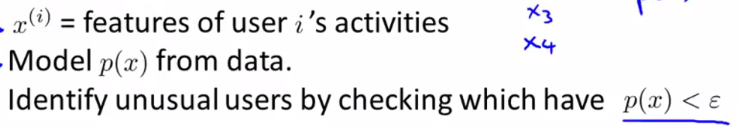


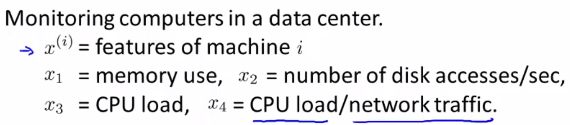
* More formally: In anomaly detection, we're give some data set’s X1-Xm examples (usually assuming these examples are normal/non-anomalous) + we want an algorithm to tell us if some new example Xtest is anomalous or not
* Given this unlabeled training set, we're going to build a model for **p(x) = probability of** x, where x are these features of aircraft engines.
* Having built a model of p(x), for a new aircraft engine, if p(x-test) < some epsilon, flag as an anomaly

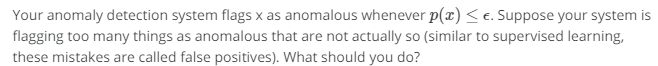


* On a plot, hopefully points lying somewhere in the middle have a pretty high p(x), + as we move out, p(x) lowers, but up to a point until they become anomalies



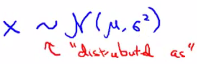
* Applications of anomaly detection:
* Fraud Detection (most common)
* If you have many users, + if each user does different activities, you can compute features of the different user activities + what build a model to check the probability of different users behaving different ways.
* Or to “check the probability of a particular vector of features of a user’s behavior”
* Examples of features of a user’s activity may be: x1 = how often this user logs in, x2 = # of pages visited, or # of transactions, x3 = # of posts on the forum, x4 = typing speed
* Can model p(x) based on this sort of data + can try to ID users behaving very strangely on your website by checking which ones have p(x) < epsilon + maybe send the profiles of those users for further review or demand additional identification from those
* This sort of technique will tend of flag users behaving *unusually*, not just users that maybe behaving *fraudulently*
* This is actually the technique used by many websites that sell things to try ID users behaving strangely = might be indicative of either fraudulent behavior or stolen accounts
* 
* Manufacturing.
* Can find unusual, say, aircraft engines + send those for further review.
* Monitoring CPUs in a Data Center
* If you have a lot of machines in a CPU cluster or data center, we can compute features at each machine like memory used, # of disc accesses, CPU load as well as more complex features the CPU load on a machine divided by amount of network traffic on that machine
* Then given a dataset of how CPUs in your data center usually behave, you can model p(x( = probability of these machines having different amounts of memory use or different numbers of disc accesses or different CPU loads + so on
* If you ever have a machine whose p(x) is very small, you know that machine is behaving unusually + might be about to go down + you can flag it for review by a sysadmin.
* This is actually being used today by various data centers to watch out for unusual things happening on their machines.



* 
* 

**II. GAUSSIAN DISTRIBUTION**

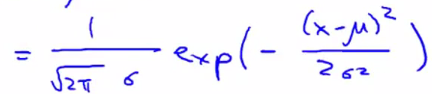
* **Gaussian distribution** = the normal distribution
* Say x is a real-valued random variable **Ɍ**
* If the probability distribution of x is Gaussian w/ mean μ + variance δ^2, write x ~ N(μ, δ^2)



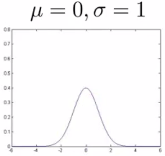
* Tilde = “distributed as.”
* N() = normal since Gaussian + normal are synonyms.
* Gaussian distribution is parameterized by 2 parameters, a mean μ + a variance δ^2
* Center of its bell-shaped curve plot = μ + the width is, δ (1 SD)
* This plot specifies the probability of x taking on different values (x values further from the middle are diminishing in probability)
* Formula for the Gaussian distribution:



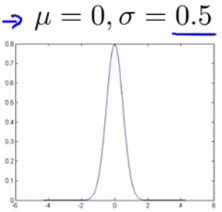
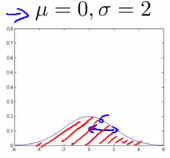
* probability of X is parameterized by the 2 parameters μ + δ squared
* Formula for the Gaussian density



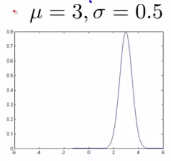
* This is formula for the bell-shaped curve
* The curve is what you get if you take a fixed value of μ + a fixed value of δ, + plot p(x)
* p(x) is plotted as a function of X for a fixed value of μ + of δ squared.
* Sometimes it's easier to think in terms of δ squared (variance) + sometimes is easier to think in terms of δ (SD)
* If μ = 0, δ =1. we have a Gaussian distribution centered around 0,



* Width of a Gaussian is controlled by δ
* Let’s say μ = 0 + δ = 0.5 or 2 (variance δ squared would therefore be 0.25 or Sqrt(2))

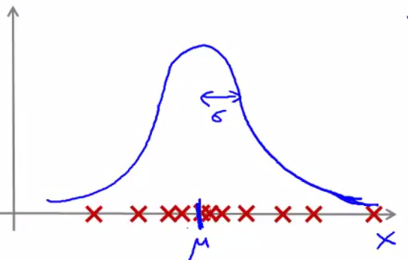
* B/c this is a probability distribution, the AUC for each curve must integrate to 1
* So thinner but much taller density = same area as 1st plot
* Gaussian distribution centered at μ = 3 shifts over the entire distribution.



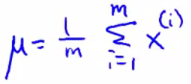
* **Parameter estimation problem**.
* Ex: We have a dataset of m examples + each example is a real number.



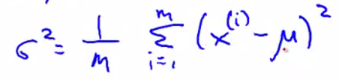
* Let's say I suspect that these examples X(i) came from a Gaussian distribution: X(i) ~ N(μ, δ^2), but I don't know what the parameters are.
* Given my data set, I want to estimate the values of μ + δ squared.

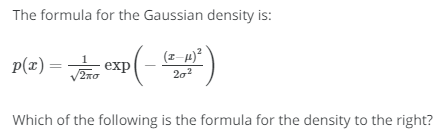
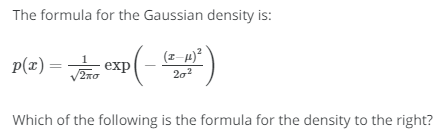
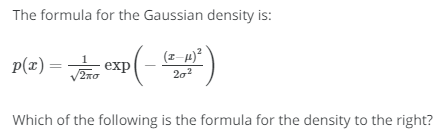


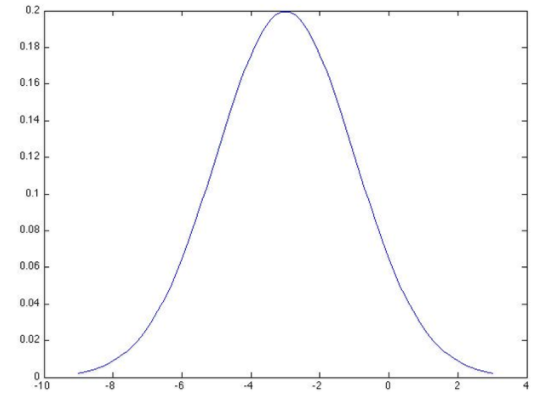
* Maybe the above might be roughly the Gaussian distribution it came from
* Seems like a reasonable fit to the data b/c it looks like data has a very high probability of being in the central region, + a lower probability of being further out
* Standard formulas for estimating the parameters μ + δ squared:
* μ = the average of my examples.



* Then use that μ to calculate δ squared



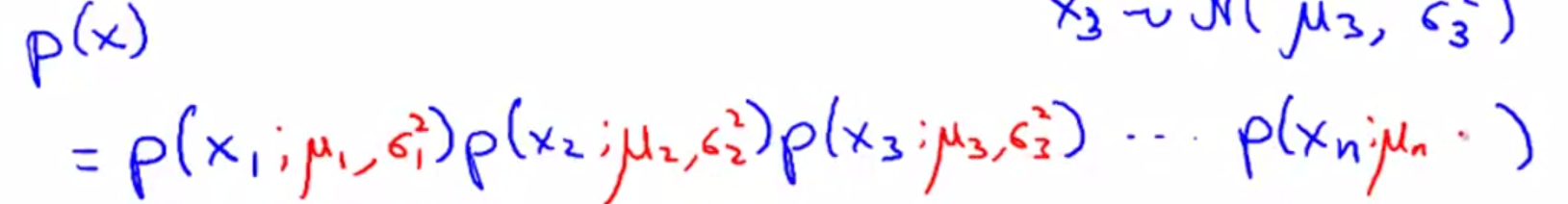
* If you've heard of **maximum likelihood estimation**, these parameter estimates are actually the maximum likelihood estimates of the **primes** of μ + δ squared
* In ML people tend to use 1/m instead of 1/m-1, but in practice it makes essentially no difference assuming m is a reasonably large training set size.
*   

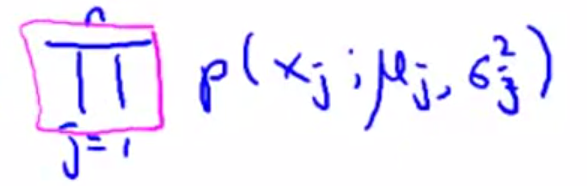
****



**III. ALGORITHM**

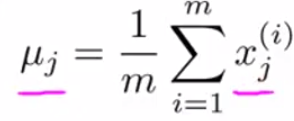
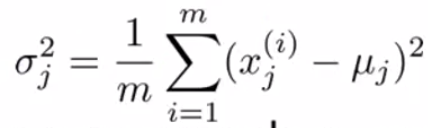
* Say we have an unlabeled training set of m examples + each is a feature in Rn
* To address anomaly detection, we model p(x) from this dataset to figure out the high + low probability features.
* X is a vector + model p(x) as: p(x1)\*p(x2)\* … \*p(x(n)),
* To model each of these terms, p(x1), p(x2), + so on, assume the feature is distributed as a Gaussian distribution w/ some mean μ1 + some variance δ^2 1,
* x2 has a different set of parameters, μ2 + δ^2



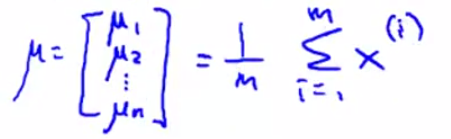
* This actually corresponds to an **independence assumption** on the values of features x1-x(n), but in practice, this algorithm works just fine whether or not these features are anywhere close to independent (even if independence assumption doesn't hold true this algorithm works just fine)
* Can write this more compactly + as a *product* from J = 1-N of p(x(j) parameterized by μ(j) + δ^2(j)
* 
* Estimating this distribution p(x) is sometimes called the **problem of density estimation**
* 



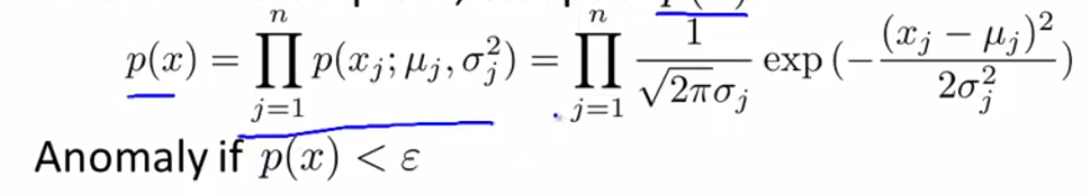
* So, putting everything together, here is our anomaly detection algorithm.
* 1) Choose/come up w/ features x(i) we think might be indicative of anomalous examples.
* Try to come up w/ features for what an anomalous example might look like
* Unusual user in a system that may be doing fraudulent things, or something funny/strange about an aircraft engine.
* Unusually large or small values.
* More generally, try to choose features that describe general properties of the things you're collecting data on.
* 2) Given a training set of m unlabeled examples X1-X(m), fit parameters, μ1-μ(n), + δ^2(1)-δ^2(n) of each jth feature

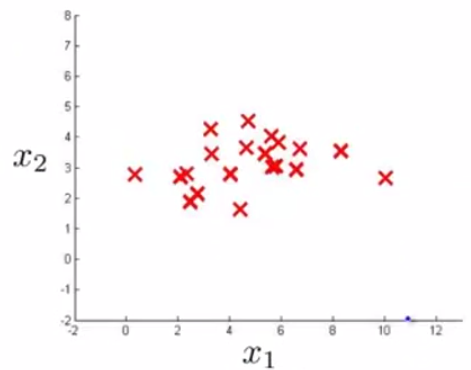
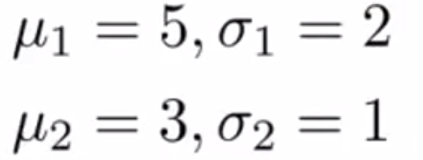
 

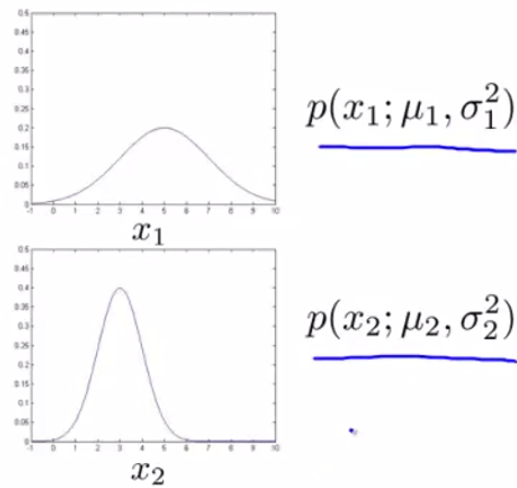
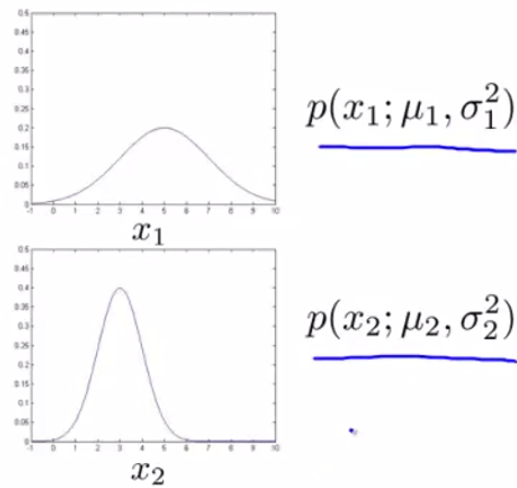
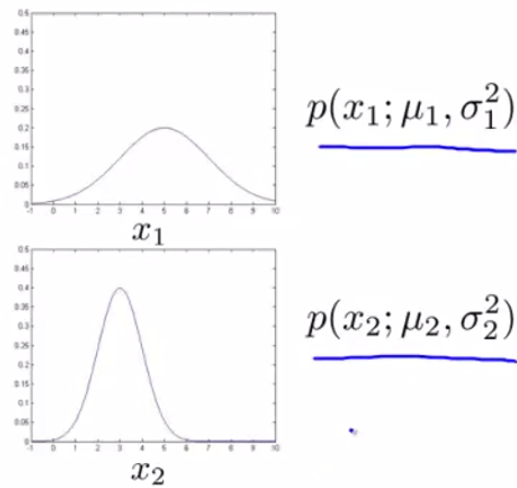
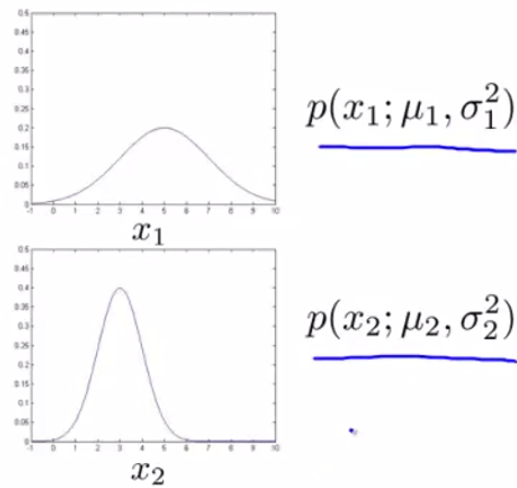
* Also possible to come up w/ vectorized versions of these
* Think of μ as a vector, a vectorized version of this set of parameters can be written



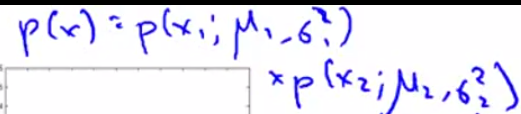
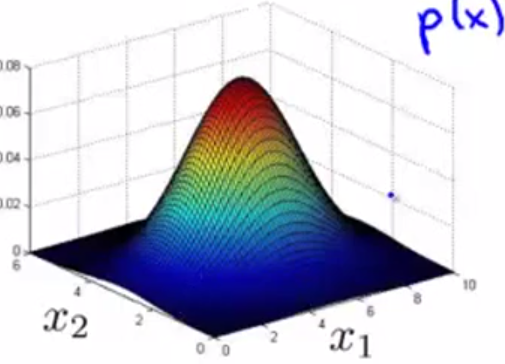
* This estimates μ for all values of n simultaneously.
* 3) When given a new example, compute p(x)/probability this new example is anomalous



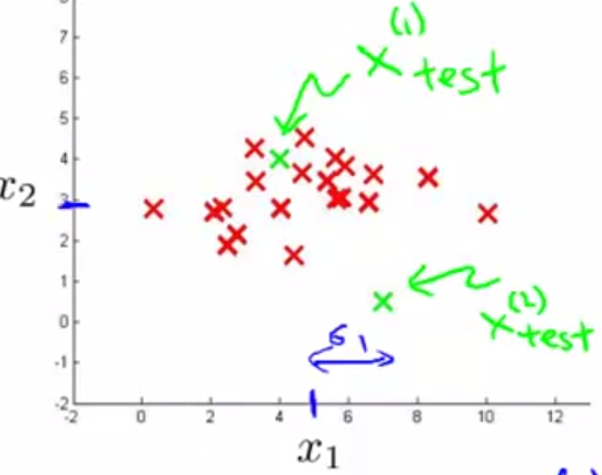
* Compute the formula for the Gaussian probability
* Example of an application of this method:
*  
* Feature x1 looks to have mean ~5 + SD of maybe 2 + x2 has an average ~3 + SD = ~1.
* In pictures, here’s p(x)1 parametrized by μ1 + δ^2(1) + p(x)2 is parametrized by μ2 + δ^2(2) + they look like these 2 distributions:

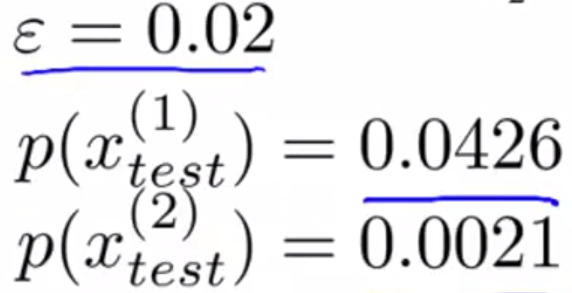
* Surface plot of p(x) (product of the 2 distributions above):



* Height = a particular point p(x) given particular x1 + x2 values of
* Now, w/ a new example, is it an anomaly or not?



* Set some value for Epsilon, say 0.02, + compute p(x1)\_test + p(x2)\_test to see if they’re >= epsilon
* p(x1)\_test has a pretty high probability, > epsilon, so we say X1\_test is NOT an anomaly.
* p(x2)\_test is a much smaller value < epsilon + so we say it is indeed an anomaly,



* What this is really saying is that, you look at the 3D-surface plot, all values of x1 + x2 that have a height high on the surface correspond to a non-anomalous example
* Whereas all points on the outer rim of the bottom of the hill have very low probability, so we are going to flag those points as anomalous
* It's defines some region such that everything outside it is flagged as anomalous

