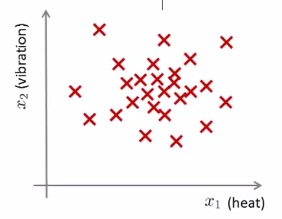
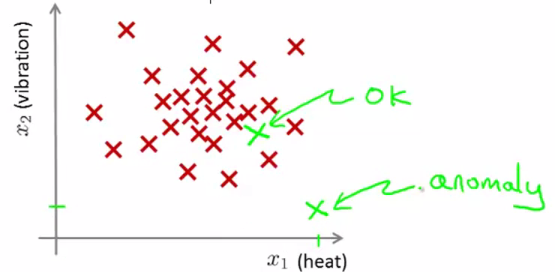
***Density Estimation***

**I. PROBLEM MOTIVATION**

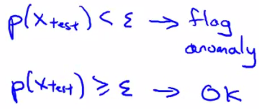
* **Anomaly Detection** = a reasonably commonly used type of ML.
* **1** interesting aspect: it's mainly for unsupervised problems, but w/ some aspects that are very similar supervised learning problems.
* Ex: Manufacturer of aircraft engines + as engines roll off the assembly line, you're doing QA + 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



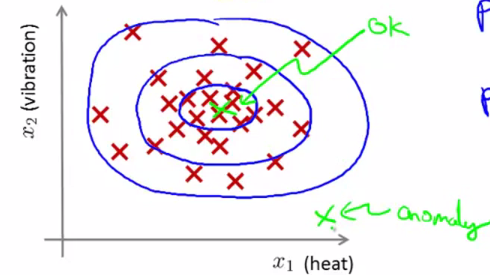
* The next day, a new aircraft engine rolls off the assembly line w/ some set of features X(test)
* Anomaly detection wants to know if this aircraft engine is **anomalous** in any way 🡪 should it undergo further testing or is it okay to ship it to a customer w/out further testing.

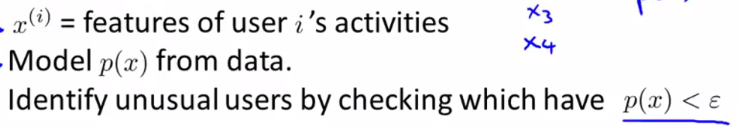


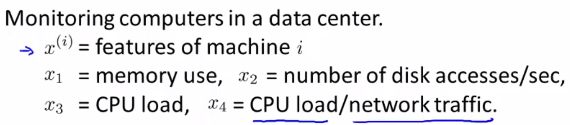
* More formally: In anomaly detection, we're give some data set’s X1-Xm examples (usually assuming these examples are normal/non-anomalous) + we want an algorithm to tell us if some new example Xtest is anomalous or not
* Given this unlabeled training set, we're going to build a model for p(x) = probability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

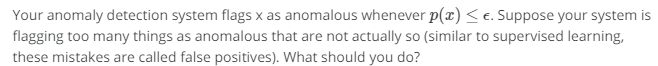


* On a plot, hopefully points lying somewhere in the middle have a pretty high p(x), + as we move out, p(x) lowers, but up to a point until they become anomalies



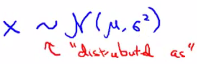
* Applications of anomaly detection:
* Fraud Detection (most common)
* If you have many users, + if each user does different activities, you can compute features of the different user activities + what build a model to check the probability of different users behaving different ways.
* Or to “check the 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. I
* 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 your 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**

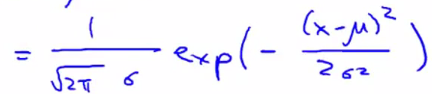
* **Gaussian distribution** = the normal distribution
* Say x is a real-valued random variable **Ɍ**
* If the probability distribution of x is Gaussian w/ mean μ + variance δ^2, write x ~ N(μ, δ^2)



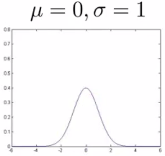
* Tilde = “distributed as.”
* N() = normal since Gaussian + normal are synonyms.
* Gaussian distribution is parameterized by 2 parameters, a mean μ + a variance δ^2
* Center of its bell-shaped curve plot = μ + the width is, δ (1 SD)
* This plot specifies the probability of x taking on different values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



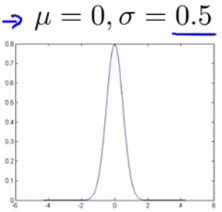
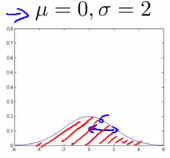
* probability of X is parameterized by the 2 parameters μ + δ squared
* Formula for the Gaussian density



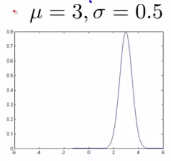
* This is formula for the bell-shaped curve
* The curve is what you get if you take a fixed value of μ + a fixed value of δ, + plot p(x)
* p(x) is plotted as a function of X for a fixed value of μ + of δ squared.
* Sometimes it's easier to think in terms of δ squared (variance) + sometimes is easier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

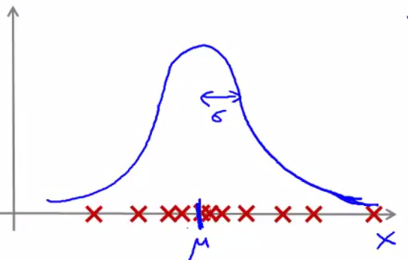
* B/c this is a probability distribution, the AUC for each curve must integrate to 1
* So thinner but much taller density = same area as 1st plot
* Gaussian distribution centered at μ = 3 shifts over the entire distribution.



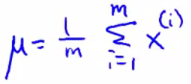
* **Parameter estimation problem**.
* Ex: We have a dataset of m examples + each example is a real number.



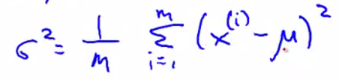
* Let's say I suspect that these examples X(i) came from a Gaussian distribution: X(i) ~ N(μ, δ^2), but I don't know what the parameters are.
* Given my data set, I want to estimate the values of μ + δ squared.

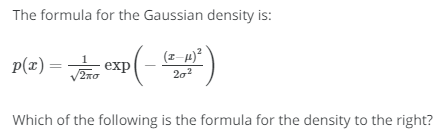
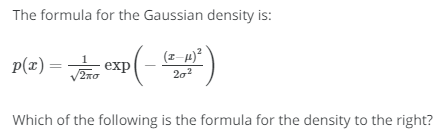
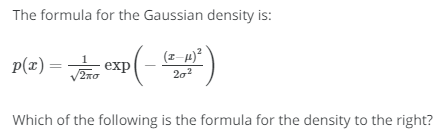


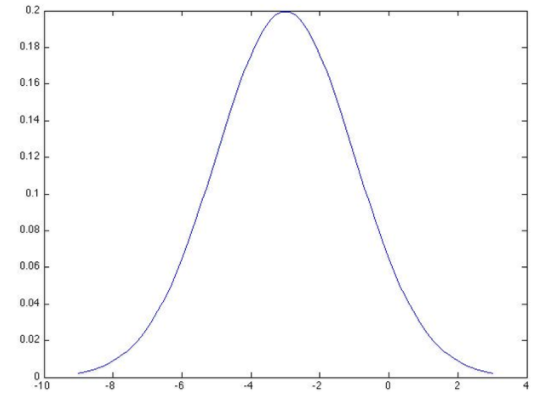
* Maybe the above might be roughly the Gaussian distribution it came from
* Seems like a reasonable fit to the data b/c it looks like data has a very high probability of being in the central region, + a lower probability of being further out
* Standard formulas for estimating the parameters μ + δ squared: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