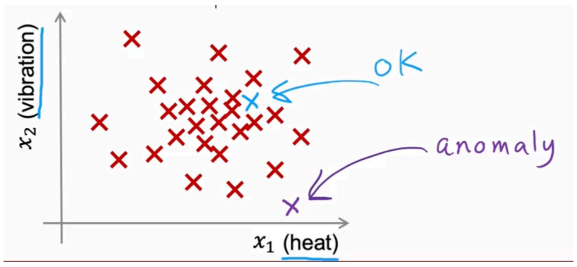
Anomaly detection

- **Anomaly detection** is an unsupervised learning algorithm used to find unusual or abnormal data points (anomalies) in a dataset.
- It is often used when you have lots of examples of "normal" behavior, but very few or no examples of "anomalous" behavior.
- Common applications: manufacturing (detecting faulty parts), fraud detection, monitoring computer systems, etc

Finding unusual events

Anomaly detection example

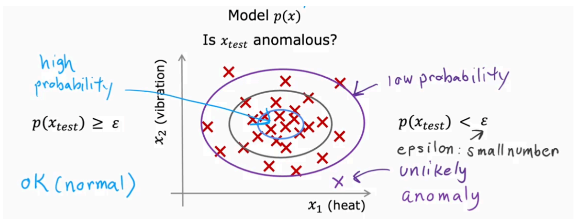
- **Features:** Each engine is measured for things like **heat generated** (x_1) and **vibration intensity** (x_2).
- **Plot:** Each **red X** is an engine.
  - The **blue X** (in the middle) is a new engine that looks similar to others (OK).
  - The purple **X** (far away) is very different (anomaly) and should be inspected.
- **Dataset:** Collect these features for many engines (most are normal).
  - $x^{(1)}, \ldots, x^{(m)}$



Density Estimation

Use a model to estimate the probability of seeing a data point with certain features (heat, vibration).

- **High probability area:** Most normal engines fall in the center (high density, high probability).



- **Low probability area:** Engines far from the center are rare (low probability).

- **Rule:** If the probability of a new engine's features is very low (below a threshold called epsilon, ϵ), flag it as an anomaly.

$$\text{anomaly} = p(x_{test}) < \epsilon$$

Another example

Fraud detection

- $x^{(i)}$ = features of user i's activities
- Model $p(x)$ from data.
- Identify unusual users by checking which have $p(x) < \varepsilon$

Feature could be:

- how often log in?
- how many web pages visited?
- transactions?
- posts? typing speed?

⇒ If $p(x) < \varepsilon$, then we can perform additional checks to identify real fraud vs. false alarms.

Manufacturing:

- $x^{(i)}$ = features of product i

Feature could be:

- airplane engine
- circuit board
- smartphone
- ratios

Monitoring computers in a data center:

- $x^{(i)}$ = features of machine i
  - x_1 = memory use
  - x_2 = number of disk accesses/sec
  - x_3 = CPU load
  - x_4 = CPU load/network traffic

Gaussian (normal) distribution

Anomaly detection algorithm

Density Estimation

Training set: $\{\vec{x}^{(1)}, \vec{x}^{(2)}, \dots, \vec{x}^{(m)}\}$. Each training example $\vec{x}^{(i)}$ has n features.

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

> The core concept is to model our data using a probability distribution. More specifically, we define a joint probability density function (pdf) based on our features, treating them as random variables. A common choice is the normal distribution, characterized by parameters μ (mean) and σ^2 (variance), to represent the data. If we assume that the features—denoted as random variables $X_1, X_2, \dots, X_n$ —are independent, we can express the joint distribution accordingly:

$$f_{X_1,X_2,...,X_n}(x_1,x_2,\dots,x_n) = f_{X_1}(x_1) f_{X_2}(x_2)\dots f_{X_n}(x_n)$$

We can define our model as the joint probability density function:

$$p(\vec{x}) = p(x_1;\mu_1,\sigma_1^2)\,p(x_2;\mu_2,\sigma_2^2)\;\dots\;p(x_n;\mu_n,\sigma_n^2) = \prod_{j=1}^{n} p(x_j;\mu_j,\sigma_j^2)$$

Put it all together

1. Choose n features x_i that might be indicative of anomalous examples.
2. Fit parameters $\mu_1, \mu_2, \dots \mu_n$ and $\sigma_1^2, \sigma_2^2, \dots, \sigma_n^2$:

$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)} \qquad \sigma_j^2 = \frac{1}{m}\sum_{i=1}^{m}\left(x_j^{(i)} - \mu_j\right)^2$$

or **Vectorized formula:**

$$\vec{\mu} = \frac{1}{m}\sum_{i=1}^{m}\vec{x}^{(i)} \qquad \vec{\mu} = \begin{bmatrix}\mu_1\\ \mu_2\\ \vdots\\ \mu_n\end{bmatrix}$$

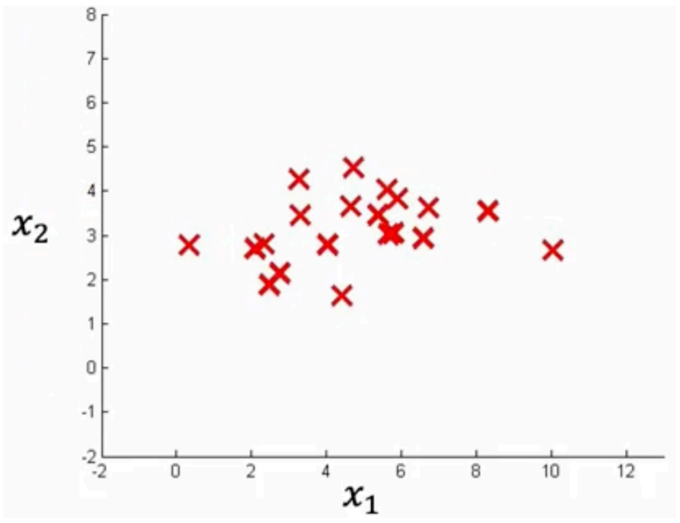
3. Given new example x, compute $p(x)$:

$$p(\vec{x}) = \prod_{j=1}^{n} p(x_j;\mu_j,\sigma_j^2) = \prod_{j=1}^{n}\left[\frac{1}{\sigma\sqrt{2\pi}}\cdot exp\left(-\frac{(x_j-\mu_j)^2}{2\sigma^2}\right)\right]$$

⇒ Anomaly if $p(\vec{x}) < \epsilon$

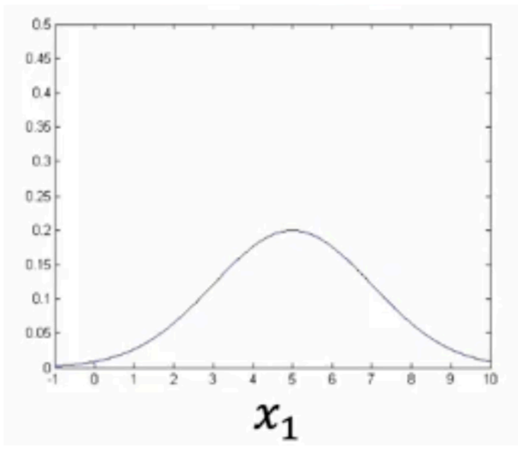
Example

Data of x_1, x_2



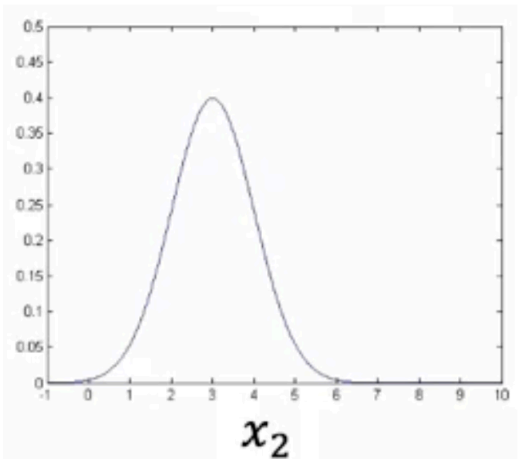
Suppose:

- Feature heat has $\mu_1 = 5, \sigma_1 = 2$



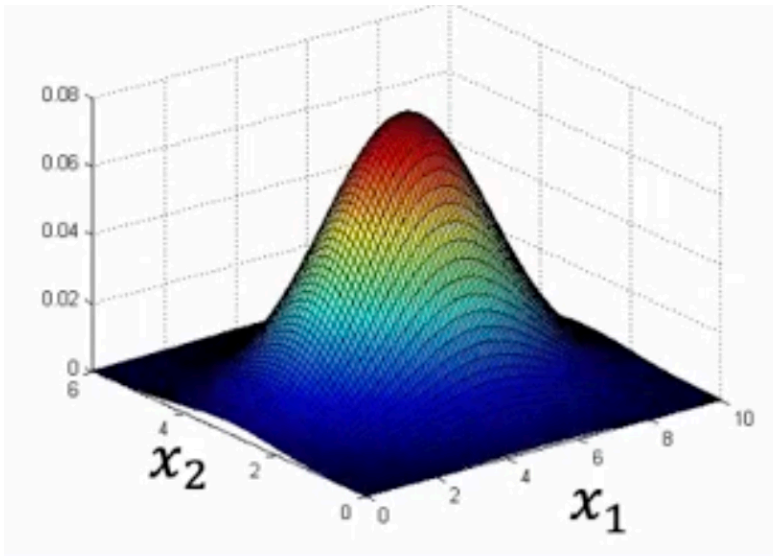
$p(x_1, \mu_1, \sigma_1^2)$

- Feature vibration has $\mu_2 = 3, \sigma_2 = 1.$

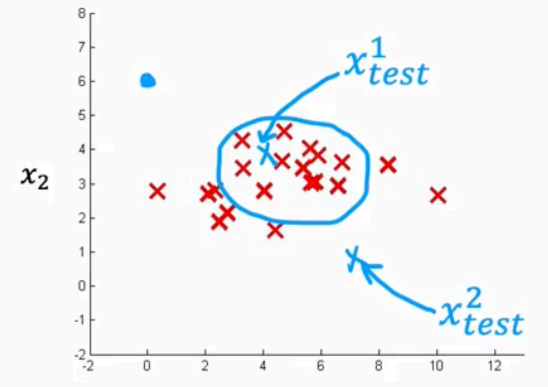


$p(x_2, \mu_2, \sigma_2^2)$

With the plot of $p(x)$ as follows:



Then $p(x)$ is computed by multiplying the Gaussians for heat and vibration. A new example with $heat = 8$ (far from mean) will get a very small probability $p(x)$, potentially flagged as anomaly.



With $\epsilon = 0.02$:

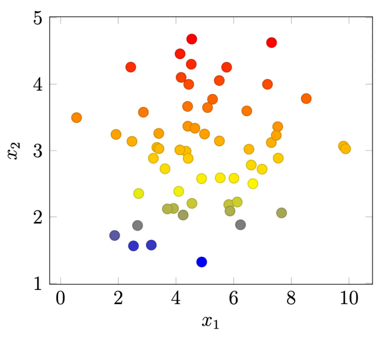
- $p(x_{test}^{(1)}) = 0.046 \rightarrow \text{OK}$
- $p(x_{test}^{(2)}) = 0.0021 \rightarrow \text{Anomaly}$

- **Choosing** ϵ
  - Can be chosen based on cross-validation or domain knowledge to balance false positives and false negatives.

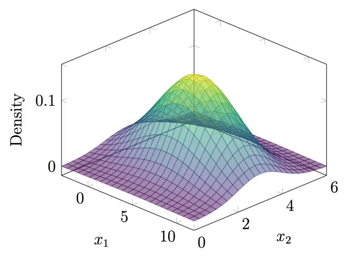
Another example

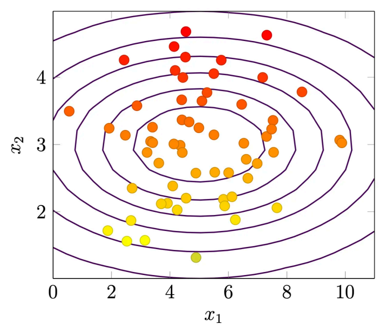
source: https://github.com/parsabsh/ML-specialization-notes/tree/main

2 **features**	**Fitting** with $\mu_1 = 5, \sigma_1 = 2$ and $\mu_2 =$	**Gaussian distribution**



$3, \sigma_2 = 1$





Developing and evaluating an anomaly detection system

The importance of real-number evaluation

When building a learning algorithm—like selecting the right features—it becomes much easier to make decisions if we can properly evaluate its performance.

Assume we have labeled data that includes both **anomalous** ($y = 1$) and **non-anomalous** or **normal** ($y = 0$) examples:

Training Set

$x^{(1)}, x^{(2)}, \ldots, x^{(m)}$ (all training data consists of **normal** (non-anomalous) examples.)

- So, for every training example: $y = 0$

Cross-Validation Set and Test Set

$$\begin{cases} (x_{\text{cv}}^{(1)}, y_{\text{cv}}^{(1)}), \ldots, (x_{\text{cv}}^{(m_{\text{cv}})}, y_{\text{cv}}^{(m_{\text{cv}})}) \\ (x_{\text{test}}^{(1)}, y_{\text{test}}^{(1)}), \ldots, (x_{\text{test}}^{(m_{\text{test}})}, y_{\text{test}}^{(m_{\text{test}})}) \end{cases}$$

- The cross-validation and test sets include **mostly normal examples** ($y = 0$)
- However, also contain **some anomalous examples** ($y = 1$) to evaluate the algorithm's ability to detect them.

Aircraft engines monitoring example

Context:

- **Total Data:** $10,000$ good (normal) engines ($y = 0$) + 20 flawed (anomalous) engines ($y = 1$).
- **Key Challenge:** Very few anomalies (typically $2 - 50$ known anomalies). This makes traditional supervised learning difficult for training.
- **Algorithm Type:** Primarily unsupervised learning (training set has no explicit labels for anomaly, only "normal").

Standard Data Split Strategy (Preferred if enough anomalies)

1. **Training Set (for `p(x)` model):**
   - $6,000$ good engines ($y = 0$).
   - Purpose: **Train the algorithm** by fitting Gaussian distributions to features. (Note: A few accidental anomalies here are acceptable).
2. **Cross-Validation (CV) Set (for tuning):**
   - $2,000$ good engines ($y = 0$).
   - 10 anomalous engines ($y = 1$).
   - Purpose: **Tune hyperparameters** like ϵ (threshold) and **refine features** (x_j).
   - How to tune:
     1. Compute $p(x)$ for all *CV* examples.
     2. Flag as anomaly if $p(x) < \epsilon$.

Alternative Data Split Strategy (When very few labeled anomalies, e.g., only 2)

Rationale: If you have extremely few known anomalies (e.g., 2), you might not have enough to create a separate test set distinct from the cross-validation set.

1. **Training Set (for `p(x)` model):**
   - $6,000$ good engines ($y = 0$).
2. **Cross-Validation (CV) Set (for tuning & final evaluation):**
   - $4,000$ remaining good engines ($y = 0$).
   - All 20 anomalous engines ($y = 1$).
   - Purpose: **Tune ϵ and features (x_j)**, similar to the standard CV set.
   - **Downside:**
     - **Higher risk of overfitting** to the CV set because

    3. Evaluate performance: Aim for high detection of 10 anomalies without misclassifying too many of the $2,000$ good engines.

3. **Test Set (for final evaluation):**
    - $2,000$ good engines ($y = 0$).
    - 10 anomalous engines ($y = 1$).
    - Purpose: **Unbiased evaluation** of the tuned algorithm's performance on unseen data.

there's no independent test set for final, unbiased evaluation.

- Performance on future, real-world data might not be as good as expected based on CV results.
- Still, often the best approach when labeled anomaly data is extremely scarce.

Algorithm Evaluation

- Fit a model $p(x)$ using the training set $x^{(1)}, x^{(2)}, \dots, x^{(m)}$.
- For each cross-validation or test example x, make a prediction using:

D

Possible Evaluation Metrics:

- True Positive, False Positive, False Negative, True Negative
- Precision / Recall
- F_1-score

Use cross-validation set to choose parameter ϵ

Anomaly detection vs. supervised learning

Anomaly detection

Supervised learning

Very small number of positive examples ($y = 1$) (0-20 is common). Large number of negative examples ($y = 0$)	Large number of positive and negative examples
Many different “**types**” of anomalies. Hard for any algorithm to learn from positive examples what the anomalies look like; **future anomalies may look nothing like any of the anomalous examples we have seen so far.**	Enough positive examples for the algorithm to get a sense of what positive examples are like; **future positive examples are likely to be similar to ones in the training set.**

Example

Anomaly detection

- Fraud detection
- Manufacturing - Finding new previously unseen defects (Aircraft engines)
- Aircraft engines defect detection
- Monitoring machines in a data center

Supervised learning

- Email spam classification
- Manufacturing - Finding known, previously seen defects (the scratches of the phone’s cover)
- Weather prediction
- Diseases classification

Choosing what features to use

Non-gaussian features

- Choosing good features very important for anomaly detection.
- More important than supervised learning because no labels.

- Algorithm learn only from data, so right features help a lot.

Example

⇒ Good if features look like **Gaussian** (normal) distribution. If not Gaussian, can try transform data to make it more **Gaussian**-like.

$x \leftarrow \log(x)$ $\quad$ $x \leftarrow \log(x + c)$ $\quad$ $x \leftarrow \sqrt{x}$ $\quad$ $x \leftarrow e^x$

TRICK: When we assign $\log(x) \rightarrow x$, it can lead to error because the possible minimum value of x could be 0 ⇒ We should add a little number (e.g, 0.0001) to x before calculating $\log(x)$.

`plt.hist(np.log(x+0.001),bins=50);`

Just as a reminder, whatever transformation you apply to the training set, remember to apply **the same transformation** to your **cross validation** and **test** set data as well.

Error analysis for anomaly detection

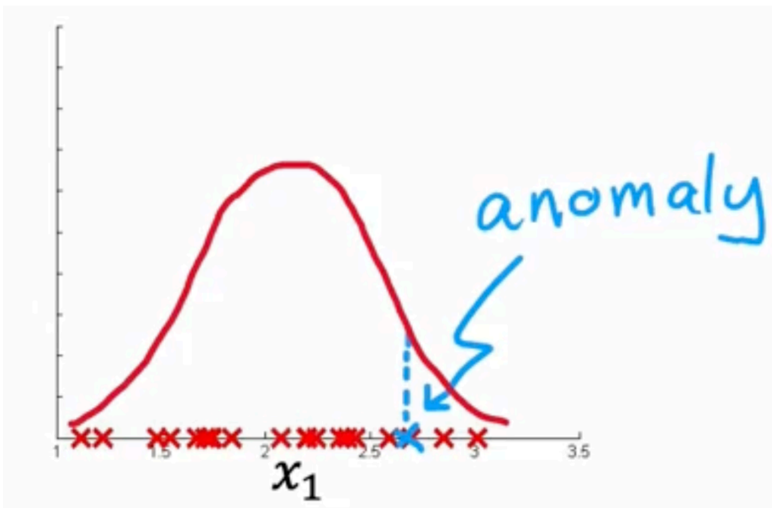
After training, if model not find anomalies well, need **check errors.**
⇒ ***Error analysis*** can help make new features.

What we want

1. $p(x)$ is large for normal examples. $p(x) \geq \epsilon$
2. $p(x)$ is small for anomalous examples. $p(x) < \epsilon$

Most common problem regarding error analysis

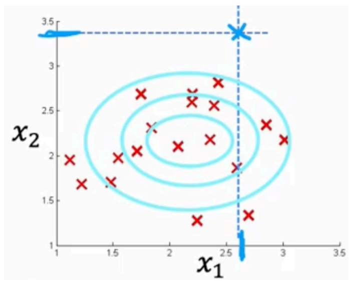
$p(x)$ is comparable for normal and anomalous examples (usually a **large value** for both).



New features can help tell anomaly and normal apart.

Example: With above plot we have x_1 : $\text{number of transactions}$. We can add more features x_2 : typing speed. It can help to solve the problem.

- The anomalous example (the **blue** one - X) stands out much more clearly.



Example: Monitoring computer in data center

Choose features that might take on unusually large or small values in the event of an anomaly.

- x_1= memory use of computer
- x_2 = number of disk accesses/sec
- x_3 = CPU load
- x_4 = network traffic

Problem: After training, the algorithm may flag certain instances as anomalies. For example, a computer might show a **high CPU load** but **low network traffic**, which is **unusual** for a data center that typically has high traffic.

We should create new feature that capture unusual patterns:

D

or:

$$x_6 = \frac{(\text{CPU load})^2}{\text{network traffic}}$$

Deciding feature choice based on $p(x)$.

- Large for normal examples;
- Becomes small for anomaly in the cross validation set

❓ **QUES:** You are building a system to detect if computers in a data center are malfunctioning. You have 10,000 data points of computers functioning well, and no data from computers malfunctioning. What type of algorithm should you use?

- Anomaly detection 👍
- Supervised learning

Explain: Creating an anomaly detection model does not require labeled data.

❓ **QUES:** You are building a system to detect if computers in a data center are malfunctioning. You have 10,000 data points of computers functioning well, and 10,000 data points of computers malfunctioning. What type of algorithm should you use?

- Anomaly detection
- Supervised learning 👍

Explain: You have a sufficient number of anomalous examples to build a supervised learning model.

❓ **QUES:** Say you have 5,000 examples of normal airplane engines, and 15 examples of anomalous engines. How would you use the 15 examples of anomalous engines to evaluate your anomaly detection algorithm?

- Put the data of anomalous engines (together with some normal engines) in the cross-validation and/or test sets to measure if the learned model can correctly detect anomalous engines.
- You cannot evaluate an anomaly detection algorithm because it is an unsupervised learning algorithm.
- Use it during training by fitting one Gaussian model to the normal engines, and a different Gaussian model to the anomalous engines.
- Because you have data of both normal and anomalous engines, don't use anomaly detection. Use supervised learning instead.

❓ **QUES:** Say you have 5,000 examples of normal airplane engines, and 15 examples of anomalous engines. How would you use the 15 examples of anomalous engines to evaluate your anomaly detection algorithm?

- Put the data of anomalous engines (together with some normal engines) in the cross-validation and/or test sets to measure if the learned model can correctly detect anomalous engines. 👍
- Because you have data of both normal and anomalous engines, don't use anomaly detection. Use supervised learning instead.
- Use it during training by fitting one Gaussian model to the normal engines, and a different Gaussian model to the anomalous engines.
- You cannot evaluate an anomaly detection algorithm because it is an unsupervised learning algorithm.

Explain: Anomalous examples are used to evaluate rather than train the model.

❓ **QUES:** Anomaly detection flags a new input xxx as an anomaly if $p(x) < \epsilon$. If we reduce the value of ϵ, what happens?

- The algorithm is more likely to classify new examples as an anomaly.
- The algorithm is less likely to classify new examples as an anomaly. 👍
- The algorithm is more likely to classify some examples as an anomaly, and less likely to classify some examples as an anomaly. It depends on the example x.
- The algorithm will automatically choose parameters μμmu and σσsigma to decrease $p(x)$ and compensate.

Explain: When ϵ is reduced, the probability of an event being classified as an anomaly is reduced.

❓ **QUES:** You are monitoring the temperature and vibration intensity on newly manufactured aircraft engines. You have measured 100 engines and fit the Gaussian model described in the video lectures to the data. The 100 examples and the resulting distributions are shown in the figure below.

The measurements on the latest engine you are testing have a temperature of 17.5 and a vibration intensity of 48. These are shown in magenta on the figure below. What is the probability of an engine having these two measurements?

- 17.5 + 48 = 65.5
- 0.0738 + 0.02288 = 0.0966
- 0.0738 ∗ 0.02288 = 0.00169 👍
- 17.5 ∗ 48 = 840

Explain: According to the model described in lecture, $p(A, B) = p(A) * p(B)$.