Detecção de Outliers

Neste exercício, você vai implementar um algoritmo de detecção de outliers e aplicá-lo para detectar falhas em servidores em uma rede.

Arquivos incluídos:

- det_anomalia.m - Script de suporte para o trabalho
- dadosrede.mat - Dataset para o primeiro exemplo
- dadosrede 2.mat - Dataset para o Segundo exemplo
- multivariateGaussian.m – Função auxiliar para calcular a função de densidade da Gaussiana multivariada no Octave
- visualizarAderencia.m – Plot 2D de uma distribuição Gaussiana e um conjunto de dados
- estimarGaussiana.m - Estima os parâmetros de uma distribuição Gaussiana com uma matriz de covariância diagonal (essa função deve ser programada por você)
- selecionarThreshold.m – Encontrar o threshold para detecção de anomalia (essa função deve ser programada por você)

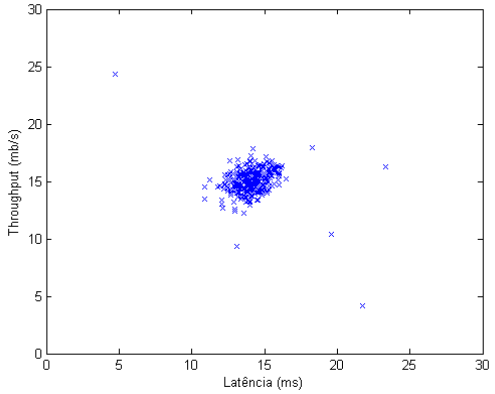
O script det_anomalia carrega os dados e chama as funções que você programou. Plota os gráficos e imprime mensagens que ajudam você ver se está no caminho certo.

As features desse problema medem o throughput (mb/s) e a latência (ms) das respostas de cada servidor. No primeiro arquivo (dadosrede), foram coletados m=307 exemplos do comportamento dos servidores colocadas em um dataset aqui representados por $\{x^{(1)}; \ldots ; x^{(m)}\}$. A grande maioria desses exemplos correspondem a comportamento normal dos servidores (não anômalo), mas eles podem incluir algumas anomalias.

Você deve usar um modelo Gaussiano para detectar os exemplos anômalos no primeiro dataset. Esse primeiro dataset é bidimensional, o que permite visualizar os dados e os resultados, em um plot 2D. Primeiro você vai ajustar uma distribuição Gaussiana e então encontrar valores com baixa probabilidade e que serão classificados como anomalias. Depois você vai aplicar o algoritmo de detecção de anomalia para um dataset com maior número de dimensões.

Parte 1

A primeira parte do script plota o primeiro dataset, mostrando a seguinte figura.



Para realizar a detecção de anomalias é necessário ajustar um modelo à distribuição dos dados. Dado o conjunto de treinamento $\{x^{(1)}; \ldots ; x^{(m)}\}$ onde $x^{(i)} \in R^n$, é necessário estimar a Gaussiana para cada feature x_i. Para cada feature i = 1, ..., n, é necessário encontrar os parâmetros μ_i e σ_i^2 da feature i. A função de densidade da distribuição Gaussiana com parâmetros μ e σ^2, é dada por:

$$f(x, \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2} \right]$$

onde μ e σ^2 correspondem, respectivamente, à média e à variância. Para estimar os parâmetros μ e σ^2, usamos as seguintes equações:

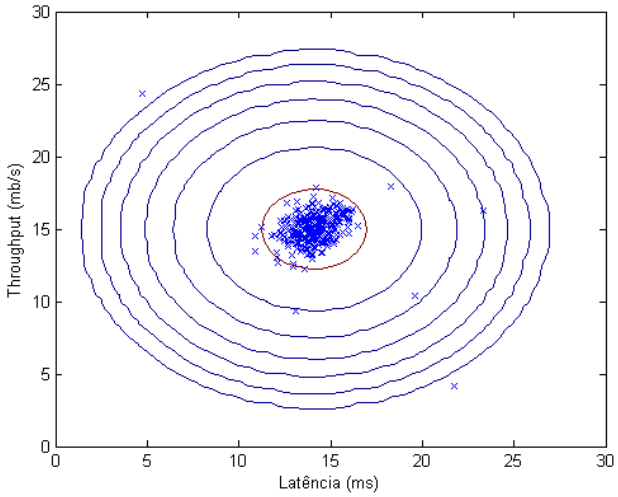
$$\mu = \frac{\sum_{i=1}^{m} x_i}{m} \qquad \text{e} \qquad \sigma^2 = \frac{\sum_{i=1}^{m} (x_i - \bar{X})^2}{m}$$

Você deve completar o código da função estimarGaussiana.m. Essa função recebe como argumento a matriz de dados X, e deve retornar o vetor n-dimensional mu que contém as médias estimadas de todas as n features do problema, e o vetor n-dimensional sigma2 que contém as variâncias estimadas de todas as n features do problema.

Observação: cuidado que a função var do MatLab pode retornar qualquer uma das seguintes fórmulas, dependendo de como é chamada. Você deve usar a segunda fórmula.

$$\sigma^2 = \frac{\sum_{i=1}^{m} (x_i - \bar{X})^2}{m-1} \qquad \text{ou} \qquad \sigma^2 = \frac{\sum_{i=1}^{m} (x_i - \bar{X})^2}{m}$$

Depois de completar o código da função estimarGaussiana.m, o script vai visualizar os contornos da distribuição Gaussina em torno dos dados, apresentando a seguinte figura:



É possível observar que a maior parte dos exemplos se encontra nas regiões com probabilidades mais altas, enquanto que es exemplos anômalos encontram-se nas regiões com probabilidades mais baixas.

Parte 2

Agora que o modelo foi calculado (mu e sigma2), é possível investigar quais exemplos possuem alta probabilidade de ocorrência, e quais são potenciais anomalias. Os exemplos com baixa probabilidade são potencialmente anomalias. Para identificá-los, vamos escolher um limiar (threshold) usando um conjunto com dados de validação cruzada (cross-validation).

Nessa parte do trabalho você vai implementar um algoritmo para selecionar o threshold, usando a métrica F1, calculada a partir dos dados que estão no arquivo cross-validation. Para isso você deve completar a função selecionarThreshold.m. Os dados no arquivo de cross-validation estão estruturados do seguinte modo: $\{(x_{cv}^{(1)}, y_{cv}^{(1)}), ..., (x_{cv}^{(mv)}, y_{cv}^{(mv)})\}$, incluindo os valores das features, que estão registrados nos nos vetores $x_{cv}^{(1)}, ..., x_{cv}^{(mv)}$, e um conjunto de rótulos $y_{cv^{(1)}}$; ... ; $y_{cv^{(m_{cv})}}$. A cada exemplo corresponde um rótulo y, do seguinte modo: quando y = 1, o exemplo corresponde a uma anomalia, e quando y = 0, o exemplo corresponde a uma situação normal (não anomalia).

Para cada exemplo no arquivo de cross-validation, calcula-se $p(x_{cv^{(i)}})$. O vetor com todas essas probabilidades pval = $p(x_{cv^{(1)}})$; …; $p(x_{cv_{(m_{cv})}})$ é passado como argumento para a função selecionarThreshold.m. Os rótulos correspondentes $y_{cv^{(1)}}$; ... ; $y_{cv^{(m_{cv})}}$ são passados como argumento para a mesma função no vetor yval.

A função selecionaThreshold.m deve retornar dois valores; o primeiro é o threshold escolhido ε. Se um exemplo x tem probabilidade p(x) < ε, então ele é considerado uma anomalia. A função também deve retornar o métrica F1, que mede o quão bem o algoritmo está classificando as verdadeiras anomalias. Para vários valores diferentes de ε, deve-se calcular a métrica F1, usando-se duas outras métricas, precisão (prec) e recall (rec):

$$F_1 = \frac{2 \cdot prec. \cdot rec}{prec + rec}$$

Prec e recall são calculados por:

$$prec = \frac{tp}{tp + fp} \qquad\qquad rec = \frac{tp}{tp + fn}$$

Onde

- tp é a quantidade de positivos verdadeiros: o rótulo diz que um exemplo é uma anomalia, e o algoritmo classificou o exemplo como uma anomalia.
- fp é a quantidade de falsos positivos: o rótulo diz que um exemplo não é uma anomalia, e o algoritmo classificou (erroneamente) o exemplo como uma anomalia.
- fn é a quantidade de falsos negativos: o rótulo diz que um exemplo é uma anomalia, e o algoritmo classificou o exemplo como uma não anomalia.

No código fornecido em selecionarThreshold.m já existe um loop para tentar vários valores diferentes de ε, e selecionar o melhor com base na métrica F_1. Você deve completar o código. Com a execução do script, você deve obter um valor de epsilon de

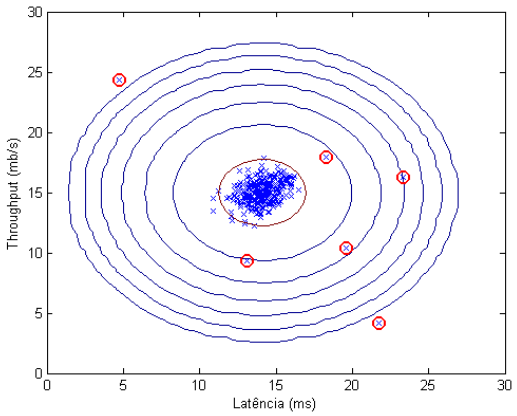
aproximadamente 8.99e-05. O script vai imprimir o valor de épsilon obtido pela sua implementação e o valor esperado de épsilon:

Melhor epsilon encontrado usando cross-validation: 8.986095e-005

Melhor F1 obtido com os dados de cross-validation: 0.800000

(o valor de epsilon deve ser aproximadamente 8.99e-05)

Uma vez codificada a função selecionarThreshold.m, o próximo passo do script é executar o seu código de detecção de anomalia e marcar com um círculo as anomalias em um gráfico.



A última parte do script executa o algoritmo implementado em um dataset mais realista, onde cada exemplo é descrito por 11 features, capturando mais propriedades dos servidores. O script vai usar o seu código para estimar os parâmetros da Gaussiana (μ e σ^2), para avaliar as probabilidades dos dados X do conjunto de treinamento, e para avaliar usando o conjunto de cross-validation Xval. Também vai usar a função selecionaThreshold para encontrar o melhor threshold epsilon. Você deve ver uma mensagem informando o seu valor de épsilon, e o valor esperado de aproximadamente 1.38e-18, com 117 anomalias detectadas.

Melhor epsilon encontrado usando cross-validation: 1.377229e-018
Melhor F1 obtido com os dados de cross-validation: 0.615385
Anomalias encontradas: 117
(o valor de epsilon deve ser aproximadamente 1.38e-18)

Parte 3
O script det_anomalia1 chama a função estimarGaussiana1. Observe que ele não seleciona a diagonal da matriz da matriz retornada por estimarGaussiana1. A diferença é que ele está usando a função com a matriz de covariância completa, que pode ser calculada pelo comando cov(X).

Modifique a função estimarGaussiana usando o comando cov(X), salve como estimarGaussiana1 e execute o script det_anomalia1

Colocar no blackboard as funções que você modificou.