

Sebastian Raschka
last modified: 03/31/2014

Problem Category

- Statistical Pattern Recognition
 - Supervised Learning
 - Parametric Learning
 - Bayes Decision Theory
 - Univariate data
 - 2-class problem
 - equal variances
 - equal priors
 - Gaussian model (2 parameters)
 - No Risk function
-

Sections

- [Given information](#)
 - [Deriving the decision boundary](#)
 - [Plotting the class conditional densities, posterior probabilities, and decision boundary](#)
 - [Classifying some random example data](#)
 - [Calculating the empirical error rate](#)
-

Given information:

[\[back to top\]](#)

model: continuous univariate normal (Gaussian) density

$$p(x|\omega_j) \sim N(\mu|\sigma^2)$$

$$p(x|\omega_j) \sim \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[-\frac{1}{2} \left(\frac{x-\mu}{\sigma} \right)^2 \right]$$

parameters:

$$P(\omega_1) = P(\omega_2) = 0.5$$

$$\sigma_1^2 = \sigma_2^2 = 1$$

$$\mu_1 = 4, \quad \mu_2 = 10$$

Deriving the decision boundary

[\[back to top\]](#)

Bayes' Rule:

$$P(\omega_j|x) = \frac{p(x|\omega_j) * P(\omega_j)}{p(x)}$$

Bayes' Decision Rule:

Decide ω_1 if $P(\omega_1|x) > P(\omega_2|x)$ else decide ω_2 .

$$\Rightarrow \frac{p(x|\omega_1) * P(\omega_1)}{p(x)} > \frac{p(x|\omega_2) * P(\omega_2)}{p(x)}$$

We can drop $p(x)$ since it is just a scale factor.

$$\Rightarrow P(x|\omega_1) * P(\omega_1) > p(x|\omega_2) * P(\omega_2)$$

$$\Rightarrow \frac{p(x|\omega_1)}{p(x|\omega_2)} > \frac{P(\omega_2)}{P(\omega_1)}$$

$$\Rightarrow \frac{p(x|\omega_1)}{p(x|\omega_2)} > \frac{0.5}{0.5}$$

$$\Rightarrow \frac{p(x|\omega_1)}{p(x|\omega_2)} > 1$$

$$\Rightarrow \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp \left[-\frac{1}{2} \left(\frac{x-\mu_1}{\sigma_1} \right)^2 \right] > \frac{1}{\sqrt{2\pi\sigma_2^2}} \exp \left[-\frac{1}{2} \left(\frac{x-\mu_2}{\sigma_2} \right)^2 \right]$$

Since we have equal variances, we can drop the first term completely.

$$\Rightarrow \exp \left[-\frac{1}{2} \left(\frac{x-\mu_1}{\sigma_1} \right)^2 \right] > \exp \left[-\frac{1}{2} \left(\frac{x-\mu_2}{\sigma_2} \right)^2 \right] \quad \left| \ln, \quad \mu_1 = 4, \quad \mu_2 = 10, \quad \sigma = 1 \right.$$

$$\Rightarrow -\frac{1}{2} (x-4)^2 > -\frac{1}{2} (x-10)^2 \quad \left| \times (-2) \right.$$

$$\Rightarrow (x-4)^2 < (x-10)^2$$

$$\Rightarrow x^2 - 8x + 16 < x^2 - 20x + 100$$

$$\Rightarrow 12x < 84$$

$$\Rightarrow x < 7$$

Plotting the class conditional densities, posterior probabilities, and decision boundary

[\[back to top\]](#)

```
In [1]: %pylab inline

import numpy as np
from matplotlib import pyplot as plt

def pdf(x, mu, sigma):
    """
    Calculates the normal distribution's probability density
    function (PDF).

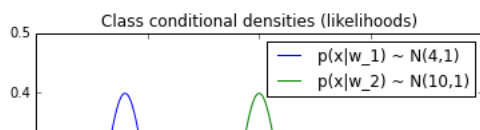
    """
    term1 = 1.0 / ( math.sqrt(2*np.pi) * sigma )
    term2 = np.exp( -0.5 * ( (x-mu)/sigma )**2 )
    return term1 * term2

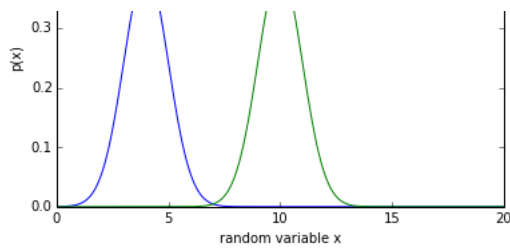
# generating some sample data
x = np.arange(0, 100, 0.05)

# probability density functions
pdf1 = pdf(x, mu=4, sigma=1)
pdf2 = pdf(x, mu=10, sigma=1)

# Class conditional densities (likelihoods)
plt.plot(x, pdf1)
plt.plot(x, pdf2)
plt.title('Class conditional densities (likelihoods)')
plt.ylabel('p(x)')
plt.xlabel('random variable x')
plt.legend(['p(x|w_1) ~ N(4,1)', 'p(x|w_2) ~ N(10,1)'], loc='upper right')
plt.ylim([0,0.5])
plt.xlim([0,20])
plt.show()
```

Populating the interactive namespace from numpy and matplotlib



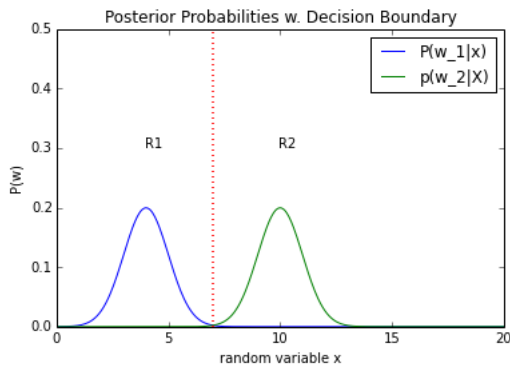


```
In [4]: def posterior(likelihood, prior):
        """
        Calculates the posterior probability (after Bayes Rule) without
        the scale factor p(x) (=evidence).

        """
        return likelihood * prior

# probability density functions
posterior1 = posterior(pdf(x, mu=4, sigma=1), 0.5)
posterior2 = posterior(pdf(x, mu=10, sigma=1), 0.5)

# Class conditional densities (likelihoods)
plt.plot(x, posterior1)
plt.plot(x, posterior2)
plt.title('Posterior Probabilities w. Decision Boundary')
plt.ylabel('P(w)')
plt.xlabel('random variable x')
plt.legend(['P(w_1|x)', 'p(w_2|X)'], loc='upper right')
plt.ylim([0,0.5])
plt.xlim([0,20])
plt.axvline(7, color='r', alpha=0.8, linestyle=':', linewidth=2)
plt.annotate('R1', xy=(4, 0.3), xytext=(4, 0.3))
plt.annotate('R2', xy=(10, 0.3), xytext=(10, 0.3))
plt.show()
```



Classifying some random example data

[\[back to top\]](#)

Note on generating univariate random data from a Normal Distribution

We can generate random samples drawn from a Normal distribution via the `np.random.randn()` function. Its default is a standard Normal distribution with $\mu = 0$ and $\sigma^2 = 1$. In order to draw random data from $N(\mu, \sigma^2)$, we use

$$\text{sigma} * \text{np.random.randn}(...) + \text{mu}$$

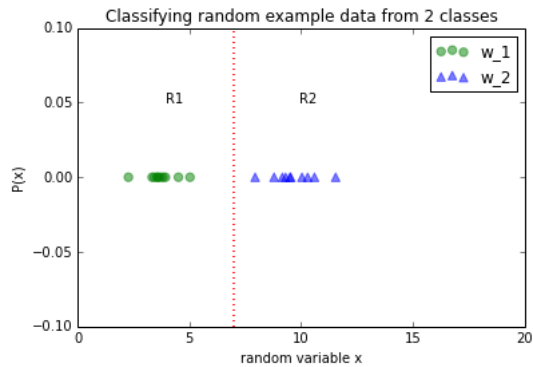
```
In [17]: # Parameters
mu_1 = 4
mu_2 = 10
sigma_1_sqr = 1
sigma_2_sqr = 1

# Generating 10 random samples drawn from a Normal Distribution for class 1 & 2
x1_samples = sigma_1_sqr**0.5 * np.random.randn(10) + mu_1
x2_samples = sigma_1_sqr**0.5 * np.random.randn(10) + mu_2
y = [0 for i in range(10)]

# Plotting sample data with a decision boundary

plt.scatter(x1_samples, y, marker='o', color='green', s=40, alpha=0.5)
plt.scatter(x2_samples, y, marker='^', color='blue', s=40, alpha=0.5)
plt.title('Classifying random example data from 2 classes')
```

```
plt.ylabel('P(x)')
plt.xlabel('random variable x')
plt.legend(['w_1', 'w_2'], loc='upper right')
plt.ylim([-0.1,0.1])
plt.xlim([0,20])
plt.axvline(7, color='r', alpha=0.8, linestyle=':', linewidth=2)
plt.annotate('R1', xy=(4, 0.05), xytext=(4, 0.05))
plt.annotate('R2', xy=(10, 0.05), xytext=(10, 0.05))
plt.show()
```



Calculating the empirical error rate

[\[back to top\]](#)

```
In [24]: w1_as_w2, w2_as_w1 = 0, 0
for x1,x2 in zip(x1_samples, x2_samples):
    if x1 >= 7:
        w1_as_w2 += 1
    if x2 < 7:
        w2_as_w1 += 1

emp_err = (w1_as_w2 + w2_as_w1) / float(len(x1_samples) + len(x2_samples))

print('Empirical Error: {}'.format(emp_err * 100))

Empirical Error: 0.0%
```

In []: