G. COWAN
15.11.21

SDA Discussion notes week 7

Prob. Sheet 4 solutions:

1a) $X_N = \sum_{i=1}^{N} r_i$, r_i indep. & $\sim U[0,1]$

$$E[X_N] \equiv \mu_N = E\left[\sum_{i=1}^{N} r_i\right]$$

$$= \sum_{i=1}^{N} E[r_i] = \sum_{i=1}^{N} \frac{1}{2} = \frac{N}{2}$$

($E[r_i] = \frac{1}{2}$ from lect. notes)

$$V[X_N] = V\left[\sum_{i=1}^{N} r_i\right]$$

$$= \sum_{i=1}^{N} V[r_i] \quad \text{(since } r_i \text{ indep.)}$$

($V[r_i] = \frac{1}{12}$ from lecture notes)

$$= \frac{N}{12}$$

$$\Rightarrow \sigma_N = \sqrt{\frac{N}{12}}$$

$\Rightarrow$ By construction $y_N = \dfrac{X_N - \mu_N}{\sigma_N} = \sqrt{\dfrac{12}{N}}\left(\sum_{i=1}^{N} r_i - \dfrac{N}{2}\right)$

has $E[y_N] = 0$, $V[y_N] = 1$ ("standardized").

2) $f(x) = 4x^3 \quad , \quad 0 \le x \le 1$

2a) Transformation method

First find cumulative distribution

$$F(x) = \int_0^x 4x'^3 \, dx' = x^4$$

Set $F(x) = x^4 = r$

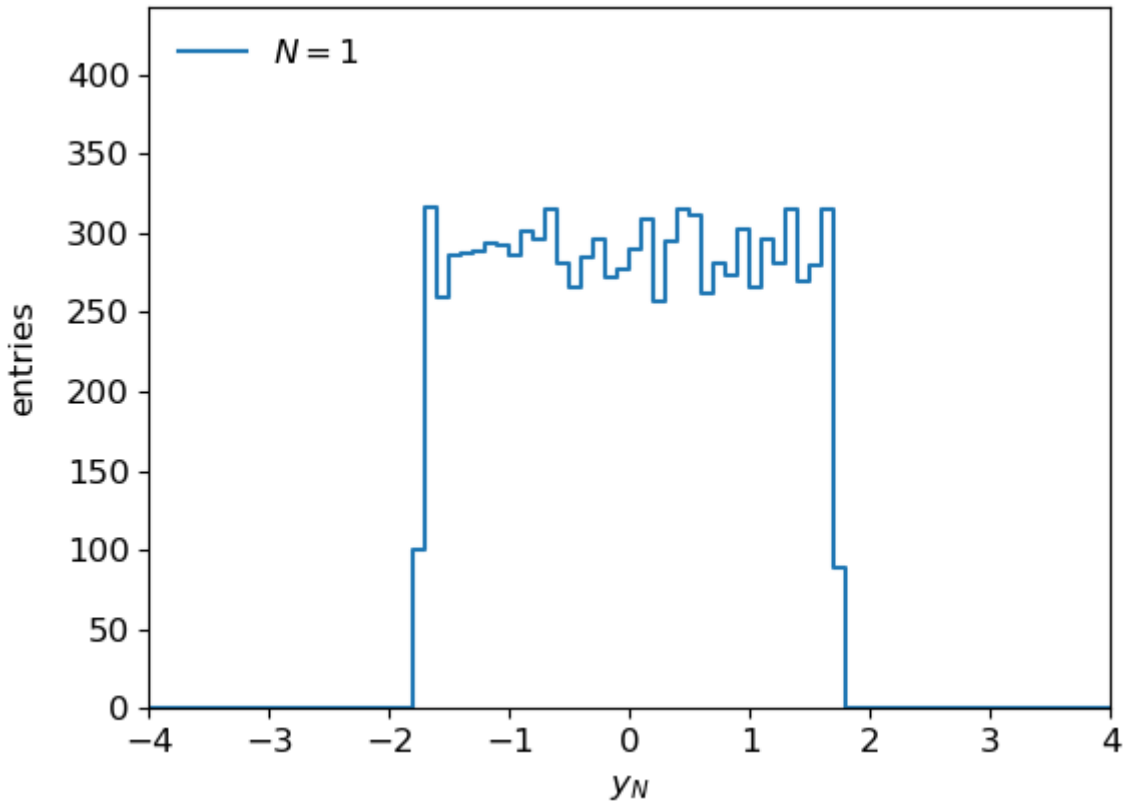
$$\Rightarrow \quad x(r) = r^{1/4}$$

In [1]:
```
# simpleMC.py -- simple Monte Carlo program to make histogram of uniformly
# distributed random values and plot
# G. Cowan, RHUL Physics, November 2021

import matplotlib
import matplotlib.pyplot as plt
import numpy as np
```

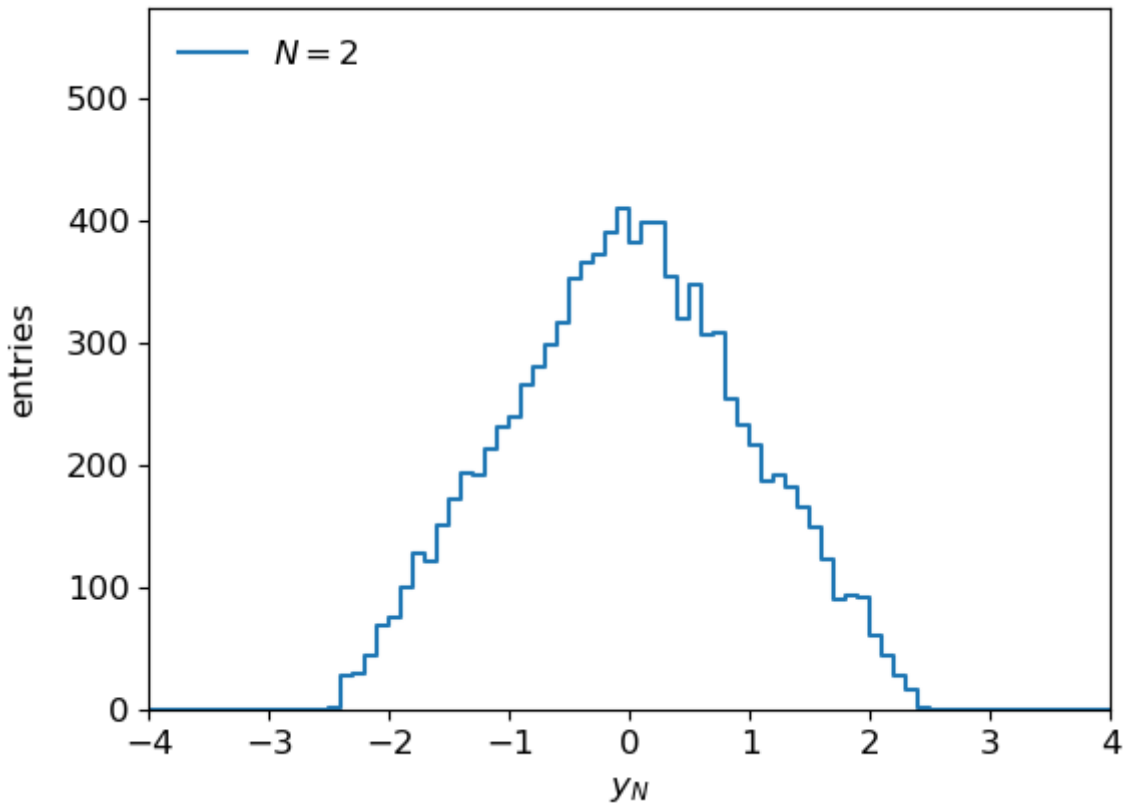
In [2]:
```
# Solution to problem sheet 4

# First make a simple routine to plot histograms
def plotHist (hist, bin_edges, xLabel, yLabel, plotLabel):
    matplotlib.rcParams.update({'font.size':12})     # set all font sizes
    xMin = bin_edges[0]
    xMax = bin_edges[len(bin_edges)-1]
    yMin = 0.
    yMax = np.max(hist)*1.4
    binLo, binHi = bin_edges[:-1], bin_edges[1:]
    xPlot = np.array([binLo, binHi]).T.flatten()
    yPlot = np.array([hist, hist]).T.flatten()
    fig, ax = plt.subplots(1,1)
    plt.gcf().subplots_adjust(bottom=0.15)
    plt.gcf().subplots_adjust(left=0.15)
    ax.set_xlim((xMin, xMax))
    ax.set_ylim((yMin, yMax))
    plt.xlabel(xLabel, labelpad=5)
    plt.ylabel(yLabel, labelpad=10)
    plt.plot(xPlot, yPlot, label=plotLabel)
    ax.legend(loc='upper left', frameon=False)
    plt.show()
```

In [3]:
```
# The function for yN returns an array of values
def yArr(N, numVal):
    rSum = np.zeros(numVal)
    for i in range (N):
        rSum += np.random.uniform(0., 1., numVal)
    return np.sqrt(12./N)*(rSum - N/2.)
```

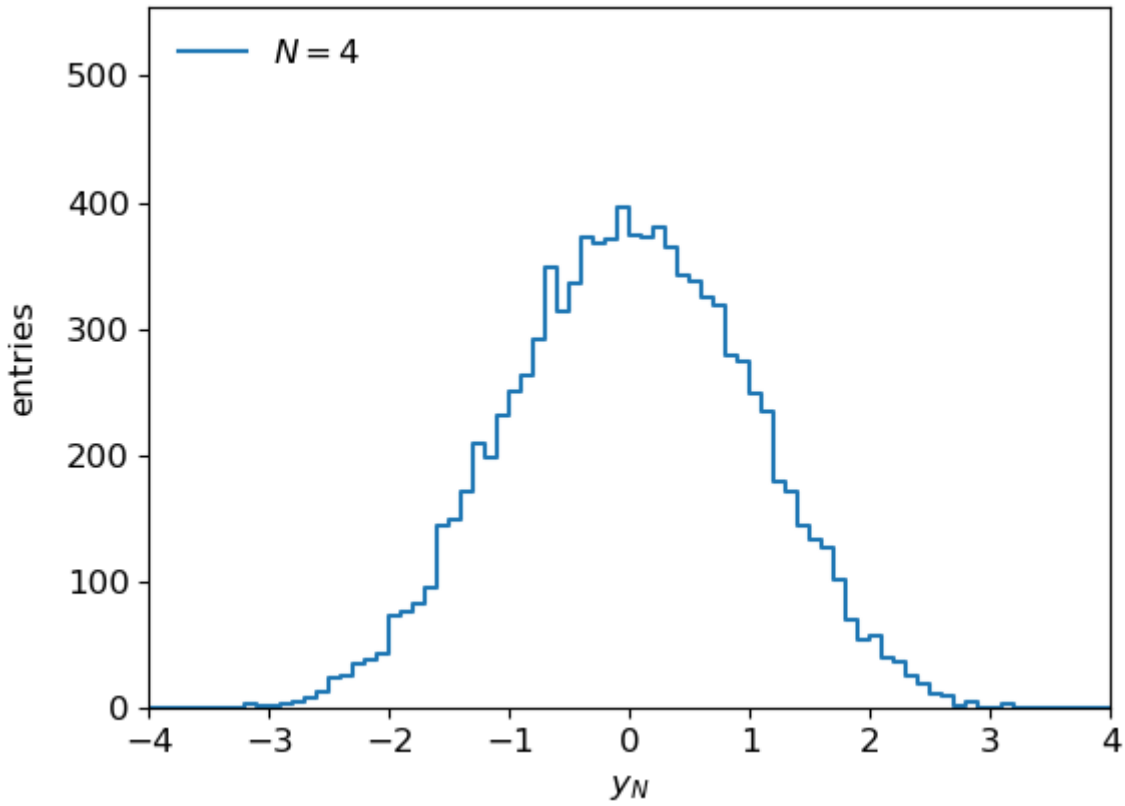
In [4]:
```
numVal = 10000
nBins = 80
yMin = -4.
yMax = 4.

for N in [1, 2, 4, 12]:
    y = yArr(N, numVal)
    histLabel = '$N = $' + str(N)
    yHist, bin_edges = np.histogram(y, bins=nBins, range=(yMin, yMax))
    plotHist(yHist, bin_edges, r'$y_{N}$', r'entries', histLabel)
    plt.figtext(0.6, 0.81, f'mean = {y.mean():.4f}')
    plt.figtext(0.6, 0.74, f'std. dev. = {y.std():.4f}')
```



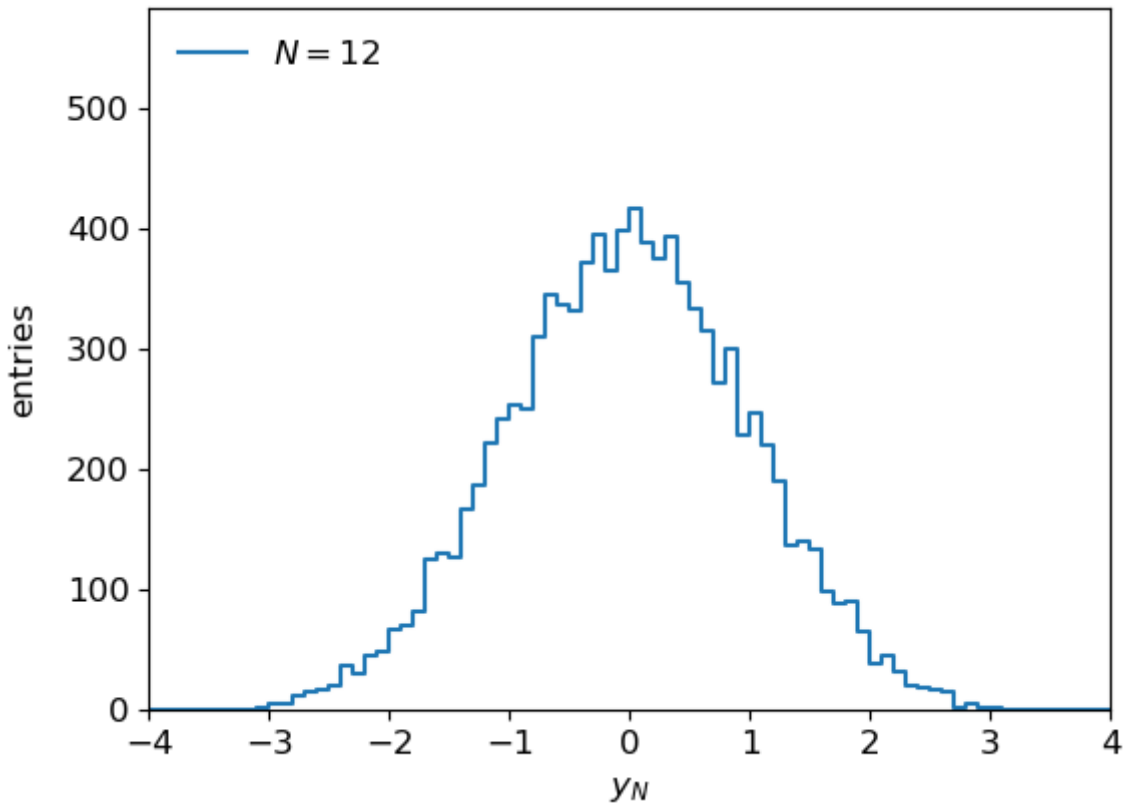

```
<Figure size 640x480 with 0 Axes>
```




```
<Figure size 640x480 with 0 Axes>
```




```
<Figure size 640x480 with 0 Axes>
```

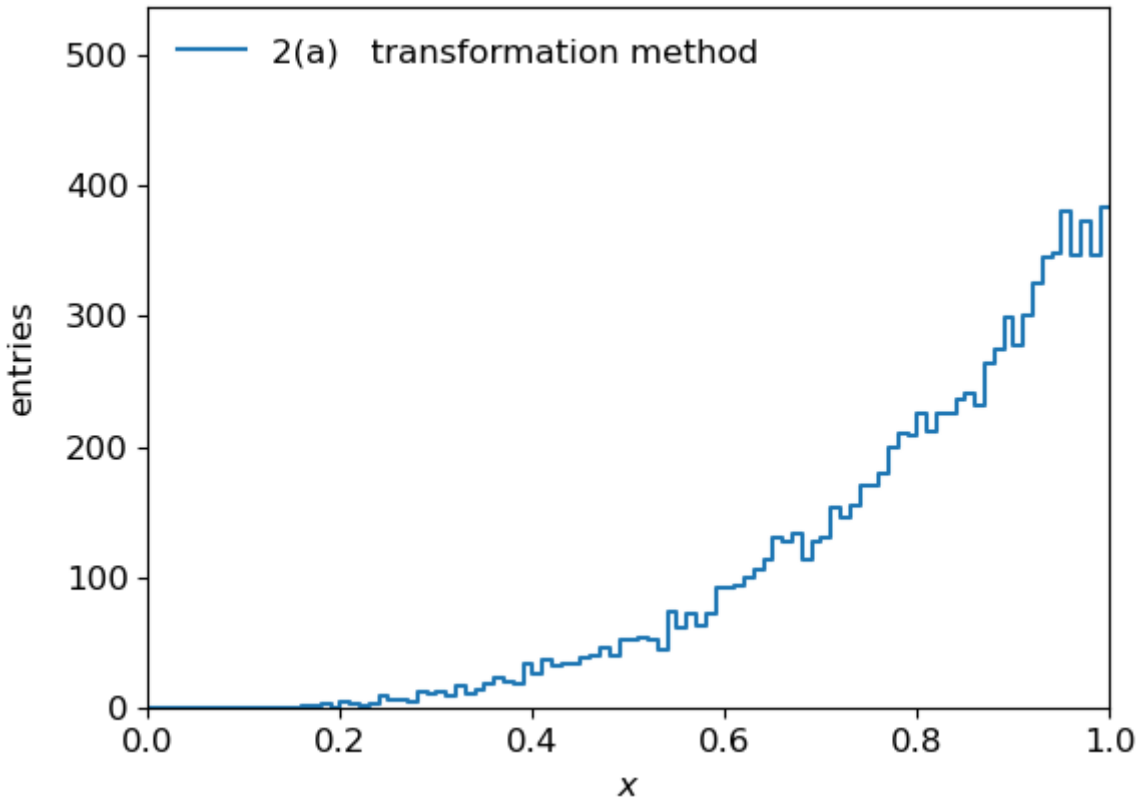


In [5]:

```
# 2(a) Generate random values according to f(x) = 4x^3 (0 < x < 1)
# with transformation method; here x = r**(1./4.)
numVal = 10000
nBins = 100
rData = np.random.uniform(0., 1., numVal)
xMin=0.
xMax=1.
xData = pow(rData, 0.25)
xHist, bin_edges = np.histogram(xData, bins=nBins, range=(xMin, xMax))
plotHist(xHist, bin_edges, r'$x$', r'entries', r'2(a)   transformation method
```

```
plt.figtext(0.18, 0.74, r'$f(x) = 4x^{3}, \; 0 \leq x \leq 1$')
plt.show()
```

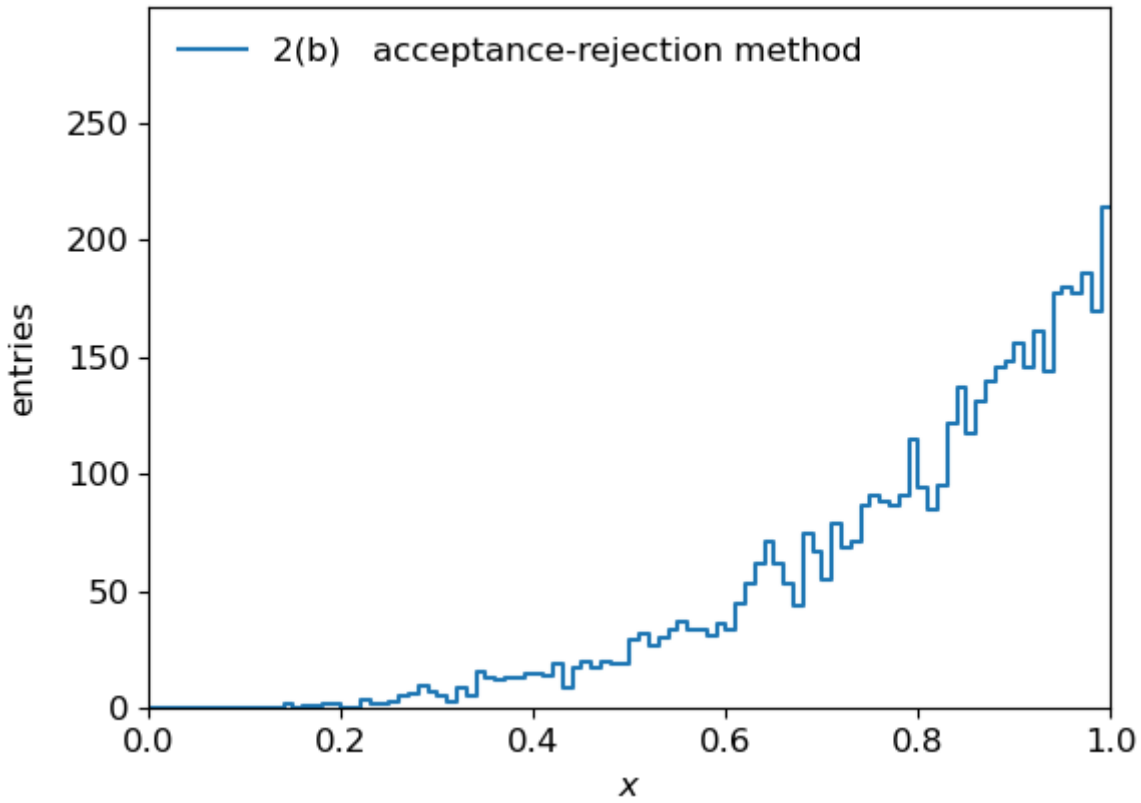
```
<Figure size 640x480 with 0 Axes>
```




```
<Figure size 640x480 with 0 Axes>
```

In [6]:

```
# 2(b) Repeat using the acceptance rejection method
numVal = 20000
nBins = 100

def f(x):
    return 4. * x**3

xMin = 0.
xMax = 1.
fMax = 4.
r1 = np.random.uniform(0., 1., numVal)
x = xMin + r1*(xMax - xMin)
r2 = np.random.uniform(0., 1., numVal)
u = fMax*r2
xData = x[u<f(x)]
nBins = 100
xHist, bin_edges = np.histogram(xData, bins=nBins, range=(xMin, xMax))
plotHist(xHist, bin_edges, r'$x$', r'entries', r'2(b)   acceptance-rejection r
plt.figtext(0.18, 0.74, r'$f(x) = 4x^{3}, \; 0 \leq x \leq 1$')
plt.show()
```




```
<Figure size 640x480 with 0 Axes>
```

In [ ]:

3) $\vec{x} \sim \text{Gauss}(\vec{\mu}_k, V)$, $k = 0, 1$

($\vec{x} = (x_1, \ldots, x_n)$; V = covariance matrix)

i.e. $f(\vec{x}|\vec{\mu}_k) = \frac{1}{(2\pi)^{n/2}|V|^{1/2}} \exp\left[-\frac{1}{2}(\vec{x}-\vec{\mu}_k)^T V^{-1}(\vec{x}-\vec{\mu}_k)\right]$

Test statistic $t(x) = \ln \frac{f(\vec{x}|\vec{\mu}_1)}{f(\vec{x}|\vec{\mu}_0)}$

$$\ln \frac{f(\vec{x}|\vec{\mu}_1)}{f(\vec{x}|\vec{\mu}_0)} = -\frac{1}{2}\left[(\vec{x}-\vec{\mu}_1)^T V^{-1}(\vec{x}-\vec{\mu}_1) - (\vec{x}-\vec{\mu}_0)^T V^{-1}(\vec{x}-\vec{\mu}_0)\right]$$

$$= -\frac{1}{2}\left[\vec{x}^T V^{-1}\vec{x} - \vec{\mu}_1^T V^{-1}\vec{x} - \vec{x}^T V^{-1}\vec{\mu}_1 + \vec{\mu}_1^T V^{-1}\vec{\mu}_1 - \vec{x}^T V^{-1}\vec{x} + \vec{\mu}_0^T V^{-1}\vec{x} + \underbrace{\vec{x}^T V^{-1}\vec{\mu}_0}_{= \vec{\mu}_0^T V^{-1}\vec{x}} - \vec{\mu}_0^T V^{-1}\vec{\mu}_0\right]$$

since scalar $() = ()^T$ and $V^{-1} = (V^{-1})^T$

$$= \underbrace{-\frac{1}{2}\left[\vec{\mu}_1 V^{-1}\vec{\mu}_1 - \vec{\mu}_0^T V^{-1}\vec{\mu}_0\right]}_{w_0} + \underbrace{(\vec{\mu}_1 - \vec{\mu}_0)^T V^{-1}}_{\vec{w}^T}\vec{x}$$

$$= w_0 + \vec{w}^T\vec{x}$$

$$\vec{w} = \left[(\vec{\mu}_1 - \vec{\mu}_0)^T \underbrace{V^{-1}}_{= V^{-1}}\right]^T = (V^{-1})^T(\vec{\mu}_1 - \vec{\mu}_0)$$

If $W = V + V$, $W^{-1} = \frac{1}{2}V^{-1}$

$$\Rightarrow \vec{w} = 2W^{-1}(\vec{\mu}_1 - \vec{\mu}_0)$$