Margot Wagner
A53279875
CSE 250a HW6
Due: 11/17/20

6.2 EM Algorithm   16 pts

(a) $P(a,c \mid b,d)$

$$P(a,c \mid b,d) = \frac{P(a,b,c,d)}{P(b,d)}$$

$$P(b,d) = \sum_{a',c'} P(a=a', b, c=c', d)$$

$$= \sum_{a',c'} P(a=a')P(b \mid a=a')P(c=c' \mid a=a', b)P(d \mid a=a', b, c=c')$$

$$P(a,b,c,d) = P(a)P(b \mid a)P(c \mid a,b)P(d \mid a,b,c)$$

$$P(a,c \mid b,d) = \frac{P(a)P(b \mid a)P(c \mid a,b)P(d \mid a,b,c)}{\sum_{a',c'} P(a=a')P(b \mid a=a')P(c=c' \mid a=a', b)P(d \mid a=a', b, c=c')}$$

(b) $P(a \mid b,d)$ + $P(c \mid b,d)$

$$P(a \mid b,d) = \sum_{c'} P(a, c=c' \mid b,d)$$

$$P(c \mid b,d) = \sum_{a'} P(a=a', c \mid b,d)$$

(c) Log-likelihood

$$\mathcal{L} = \sum_t \log P(B=b_t, D=d_t)$$

$$= \sum_t \log \sum_{a,c} P(A=a, B=b_t, C=c, D=d_t)$$

$$= \sum_t \log \sum_{a,c} P(A=a)P(B=b_t \mid A=a)P(C=c \mid A=a, B=b_t)P(D=d_t \mid A=a, B=b_t, C=c)$$

(d) EM Algorithm

From lecture:

root: $$P(X_i = x) = \frac{1}{T}\sum_t P(X_i = x \mid \overset{\text{evidence}}{V_t = v_t})$$

child: $$P(X_i = x \mid pa_i = \pi) = \frac{\sum_t P(X_i = x, pa_i = \pi \mid V_t)}{\sum_t P(pa_i = \pi \mid V_t)}$$

$$P(A=a) \leftarrow \frac{1}{T}\sum_t P(A=a \mid b_t, d_t)$$

$$P(A=a) \leftarrow \frac{1}{T} \sum_t P(A=a \mid b_t, d_t)$$

$$P(B=b \mid A=a) = \frac{\sum_t P(B=b, A=a \mid b_t, d_t)}{\sum_t P(A=a \mid b_t, d_t)}$$

$$P(B=b, A=a \mid b_t, d_t) = P(A=a \mid b_t, d_t)\, P(B=b \mid A=a, b_t, d_t)$$

$$= P(A=a \mid b_t, d_t)\, I(b, b_t)$$

$$P(B=b \mid A=a) \leftarrow \frac{\sum_t P(A=a \mid b_t, d_t)\, I(b, b_t)}{\sum_t P(A=a \mid b_t, d_t)}$$

$$P(C=c \mid A=a, B=b) = \frac{\sum_t P(A=a, B=b, C=c \mid b_t, d_t)}{\sum_t P(A=a, B=b \mid b_t, d_t)}$$ prev part

$$P(A=a, B=b, C=c \mid b_t, d_t) = P(B=b \mid b_t, d_t)\, P(A=a, C=c \mid b_t, d_t)$$

$$= P(A=a, C=c \mid b_t, d_t)\, I(b, b_t)$$

$$P(C=c \mid A=a, B=b) \leftarrow \frac{\sum_t P(A=a, C=c \mid b_t, d_t)\, I(b, b_t)}{\sum_t P(A=a \mid b_t, d_t)\, I(b, b_t)}$$

$$P(D=d \mid A=a, B=b, C=c) = \frac{\sum_t P(A=a, B=b, C=c, D=d \mid b_t, d_t)}{\sum_t P(A=a, B=b, C=c \mid b_t, d_t)}$$

$$= \frac{\sum_t P(B=b \mid b_t, d_t)\, P(D=d \mid B=b, b_t, d_t)\, P(A=a, C=c \mid b_t, d_t)}{\sum_t P(B=b \mid b_t, d_t)\, P(A=a, C=c \mid b_t, d_t)}$$

$$P(D=d \mid A=a, B=b, C=c) \leftarrow \frac{\sum_t I(b, b_t)\, I(d, d_t)\, P(A=a, C=c \mid b_t, d_t)}{\sum_t P(A=a, C=c \mid b_t, d_t)\, I(b, b_t)}$$

6.3 Auxilary Function

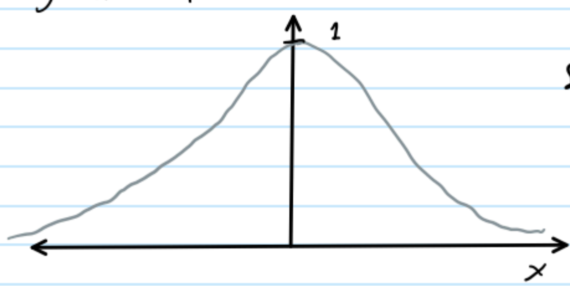
(a) $f(x) = \log\cosh(x)$

$f'(x) = \tanh(x) \rightarrow \tanh(0) = 0$

$f''(x) = \text{sech}^2(x) \rightarrow \text{sech}^2(0) = 1 > 0$

so it is a minimum.

(b) $f''(x) \le 1$



$\text{sech}^2(x)$

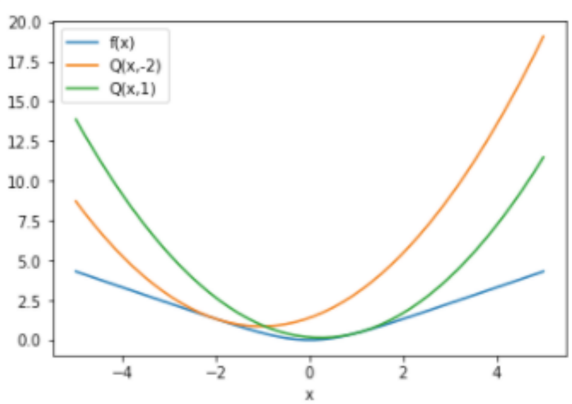
$h(x) = \text{sech}^2(x)$

$h'(x) = -2\tanh(x)\,\text{sech}^2(x) = 0$ when $x = 0$

$h''(x) = 4\tanh^2(x)\,\text{sech}^2(x) - 2\,\text{sech}^4(x) < 0$ when $x = 0$

maximum at $x = 0$ where $\text{sech}^2(0) = 1$

(c) graph



(d) i $Q(x,x) = f(x) + f'(x)(\cancel{x - x}) + \frac{1}{2}(\cancel{x - x})^2$

$\boxed{Q(x,x) = f(x)}$

ii

$$f(x) = f(y) + \int_y^x du \left[f'(y) + \int_y^u dv\, f''(v) \right]$$

$$= f(y) + f'(y)(x-y) + \int_y^x du \int_y^u dv\, f''(v) \quad \text{subst } f''(x) \leq 1$$

$$\leq f(y) + f'(y)(x-y) + \int_y^x du \int_y^u dv\,(1)$$

$$= f(y) + f'(y)(x-y) + \tfrac{1}{2}(x-y)^2 = Q(x,y)$$

$$\therefore \; f(x) \leq Q(x,y)$$

(e) $x_{n+1} = \text{argmin}_x Q(x, x_n)$

$$\frac{\partial}{\partial x} Q(x, x_n) = \frac{\partial}{\partial x}\left[f(x_n) + f'(x_n)(x - x_n) + \tfrac{1}{2}(x - x_n)^2 \right]$$

$$0 = 0 + f'(x_n) + (x - x_n)$$

$$f'(x_n) + x - x_n$$

$$x_n - f'(x_n) = x$$

$$x_{n+1} = x_n - f'(x_n)$$

(f)

In [35]:
```
x = np.linspace(-5, 5, 50)
```
executed in 3ms, finished 11:40:06 2020-11-16

In [36]:
```
def f(x):
    return np.log(np.cosh(x))
```
executed in 3ms, finished 11:40:06 2020-11-16

In [37]:
```
def f_prime(x):
    return np.tanh(x)
```
executed in 2ms, finished 11:40:06 2020-11-16

In [44]:
```
def f_double_prime(x):
    return sech(x)**2
```
executed in 2ms, finished 11:44:21 2020-11-16

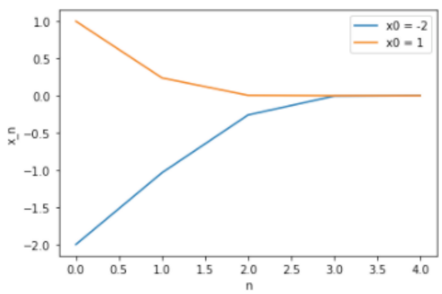
In [38]:
```
def Q(x, y):
    return f(y) + f_prime(y)*(x - y) + 0.5*(x - y)**2
```
executed in 3ms, finished 11:40:06 2020-11-16

6.3 (f)

In [40]:
```
def update_e(x0, iters):
    x = np.zeros(iters)

    x[0] = x0

    for n in range(iters - 1):
        x[n+1] = x[n] - f_prime(x[n])

    return x
```
executed in 4ms, finished 11:40:09 2020-11-16

In [41]:
```
x0 = [-2, 1]
iters = 5
x = np.zeros((len(x0), iters))

for i in range(len(x0)):
    x[i, :] = update_e(x0[i], iters)
```
executed in 4ms, finished 11:40:10 2020-11-16

In [42]:
```
plt.plot(np.arange(0, iters, 1), x[0, :], label='x0 = -2')
plt.plot(np.arange(0, iters, 1), x[1, :], label='x0 = 1')
plt.ylabel('x_n')
plt.xlabel('n')
plt.legend()
plt.show()
```
executed in 121ms, finished 11:40:10 2020-11-16



(g) $x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)} = x_n - \frac{\tanh(x_n)}{\operatorname{sech}^2(x_n)}$

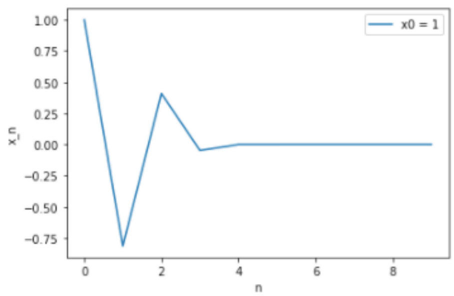
x_n does not converge for $x_0 = -2$ because it becomes undefined (divide by zero).

$|x_0| < 1.0894$

In [45]:
```python
def update_g(x0, iters):
    x = np.zeros(iters)

    x[0] = x0

    for n in range(iters - 1):
        x[n+1] = x[n] - f_prime(x[n])/f_double_prime(x[n])

    return x
```
executed in 4ms, finished 11:47:32 2020-11-16

In [49]:
```python
x0 = [1]
iters = 10
x = np.zeros((len(x0), iters))

for i in range(len(x0)):
    x[i, :] = update_g(x0[i], iters)
```
executed in 4ms, finished 11:49:05 2020-11-16

In [54]:
```python
plt.plot(np.arange(0, iters, 1), x[0, :], label='x0 = 1')
plt.ylabel('x_n')
plt.xlabel('n')
plt.legend()
plt.show()
```
executed in 114ms, finished 11:50:14 2020-11-16



In [73]:
```python
x0 = np.linspace(1.05,1.09,1000)
iters = 10
x = np.zeros((len(x0), iters))

for i in range(len(x0)):
    try:
        x[i, :] = update_g(x0[i], iters)
    except ZeroDivisionError:
        print(x0[i])
        break
```
executed in 450ms, finished 11:54:48 2020-11-16

```
1.0893993993993996
```

(h) graph

The graph still shows a clear minimum, but the exact value is not as easy to determine exactly just by looking at the graph since it isn't zero.

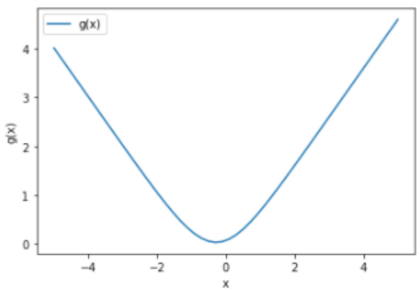
6.3 (h)

In [76]:
```python
def g(x):
    sumk = 0
    for k in range(1,11):
        sumk += np.log(np.cosh(x + (1/k)))

    return sumk/10
```

executed in 3ms, finished 11:59:26 2020-11-16

In [77]:
```python
x = np.linspace(-5, 5, 50)
plt.plot(x, g(x), label='g(x)')
plt.xlabel('x')
plt.ylabel('g(x)')
plt.legend()
plt.show()
```

executed in 113ms, finished 11:59:57 2020-11-16



(i) $R(x,x) = g(x)$

$$R(x,x) = g(x) + g'\cancel{(x)(x-x)} + \tfrac{1}{2}\cancel{(x-x)^2}$$

$$\boxed{R(x,x) = g(x)} \checkmark$$

$$g(x) = \frac{1}{10}\sum_{k=1}^{10} \log\cosh\left(x + \frac{1}{k}\right)$$

$$= \frac{1}{10}\sum_{k=1}^{10} f\left(x + \frac{1}{k}\right)$$

$\therefore\ g''(x) \le 1 \ \forall x$ since $f''(x) \le 1 \ \forall x$

Using this, using the same logic in part d, we can say

$$R(x,y) \ge g(x)$$

$\therefore R(x,y)$ is an auxillary function for $g(x)$

(j) $\frac{\partial}{\partial x} R(x, x_n) = 0 = g'(x_n) + x - x_n$

$$x_{n+1} = x_n - g'(x_n)$$

(k) $x_{min} = -0.2836$

$g(x_{min}) = 0.03266$

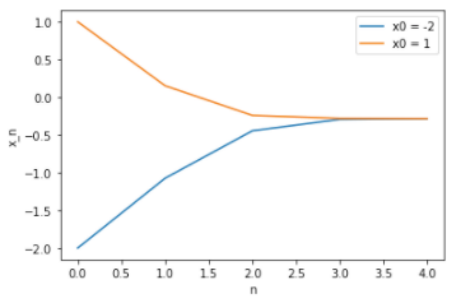
6.3 (k)

In [78]:
```
def g_prime(x):
    sumk = 0
    for k in range(1,11):
        sumk += np.tanh(x + (1/k))

    return sumk/10
```
executed in 3ms, finished 12:14:37 2020-11-16

In [85]:
```
def update_k(x0, iters):
    x = np.zeros(iters)

    x[0] = x0

    for n in range(iters - 1):
        x[n+1] = x[n] - g_prime(x[n])

    return x
```
executed in 3ms, finished 12:19:51 2020-11-16

In [86]:
```
x0 = [-2, 1]
iters = 5
x = np.zeros((len(x0), iters))

for i in range(len(x0)):
    x[i, :] = update_k(x0[i], iters)
```
executed in 4ms, finished 12:19:54 2020-11-16

In [87]:
```
plt.plot(np.arange(0, iters, 1), x[0, :], label='x0 = -2')
plt.plot(np.arange(0, iters, 1), x[1, :], label='x0 = 1')
plt.ylabel('x_n')
plt.xlabel('n')
plt.legend()
plt.show()
```
executed in 121ms, finished 12:19:54 2020-11-16



In [90]:
```
print("The minimum of g(x) is {0} at x={1}".format(g(x[0,-1]), x[0,-1]))
```
executed in 4ms, finished 12:24:53 2020-11-16

```
The minimum of g(x) is 0.03265729378239994 at x=-0.2836231987205744
```

6.4

orig: $P(Y=1|x) = 1 - \prod_{i=1}^{n} (1-p_i)^{x_i}$

now (w/ z) $\begin{cases} P(z_i=1|x_i=0) = 0 \\ P(z_i=1|x_i=1) = p_i \end{cases}$

$P(Y=1|z) = \begin{cases} 1 & \text{if any } z_i=1 \\ 0 & \text{all } z_i=0 \end{cases}$

(a) $P(Y=1|x) = \sum_{z\in\{0,1\}^n} P(Y=1, z|x)$ marginalization

$= \sum_{z\in\{0,1\}^n} P(z|x) P(Y=1|z,\cancel{x})$ product rule, cond ind

$= \sum_{z\in\{0,1\}^n} P(z|x)(1 - I(z,\{0\}^n))$ I = 1 when all z's are 0

$= \sum_{z\in\{0,1\}^n} P(z|x) - \sum_{z\in\{0,1\}^n} P(z|x) I(z,\{0\}^n)$ distribute

$= \sum_{z\in\{0,1\}^n} P(z|x) - \sum_{z\in\{0,1\}^n} P(z=\{0\}^n|x)$

$= 1 - \prod_{i=1}^{n} P(z_i=0|x_i)$ cond ind

$= 1 - \prod_{i=1}^{n} (1-p_i)^{x_i} \cancelto{1}{(1)^{1-x_i}}$

$= 1 - \prod_{i=1}^{n} (1-p_i)^{x_i}$

(b) $P(z_i=1, x_i=1 | x, y) = I(x_i,1) P(z_i=1|x,y)$ logical consistency

$= I(x_i,1) \dfrac{P(z_i=1|x) P(y|z_i=1,\cancel{x})}{P(y|x)}$ Bayes

$= I(x_i,1) \dfrac{P(z_i=1|x) I(y,1)}{P(y|x)}$ logical or

$= I(x_i,1) I(y,1) \dfrac{[\cancelto{0}{P(z_i=1|x_i=0)} + P(z_i=1|x_i=1)]}{P(y|x)}$ definition

$= \dfrac{I(x_i,1) I(y,1)\, p_i}{P(y|x)}$ answer from (a) (note: everything is 0 if y=0)

$= \dfrac{y x_i p_i}{1 - \prod_j (1-p_j)^{x_j}}$

(c) p_i

Generally

$$P(X_i = x \mid pa_i = \pi) \leftarrow \frac{\sum_{t=1} P(X_i = x, pa_i = \pi \mid V_t = v_t)}{\sum_{t=1}^{T} P(pa_i = \pi \mid V_t = v_t)}$$ definition

$$P(Z_i = 1 \mid X_i = 1) \leftarrow \frac{\sum_t P(X_i = 1, Z_i = 1 \mid x_t, y_t)}{\sum_t P(X_i = 1 \mid x_t, y_t)}$$ subst

$$= \frac{\sum_t P(X_i = 1, Z_i = 1 \mid x_t, y_t)}{\sum_t I(x, x^t)}$$ I def.

$$p_i \leftarrow \frac{1}{T_i} \sum_t P(Z_i = 1, X_i = 1 \mid x_t, y_t)$$ $\sum_t I(x, x^t) = T_i$

6.4d

In [165]:
```
X = np.loadtxt('noisyOr_X.txt')
y = np.loadtxt('noisyOr_Y.txt')
Z = np.zeros_like(X)
T = X.shape[0] # number of examples
n = X.shape[1] # number of inputs
p = np.asarray([1/n]*n)
```
executed in 11ms, finished 20:32:49 2020-11-16

In [150]:
```
def p_y1_x(t):
    '''
    Probability y is 1 given x based on equation for 6.4a
    '''
    
    return 1 - prod([(1 - p[i])**X[t,i] for i in range(n)])
```
executed in 3ms, finished 20:18:03 2020-11-16

In [151]:
```
def p_y0_x(t):
    '''
    Probability y is 0 given x based on equation for 6.4a
    '''
    
    return 1 - p_y1_x(t)
```
executed in 2ms, finished 20:18:04 2020-11-16

In [160]:
```
def p_y_x(yt, t):
    '''
    Probability y given x
    '''
    if yt == 1:
        return p_y1_x(t)
    else:
        return p_y0_x(t)
```
executed in 3ms, finished 20:19:04 2020-11-16

In [153]:
```
def mistakes(y, yPred):
    '''
    Returns the number of mistakes between actual y and predicted y
    '''
    return len(y) - sum(y==yPred)
```
executed in 2ms, finished 20:18:05 2020-11-16

In [135]:
```
def predictions():
    '''
    Make predictions based on the equation for 6.4a
    '''
    return [1 if p_y1_x(t) >= 0.5 else 0 for t in range(T)]
```
executed in 3ms, finished 20:14:13 2020-11-16

In [161]:
```
def log_likely():
    '''
    Gives the normalized log-likelihood as defined in 6.4b
    '''
    return sum(np.log([p_y_x(y[t], t) for t in range(T)]))/T
```
executed in 2ms, finished 20:19:07 2020-11-16

In [162]:
```
def update():
    '''
    Update p according to 6.4c
    '''
    Ti = np.zeros(n)
    p_new = np.zeros_like(p)
    for i in range(n):
        Ti[i] = sum(X[:,i])
        p_new[i] = (1 / Ti[i])*sum([((y[t]*X[t,i]*p[i])/p_y1_x(t)) for t in range(T)])


    return p_new
```
executed in 3ms, finished 20:19:08 2020-11-16

In [164]:
```
# EM Algo
n_iterations = 257
i_report = [0]
i_report.extend([2**(i) for i in range(9)])
for i in range(n_iterations):
    if i in i_report:
        print(i, mistakes(y, predictions()), log_likely(), sep='\t')

    p = update()
```
executed in 31.4s, finished 20:19:41 2020-11-16

```
0	195	-1.044559748133717
1	60	-0.504940510120726
2	43	-0.410763774177962
4	42	-0.3651271742872333
8	44	-0.34766321194257643
16	40	-0.33467666667097906
32	37	-0.32259268945106767
64	37	-0.31483106238579917
128	36	-0.3111558174240999
256	36	-0.3101611042419866
```

Iterations	N Mistakes	Log-likelihood
0	195	-1.044559748133717
1	60	-0.504940510120726
2	43	-0.410763774177962
4	42	-0.3651271742872333
8	44	-0.34766321194257643
16	40	-0.33467666667097906
32	37	-0.32259268945106767
64	37	-0.31483106238579917
128	36	-0.3111558174240999
256	36	-0.3101611042419866