Solution to Problem 1.

The empirical risk objective function:

$$\hat{R}_n(w) = \frac{1}{n}\sum_{i=1}^{n} \ell_{\text{logistic}}(y, w)$$
$$= \frac{1}{n}\sum_{i=1}^{n} \log\left(1 + e^{-yw^T x}\right)$$

By defining $y_i' = \begin{cases} 1 & y_i = 1 \\ 0 & y_i = -1 \end{cases}$, the NLL is:

$$NLL(w) = -\left[\sum_{i=1}^{n} y_i' \log p(y=1 \mid x; w) + (1-y_i') \log\left(1 - p(y=1 \mid x; w)\right)\right]$$

① if $y_i = 1$, then $y_i' = 1$

$$NLL(w) = -\sum_{i=1}^{n} \log p(y_i = 1 \mid x_i; w) = \sum_{i=1}^{n} \log\left(1 + e^{-y_i w^T x_i}\right) = n\hat{R}_n(w)$$

② if $y_i = -1$, then $y_i' = 0$

$$NLL(w) = -\sum_{i=1}^{n} \log\left(1 - p(y_i = 1 \mid x_i; w)\right) = \sum_{i=1}^{n} \log\left(\frac{e^{-w^T x_i}}{1 + e^{-w^T x_i}}\right)^{-1}$$
$$= \sum_{i=1}^{n} \log\left(1 + e^{-y_i w^T x_i}\right) = n\hat{R}_n(w)$$

To sum up, it's proved by ①② that:

to minimize the negative log-likelihood (NLL) of the data is to minimize the empirical risk.

These two approaches are then equivalent to each other.

Solution to Problem 2.

The logistic regression:

$$f(w^T x) = \frac{1}{1 + e^{-w^T x}}$$

① if $f(w^T x) \geq 0.5$, then $y = 1$

where $1 + e^{-w^T x} \leq 2 \Rightarrow w^T x \geq 0$

② if $f(w^T x) \leq 0.5$, then $y = -1$

where $1 + e^{-w^T x} \geq 2 \Rightarrow w^T x \leq 0$

Hence, the decision boundary is: $\{x : w^T x = 0\}$.

Solution to Problem 3

Notice that in solution to problem 1

It's proved that:

$$\log L(cw) = -\sum_{i=1}^{n} \log(1 + e^{-y_i c w^T x_i})$$

$$\frac{\partial \log L(cw)}{\partial c} = \sum_{i=1}^{n} \frac{e^{-y_i c w^T x_i}}{1 + e^{-y_i c w^T x_i}} \cdot y_i w^T x_i$$

Here, $\frac{e^{-y_i c w^T x_i}}{1 + e^{-y_i c w^T x_i}}$ is always positive; Besides, since all examples are classified correctly, $y_i w^T x_i \geq 0$ for all i. Then $\frac{\partial \log L(cw)}{\partial c} \geq 0$

It's proved that: by increasing a scalar c on $\hat{w}$ unboundedly, we can increase the likelihood!

Solution to Problem 4

In the note, it's given that $\lambda \|w\|^2$ and Log-Sum-Exp are convex functions

Notice that $\log(1 + e^{-y_i w^T x_i})$ is in the form of Log-Sum-Exp.

$J(w)$ is then a non-negative weighted sum of the convex functions. Thus, $J(w)$ is a convex function.

Solution to Problem 5

In [1]:

```python
def f_objective(theta, X, y, l2_param=1):
    '''
    Args:
        theta: 1D numpy array of size num_features
        X: 2D numpy array of size (num_instances, num_features)
        y: 1D numpy array of size num_instances
        l2_param: regularization parameter

    Returns:
        objective: scalar value of objective function
    '''
    features_num = X.shape[1]
    samples_num = X.shape[0]
    y = list(y[0])

    loss = 0
    for i in range(samples_num):
        loss += np.logaddexp(0, - y[i] * (theta @ X[i]))
    loss = loss / samples_num

    reg = l2_param * (theta @ theta)
    return loss + reg
```

executed in 4ms, finished 17:36:50 2021-03-31

Solution to Problem 6.

In [2]:
```
from scipy import optimize
from functools import partial

def fit_logistic_reg(X, y, objective_function, l2_param=1):
    '''
    Args:
        X: 2D numpy array of size (num_instances, num_features)
        y: 1D numpy array of size num_instances
        objective_function: function returning the value of the objective
        l2_param: regularization parameter

    Returns:
        optimal_theta: 1D numpy array of size num_features
    '''
    objective_func = partial(objective_function, X=X, y=y, l2_param=l2_param)
    w0 = np.ones(X.shape[1])
    w_opt = optimize.minimize(objective_func, w0).x

    return w_opt
```

executed in 532ms, finished 17:36:50 2021-03-31

In [4]:
```
x_train = pd.read_csv("X_train.txt", header=None)
y_train = pd.read_csv("y_train.txt", header=None)

# Standardize features by removing the mean and scaling to unit variance
scalar = StandardScaler()
X_train = scalar.fit_transform(x_train)

# Add a column for the bias term
X_train = np.hstack((X_train, np.ones((X_train.shape[0], 1))))
y_train[y_train==0] = -1
```

executed in 38ms, finished 17:36:51 2021-03-31

In [5]:
```
w_opt = fit_logistic_reg(X_train, y_train, f_objective, l2_param=1)
w_opt
```

executed in 2.15s, finished 17:36:53 2021-03-31

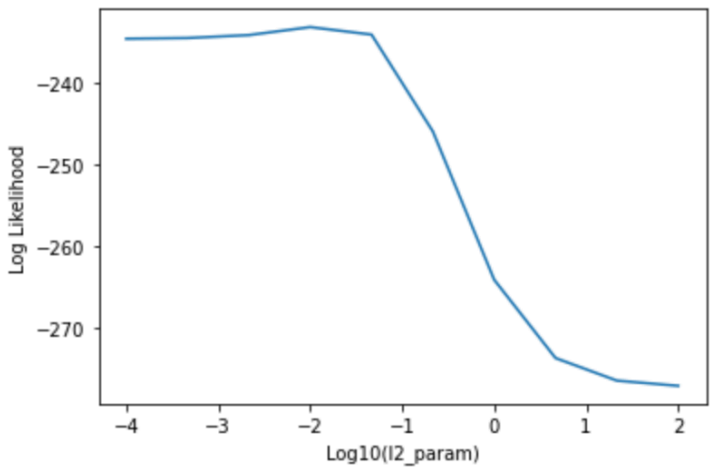
Out[5]:
```
array([ 0.00095627, -0.0002996 ,  0.00302681,  0.10532759, -0.00358736,
       -0.00135853, -0.00385287, -0.00079007, -0.00114404, -0.0717843 ,
        0.00654889, -0.00451094,  0.01124927, -0.00386438, -0.00271265,
        0.00150358, -0.00278398, -0.00919054, -0.00682277, -0.01027481,
        0.00281872])
```

Solution to Problem 7

```
def log_likelihood(theta, X, y):
    '''
    Args:
        theta: 1D numpy array of size num_features
        X: 2D numpy array of size (num_instances, num_features)
        y: 1D numpy array of size num_instances
        l2_param: regularization parameter

    Returns:
        objective: scalar value of objective function
    '''
    features_num = X.shape[1]
    samples_num = X.shape[0]
    y = list(y[0])

    nega_log_likeli = 0
    for i in range(samples_num):
        nega_log_likeli += np.logaddexp(0, - y[i] * (theta @ X[i]))

    return - nega_log_likeli
```

```
x_val = pd.read_csv("X_val.txt", header=None)
y_val = pd.read_csv("y_val.txt", header=None)

# Standardize features by removing the mean and scaling to unit variance
X_val = scalar.fit_transform(x_val)

# Add a column for the bias term
X_val = np.hstack((X_val, np.ones((X_val.shape[0], 1))))
y_val[y_val==0] = -1
```

```
log_likeli_lst = []
l2_param_lst = [10**i for i in np.linspace(-4, 2, 10)]
for i in l2_param_lst:
    w_opt = fit_logistic_reg(X_train, y_train, objective_function=f_objective, l2_param=i)
    log_likeli = log_likelihood(w_opt, X_val, y_val)
    log_likeli_lst.append(log_likeli)
```

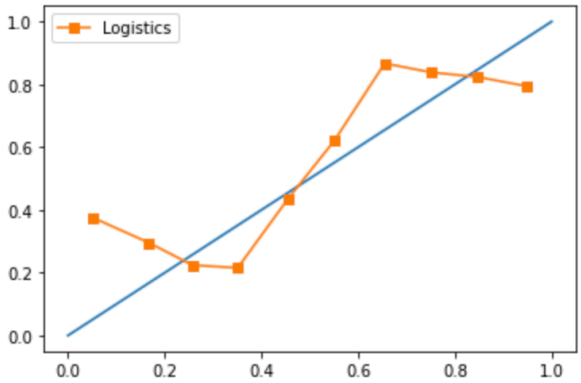


The l2-regularization parameter that minimizes the log-likelihood on the validation set is l2-regu = 0.01

Solution to Problem 8

In [11]:
```
from sklearn.calibration import calibration_curve

plt.plot([0,1], [0,1])
samples_num, features_num = X_val.shape
w_opt = fit_logistic_reg(X_train, y_train, f_objective, 10**(-2))

positive_prob = X_val @ w_opt
for i in range(samples_num):
    positive_prob[i] = 1 / (1 + np.exp(-positive_prob[i]))
positive_prob = (positive_prob - positive_prob.min()) / (positive_prob.max() - positive_prob.min())

fraction_of_posi, mean_predict = calibration_curve(y_val, positive_prob, n_bins=10)
plt.plot(mean_predict, fraction_of_posi, 's-', label='Logistics')
plt.legend()
plt.show();

fraction_of_posi
```

executed in 3.94s, finished 17:37:33 2021-03-31



Out[11]:
```
array([0.375     , 0.29545455, 0.22413793, 0.21538462, 0.43396226,
       0.62162162, 0.86666667, 0.83783784, 0.82352941, 0.79411765])
```

- From the plot, when the validation set is broken into 10 groups, the prediction is not well caliberated.

Solution to Problem 9

$$p(w|D) = \frac{p(D|w)\cdot p(w)}{p(D)} = \frac{\prod_{i=1}^{n} p(y_i|x_i;w)\cdot p(w)}{p(D)}$$

recall from the solution to problem 1 that:

$$NLL(w) = -\left[\sum_{i=1}^{n} y_i' \log p(y=1|x;w) + (1-y_i')\log(1-p(y=1|x;w)\right.$$

$$\prod_{i=1}^{n} p(y_i|x_i;w) = e^{-NLL(w)}$$

Hence: $p(w|D) \propto p(w)\cdot e^{-NLL(w)}$

Solution to Problem 10.

No.

From solution to problem 9.

$$p(w|D) \propto e^{-NLL(w)}\cdot p(w)$$

$$\propto e^{-NLL(w)}\cdot \frac{1}{1+e^{-yw^Tx}}$$

This doesn't look like any form of ganssian distribution.

Solution to Problem 11.

proof:

$$p(w) = \frac{1}{\sqrt{2\pi|\Sigma|}} e^{\left(-\frac{w^T w}{2|\Sigma|}\right)}$$

$$\begin{aligned}
w_{MAP} &= \text{argmax } e^{-NLL_D(w)} p(w) \\
&= \text{argmax } \log\left(e^{-NLL_D(w)} p(w)\right) \\
&= \text{argmin } \left(-\log\left(e^{-NLL_D(w)} p(w)\right)\right) \\
&= \text{argmin } \left(-\log e^{-NLL_D(w)} - \log p(w)\right) \\
&= \text{argmin } \left(NLL_D(w) - \log\left(\frac{1}{\sqrt{2\pi|\Sigma|}}\right) + \left(\frac{1}{2} w^T \Sigma^{-1} w\right)\right) \\
&= \text{argmin } \left(n \hat{R}(w) + \frac{1}{2} w^T \Sigma^{-1} w\right) \\
&= \text{argmin } \left(\hat{R}(w) + \frac{1}{2n} w^T \Sigma^{-1} w\right)
\end{aligned}$$

let: $\Sigma = \frac{1}{2n\lambda} I$

$$w_{MAP} = \text{argmin } \left(\hat{R}(w) + \lambda \|w\|^2\right)$$

Thus, MAP estimate for w after observing data D is the same as the minimizer of the regularized logistic regression function.

Solution to Problem 12

From the previous question,

we know that: we need to let $\Sigma = I$

$$\Sigma = \frac{1}{2n\lambda} I = I$$

Hence, $\lambda = \frac{1}{2n}$.

Solution to Problem 13

since $p(x=H \mid z=H) = 1 \quad p(x=H \mid z=T) = 0$

$$p(x=H \mid \theta_1, \theta_2)$$

$$= p(x=H, z=H \mid \theta_1, \theta_2) + p(x=H, z=T \mid \theta_1, \theta_2)$$

$$= p(x=H, z=H \mid \theta_1, \theta_2) + 0$$

$$= p(x=H \mid z=H; \theta_2)\, p(z=H \mid \theta_1)$$

$$= \theta_1 \theta_2$$

Solution to Problem 14

the likelihood for data D_r is:

$$L_{D_r}(\theta) = P(D_r \mid \theta_1, \theta_2)$$

$$= (\theta_1\theta_2)^{n_h}(1-\theta_1\theta_2)^{n_t}$$

Solution to Problem 15

$$\log L_D(\theta) = \log(\theta_1\theta_2)^{n_h}(1-\theta_1\theta_2)^{n_t}$$

$$= n_h \log(\theta_1\theta_2) + n_t \log(1-\theta_1\theta_2)$$

We cannot estimate θ_1 and θ_2 using MLE

Proof:

take the derivative of the likelihood func:

$$\frac{\partial \log L_D(\theta)}{\partial \theta_1} = \frac{n_h\theta_2}{\theta_1\theta_2} - \frac{n_t\theta_2}{1-\theta_1\theta_2}$$

Here, we have two parameters to be solved, but we only have one equation, thus we are not able to estimate θ_1 and θ_2 using MLE

Solution to Problem 16

$$L(\theta_1, \theta_2) = p(D_r, D_c \mid \theta_1, \theta_2)$$
$$= p(D_c \mid \theta_1)\, p(D_r \mid \theta_1, \theta_2)$$

For $p(D_c \mid \theta_1)$

$$p(D_c \mid \theta_1) = \theta_1^{C_h} (1-\theta_1)^{C_t}$$

$$\frac{\partial \log p(D_c \mid \theta_1)}{\partial \theta_1} = \frac{C_h}{\theta_1} - \frac{C_t}{1-\theta_1} = 0$$

$$\hat{\theta}_1 = \frac{C_h}{C_h + C_t}$$

For $p(D_r \mid \theta_1, \theta_2)$

$$p(D_r \mid \theta_1, \theta_2) = (\theta_1\theta_2)^{n_h} (1-\theta_1\theta_2)^{n_t}$$

$$\frac{\partial \log p(D_r \mid \theta_1, \theta_2)}{\partial \theta_2} = \frac{n_h}{\theta_2} - \frac{n_t \theta_1}{1-\theta_1\theta_2} = 0$$

plug in that: $\hat{\theta}_1 = \frac{C_h}{C_h + C_t}$

Hence: $\hat{\theta}_1 \hat{\theta}_2 = \frac{n_h}{n_h + n_t}$

$$\Rightarrow \quad \hat{\theta}_2 = \frac{(C_h + C_t)\, n_h}{(n_h + n_t)\, C_h} = \frac{N_c \cdot n_h}{N_r \cdot C_h}$$

Solution to Problem 17.

Given that: $\theta_1 \sim \text{Beta}(h, t)$

hence, $p(\theta_1) = \frac{1}{B(h,t)} \theta_1^{h-1} (1-\theta_1)^{t-1}$

$$MAP(\theta_1, \theta_2) = p(\theta_1, \theta_2 | D_r, D_c)$$

$$= \frac{p(D_r, D_c | \theta_1, \theta_2)\, p(\theta_1)}{p(D_r, D_c)}$$

$$= \frac{p(D_c|\theta_1)\, p(D_r|\theta_1, \theta_2)\, p(\theta_1)}{p(D_r, D_c)}$$

recall that:

$$p(D_c|\theta_1) = \theta_1^{C_h} (1-\theta_1)^{C_t}$$

For $\ell = \log p(D_c|\theta_1) p(\theta_1)$

$$= (C_h + h - 1) \log \theta_1 + (C_t + t - 1) \log (1-\theta_1)$$

$$\hat{\theta}_{1\,MAP} = \frac{C_h + h - 1}{(C_h + h - 1) + (C_t + t - 1)}$$

$$\hat{\theta}_{2\,MAP} = \frac{((C_h + h - 1) + (C_t + t - 1))\, n_h}{(C_h + h - 1)(n_h + n_t)}$$