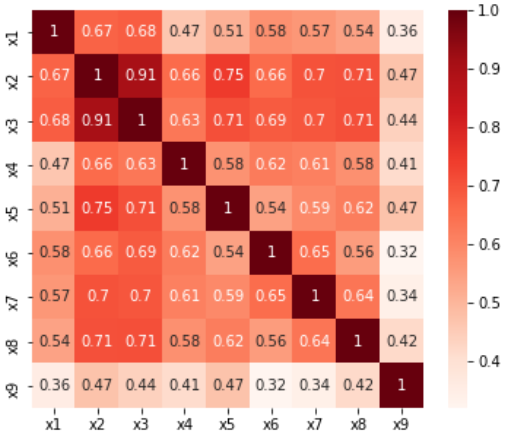
Statistical Learning Theory(01)

Assignment 1 - Logistic Regression

1970030 park guen a

①. Define what your features/attributes are in your model. Appropriate features should be selected (e.g., [x1, x2, x7] or all of them) or defined (e.g., x1•x2, x12]) to improve the model's performance.

저는 우선 feature 사이의 상관관계를 알아보기 위해 1-filter method를 사용해서 heatmap을 그려보고 결과를 통해 feature selection을 수행하기로 했습니다. 그 결과 다음과 같은 그래프가 나왔습니다.



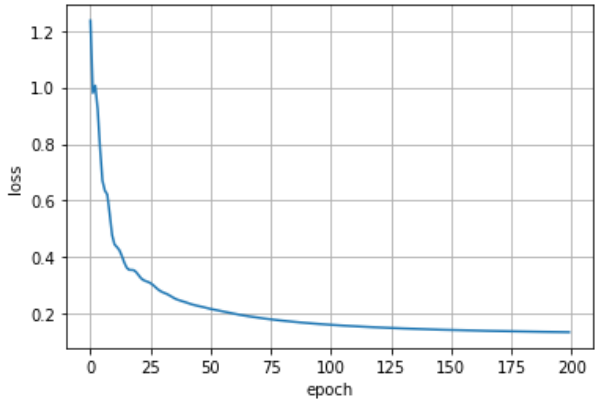
binary classification을 써야하는 문제인만큼 결과값은 빼고 feature 사이의 상관계수만 구하면 된다고 생각했습니다. 결과를 보면 x2와 x3의 상관계수가 가장 높고 x5,x7,x8도 x2,x3과 높은 상관계수를 가지는 것을 확인할 수 있습니다.

결과를 이용해서 feature의 개수를 조정하려고 했는데 평균적으로 6개의 feature를 사용할 때 정확도가 높았습니다. 그리고 교수님께서 정하셨던 1,2,3,4,5,6번째 feauture를 사용한 것이 성능이 좋아 최대한 이용하고자 했습니다. 그래서 우선 상관계수가 낮은 9번째 feature를 넣으면서 2,3번째 feature 중 하나는 빼려고 했습니다. 그리고 feature 6개의 개수를 맞추면서 5,7,8번째 feature 중에서 제외하기로 했더니 1,3,4,5,6,9번째 feature를 이용하는 모델과 1,2,4,5,6,9번째 feature를 이용하는 모델 이렇게 2가지가 후보로 나왔습니다. 결론적으로 1,2,4,5,6,9번째 feature를 이용하는 것이 성능이 가장 좋았습니다.

②. Report your logistic regression model's train accuracy (%), test accuracy (%), and show a plot of loss log over epoch during training.

저는 feature를 select한 뒤에 epoch과 learning rate를 바꾸고 adam으로 optimizer 바꾸어 계산을 수행했습니다. 그 결과 train accuracy = 95.0%, test accuracy = 99.49748743718592%로 나왔습니다.

다음은 training과정의 loss log입니다.



③. Select the appropriate types (L1, L2, or L1+L2) of regularization parameter λ and its value (e.g., λ=10), and justify your choice of λ.

저는 adam optimizer의 weight decay를 이용해 regularization을 적용해보았습니다. Adam optimizer의 weight decay는 L2-regularization과 비슷한 역할을 한다고 알려져 있어 이 값을 변경해가며 연산을 수행했습니다. L2에서는 큰 W의 값들이 더 크게 영향을 받습니다. 하지만 W값들 사이의 크기차이가 크지않아서인지 λ를 10으로 설정했더니 loss가 별로 줄어들지 않았습니다. 그래서 조금씩 λ값을 줄여가며 연산을 수행했고 λ가 0.001 이나 0.0001일때는 regularization을 진행하지 않고 구한 accuracy와 비슷하게 결과가 나왔습니다.

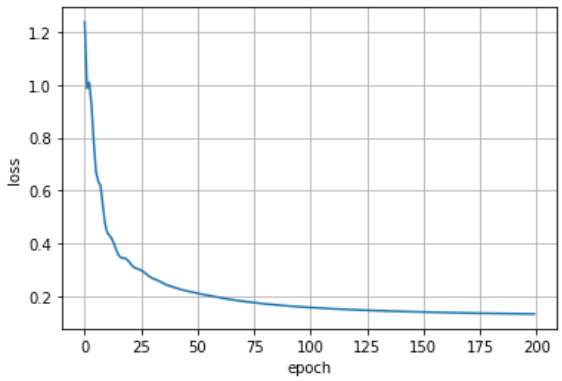
④. Report your regularized logistic regression model's train accuracy (%), test accuracy (%), and show a plot of loss log over epoch during training.

λ를 0.0001로 적용했을 때 train과 test accuracy가 다음과 같이 나왔습니다.

train accuracy = 95.19999999999999%, test accuracy = 99.49748743718592%

λ가 0.001일때보다는 train accuracy가 오른 것을 확인할 수 있었습니다.

다음은 loss log입니다.



⑤. What efforts have you made to improve accuracy of your final model? (e.g., I defined new feature using the given features or chose … attributes as a feature for my model as follows…) Report your final best model's test accuracy (%).

②에서 언급했듯이 여러가지 파라미터들을 조정하고 optimizer를 바꾸었습니다. Learning rate는 너무 늘리니 overshooting이 발생해서 점점 줄여나가다가 undershooting이 발생하지 않도록 0.09로 정하였습니다. 그리고 epoch은 200회 정도가 정확도가 가장 높았습니다. Optimizer는 SGD보다 성능이 좋다고 알려진 adam optimizer로 바꾸었지만 정확도에 큰 변화가 있지는 않았습니다.

그리고 6번 feature의 '?'부분을 0으로 대체하지 않고 randint 함수를 이용해 1~10사이의 임의의값을 조정했더니 정확도가 조금 올랐습니다.

Data의 수가 많지 않아 계속해서 정확도가 99.49748743718592%로 거의 고정이 되었습니다. 초기 W 값을 잘 정해주기 위해서 Xavier initialization과 함께 layer를 추가해보려고 했는데 성능이 좋지 않았습니다. Data augmentation과 함께 hidden layer의 수를 튜닝해주면 조금 더 나은 결과가 나올 수 있었을 것으로 예상됩니다.

```
class LogisticRegression(nn.Module):
  def __init__(self, input_size, output_size):
    super().__init__()
    self.linear1 = nn.Linear(input_size, 250)
    self.linear2 = nn.Linear(250, 125)
    self.linear3 = nn.Linear(125, output_size)

    # xavier initialization
    nn.init.xavier_uniform_(self.linear1.weight)
    nn.init.xavier_uniform_(self.linear2.weight)
    nn.init.xavier_uniform_(self.linear3.weight)

  def forward(self, x):

    first = self.linear1(x)
    second =self.linear2(first)
    pred = torch.sigmoid(self.linear3(second))
    return pred   #probability (not direct value)
```

결과적으로 제가 수정한 모델의 test accuracy는 99.49748743718592%이 나왔습니다.