

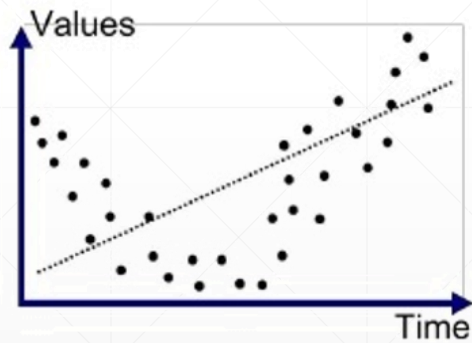


Regularization

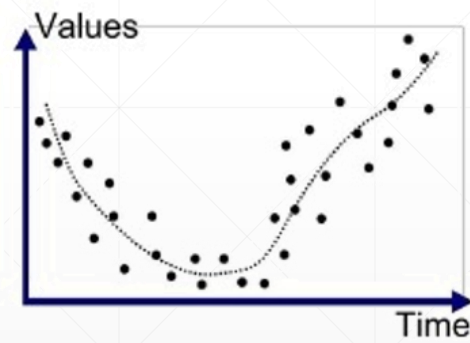
主讲人：龙良曲

Occam's Razor

- More things should not be used than are necessary.*



Underfitted



Good Fit/Robust



Overfitted

Reduce Overfitting

- More data
 - Constraint model complexity
 - shallow
 - regularization
 - Dropout
 - Data argumentation
 - Early Stopping
-

Regularization



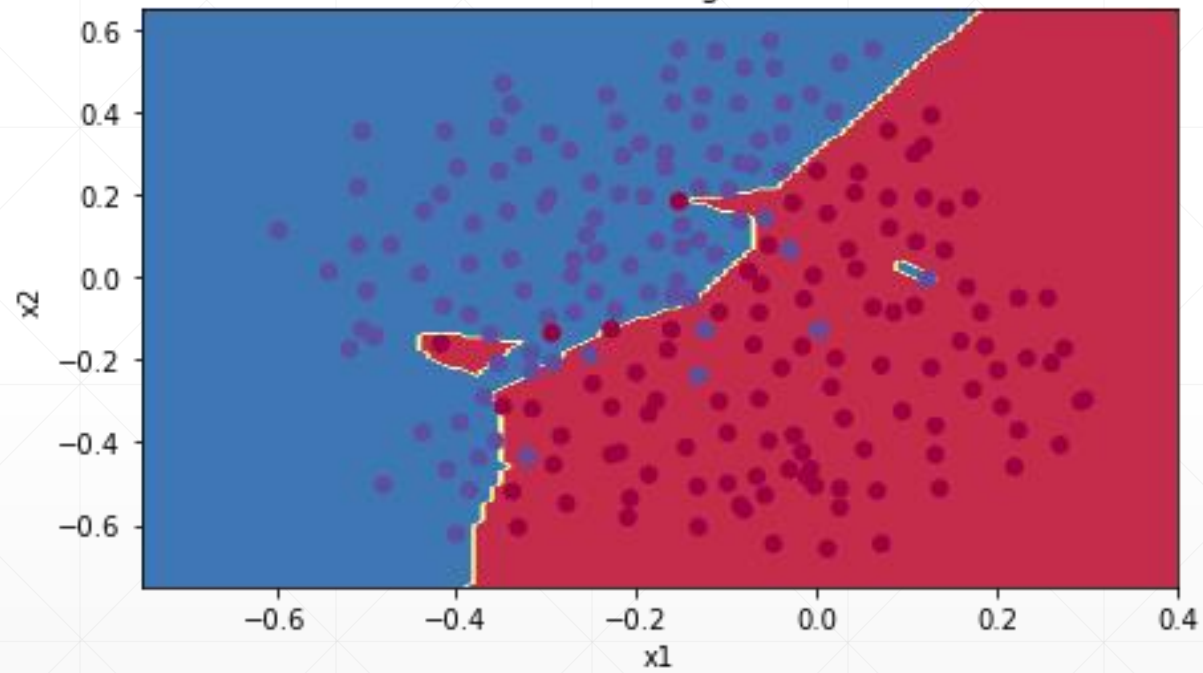
$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m [y_i \ln \hat{y}_i + (1 - y_i) \ln(1 - \hat{y}_i)]$$

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \dots + \beta_n x^n + \varepsilon.$$

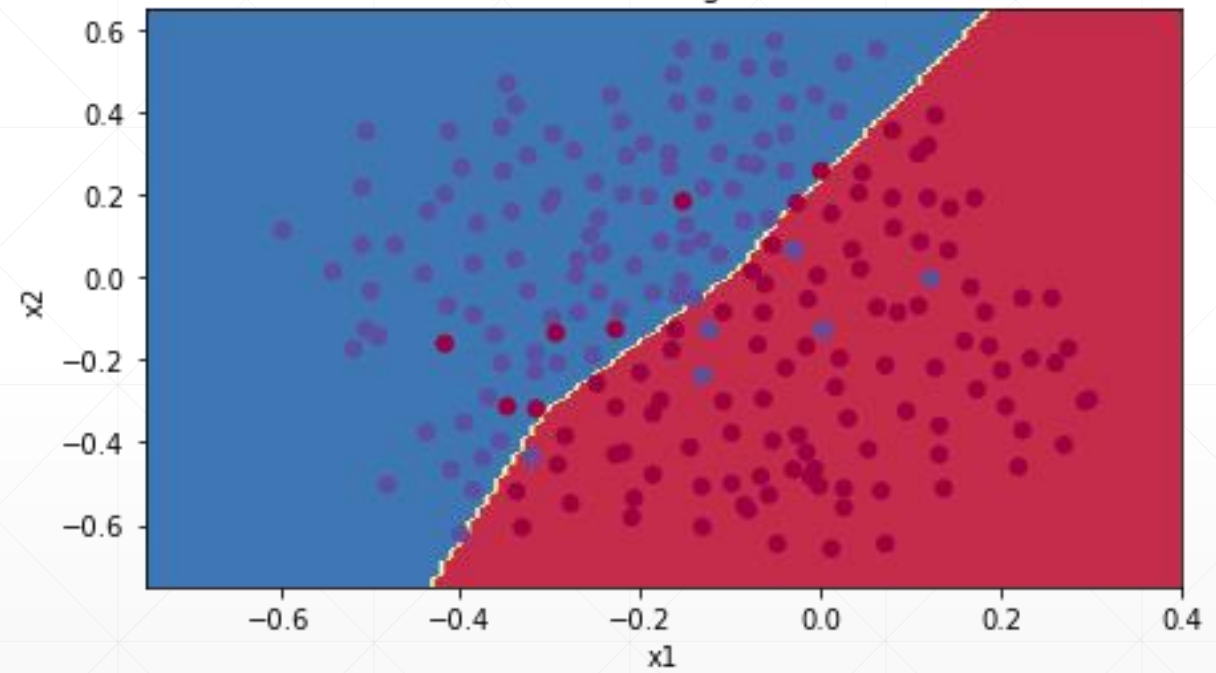
Enforce Weights close to 0

Intuition

Model without regularization



Model with L2-regularization



How

- L1-regularization

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m [y_i \ln \hat{y}_i + (1 - y_i) \ln(1 - \hat{y}_i)] + \lambda \sum_{i=1}^n |\theta_i|$$

- L2-regularization

$$J(W; X, y) + \frac{1}{2} \lambda \cdot ||W||^2$$



L2-regularization



```
device = torch.device('cuda:0')  
net = MLP().to(device)  
optimizer = optim.SGD(net.parameters(), lr=learning_rate, weight_decay=0.01)  
criterion = nn.CrossEntropyLoss().to(device)
```

L1-regularization



```
regularization_loss = 0
for param in model.parameters():
    regularization_loss += torch.sum(torch.abs(param))

classify_loss = criterion(logits, target)
loss = classify_loss + 0.01 * regularization_loss

optimizer.zero_grad()
loss.backward()
optimizer.step()
```




下一课时

动量与学习率衰减

Thank You.
