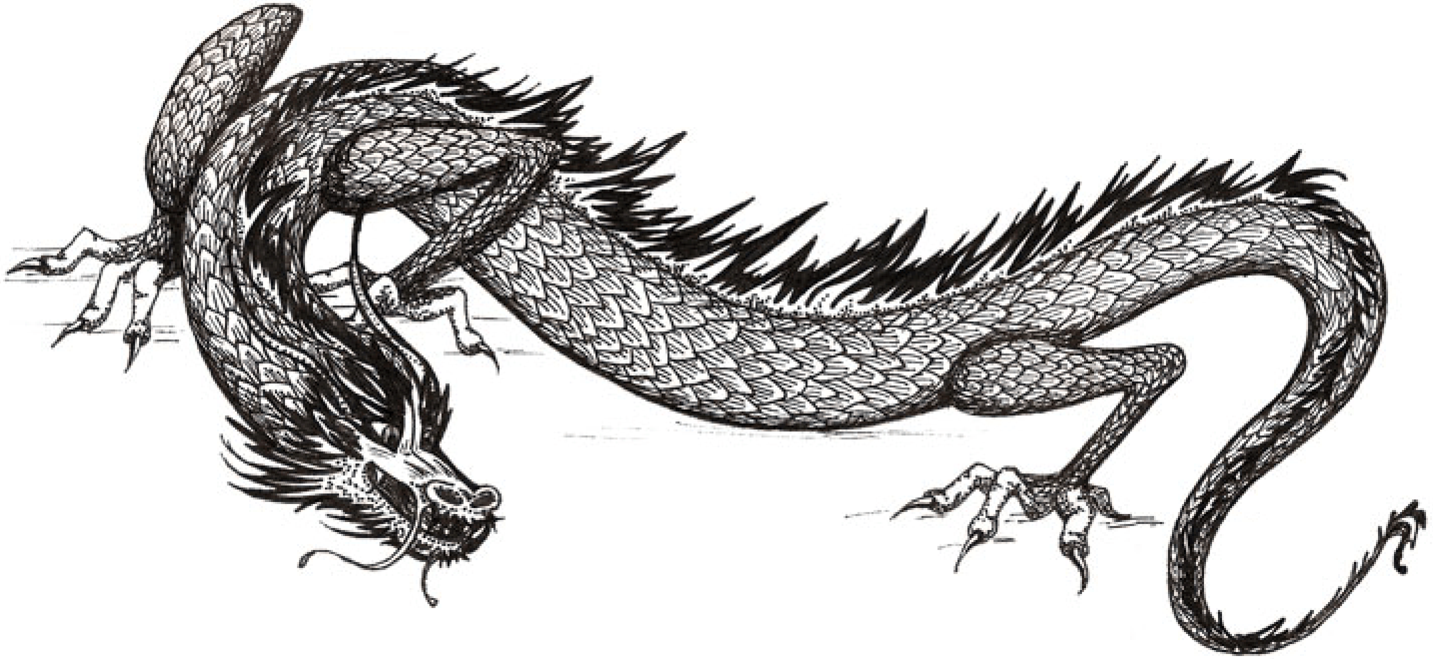
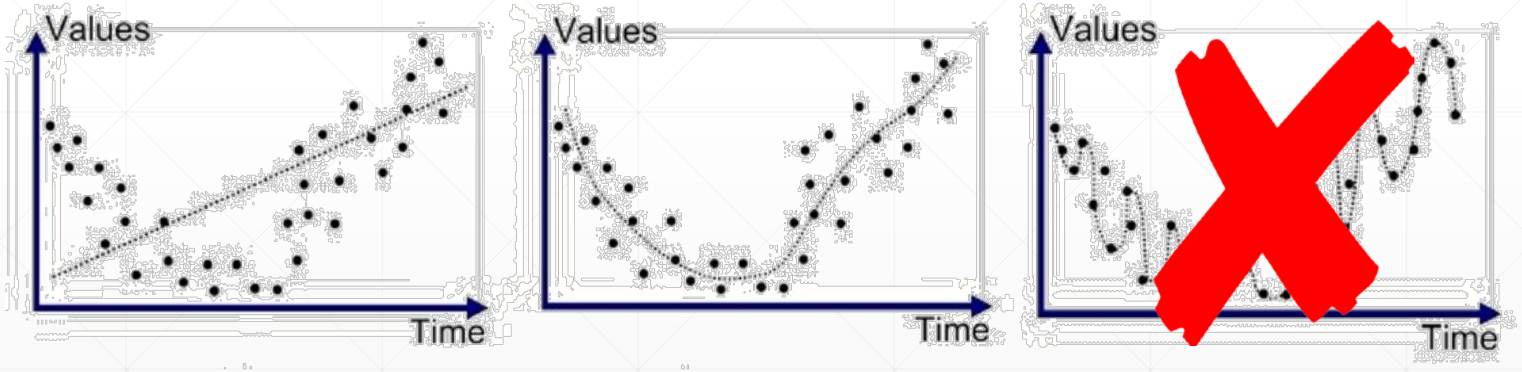


# Regularization

主讲：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

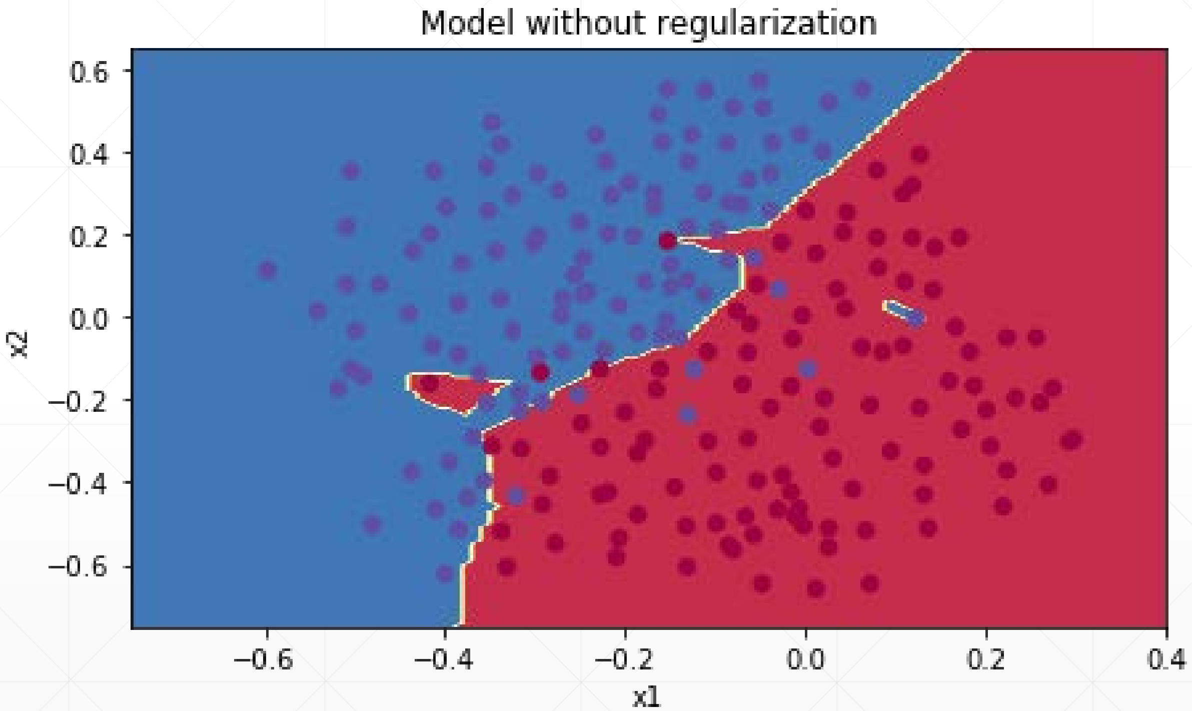
Weight Decay

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

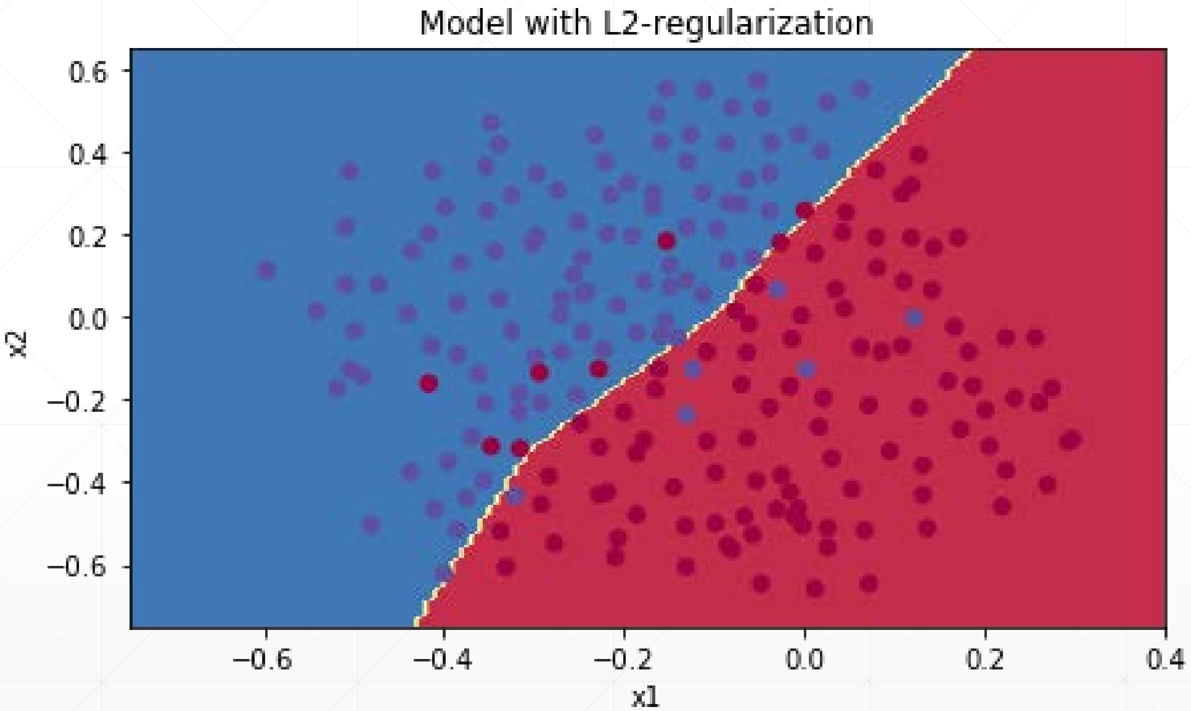
$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \cdots + \beta_n x^n + \varepsilon.$$

Enforce Weights close to 0

# Intuition



Model without regularization



Model with L2-regularization

# How

- L1-regularization

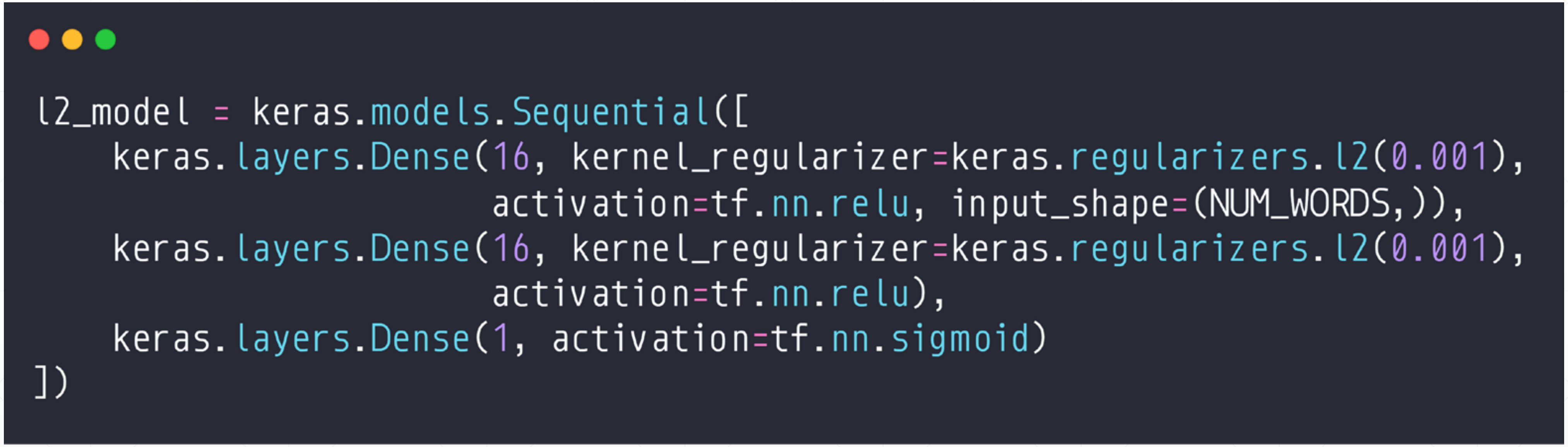
$$J\left(\theta\right) = -\frac{1}{m}\sum_{i=1}^{m}\left[y_i \ln \hat{y}_i + (1 - y_i)\ln(1 - \hat{y}_i)\right] + \lambda\sum_{i=1}^{n}|\theta_i|$$

- L2-regularization

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

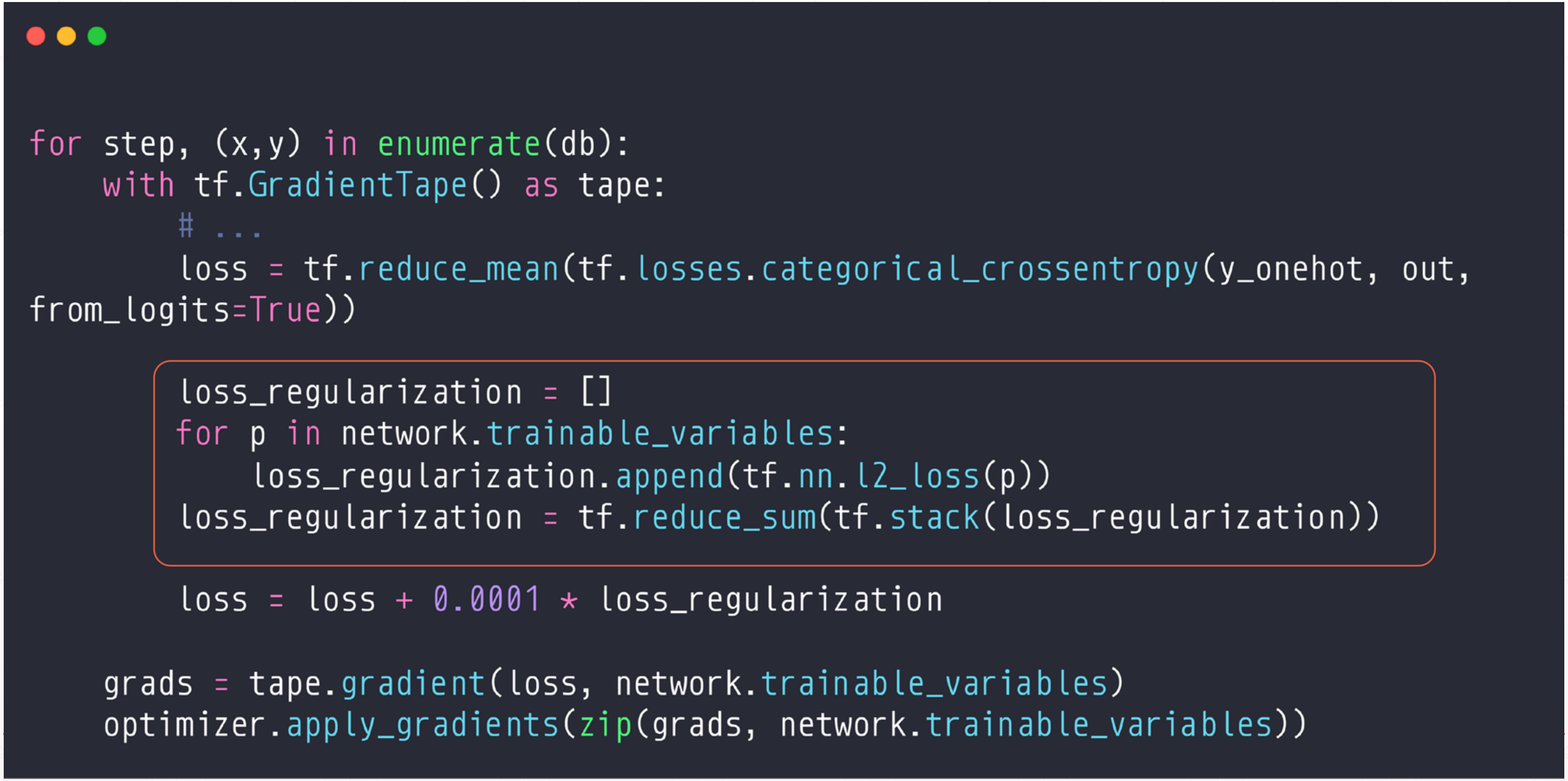
lambda

# One-by-one regularization



```
l2_model = keras.models.Sequential([
    keras.layers.Dense(16, kernel_regularizer=keras.regularizers.l2(0.001),
                       activation=tf.nn.relu, input_shape=(NUM_WORDS,)),
    keras.layers.Dense(16, kernel_regularizer=keras.regularizers.l2(0.001),
                       activation=tf.nn.relu),
    keras.layers.Dense(1, activation=tf.nn.sigmoid)
])
```

# Flexible regularization



```
for step, (x,y) in enumerate(db):
    with tf.GradientTape() as tape:
        # ...
        loss = tf.reduce_mean(tf.losses.categorical_crossentropy(y_onehot, out,
from_logits=True))

        loss_regularization = []
        for p in network.trainable_variables:
            loss_regularization.append(tf.nn.l2_loss(p))
        loss_regularization = tf.reduce_sum(tf.stack(loss_regularization))

        loss = loss + 0.0001 * loss_regularization

    grads = tape.gradient(loss, network.trainable_variables)
    optimizer.apply_gradients(zip(grads, network.trainable_variables))
```

# 下一课时

学习率与动量

# Thank You.