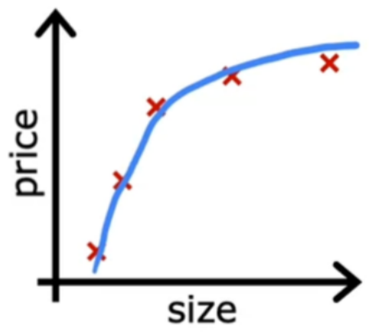
Day-24, Oct 26, 2024.

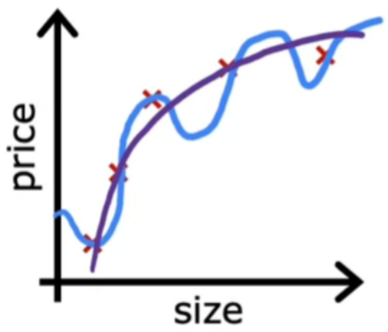
Lasso Regularization.

Feature Selection.

Intuition



$w_1x + w_2x^2 + b$



$w_1x + w_2x^2 + w_3x^3 + w_4x^4 + b$

$w_3 \approx 0$ $w_4 \approx 0$

make w_3, w_4 really small (≈ 0)

$$\min_{\vec{w},b} \frac{1}{2m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)^2 + 1000\,w_3^2 + 1000\,w_4^2$$

$w_3 = 0.001$, $w_4 = 0.002$

Stanford ONLINE DeepLearning.AI Andrew Ng

Regularization

small values $w_1, w_2, \cdots, w_n, b$

simpler model

less likely to overfit

$w_3 \approx 0$

$w_4 \approx 0$

size	bedrooms	floors	age	avg income	...	distance to coffee shop	price
x_1	x_2	x_3	x_4	x_5		x_{100}	y

n features

n = 100

$w_1, w_1, w_2, \cdots, w_{100}, b$

regularization term

$$J(\vec{w}, b) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)^2 + \frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2 + \frac{\lambda}{2m} b^2\right]$$

can include or exclude b

"lambda"

regularization parameter $\lambda > 0$

$y = \beta_0 + \beta_1 x_1^2 + \beta_1 x_2^3 + \cdots + \beta_n x_n$

So, the main idea is to add penalty to the model itself.

It all depends upon λ.

$\lambda = 0$ (overfit)

λ = Optimal (good)

λ = large (underfit)

Regularization

mean squared error

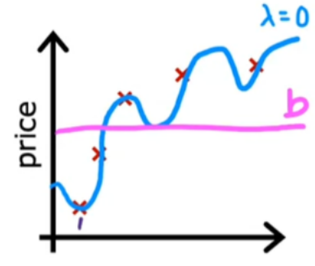
regularization term

$$\min_{\vec{w},b} J(\vec{w}, b) = \min_{\vec{w},b}\left[\frac{1}{2m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)^2 + \frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2\right]$$

fit data

Keep w_j small

λ balances both goals



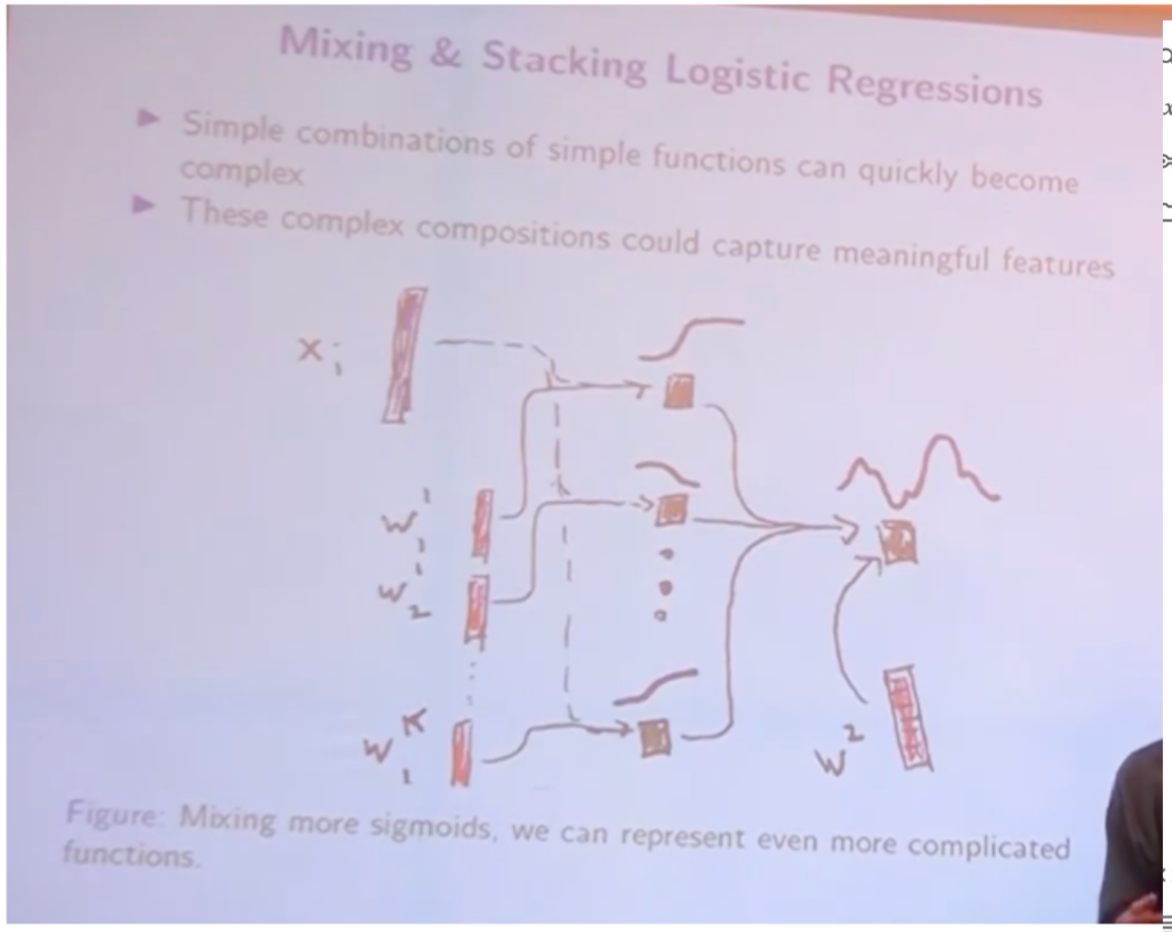
choose $\lambda = 10^{10}$

$f_{\vec{w},b}(\vec{x}) = \underset{\approx 0}{w_1 x} + \underset{\approx 0}{w_2 x^2} + \underset{\approx 0}{w_3 x^3} + \underset{\approx 0}{w_4 x^4} + b$

$f(x) = b$

Choose λ

Mixing and Stacking the Logistic Regression



Key Points:

Simple Combinations, Complex Functions: The slide emphasizes that even simple combinations of logistic regressions can create complex functions.

Capturing Meaningful Features: The slide suggests that these complex compositions can capture meaningful features from the data.

Figure: The figure illustrates the idea of stacking multiple logistic regressions. The output of one logistic regression is used as input to another, creating a network of interconnected functions.

Interpretation:

The diagram shows a series of logistic regression units (represented by the vertical rectangles) connected together. Each unit takes an input, applies a logistic function (sigmoid), and produces an output. The outputs of multiple units can be combined to create more complex functions.

By stacking multiple layers of these logistic regression units, we can create deep neural networks that are capable of learning highly complex patterns in data. This is the fundamental idea behind deep learning.

In essence, the image demonstrates how simple building blocks (logistic regressions) can be combined to create powerful and flexible models capable of learning complex representations from data.

Key Points:

Simple Combinations, Complex Functions: The slide emphasizes that even simple combinations of logistic regressions can create complex functions.

Capturing Meaningful Features: The slide suggests that these complex compositions can capture meaningful features from the data.

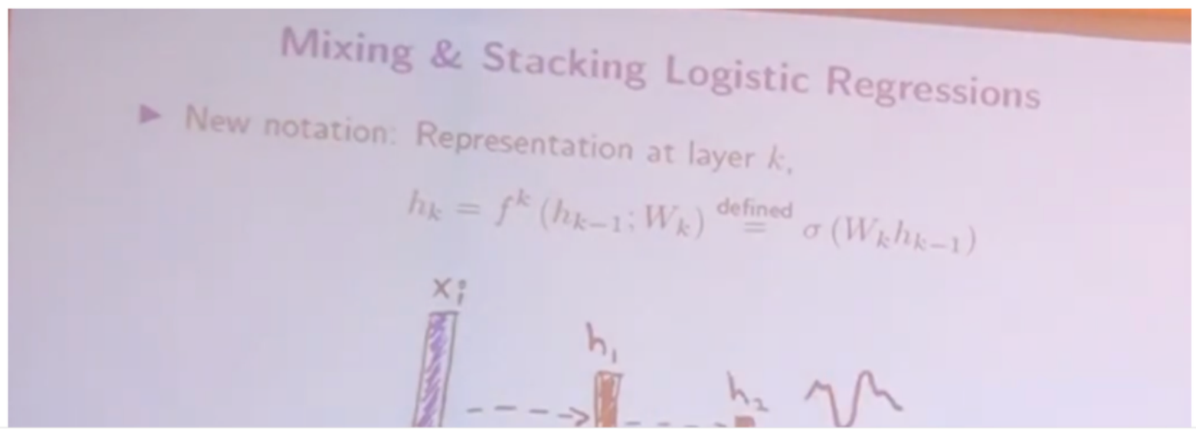
Figure: The figure illustrates the idea of stacking multiple logistic regressions. The output of one logistic regression is used as input to another, creating a network of interconnected functions.

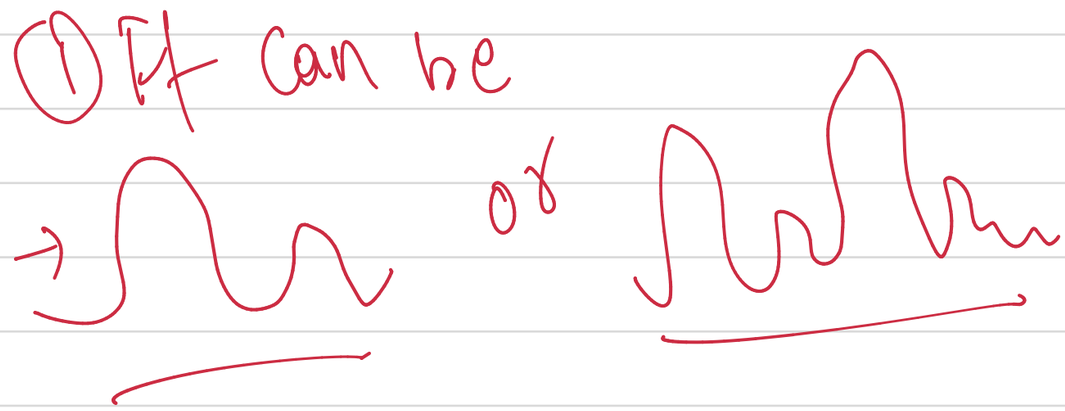
Interpretation:

The diagram shows a series of logistic regression units (represented by the vertical rectangles) connected together. Each unit takes an input, applies a logistic function (sigmoid), and produces an output. The outputs of multiple units can be combined to create more complex functions.

By stacking multiple layers of these logistic regression units, we can create deep neural networks that are capable of learning highly complex patterns in data. This is the fundamental idea behind deep learning.

In essence, the image demonstrates how simple building blocks (logistic regressions) can be combined to create powerful and flexible models capable of learning complex representations from data.





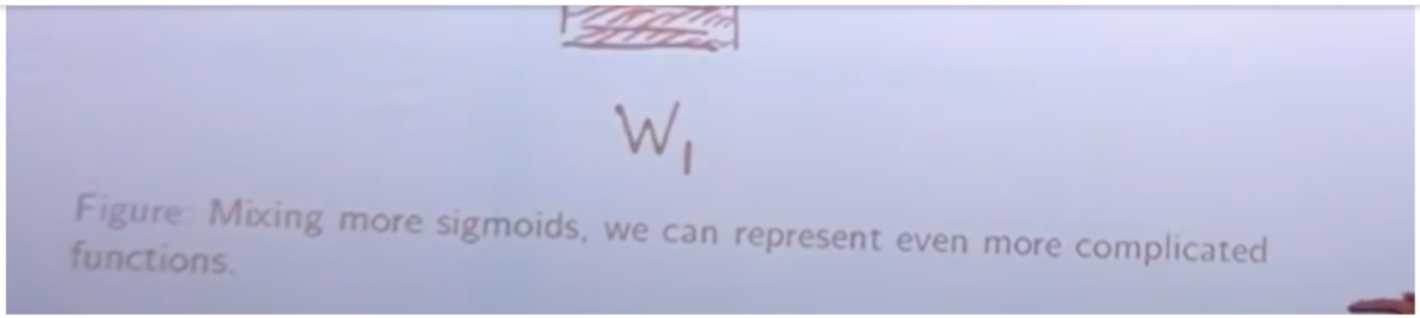


Image Interpretation

The image representation on machine learning, specifically focusing on the concept of stacking logistic regressions to create more complex models.

Key Points:

New Notation: The slide introduces a new notation for representing the representation at layer k: hk = fk(hk-1; Wk). This notation indicates that the representation at layer k is a function fk of the previous layer's representation hk-1 and the weights Wk.

Mixing & Stacking: The slide emphasizes that by stacking multiple layers of logistic regressions, we can create more complex functions.

Figure: The figure illustrates the idea of stacking two logistic regressions. The output of the first layer (h1) is used as input to the second layer (h2), creating a network of interconnected functions. Relevance to Machine Learning

This slide highlights the concept of model stacking, which is a technique for combining multiple models to improve overall performance. By stacking logistic regressions, we can create more expressive models that can capture complex patterns in the data.

Additional Insights:

The slide suggests that stacking logistic regressions can be a powerful technique for creating non-monotonic functions.

This technique can be used in various machine learning tasks, including classification and regression.

In essence, the image demonstrates how simple building blocks (logistic regressions) can be combined to create powerful and flexible models capable of learning complex representations from data.

MultiLayer Perceptrons or Deep Learning: Concept of stacking logistic regressions to create more complex models.

Yes, we can stack and combine the multiple layers.

Concept of Deep Learning

Simple Combinations, Complex Functions: The slide emphasizes that even simple combinations of logistic regressions can create complex functions. Capturing Meaningful Features: The slide suggests that these complex compositions can capture meaningful features from the data.

Figure: The figure illustrates the idea of stacking multiple logistic regressions. The output of one layer is used as input to the next layer, creating a network of interconnected functions. Relevance to Deep Learning

This image represents the core concept of deep learning. By stacking multiple layers of logistic regressions (or other non-linear activation functions), we create deep neural networks that are capable of learning highly complex patterns in data.

Additional Insights:

The intermediate representations (h1, h2, ..., hk-1) learned by the network are crucial for capturing complex features.

The final layer (hk) often uses a softmax function to produce probabilities for different classes in a classification task.

This architecture allows the network to learn hierarchical representations of data, where lower layers capture simple features and higher layers capture more complex and abstract features.

In essence, the image demonstrates how simple building blocks (logistic regressions) can be combined to create powerful and flexible models capable of learning complex representations from data.

Summary of Regularization in Machine Learning(Coursera)

Regularization helps prevent overfitting by adding a penalty to the cost function for large parameter values. Instead of just minimizing the cost function for linear regression, we modify it to include terms that penalize specific parameters (e.g., 1000x + 1000x^2 + 1000x^3). This encourages smaller parameter values, leading to a simpler model.

Key Points:

Generalization: Instead of penalizing specific parameters, regularization is applied to all parameters using the term , where λ controls the penalty strength.

Effects of λ:

(λ / (2m)) * Σ(W_j^2) for j=1 to n

λ=0: No regularization, leading to potential overfitting.

Large λ: Strong penalty, resulting in underfitting (simplistic model).

Connected to Python 3 Google Compute Engine backend

λ=0: No regularization, leading to potential overfitting.

Large λ: Strong penalty, resulting in underfitting (simplistic model).

Optimal λ: Balances fitting the data and keeping parameters small.

The goal is to find a λ that allows the model to capture important features without overfitting. In upcoming videos, we will explore how to implement regularization in linear and logistic regression models effectively.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression

# Generate synthetic data
np.random.seed(0)
X = np.random.rand(100, 1) * 10  # Feature values
y = np.sin(X) + np.random.normal(0, 0.5, X.shape)  # Target values with noise

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Fit a polynomial regression model (degree 15) to create overfitting
poly = PolynomialFeatures(degree=15)
X_poly_train = poly.fit_transform(X_train)
X_poly_test = poly.transform(X_test)

model = LinearRegression()
model.fit(X_poly_train, y_train)

# Predicting on training and test sets
y_train_pred = model.predict(X_poly_train)
y_test_pred = model.predict(X_poly_test)

# Plotting the results
plt.figure(figsize=(12, 6))

# Plot training data and predictions
plt.subplot(1, 2, 1)
plt.scatter(X_train, y_train, color='blue', label='Training Data')
plt.scatter(X_train, y_train_pred, color='red', label='Predictions')
plt.title('Overfitting: Training Data and Predictions')
```

Connected to Python 3 Google Compute Engine back

See how L1 and L2 add penalties to Inputs.