

CS 331

Computer Vision

10 / 26 / 2026

Instructor: Michael Eckmann

Topics

- Questions / Comments?
- Linear Regression homework to be given later today (I'll email you when it is posted)
- Recap on Regression / Classification
- Recap on Linear regression and Logistic regression
- Generalization / Regularization
- Weight Decay

Regression / Classification

- What's the difference between regression and classification?
- Linear regression is a regression technique.
- Would you say Logistic regression is more a classification technique?

Regularization

- Now that we know how to do linear regression and logistic regression for a variety of polynomials, let me explain the idea of regularization.

Capacity

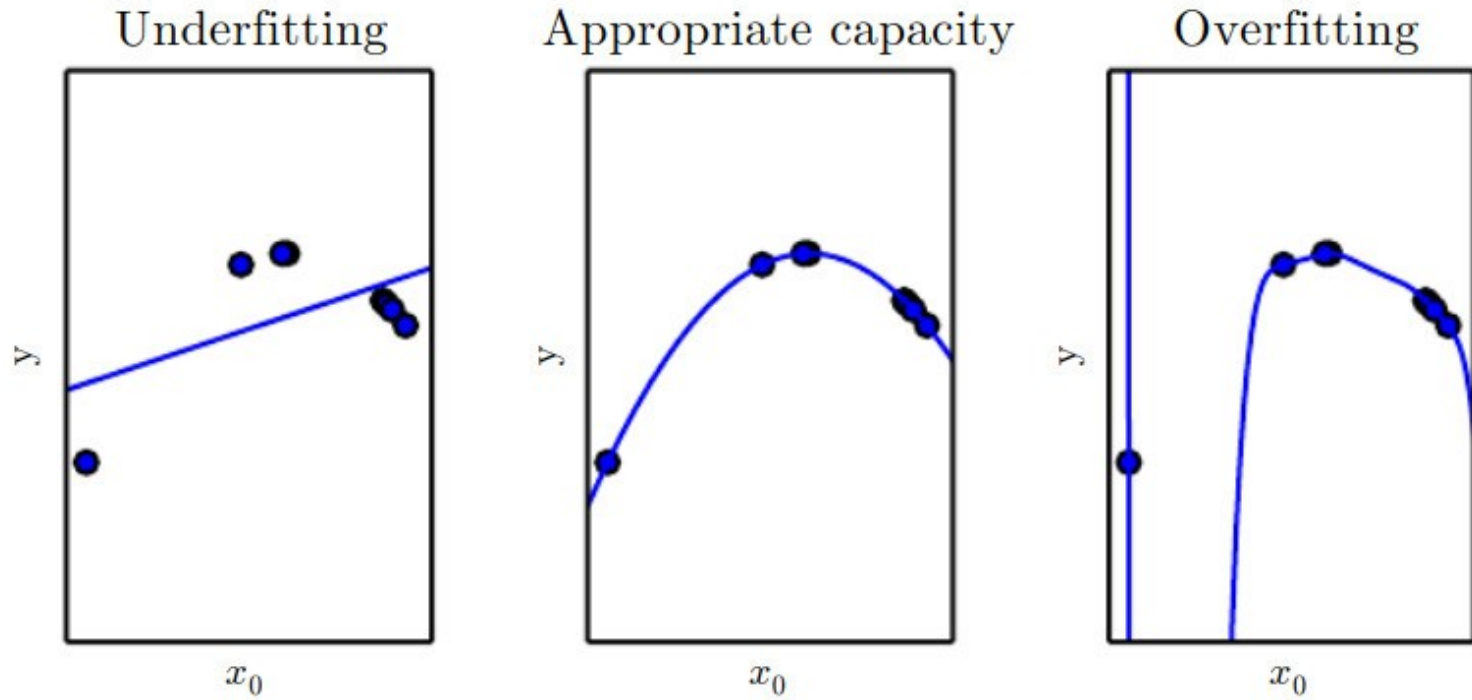
- We could try to learn potentially complex functions for either linear regression or the decision boundary for logistic regression.
- e.g. suppose for linear regression we have an input variable x_1 and an output variable y . We are unsure of the appropriate function to fit (i.e. linear, quadratic, cubic, etc.)
- If we let the linear regression algorithm learn a cubic function, it will learn a cubic function that best fits the training data. This may or may not be a good idea.

Capacity

- Recall: we provide training data for the learning algorithm to learn weights and produces a model.
- We want the model to be able to work well on data that it has not yet seen. Meaning we want the ***test*** data to be predicted correctly.
- Because the training data is only a sample of the entire input space, if we fit the training data to a high degree polynomial, when it would be more appropriate to fit it to a linear equation, then we will end up ***overfitting*** the training data. This is an example of poor generalization.

Capacity

- The consequences of **overfitting** the training data are that the model is not as **generalizable** (to unseen data) as it could/should be.



Capacity

- On the other hand, we do not want to **underfit** the training data either. That is, a lower capacity function than should be learned would cause the *test* data to also not be predicted accurately → lower generalization.

Generalization

- Generalization is the idea that we want the model to predict correctly for unseen data.

Regularization

- **Regularization** is a way for us to provide the learning algorithm a preference for some types of functions over others.
- The Deep Learning text book by Goodfellow et.al. defines regularization this way: any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error.
- Let's make sure we know what is meant by generalization and generalization error Let's look back at those graphs on the Capacity slide.

Regularization

- Regularization for linear regression.
- Recall the idea that we can augment our data by providing columns of the input squared, cubed, etc.
- If we have multiple inputs, we could provide $x_1 * x_2$ as a column, $(x_1 ** 2) * x_2$ as a column, $x_1 * (x_2 ** 2)$ as a column etc. Each with their own weight to be learned.
- In this way we would be learning potentially very complex polynomials.
- To reduce the possibility of overfitting, we could tradeoff some training error for learning a simpler function.

Regularization

- Regularization for linear regression.
- One kind of regularization for linear regression is **weight decay**. The idea is we add in new terms to the loss function such that higher weights are more lossy (cause an increase in the loss function) --- i.e. we tell the loss function to prefer to keep weights small. Small weights result in less overfitting.
- Though we want to balance this penalty for large weights while also keeping the original loss in check.

Regularization

- One kind of **weight decay** is this:
 1. add a new hyperparameter lambda which is a multiplier value that specifies how much we want to penalize large weights.
 2. Add $\lambda * w^T w$ to the loss function (since we are minimizing the loss function, this prefers the weights to have a smaller L2 norm).
 3. The change in loss function makes the partial derivatives have an added $\lambda * w_j$

Note: actual formulas two slides ahead

Regularization

- w_0 is excluded from the penalty and the partial derivative.
- Recall that w_0 is the weight multiplied by x_0 which is always 1 --- it is the constant term and we don't want to penalize it from being a large value.

Regularization

- The loss function with weight decay becomes:

$$J(w) = \frac{1}{2m} \left[\sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)})^2 + \lambda w^T w \right]$$

↑
excluding w_0 .

$$\hat{y} = w^T x.$$

- The weight updates with the partial derivatives of the loss function with weight decay are:

$$w_0 = w_0 - \text{learnRate} * \frac{1}{m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)}) \cdot x_0^{(i)}$$

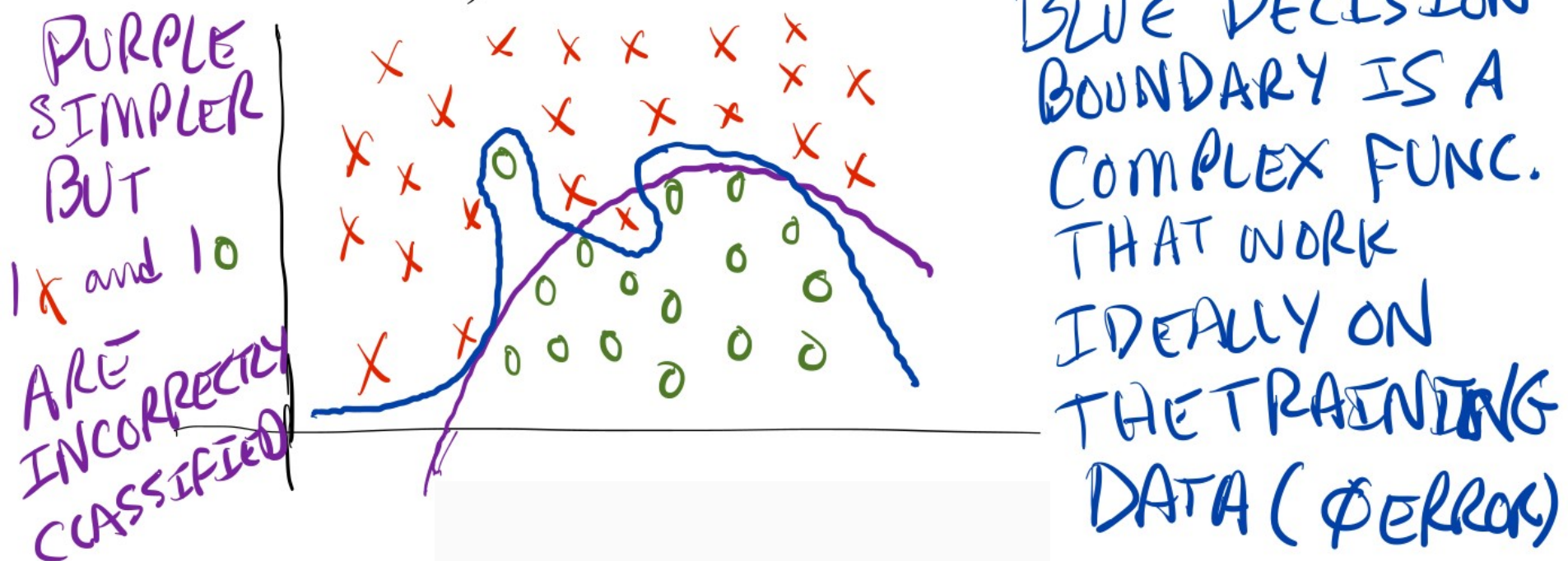
$$w_j = w_j - \text{learnRate} * \left[\frac{1}{m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)}) \cdot x_j^{(i)} + \frac{\lambda}{m} w_j \right]$$

Regularization for Logistic Regression

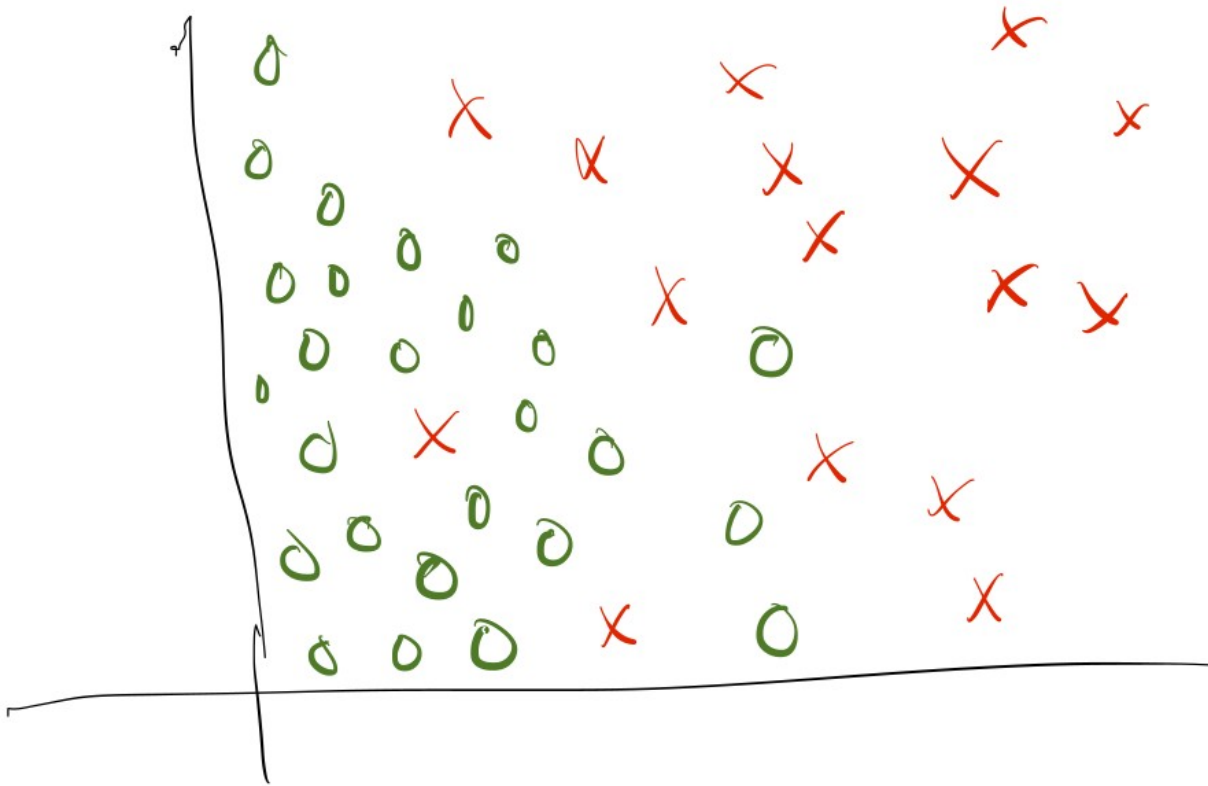
- The same idea can work for logistic regression as well.
- That is, add $\lambda * w^T w$ to the loss function (this prefers the weights to have a smaller L2 norm). And the partial derivatives have an add of $\lambda * w_j$

Regularization for Logistic Regression example

- Example: could learn a complex function and have little error on the training data, or learn a less complex function and have more error on the training data, but the model may be more generalizable (work correctly for unseen data).



Regularization for Logistic Regression example



Regularization for Logistic Regression

FOR LOGISTIC REG.

COST function as before w/ $\lambda * w^T w$.

$$\hat{y} = \frac{1}{1 + e^{-w^T x}}$$

$$w_0 = w_0 - \text{learningRate} * \frac{1}{m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)}) \cdot x_0^{(i)}$$

$$w_j = w_j - \text{learningRate} * \left[\frac{1}{m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)}) x_j^{(i)} + \frac{\lambda}{m} w_j \right]$$

Regularization for Logistic Regression

- If we rearrange the calculations in the weight update computations, we can see how this weight decay actually nudges weights lower.

Regularization for Logistic Regression

if we label $\lambda_{\text{car/keo}} = \lambda$.

$$w_j = w_j \underbrace{\left(1 - \frac{\lambda}{m}\right)}_{\text{this is a \# slightly less than 1.}} - \lambda * \frac{1}{m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)}) x_j^{(i)}$$

this is a # slightly less than 1.

Regularization for Logistic Regression

- Let's try some numbers to see what weight decay actually does.
- Try: $\alpha = 0.01$, $\lambda = 0.1$, $m = 100$
- Try: $\alpha = 0.01$, $\lambda = 0.7$, $m = 100$