Text Classification: Evaluation, Regularization, and Generalization

CS 485, Spring 2024
Applications of Natural Language Processing
https://people.cs.umass.edu/~brenocon/cs485_s24/

Brendan O'Connor
College of Information and Computer Sciences
University of Massachusetts Amherst

- Logistic regression: given features and feature weights, we can predict probabilities on new documents
- We can train the weights to maximize training data likelihood
  - But will it generalize?
  - How to evaluate a classifier model?

Held-out data for evaluation

- How well will my classifier work in the future?
  - Analogy: overfitting for curve-fitting
  - Can we look at classifier accuracy on training data?

Held-out data for evaluation

- Need to diagnose how much your model is **overfitting** the training set
- Data splits are key. Some ways to split:
  - Training set -vs- test set
  - Training set -vs- "validation"/"development" set -vs- test set
  - Cross-validation (within training set) -vs- test set

Cross-validation

- Cross-validation (within training set) -vs- test set
- Advantage: use all labeled data

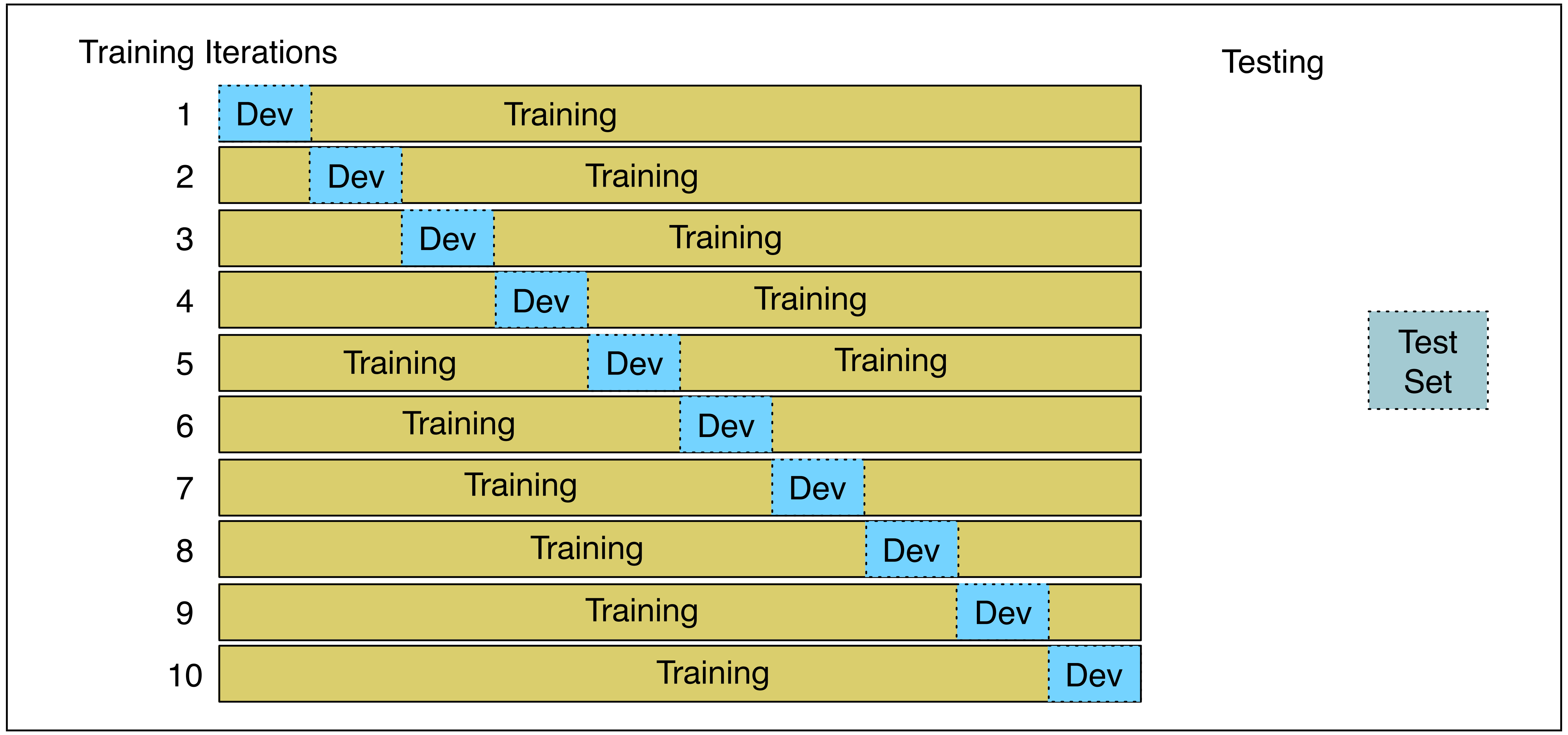


Figure 4.7 10-fold cross-validation

Regularization in Naive Bayes

Regularization in logistic regression

- If "dog" only occurs for class **k**, what weight will it get?
- Consider MLE training:

- Solution: ***regularized*** training for logistic regression

Overfitting and generalization

- Overfitting: your model performs overly optimistically on training set, but generalizes poorly to other data (even from same distribution)
  - Non-classification example: curve-fitting [blackboard]
- To diagnose: separate training set vs. test set.
- How did we regularize Naive Bayes and language modeling? For logistic regression: L2 regularization for training

Regularization tradeoffs

- No regularization <--------------> Very strong regularization

Regularization

- Just like in language models, there's a danger of overfitting the training data. (For LM's, how did we combat this?)
- One method is <u>count thresholding</u>: throw out features that occur in < L documents (e.g. L=5). This is OK, and makes training faster, but not as good as....
- <u>*Regularized* logistic regression</u>: add a new term to penalize solutions with large weights. Controls the **bias/variance tradeoff**.

$$\beta^{\text{MLE}} = \arg\max_{\beta} \; \left[\log p(y_1..y_n | x_1..x_n, \beta)\right]$$

$$\beta^{\text{Regul}} = \arg\max_{\beta} \; \left[\log p(y_1..y_n | x_1..x_n, \beta) - \lambda \underbrace{\sum_j (\beta_j)^2}_{\substack{\text{“Quadratic penalty”} \\ \text{or “L2 regularizer”:} \\ \text{Squared distance from origin}}}\right]$$

"Regularizer constant": Strength of penalty

Logistic regression wrap-up

- Given you can extract features from your text, logistic regression is the best, easy-to-use, method
  - Logistic regression with BOW features is an excellent baseline method to try at first
  - Will be a foundation for more sophisticated models, later in course
- Always regularize your LR model
- We recommend using the implementation in scikit-learn
  - Useful: CountVectorizer to help make BOW count vectors

- chalkboard photos from 2/20 follow

X: features

β: weights

Binary LR: $\sum_j \beta_j x_j$

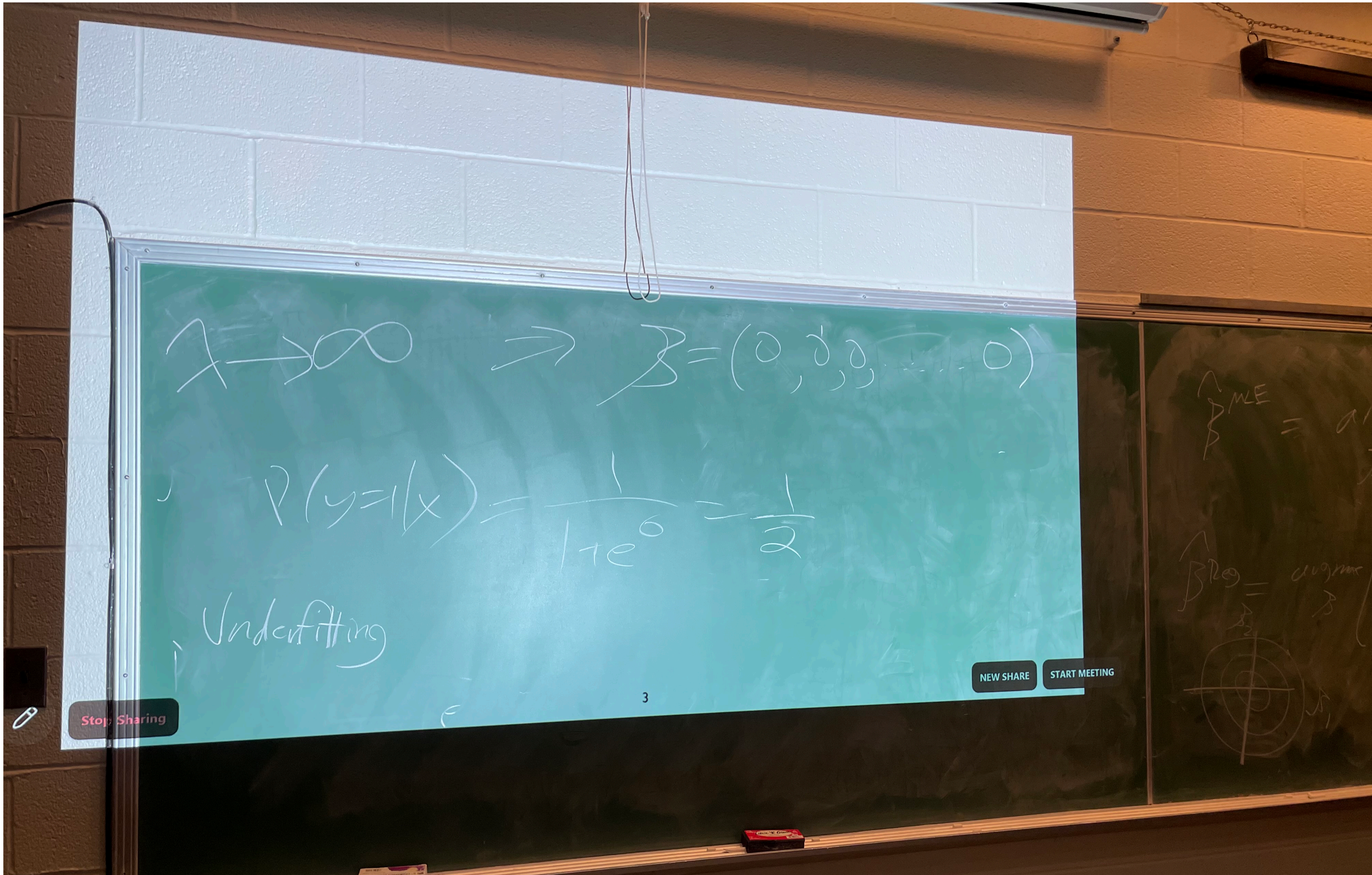
$$P(y=1|x) = g(\overbrace{\beta^T x})$$

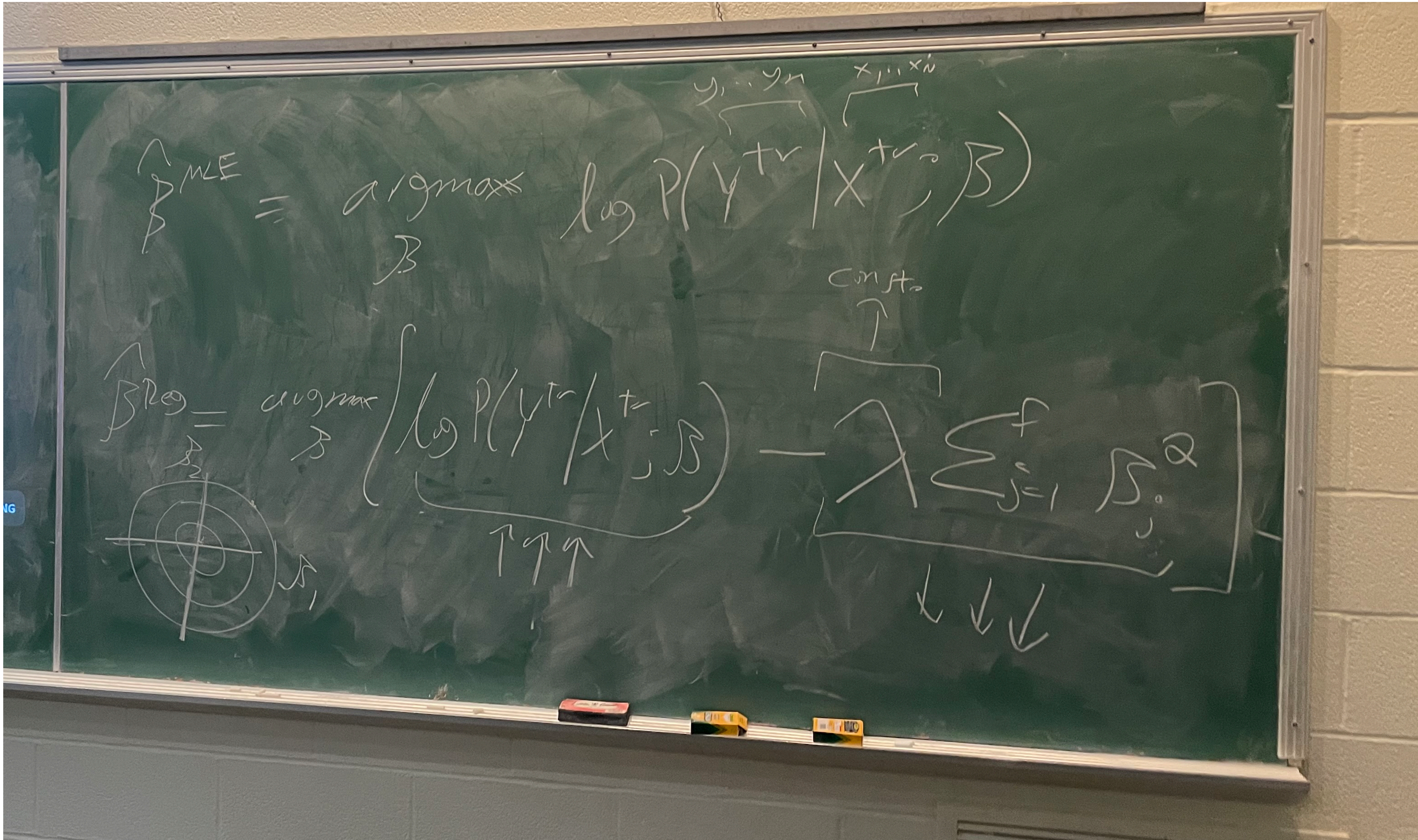
$$g(a) = \frac{e^a}{1+e^a} = \frac{1}{1+e^{-a}}$$

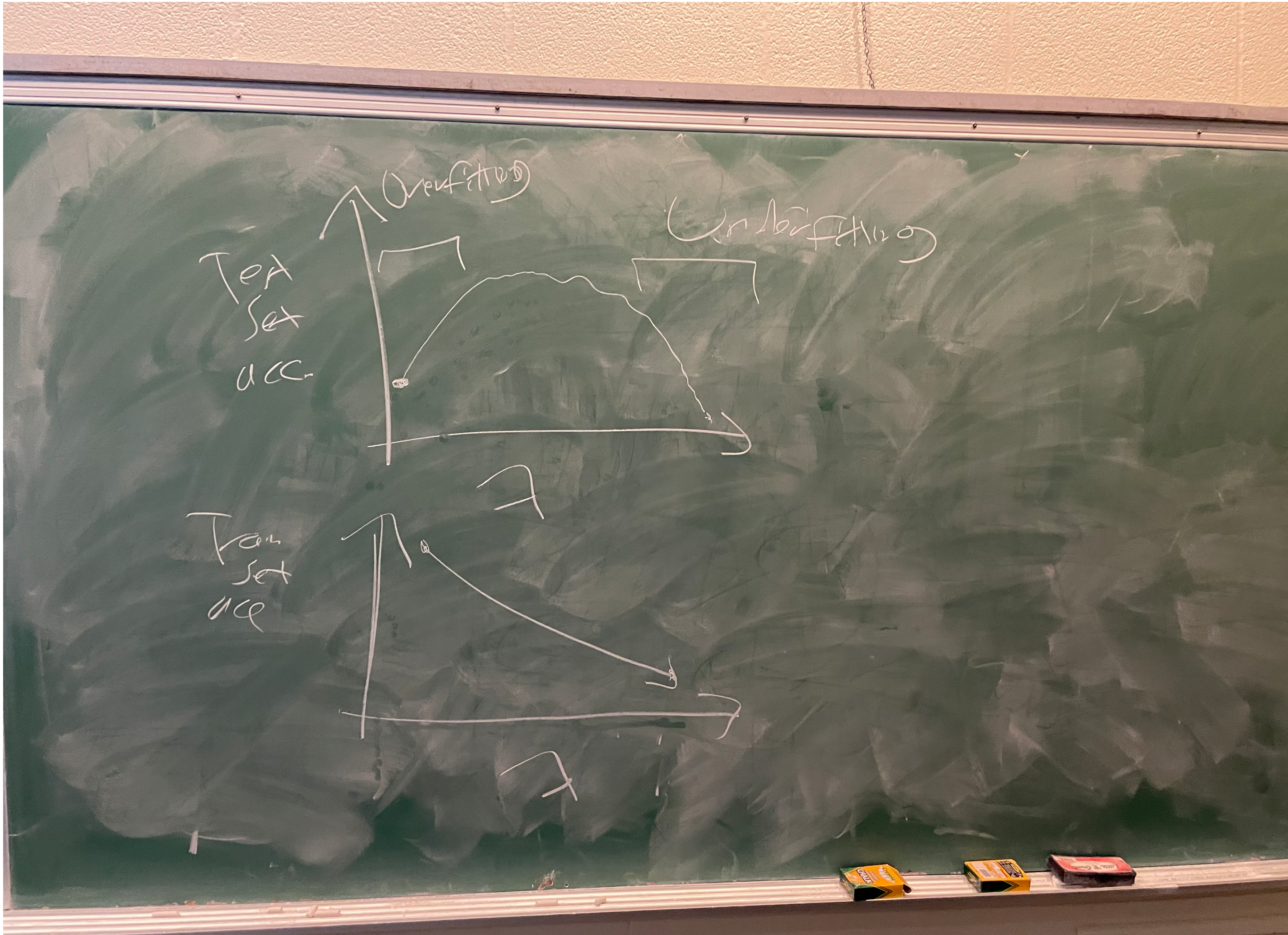
CS 485

Exercise:

- Add Name
- Turn in at end of class


λ → ∞ ⇒ β = (0, 0, 0, ..., 0)
P(y=1|x) = 1/(1+e^0) = 1/2
Underfitting
3
NEW SHARE
START MEETING
Stop Sharing


y1...yn
x1,...xn
$\hat{\beta}^{MLE} = \arg\max_{\beta} \log P(y^{tr} | X^{tr}; \beta)$
const.
$\hat{\beta}^{Reg} = \arg\max_{\beta} \left(\log P(y^{tr} | X^{tr}; \beta) - \lambda \sum_{j=1}^{p} \beta_j^2 \right)$
β_2
β_1


Overfitting
Underfitting
Test Set acc
Train Set acc

- stopped here 2/20

Evaluation metrics

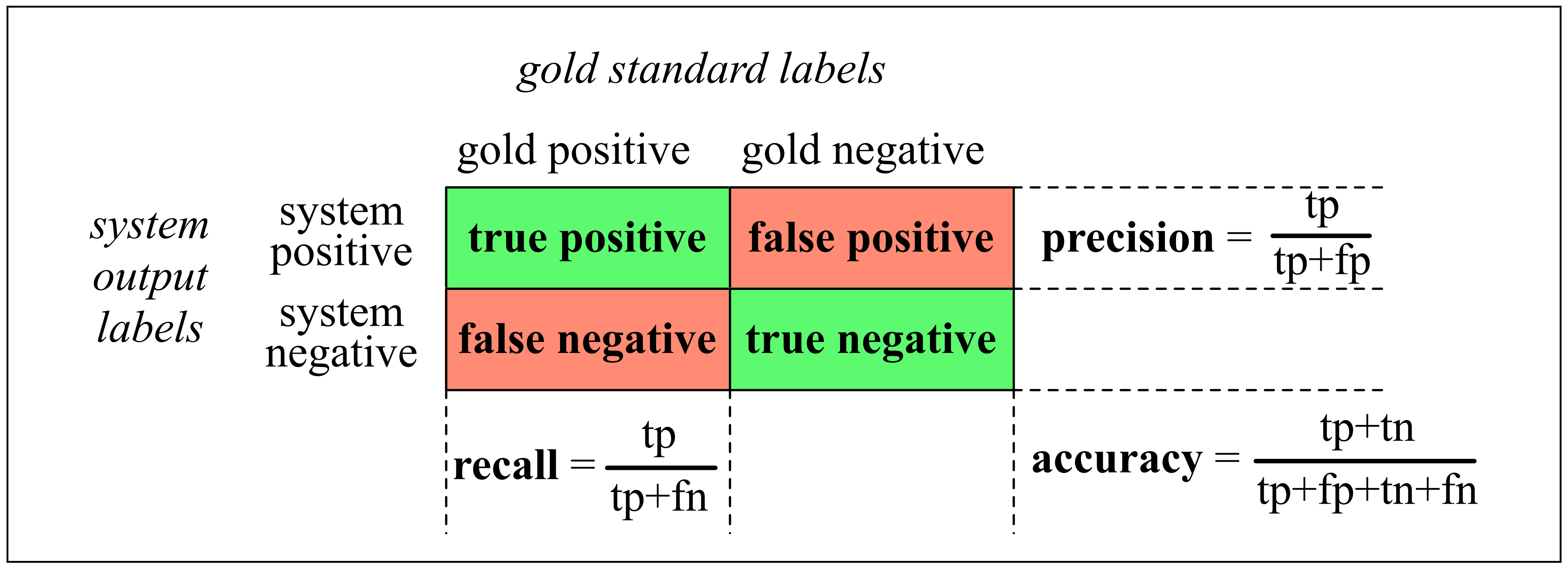


Figure 4.4 A confusion matrix for visualizing how well a binary classification system performs against gold standard labels.

- Accuracy:
  - But do we care about false positives and negatives equally?
  - What about rare classes?
- Precision, Recall, F1

Decision threshold

- Problem: you'd like a higher precision model (for class SPAM), and willing to sacrifice recall.
- Solution: predict SPAM more conservatively: only if probability exceeds a threshold
    - Compare to the default decision rule

Precision-Recall curve

- Different models may trade off precision and recall
- For a single model, different decision thresholds may trade off precision and recall
- View them jointly with a **precision-recall curve**

Do I have enough labels?

- For training, hundreds to thousands of annotations may be needed for reasonable performance
  - Current work: how to usefully make NLP models with <10 or <100 training examples. "Few-shot learning"
- Exact amounts are difficult to know in advance. Can do a **learning curve** to estimate if more annotations will be useful.
- But where do the labels come from? Next week!