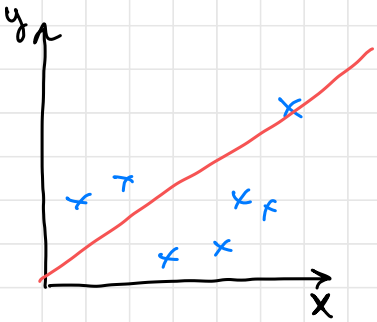
1/23/2024: Overfitting

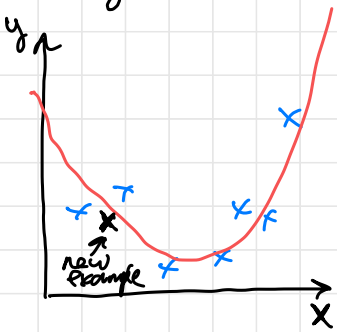


Features = $[1, x]$

"underfitting"

too simple

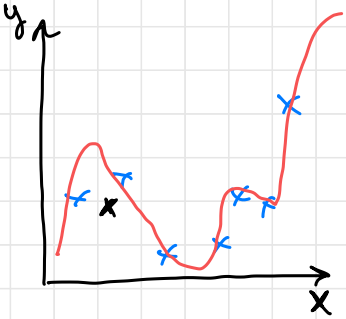
AVOID THIS



Features = $[1, x, x^2]$

good balance between underfitting & overfitting

WANT THIS



Features = $[1, x, x^2, \cdots, x^7]$

zero training loss!

learns function that is too complex.

"overfitting"

↓

can lead to worse generalization to new examples

AVOID THIS

Splitting your dataset

Dataset

≈70% → Training Data

What we use to learn parameters/ train the model

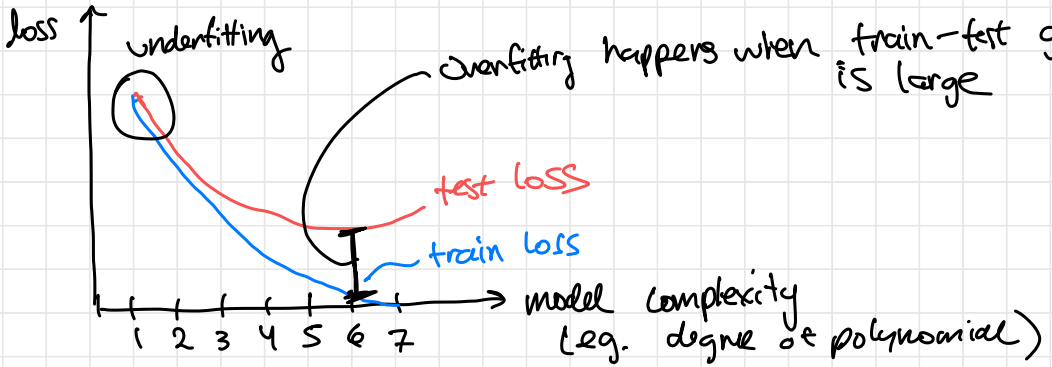
≈10% → Development Data (or "validation")

Use to choose hyperparameters

≈20% → Test Data

Evaluate how well model generalizes to new examples

Must be "held out" hidden from model during training



Big Question: How should we choose right level of model complexity?

Term: **hyperparameter** = Any setting of a ML model that is not a learned parameter

- Which features?
- Learning rate
- How long to run gradient descent

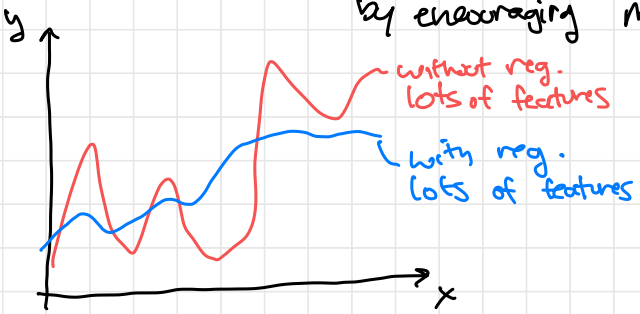
To choose hyperparameters:

1. Train models with different hyperparameter values
2. Evaluate on dev set
3. Choose model with best dev set loss (or accuracy)
4. Evaluate this model only on test set

Why not use test set to choose hyperparams?
Still a form of "cheating"
A: Model should only get one chance to take "final exam" = test set

Dev set ≈ practice exam

Regularization: A technique to reduce overfitting by encouraging models to be "simpler"



L2 Regularization:
Encourage L2 norm of parameters to be small

$$\|w\| = \sqrt{\sum_{j=1}^{d} w_j^2}$$

How? Add term to loss function. e.g. for regression:

$$L(w) = \underbrace{\frac{1}{n}\sum_{i=1}^{n}\left(w^T x^{(i)} - y^{(i)}\right)^2}_{\text{original loss}} + \lambda \|w\|^2$$

$\|w\|^2 = \sum_{j=1}^{d} w_j^2$

new hyperparameter!
Can be any number ≥ 0
$\lambda = 0$ then no regularization
larger $\lambda \Rightarrow$ stronger regularization

How does this affect gradient?

$$\nabla_w L(w) = [\text{old gradient}] + \lambda \cdot 2w$$

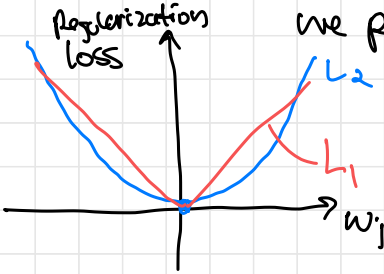
because $\frac{\partial}{\partial w_j}\sum_{k=1}^{d} w_k^2 = 2w_j$

G.D. update:

$$w \leftarrow w - \eta\left([\text{old gradient}] + \lambda 2w\right)$$

Subtract multiple of w from w
ie move towards origin
("weight decay")

L_1 Regularization: Similar to L_2 reg but



we penalize L_1 norm of w

$$= \|w\|_1 = \sum_{j=1}^{d} |w_j|$$

How? Add $\lambda \|w\|_1$ to loss

$$\text{Sign}(z) = \begin{cases} +1 & \text{if } z > 0 \\ -1 & \text{if } z < 0 \\ 0 & \text{if } z = 0 \end{cases}$$

What is $\nabla_w \|w\|_1$?

$$\frac{\partial}{\partial w_j} \sum_{k=1}^{d} |w_k| = \frac{\partial}{\partial w_j} |w_j| = \text{sign}(w_j)$$

$$\text{So } \nabla_w \|w\|_1 = \begin{bmatrix} \text{sign}(w_1) \\ \vdots \\ \text{sign}(w_d) \end{bmatrix} = \text{Sign}(w)$$

Encourages "sparse" weights

Compare with $\nabla_w \|w\|^2 = 2w$

L_1: Always take constant-sized step — Encourage some w_j's to be exactly 0

L_2: Take small step for small w_j, large step for large w_j — Avoid really big w_j's

Takeaway: for L_1, learn w_j's $= 0 \iff$ ignore feature j

So we learn to select only useful features