Pseudocode for MLBot 1.0:

Make modeling assumptions

While True:

train on training set

evaluate on validation set ← ≈ acc on unseen data

if train acc ≈ val acc ≈ good enough

declare victory & break;

else:

revise modeling assumptions and/or tweak hyperparameters ← e.g., k

eval on ~~??~~ test set

Problem?

Solution?

Data split best practice:

Labeled data:

Train	Val	Test

What fraction? Depends on data set size and noise.

Smaller train set → higher variance in trained model

larger train set → higher variance performance estimate

Alternative: For small data, take full advantage of as much data as possible:

fold 1	fold 2	fold 3	...	fold k	test

"k-fold cross-validation":

```
for i in range(k):
    train on all folds except i
    v_i ← validate on i
val_acc = mean(v_i)
```

When is this a good idea?

+ more training data → better models
+ lower variance val accuracy measurement
- need to train K times

"leave-one-out cross-validation":

$K = n$

Tools in the fight against overfitting

✓ 1. Train/val/test splits

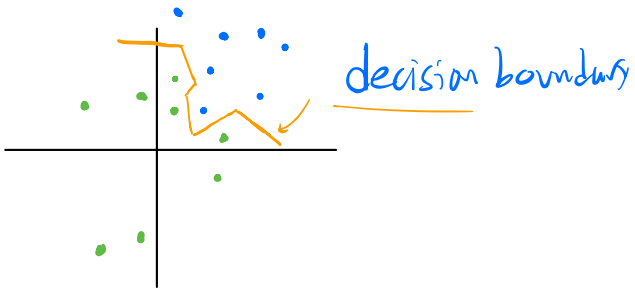
2. Regularization: build a "simplicity prior" into your model.

often, simplicity/flexibility can be controlled by hyperparameters

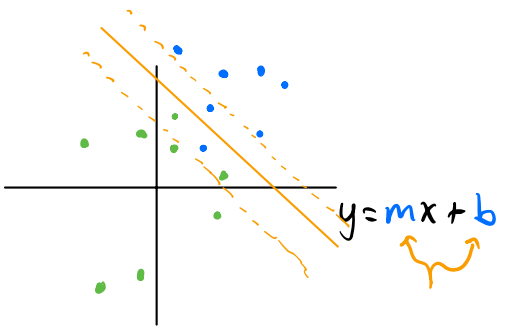
Example: K in kNN

Aside: parameters and hyperparameters

kNN is called nonparametric



Example of a parametric model:
a linear classifier.



Activity: Investigate an example of hyperparameter tuning a linear classifier.