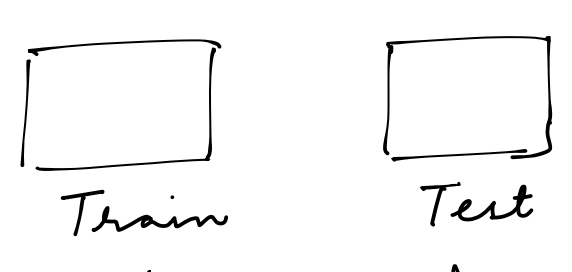
Generalizing to unseen data

Reducing Train Loss → Done?

ML → Generalize to unseen data

Overfitting & Underfitting



↳ same dist ← ML's assumption

Very good ↑ Test ↓ ← Overfitting

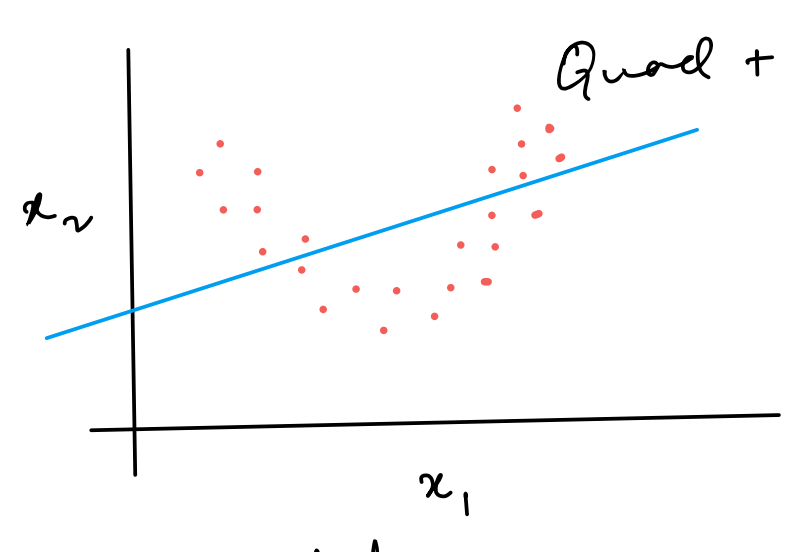
Poor on train set ↓ Test ↓ ← Underfitting

Poor train ↓ Test ↑ ? Lucky set

Bias & Variance

$\theta_0 x_0 + \theta_1 x_1 < \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_0^2 + \theta_3 x_1^2 + \theta_4 x_0 x_1$

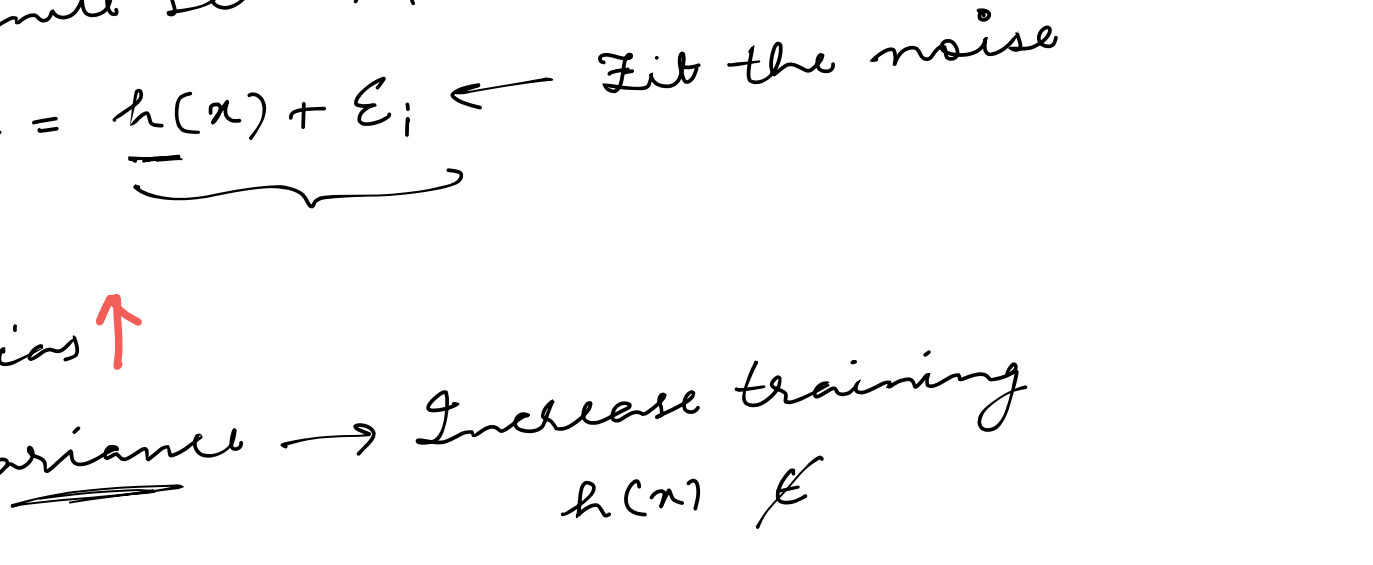
Quad + Linear



Linear model

↳ High bias → Your family of models fundamentally cannot approximate $h(x)$

$y = h(x)$



data from same dist.
↳ different models learnt.

5^{th} order

Training error $\simeq 0$

Training data → Sample of an infinite set

Finite set → Randomness

$y = h(x) + \varepsilon_i$ ← Fit the noise

Bias ↑

Variance → Increase training

$h(x)$ ε

Bias: Your test set error if you trained your model on virtually infinite data

Variance: Captures your model's sensitivity to randomness.

Model Validation / Selection

100% ← Train ✗

Train Validation **Test**

80% train
10% valid ← cal loss } Cheating
10% test ← After many iterations on the previous two.

1. Hold out cross validation

carve out a valid set.
eval on val set

Train ← Linear / Quadr

Val ← Linear beats Quad

80% 10% 10%
800 100 100

1,000,000

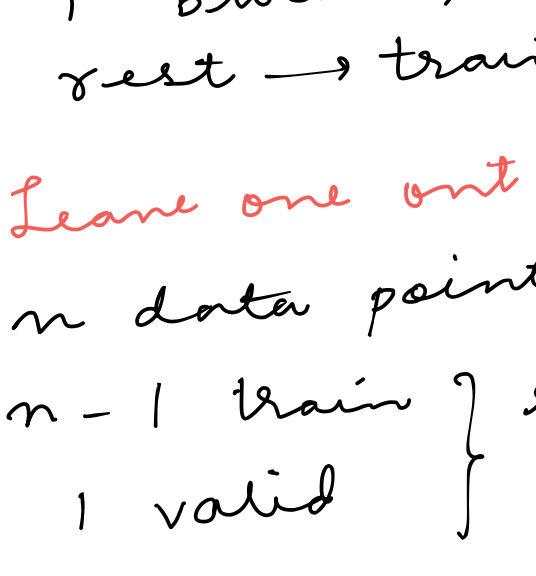
1,00,000 (Valid) 1,00,000 (Test)

{ 1% valid
{ 1% test

Numbers → 1000 data

2. K-fold cross validation

Hold out wastes data



$Km = n$

Validate

0^{th} block → valid
rest → train } val loss

1^{st} block → valid
rest → train } val loss

→ average

3. Leave one out cross val

n data points

n − 1 train
1 valid } repeat n times

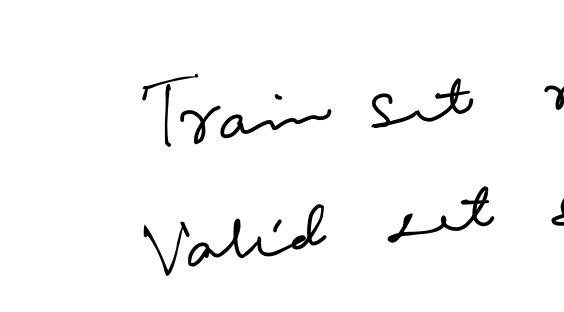
If → Model selection → Linear / Quadratic

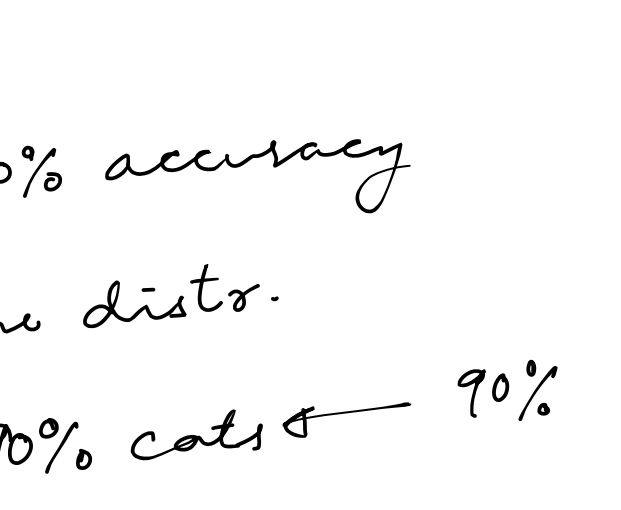
Linear → Train + val (Retrain)

Deploy.

Stratification

Respect the distribution





cat 50%
dog 50%

↑ Bad split!

1. Shuffle data
2. Stratification

80% → dogs ← dog
20% → cat

80% accuracy

Train set respects the distr.

Valid set skewed 90% cats ← 90%

60%	20%	20%
Train	Val	Test
60:20:20	60:20:20	60:20:20

F1 score

Cross val done → Overfit

Train ✓
Val ↓

1. Get more data.
2. Regularization ← Restrict your model's power.

$J(\theta) = \frac{1}{2m}\sum_{i=1}^{n} (\theta^T x^{(i)} - y^{(i)})^2 + \lambda R(\theta)$

$R(\theta)$ Regular. ↑

simple → $R(\theta)$ ↓

Regulariser $= \|\theta\|_2 = \left(\sum_{i=0}^{d} \theta_i^2\right)^{1/2}$ ← Ridge regression

$\|\theta\|_1 = \sum \|\theta_i\|$ ← Lasso regression

$J(\theta) := J(\theta) + \lambda \|\theta_i\|_2^2$

θ_i should be high
Will $R(\theta)$ still push it down.

Dataset → Extra features ← Feature selection

Length
Breadth }

$R(\theta)$ → side effect → Feature selection

→ Lasso → $\|\theta\|_1$ → $\theta \to 0$
$\|\theta_2\|$ → $\simeq 0$

Why does Lasso do feature selection & ridge regression doesn't.

---

1. Minimizing train Loss isn't the final goal.
   ↳ Generalize to unseen data.
2. Is my model overfitting or underfitting
   ↳ Bias and variance
3. Bias → your model is fundamentally incapable of representing the data.
   ↳ Pump more data? ✗
4. Variance → Sensitive to randomness in the finite data set.
   ↳ Pump more train data. ✓
   ↳ Regularize
   ↳ Use a simpler model.
5. How do get an idea of bias/var?
   ↳ Cross-validation
   1. Hold out → Single val.
   2. K-fold → K valid, K train } average K valid sets
   3. Leave one out → use 1 datum at val set } repeat for each datum

   ↳ Always shuffle your data before splitting.
   ↳ In case of classification → Stratify
6. Regularization → L_2 norm of θ → Ridge regression
   L_1 norm of θ → Lasso regression
7. Lasso automatically does feature selection.
   Why? → Resource

Code LR → scikit learn

Linear regressor

LR → Random θ
Update rule → θ improves
Convergence criteria

↳ Classes
Numpy
Pandas }