From last time

total generalized error $= E_{x,y,D\text{-trained}}\left[(h_D(\vec{x})-y)^2\right]$

$$= \iiint (h_D(\vec{x})-y)^2 f(\vec{x},y,D)\, dy\, d\vec{x}\, dD$$

★ $=$ [illegible] $\vec{x}) + \underbrace{E_{\vec{x}}\left[(\bar{h}(\vec{x})-\bar{y}(\vec{x}))^2\right]}_{\text{"bias squared of estimator"}}$

$+ \underbrace{E_{\vec{x},y}\left[(\bar{y}(\vec{x})-y)^2\right]}_{\text{"noise of estimator"}}$

sensitive your classifier is to changing from one D to some other set D'

How much are we overfitting to that particular D?

★ The estimator bias shows that even if you had infinite data your classifier would be imperfect because of its model (for example, it may be linear $[x_1]$ when it should be quadratic $\begin{bmatrix}1\\x_1\\x_1^2\end{bmatrix}$)

☆ This error is inherent to our choice of algorithm

An algorithm is bias if no amount of data will improve it because you are underfitting

☆ Classifier noise [illegible] average error of the best possible MAP [illegible] (You cannot do better than this) with respect actual labels.

Ex: two plants with exact same heights, but different diameters

$\therefore$ total generalized error = var of $h_D(\vec{x})$ + bias of any h of this form + noise

☆☆

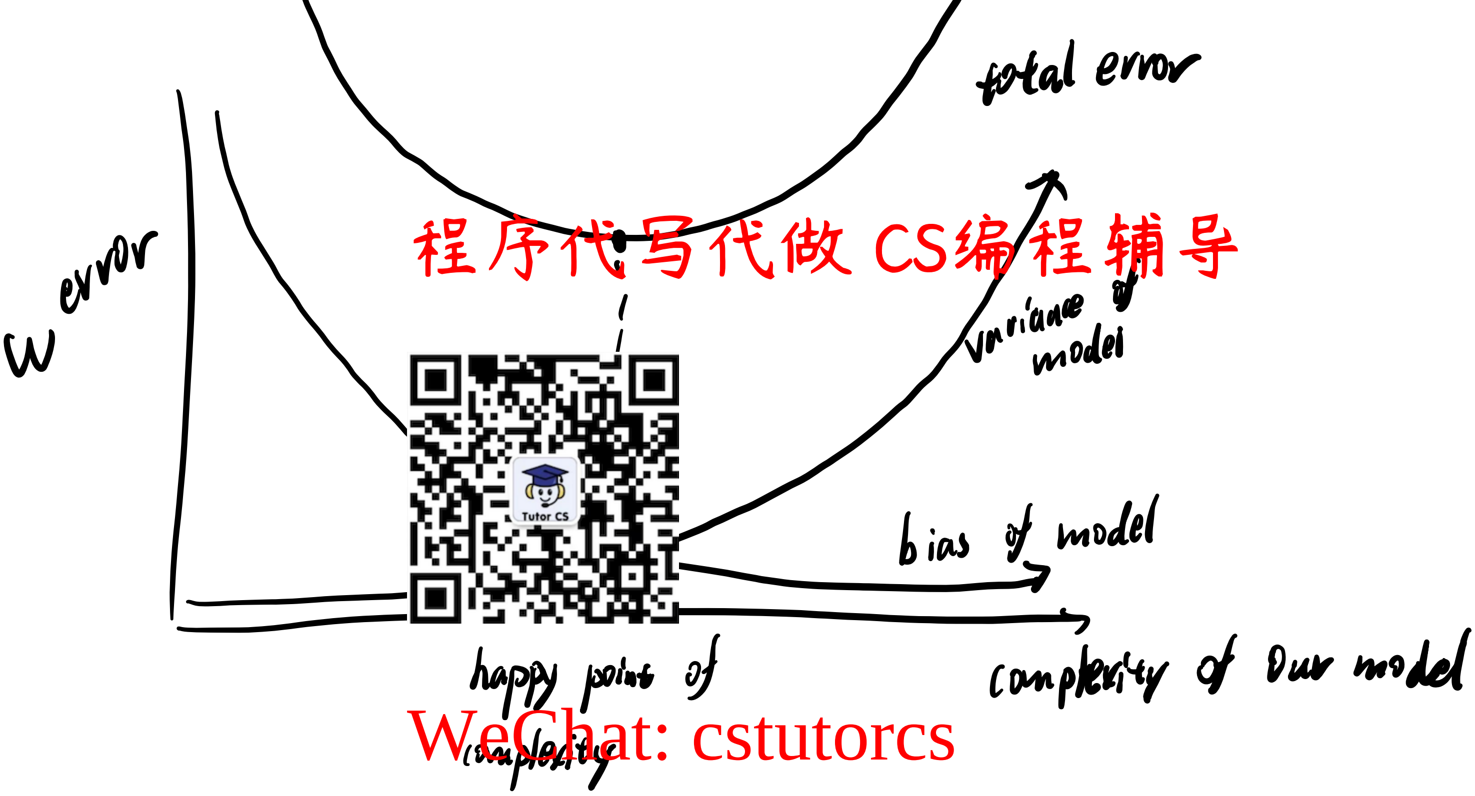
$\underbrace{\text{this form}}$

Ex linear
quadratic
cubic

So, for a given "N", the total expected (generalized) error will include two components that react in opposite direction when we change the complexity of our model :

* If we try to reduce bias by increasing complexity, then we are in danger of overfitting and high variance.

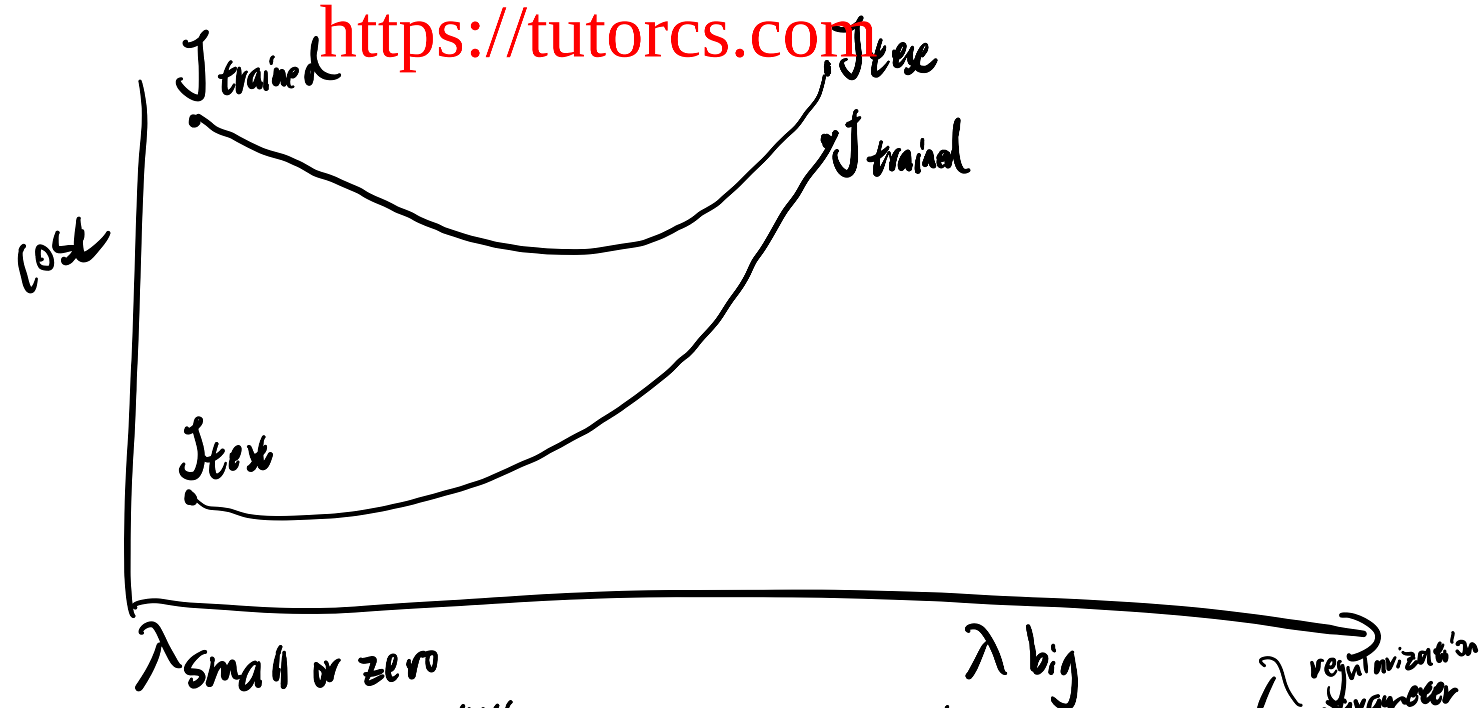
* If we try the variance by reducing complexity, th e in danger of underfitting and we increase bias



Recall, one controller of complexity we used was

REGULARIZATION

When we include $\lambda \|w\|^2$ in the minimization



high complexity overfitting

$\Rightarrow$ $J_{trained}$ small

J_{test} big

low complexity

underfitting

$\Rightarrow$ $J_{trained}$ big

J_{test} big

LEARNING CURVES

Acquiring [illegible] ble data, or replacing attribute are typically [illegible] in man-hours (surveys, polls data mining servis, sanitizing, prepping, isolating and visualing, etc)

If you're going to spend man-hours, well, first find out if it will help

Def | Learning curves = graphs that track training cost (without regularization) against test cost is "N" gets bigger

$$D = \{ \underbrace{(\vec{x}_1, y_1)\ (\vec{x}_2, y_2)\ (\vec{x}_3, y_3)}_{\text{use only 40\% of data}} \cdots (\vec{x}_n, y_n) \}$$

first round — use only 40% of data

second round — use only 50% of data

third round

use only 60% of data

etc

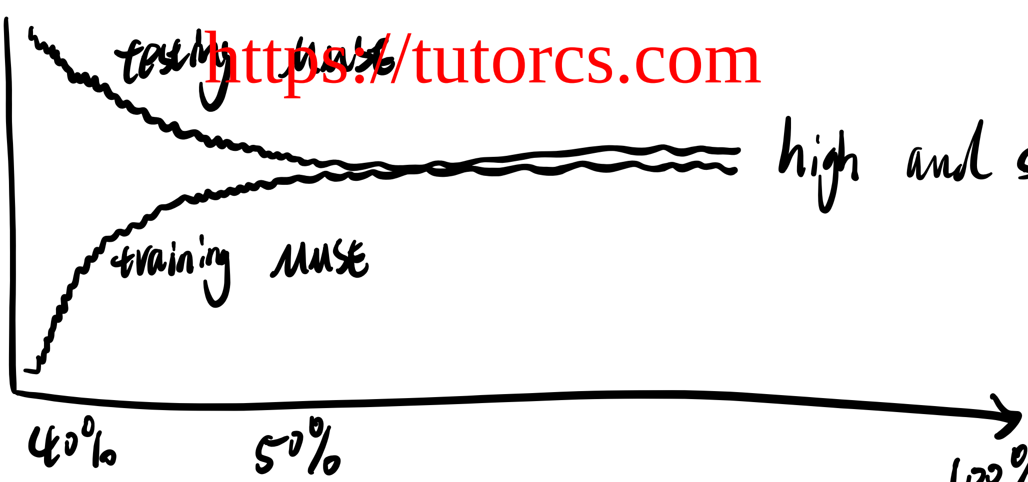
n-th round use

Note: in each round, set aside 70% for training and 30% for testing

what will we see if our model is underfitting (high bias)?

MMSE

$$\frac{1}{N_i}\sum_{k=1}^{N_i}\left(h(\vec{x})-y_k\right)^2$$



high and stay high

here

J_{train} is smaller as we are free to over fit with little data

J_{test} is big (since overfitting is apparent)

both J_{test} and J_{train} are big and above the same

$\Rightarrow$ more data will not help

what will we use when model is overfitting (high variance)?

with few data points, J_{train} will be small

J_{test} will be big

J_{train} goes higher

J_{test} will go lower

there will be a gap

SUPPORT VECTOR MACHINES (SVM)

supervised classification learning with MANY open source implementation; they are very fast.

Our training instances are the set D of $(\vec{x}^{(j)}, y_j)$

where $y_j = 1$ or $y_j = -1$

LINEAR SVM

First: Assume our training data is linearly seperable

- SVM tries to find the "WIDEST STREET" whose center is the boundary that seperates the two decision regions

- so, we want to find that optimum boundary that maximizes the margins between the bondary line (hypperplane, H) and the nearest training feature vectors on both sides.

First, we employ a mathmatical covarience:

we set the rule that for all training data:

$\vec{w}\vec{X}_+ + b \geq +1$ for all training data with $y_+ = +1$

and $\vec{w}\vec{x}_- + b \leq -1$ for all training data with $y_- = -1$

Trick to combine both into one:

$y_i(\vec{w}^T\vec{x}_i + b) \geq 1$ for all training instances (♡)

Recall, for ... fication, for boundary hyperplane H,

$g(\vec{x}) = \vec{w}^T\vec{x} + b = 0 \quad \forall \vec{x} \in H$