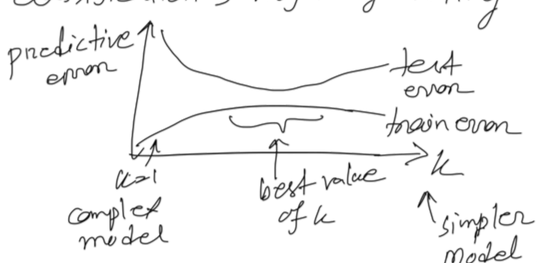
Naive Bayes

$$P(y=1 \mid x_1=c, x_2=0) = \frac{P(x_1=c \mid y=1)\, P(x_2=0 \mid y=1) P(y=1)}{P(x_1=c, x_2=0 \mid y=1) + P(x_1=c, x_2=0 \mid y=0)}$$

inductive bias
set of assumptions on the learner before it sees any data

curse of dimensionality in KNN:
In higher dimension, distance from farthest and nearest points to the query point is same.

KNN
- as $k \downarrow$ → overfit; $k \uparrow$ → smoother boundary
- regression → avg over y values
  classification → majority voting



- as m increases, the optimal value of k increases
- use validation error to select best k.

Regression

- $J(\theta) = \frac{1}{m}\sum_i (y^i - \theta x^{iT})^2$
- $\nabla J(\theta) = -\frac{2}{m}\sum_i \underbrace{[y^i - \theta x^{iT}]}_{\text{error magnitude}} \cdot \underbrace{[x_0^i, x_1^i, x_2^i \dots]}_{\text{sensitivity to each param.}}$

$$= -\frac{2}{m}\sum_i [Y^T - \theta X^T] \cdot [X]$$

- for min. MSE, $\nabla J(\theta) = 0$

$$\Rightarrow Y^T X - \theta X^T X = 0$$
$$\Rightarrow \theta = Y^T X (X^T X)^{-1}$$

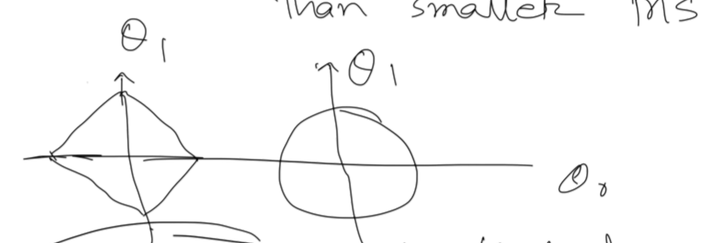
- issues with MSE choice: sensitivity to outliers
- L1 loss $= \sum_i (|y_i - \hat{y}_i|)$ L2 loss: $\sum_i (y_i - \hat{y}_i)^2$
- Best MSE constant predictor = mean of the training points
- Best MSE linear predictor = pass through the points (if 2) or three midpoints (if >2)

∴

- Bias, Variance both depend on Learner, training procedure, training size
- Detect overfitting / select model:
  use validation
- Cross Validation
  adv: lets us use more validation data so, less noisy estimate of test data
  disadv: more work, trains k model doesn't evaluate any particular predictor
- keep degree fixed, increase data size → low variance
- regularize when overfit / complexity high

$$J(\theta) = \frac{1}{2}(\underline{y} - \underline{\theta}\underline{x}^T)\cdot(\underline{y} - \underline{\theta}\underline{x}^T)^T + \underbrace{\alpha\theta\theta^T}_{\text{regularizer}}$$

α large ⇒ smaller θ more preferable than smaller MSE



L1/lasso regularization — L2/Quadratic regularizer

Smaller p	larger p
penalize small non-zero params more	less
a few big params	No big params; prefers lots of small weights

generates sparser soln than L2.

- Data increasingly separable in higher dimension

Perceptron Algo

```
while ¬done
    for each data point j:
        ŷ^j = sign(θ · x^j)
        θ = θ − (y^j − ŷ^j) x^j
```

correct pred. don't contribute

- In logistic regression, we interpret $\sigma(\theta x^T)$ as a prob. that $y=1$. The NLL loss is:

$$J(\theta) = \frac{1}{m}\sum_i \left(y_i \log(\sigma(\theta x^i)) + (1-y_i)\log(1-\sigma(\theta x^i))\right)$$

also, replace 0/1 loss with surrogate loss.

- Multiclass perceptron:

```
while ¬done:
    for each data point
        f^j = arg max_c (θ_c · x^j)
        θ_f = θ_f − α x^j
        θ_y = θ_y + α x^j
```

[if prediction f matches true class y, then no update]

NNet.

activation:
logistic $= \frac{1}{1+\exp(z)}$
ReLU $= \max(a, 0)$

SVM

primal form:

$$w^* = \arg\min_w \sum_j w_j^2$$
$$\text{s.t. } y^i(wx^i + b) \geq +1$$

adv: when lots of data

Dual:

$$\max_{\alpha \geq 0} \sum_i \left(\alpha_i - \frac{1}{2}\sum_j \alpha_i \alpha_j y^i y^j (x_i \cdot x^j)\right)$$
$$\text{st } \sum_i \alpha_i y^i = 0$$

adv: when lots of features efficient

Perceptron:

$m \uparrow$ → bias ↓ False
$n \uparrow$ → variance ↑ True

SGD adv: better for large m may avoid local optima

BGD adv: easier to set step size monotonic easier to gauge convergence
↓ may stuck in local optima

On overfitting

- $m \uparrow$ — ↓ } lin. Regression
- $n \uparrow$ — ↑

- $m \uparrow$ — ↑
- $n \uparrow$ — No effect
- Rescale — ↓ } KNN

- # hidden nodes — ↑ } Neural Net
- # sigmoid → Relu — No effect

- kernel lin → Poly — ↑
- grad descent / QP
- No effect } SVM

- R ↑ — ↓ / No eff } soft SVM

{ lin Reg ⇒ Regularization ↑ → variance ↓

feat. transform → No effect on VC Dim

lin. classifier:
overfitted, $m \uparrow$
⇒ train err ↑
test err. ↓

underfitted, $m \uparrow$
⇒ train err same
test err. same

but increasing # of features increases VC dim.