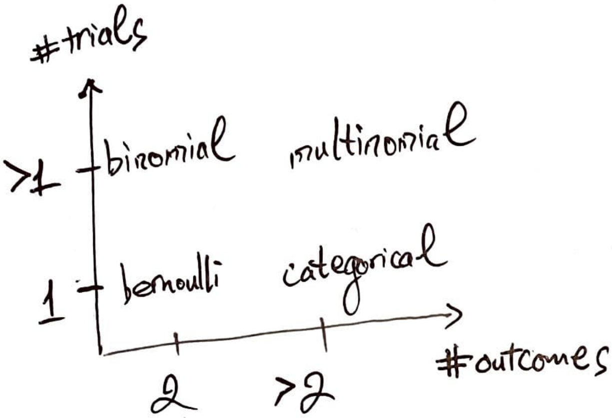
Khoi Nguyen's CheatSheet

CS 6375

Distributions

Discrete

Poisson: $P(k) = \dfrac{\lambda^k e^{-\lambda}}{k!}$



Continuous

Gaussian: $$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Beta : $$P(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)}$$

Linear Regression

Solution: (Univariate)

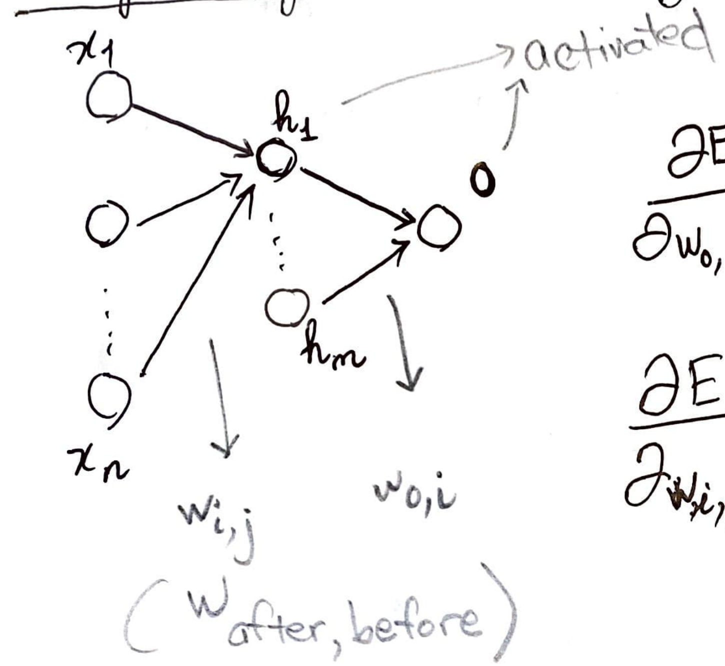
$$w_1 = \frac{m\sum_{i=1}^{m} x_i y_i - \sum_{i=1}^{m} x_i \sum_{i=1}^{m} y_i}{m\sum_{i=1}^{m} x_i^2 - \left(\sum_{i=1}^{m} x_i\right)^2}$$

$$w_0 = \frac{1}{m}\left(\sum_{i=1}^{m} y_i - w_1 \sum_{i=1}^{m} x_i\right)$$

(Multivariate) $\vec{w} = (X^T X)^{-1} X^T Y$

Neural net

2-layer's algo: GD w/ following update rules



$$\frac{\partial E}{\partial w_{o,i}} = -2(y-o)\,o(1-o)\,h_i$$

$$\frac{\partial E}{\partial w_{i,j}} = -2(y-o)\,o(1-o)\,w_{o,i} \times h_i(1-h_i)\,x_j$$

Dynamic programming:

Forward

$$h_i = a\left(\sum_{j=1}^{n} w_{i,j}\,x_j\right)$$

$$o = a\left(\sum_{i=1}^{m} w_{o,i}\,h_i\right)$$

Backward

$$\delta_i = \delta_o\,h_i(1-h_i)\,w_{o,i}$$

$(w_{i,j} \mathrel{+}= \alpha\,\delta_i\,x_j)$

$$\delta_o = (y-o)\,o(1-o)$$

$(w_{o,i} \mathrel{+}= \alpha\,\delta_o\,h_i)$

AND perceptron (0/1 input)

$$w_i = \begin{cases} +1 & \text{if } X_i \text{ in Clause} \\ -1 & \text{otherwise} \end{cases}$$

$w_0 = -k + 0.5$

(Clause $= X_1 \wedge \dots \wedge X_k \wedge \overline{X_{k+1}} \wedge \dots \wedge \overline{X_n}$)

OR perceptron

→ same

$w_0 = n - 1/2 - k$

Bias & Variance

$$\mathbb{E}\left[(y^* - h(x^*))^2\right] = \mathbb{E}\left[\left(h(x^*) - \mathbb{E}[h(x^*)]\right)^2\right] \rightarrow \text{Var.}$$

$$+ \left(\mathbb{E}[h(x^*)] - f(x^*)\right)^2 \rightarrow \text{Bias}^2$$

$$+ \mathbb{E}(\epsilon^2) \rightarrow \text{noise}$$

Bias

Low:
- Linear Reg on Linear data
- 2nd polynomial on quadratic data
- NN with many hidden units trained to completion

High:
- const
- Lin Reg on non-linear data
- NN with few hidden units on non-linear data

Variance

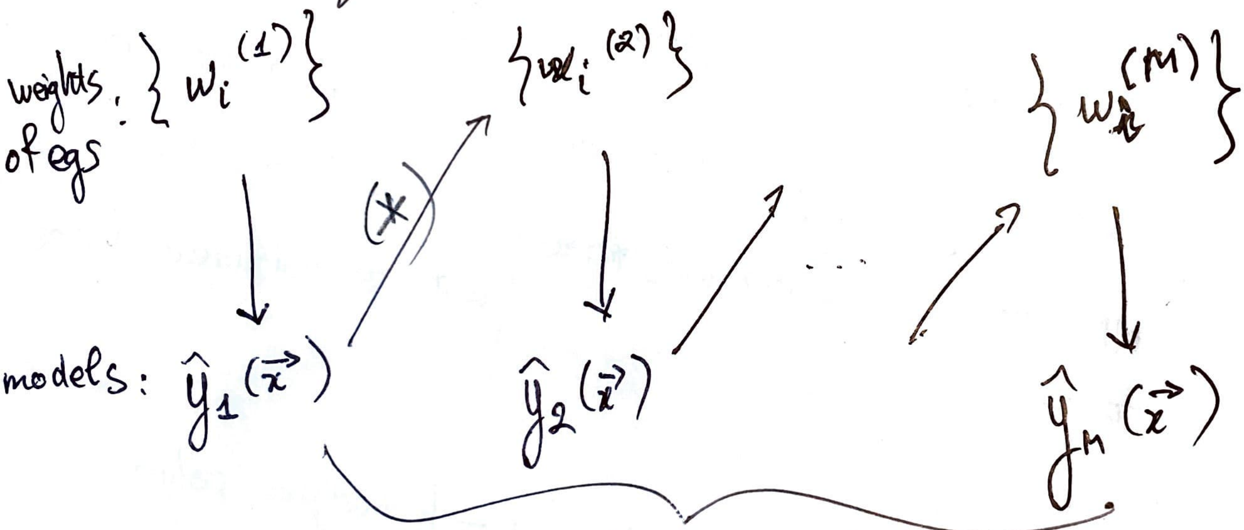
Low:
- const
- model ⊥ training data

High:
- high degree polynomial
- NN w/ many hid. units (complete training)

Ensembles

- Bagging: trained K models, each on one "bootstrap"
  (bootstrap: same-sized dataset by sampling WITH replacement)
  ⇒ Only help UNSTABLE learners (NN, DT)
  (not KNN, NB)

Boosting



$$Y(\vec{x}) = \mathrm{sgn}\left(\sum_{i=m}^{M} \alpha_m \times \hat{y}_m(\vec{x}_i)\right)$$

(**)

In AdaBoost:

$$Error_m = \frac{\sum_i^N w_i \times \mathbb{I}(y_i \neq \hat{y}_m(\vec{x}))}{\sum_i^N w_i}$$

(**): $\alpha_m = \log\left(\frac{1 - Err_m}{Err_m}\right)$

(*): $w_i \mathrel{*}= e^{\alpha_m \times \mathbb{I}(y_i \neq \hat{y}_m(\vec{x}_i))}$