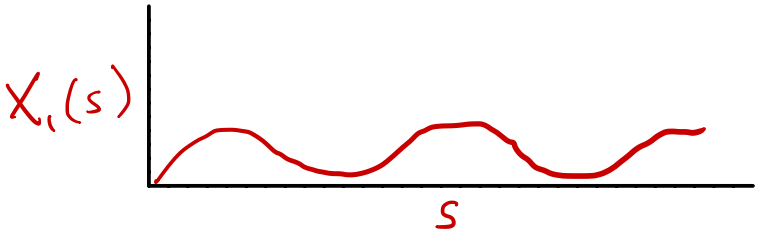
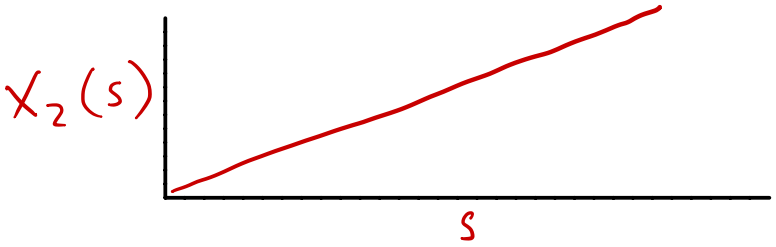
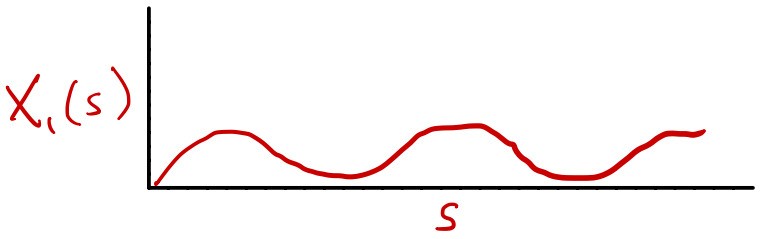
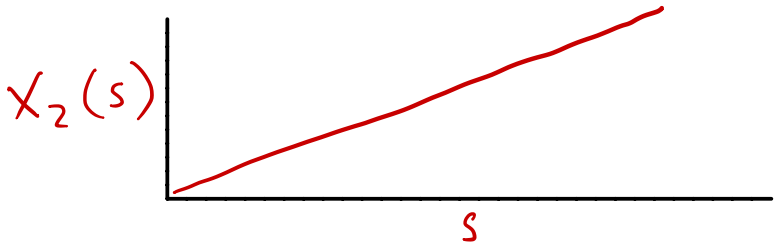




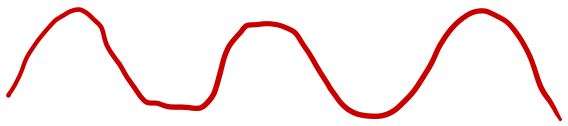
$$V(s) = w_1 X_1(s) + w_2 X_2(s)$$



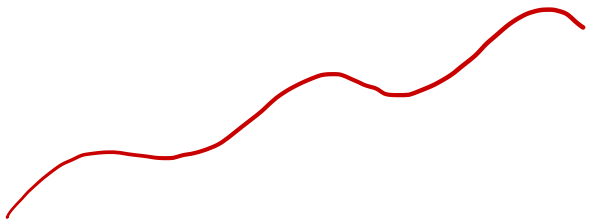


$$V(s) = w_1 X_1(s) + w_2 X_2(s)$$

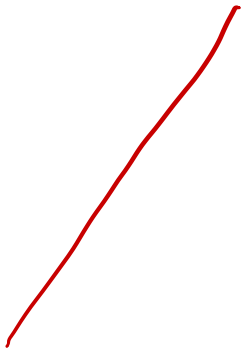
$w_1 = 2 \quad w_2 = 0$:

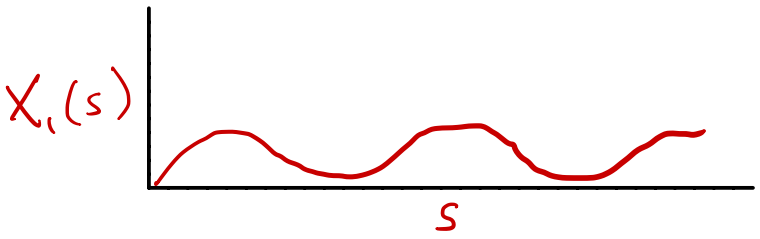


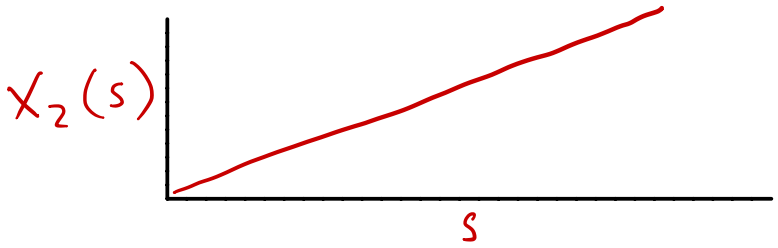
$w_1 = 1 \quad w_2 = 1$:



$w_1 = 0 \quad w_2 = 4$:







$$V(s) = w_1 X_1(s) + w_2 X_2(s)$$

Tabular equivalent:

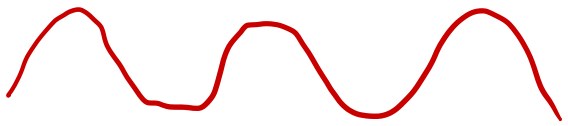
$$X_1(s) = \begin{cases} 1 & \text{at } s_1 \\ 0 & \text{elsewhere} \end{cases}$$

$$\vdots$$

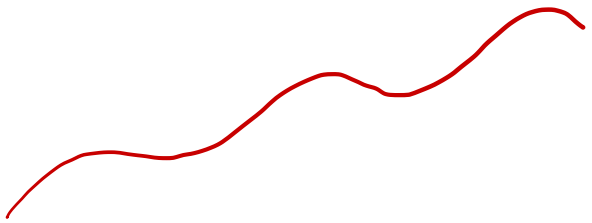
$$X_N(s) = \begin{cases} 1 & \text{at } s_N \\ 0 & \text{elsewhere} \end{cases}$$

Thus, $w_i = V(s_i)$

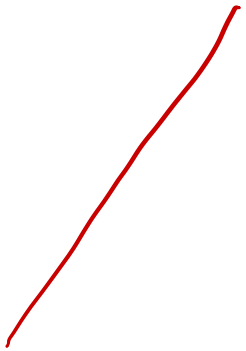
$w_1 = 2 \quad w_2 = 0$:



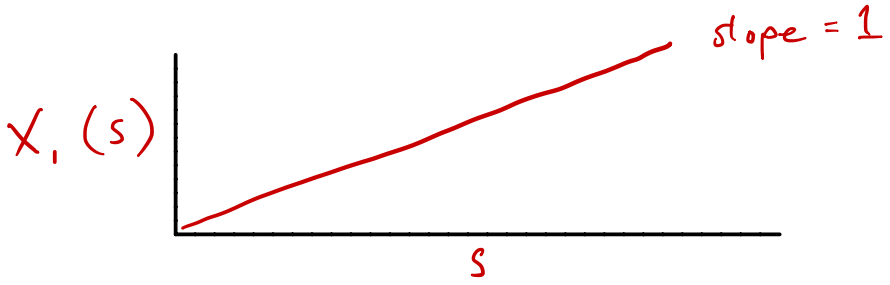
$w_1 = 1 \quad w_2 = 1$:

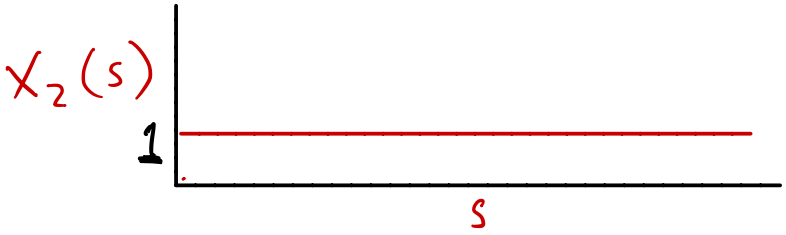


$w_1 = 0 \quad w_2 = 4$:



Linear Regression:





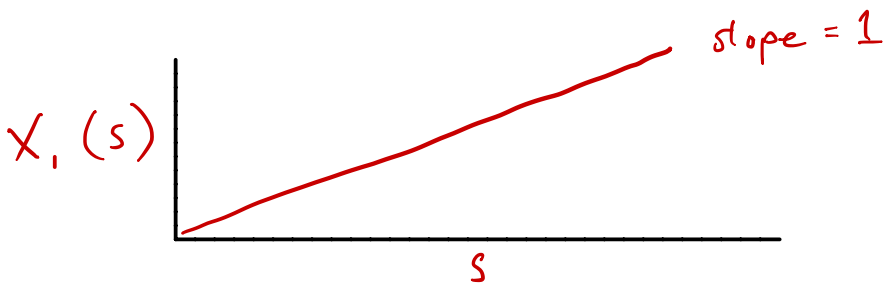
$$Y = w_1 x_1 + w_2 x_2$$

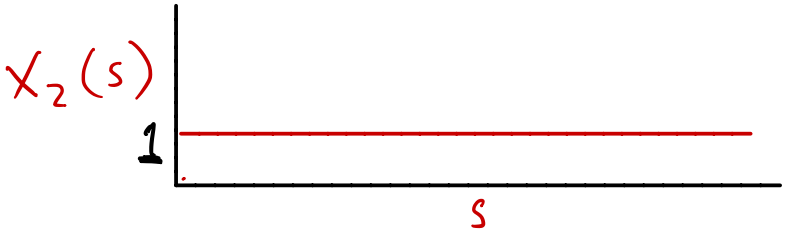
$$= w_1 x_1 + w_2 \cdot 1$$

w_1: slope

w_2: intercept

Linear Regression:





$$Y = w_1 x_1 + w_2 x_2$$

$$= w_1 x_1 + w_2 \cdot 1$$

↑ slope ↑ intercept

Supervised learning:

Given $\langle x, y \rangle$ pairs,

learn $f(x) \rightarrow y$

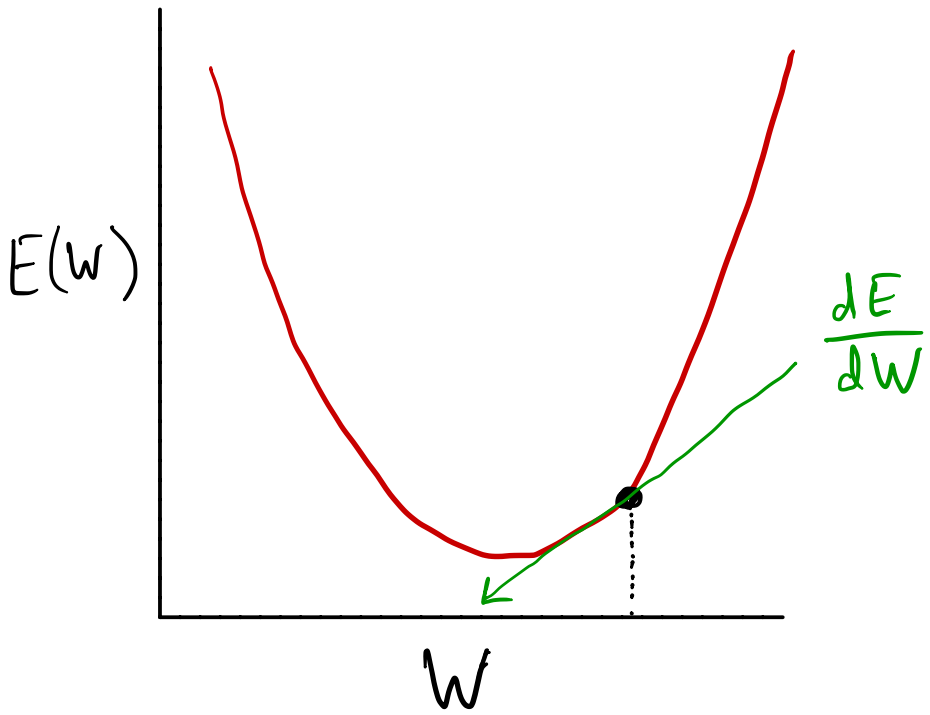
Discrete $y \rightarrow$ classification

e.g. "is this image a cat or a dog?"

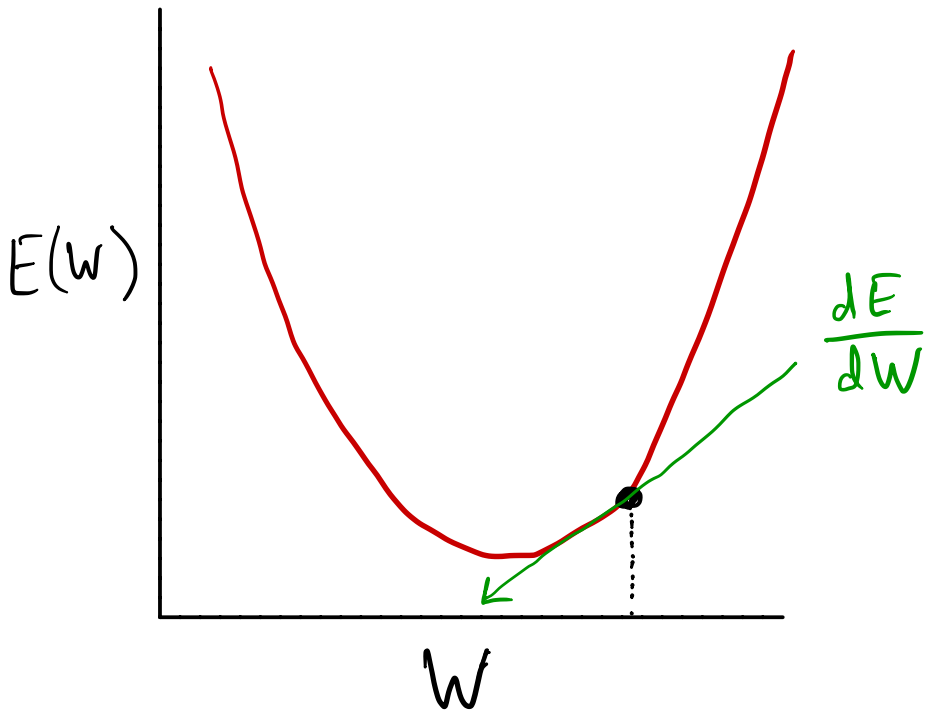
Continuous $y \rightarrow$ regression

e.g. "predict height from weight"

$$E(w) = \sum_{i=0}^{N} \left(w^T x_i - y_i \right)^2$$



$$E(W) = \sum_{i=0}^{N} \left(W^T X_i - Y_i\right)^2$$

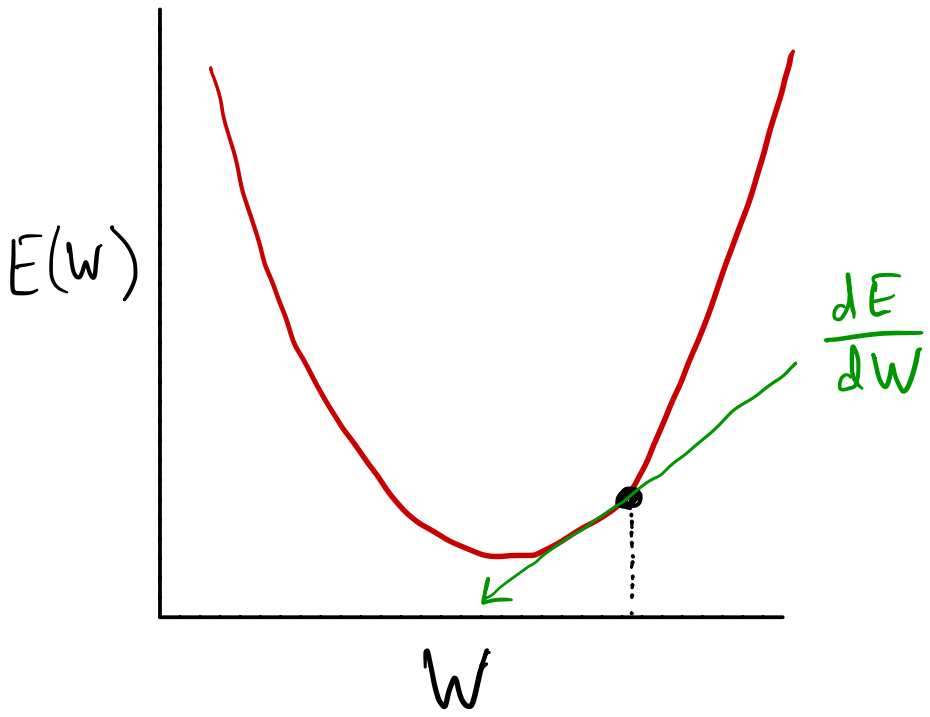


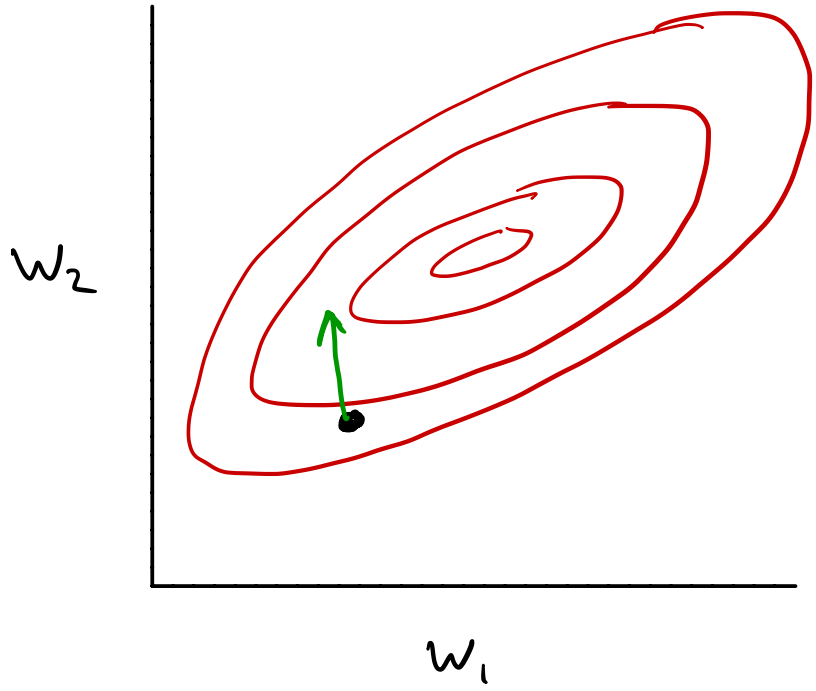
Batch least squares: Solve for $\underset{W}{\operatorname{argmin}}\, E(W)$ directly

Exact gradient descent: Compute $\frac{dE}{dW}$ exactly from all X_i

SGD: Approximate $\frac{dE}{dW}$ with a small number of samples of X, often only one.

$$E(w) = \sum_{i=0}^{N} \left(w^T x_i - y_i\right)^2$$





Batch least squares: Solve for $\underset{w}{\operatorname{argmin}}\, E(w)$ directly

Exact gradient descent: Compute $\frac{dE}{dw}$ exactly from all x_i

SGD: Approximate $\frac{dE}{dw}$ with a small number of samples of x, often only one.

RL + Function Approximation

Update Rule:

$$w \leftarrow w + \alpha \left[G_t - \hat{v}(S_t, w) \right] \nabla \hat{v}(S_t, w)$$

RL + Function Approximation

Update Rule:

$$w \leftarrow w + \alpha \left[G_t - \hat{v}(S_t, w) \right] \nabla \hat{v}(S_t, w)$$

When approximator is linear in features / bases

i.e. $\hat{v}(S_t, w) = w \cdot x(S)$, then:

$$w \leftarrow w + \alpha \left[G_t - \hat{v}(S_t, w) \right] x(S_t)$$

since $\frac{d\hat{v}(S)}{dw_i} = x_i(S)$

RL + Function Approximation

Update Rule:

$$w \leftarrow w + \alpha \left[G_t - \hat{v}(S_t, w) \right] \nabla \hat{v}(S_t, w)$$

When approximator is linear in features / bases

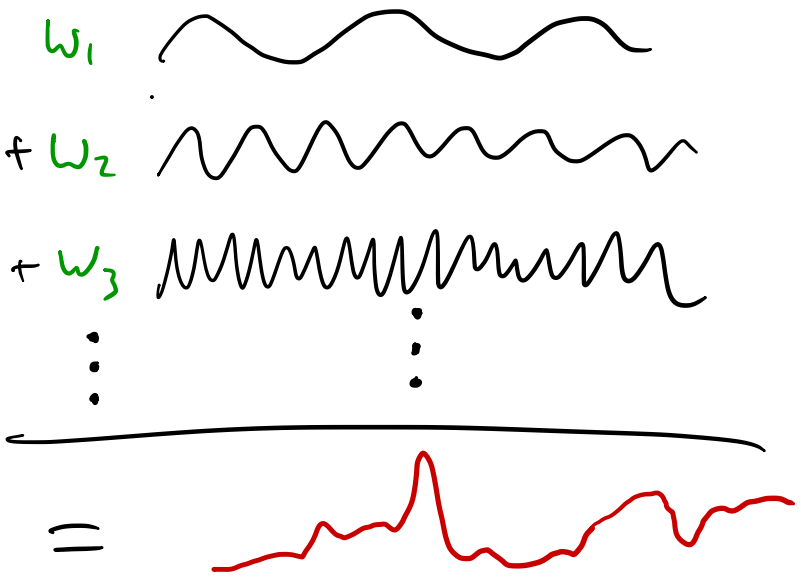
i.e. $\hat{v}(S_t, w) = w \cdot x(s)$, then:

$$w \leftarrow w + \alpha \left[G_t - \hat{v}(S_t, w) \right] x(S_t)$$

since $\frac{d\hat{v}(s)}{dw} = x_i(s)$

Assigns "credit" to most activated features for success + failure

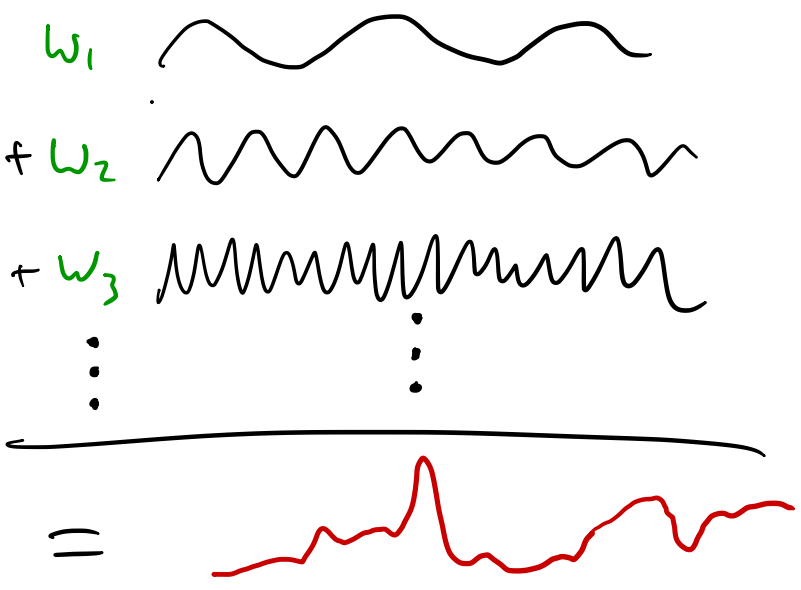
Fourier Basis



Params:

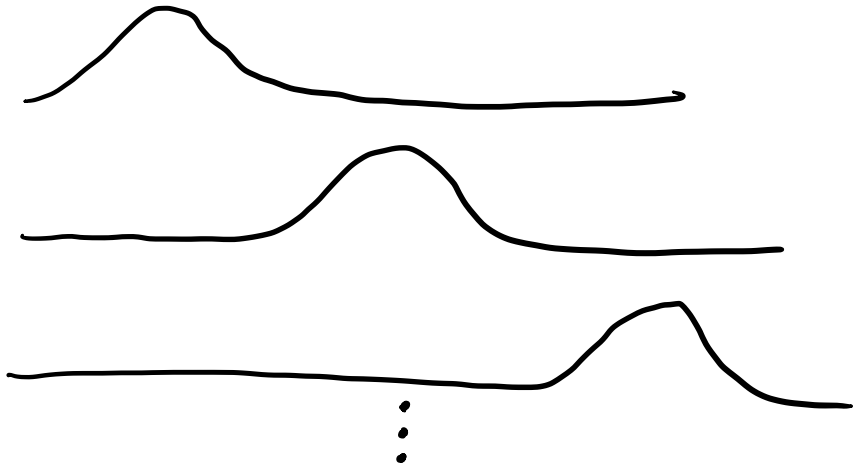
- order
- Interaction terms?

Fourier Basis



Params:
- order
- interaction terms?

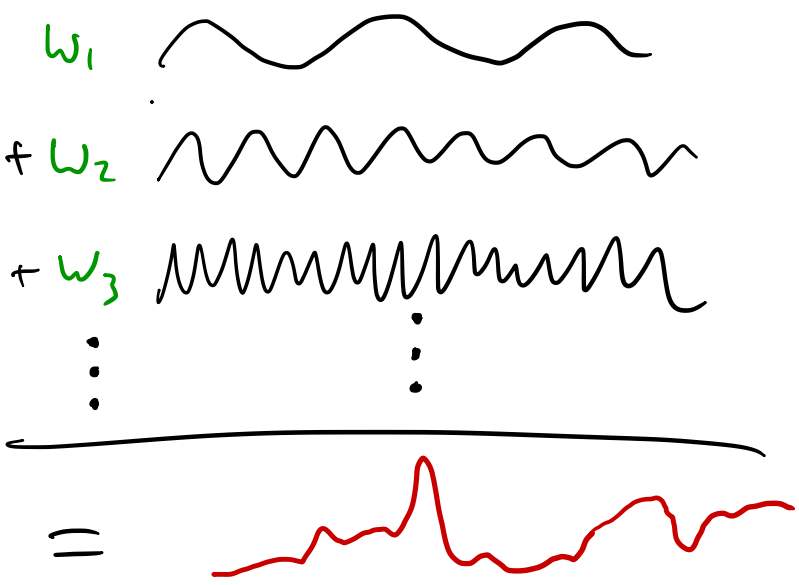
Radial Basis Functions



Params:
- Kernel width
- Tiling density

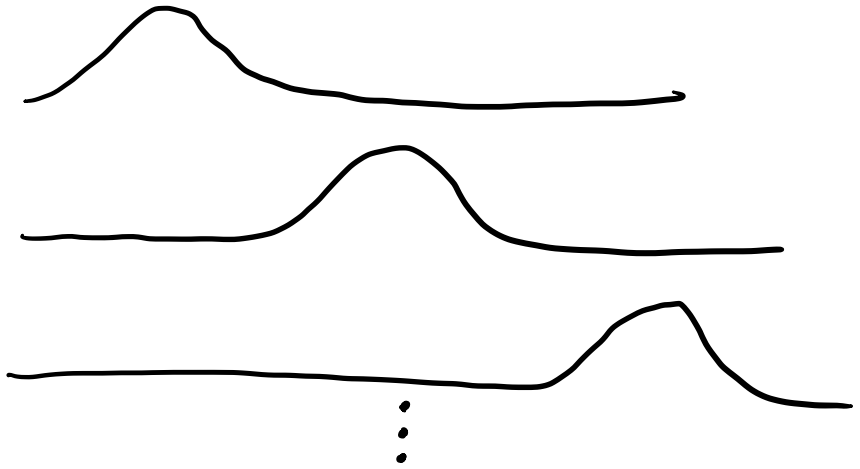
(both variable per dimension)

Fourier Basis



Params:
- order
- interaction terms?

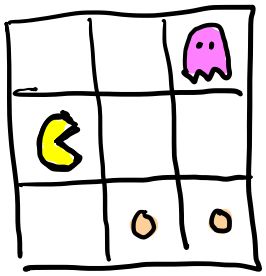
Radial Basis Functions



Params:
- Kernel width
- Tiling density

(both variable per dimension)

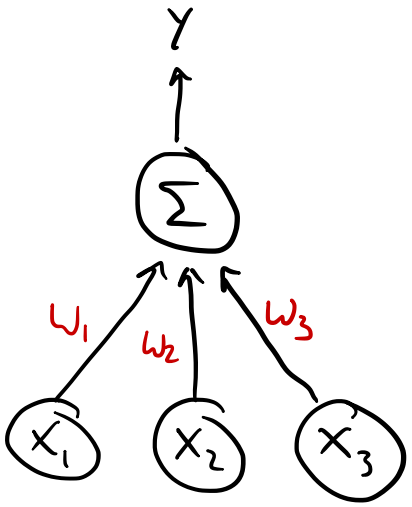
Hand-Designed Features



"Distance to nearest food pellet"
"Number of ghosts within radius of 3"
"1 if exactly this state, 0 otherwise"

→ How do choose?
Representative Power?
Locality?

Perceptron

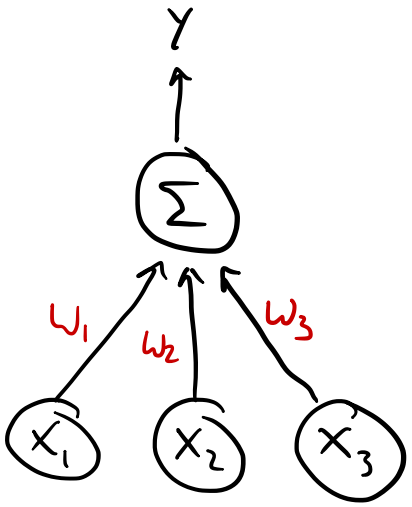


$$Y = x_1 w_1 + x_2 w_2 + x_3 w_3$$

$$\frac{dy}{dw_1} = x_1$$

Y is Linear in X

Perceptron

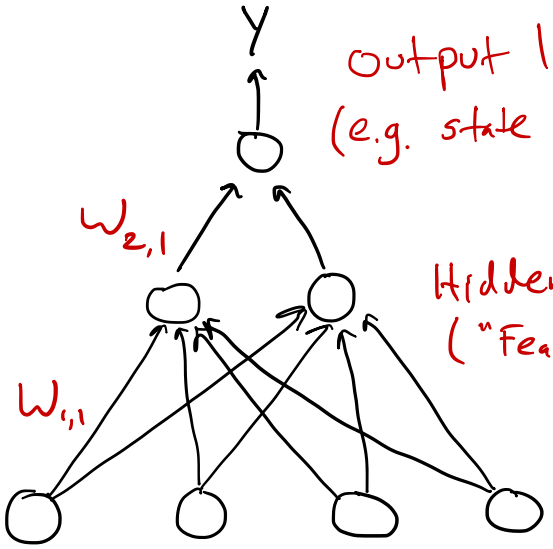


$$Y = x_1 w_1 + x_2 w_2 + x_3 w_3$$

$$\frac{dy}{dw_1} = x_1$$

Y is Linear in X

Neural Network



output layer
(e.g. state value)

Hidden layer(s)
("Features")

Input layer
(usually raw state vars)

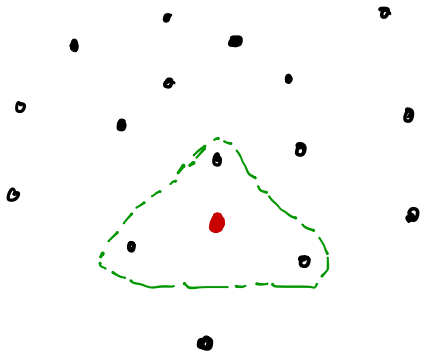
$$\frac{dy}{dw_{1,1}} = ?$$

⇒ Differentiation Via Backpropagation

Y is Nonlinear in X!

Nonparametric Methods

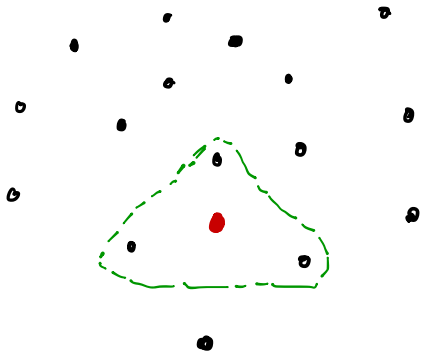
K-nearest neighbor



$$\hat{v}(s_q) = \frac{1}{k} \sum_{i=1}^{k} G(s_i)$$

Nonparametric Methods

K-nearest neighbor



$$\hat{V}(s_q) = \frac{1}{k} \sum_{i=1}^{k} G(s_i)$$

Kernel Methods

Similarity "Kernel" $K(s, s')$

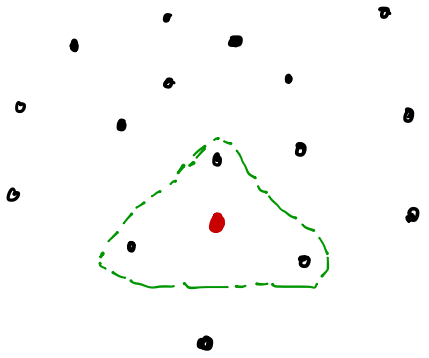
For data set of N states

And query state s_q

$$\hat{V}(s_q) = \sum_{i=1}^{N} K(s_q, s_i) G(s_i)$$

Nonparametric Methods

K-nearest neighbor



$$\hat{V}(s_q) = \frac{1}{k} \sum_{i=1}^{k} G(s_i)$$

KNN special case kernel where:

$K(s, s') = 1/k$ if s' is a KNN of s

0 otherwise

Kernel Methods

Similarity "Kernel" $K(s, s')$

For data set of N states

And query state s_q

$$\hat{V}(s_q) = \sum_{i=1}^{N} K(s_q, s_i) G(s_i)$$

"Kernal trick" eqivalent to computing $\hat{v}$ in high-dim space of features

→ Complexity of Kernel methods grows with number of data points, not features

$$w \leftarrow w - \frac{1}{2} \alpha \nabla \left[V_\pi(S_t) - \hat{v}(S_t, w) \right]^2$$

$$w \leftarrow w + \alpha \left[\mathbf{U}_t - \hat{v}(S_t, w) \right] \cdot \nabla \hat{v}(S_t, w)$$

$$w \leftarrow w - \frac{1}{2} \alpha \nabla \left[V_\pi(S_t) - \hat{v}(S_t, w) \right]^2$$

$$w \leftarrow w + \alpha \left[U_t - \hat{v}(S_t, w) \right] \cdot \nabla \hat{v}(S_t, w)$$

MC Estimate:

$$U_t = \sum_{i=t}^{T} \gamma^{i-t-1} R_i$$

- Unbiased
- fixed when w changes

converges near global opt

$$w \leftarrow w - \frac{1}{2}\alpha \nabla \left[V_\pi(S_t) - \hat{V}(S_t, w) \right]^2$$

$$w \leftarrow w + \alpha \left[U_t - \hat{V}(S_t, w) \right] \cdot \nabla \hat{V}(S_t, w)$$

MC Estimate:

$$U_t = \sum_{i=t}^{T} \gamma^{i-t-1} R_i$$

- Unbiased
- fixed when w changes

Converges near global opt

Bootstrap Estimate:

$$U_t = R_t + \gamma \hat{V}(S_{t+1}, w)$$

- Biased
- Function of w → changes with w!
- Gradient calc treated U_t as a constant!

Converges to TD fixed point (local opt)

For TD fixed point w_{TD}:

$$\overline{VE}(w_{TD}) \leq \frac{1}{1-\gamma} \min_w \left[\overline{VE}(w) \right]$$