Deriving Δ rule – $\Delta w_i = -\varepsilon \frac{\partial E}{\partial w_i}$, $\hat{y}_N = \sum_i w_i x_{in}$ Sum training case $E = \frac{1}{2}\sum_{n=1}^{N}(y_n - \hat{y}_n)^2$ $\frac{\partial E}{\partial w_i} = \sum^N (y_n - \hat{y}_n)\frac{\partial \hat{y}}{\partial w_i} \to = x_{in}$

CE $\frac{\partial E}{\partial h_k} = \partial - \sum^K t_k \ln h_k = \frac{\partial}{\partial h_k} - t_k \ln(h_k) - (1-t_k)\ln(1-h_k) = -t_k/h_k + (1-t_k)/(1-h_k) = (h_k - t_k)/h_k(1-h_k)$

Soft Max $\frac{\partial h_k}{\partial y_k} = \frac{\partial h_k}{\partial \exp(y_k)} \frac{\partial \exp(y_k)}{\partial y_k} = e^{y_k}[(\sum_k e^{y_k})^{-1} - e^{y_k}[\sum_j e^{y_j}]^{-2}] = \frac{e^{y_k}}{\sum e^{y_k}}\left[1 - \frac{e^{y_k}}{\sum e^{y_k}}\right] = h_k(1-h_k)$; $\partial - \sum_k \underset{\in[1,0]}{\partial_k} \log h_k = h_k - \partial_k$ > SSE from Linear

Sigmoid $\frac{\partial h_j}{\partial y_j} = \frac{\partial(1+\exp(-y_j))^{-1}}{\partial y_j} = -(1+e^{-y_j})^{-2}(-e^{-y_j}) = e^{-y_j}/(1+e^{-y_j})^2 = h_j(1-h_j)$ → low (linear) capacity

linear $\frac{\partial x_j}{\partial w_{ij}} = \frac{\partial b + \sum_i y_i w_{ij}}{\partial w_{ij}} = y_i$

$z = \sum_i x_i w_i$ $z \geq \theta \Rightarrow y = 1$ binary threshold

$z = \sum_i x_i w_i - \theta$ $z > 0 \Rightarrow y = 1$ perceptron

Note: $\sigma(k) = \frac{1}{1+\exp(-k)}$

Softmax $\Rightarrow y_i = e^{x_i} / \sum_j e^{x_j}$

Dist rep = Each concept rep by Many neurons, each neuron partic. in rep of many concepts.

– Coarse Coding Saves #fine neurons / # coarse = r^{k+1} r = Coarse grids, N neurons Nr vs. Nr^2 → $w_1 + w_2 \geq \theta$, $w_1, w_2 \leq \theta$, $\theta > 0$ N.L.S.

Replicate Features = Equiv., Subsampling = Invar (local Avg) → Set of weights $\exists$ ∴ linearly Separable

Convolut. NN → PERC. conv → ✓ ⇒ leave weights alone [⊗1 ⇒ Subtract input, 0 ⇒ add input] ⇒ (weight – Satisfactory $\vec{w}$)² ↓

Speed Learning → Adaptive learning rate (Global or Seperate per w_i) → Momentum $[\Delta w(t) = \alpha \Delta w(t-1) - \varepsilon \frac{\partial E}{\partial w}(t)]$

→ Stochastic GD → Congugate gradient or $-\frac{\partial E}{\partial w_i} H^{-1}[\emptyset \to \circledcirc]$ Prevent Overfit → $E[y^n - t]^2 = E[y + \sum_i w_i \varepsilon_i - t]^2 = E[(y-t) + \sum_i w_i \varepsilon_i]^2$

$= (y-t)^2 + \sum_i w_i \sigma_i^2$ } $\equiv \frac{\partial C}{\partial w_i} = \frac{\partial E}{\partial w_i} + \lambda w_i$ Bayes: $-\log P(w|D) = -\log P(D|w) - P(w) + P(D) = \frac{1}{2\sigma_D^2}\sum_c (y_c - t_c)^2 + \frac{1}{2\sigma_w^2}\sum_i w_i^2$

$= E + \frac{\sigma_D^2}{\sigma_w^2}\sum_i w_i^2$ [Assume N prior + prediction] MAKAY Q & DM – Guess σ_D^2, σ_w^2 – Learn – Set $\sigma_D^2 = \frac{1}{M}\sum(d_i - y_i)^2$, $\sigma_w^2 = \frac{1}{n}\sum w_{ij}^2$ } ↑ hold these, min C^*

– Repeat. Ass. $P(t_c|y_c) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{1}{2\sigma^2}(t_c - y_c))$, $P(w) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-w^2/2\sigma_w^2)$ } {Min Error = Max log prob.} $\frac{\partial C^*}{\partial w_i}, \frac{\partial C^*}{\partial \sigma_D} = 0$

$w_{MAP} = \arg\max_w [P(w|D)] = \max_w [P(w)P(D|w) = \max_w (P(D|w) = w_{ML}$, BAYES: $P(t|\text{input}_{test}) = \sum_g P(w_g|D) P(t|\text{input}_{test}, w_g)$ ← likelihood term

MIN Sq Err ≡ MAX log prob ~ $N(y_c = \mu)$ [y_c = models guess] $\hat{t}_i^{(MAP)} = P(t_i|x_i, w_{MAP})$

Combining Networks – use AVG of predictors y_i ($\bar{y}$). Data = t_c y_c → = mix of "regimes", use sep. model for each.

→ Cluster Cases into regimes, not inputs. Make Y_i's disagree: diff learning algthms, bagging (resample w/ replacement), Boosting

→ fit next model w/ data reweighted by how poorly prev. Model fit case.

ADABOOST → H_{final} (Strong Classifier) = $\text{Sgn}(\sum_t \alpha_t$ [weight of rule of thumb] $h_t(x)$ [Weak Classifier / perceptron]) ① Given $(x_1, y_1) \ldots$ (M datapoints)

$y_i \in \{-1, 1\}$, for t = [1, T], Make D_t on [1, M]: more weight on Misclass x_i. Find $h_t(x) \to \{-1, 1\}$ w/ $\varepsilon_t = P_{D_t}[h_t(x_i) \neq y_i) < 0.5$

Learning → Supervised → [P(Y|X)] Unsup → [P(X)] → Give low Energy to data, + ↓ R^n ⇒ ↑ learning rate

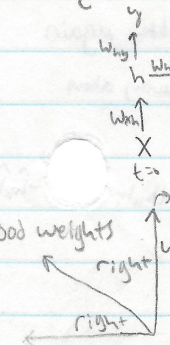
PCA → $X_n \to X_m$ ($m < n$) m ⊥ which data has most variance. Use mean for other M–n Dim. Autoenc. → linear NN w_i's span R^m; or Nonlin NN → w/ encoding Hidunit, bottleneck, N input + output units. Clustering → Force hidunit w/ $\vec{w}_i$ closest to $\vec{input}$ to have activity 1, w_i = Cluster i. $\frac{\partial E}{\partial w_{input \to hidden}} = 0$, Backprop w/ $\frac{\partial E}{\partial w_{hidden \to output}} \equiv \downarrow$ SSE ≡ / Centre of G.

Kmeans → Ass. K clusters, random ① Assign each data point to closest μ [↓SSE] ② Move μ to Centre of G of assigned data (↓SSE) ③ Repeat until -- SSE. → Try non-local split-and-merge moves, Soft K Means ≡ Each Cluster has diff responsibility for each datapoint. → P X^c is in ith Cluster

EM – use validation set or Kmeans to determine # Gaussians (N) to use. E step → Get $P(i|X^c) = P(i)P(X^c|i)/P(X$ ← Mixing proportion π_i

M step – use to update π_i, μ_i, σ_i^2 [ith N, Cth datapoint, $\in R^d$] $\pi_i^* = \sum_{c=1}^{N} P(i|X^c)/N$ $\mu_i^* = \sum_c P(i|X^c)X^c / \sum_c P(i|X^c)$

$\sigma_{id}^{2*} = \sum_c P(i|X^c)\|X_d^2 - \mu_{id}^*\|^2 / \sum_c P(i|X^c)$ Note: $P(X^c) = \sum_j P(j)P(X^c|j)$, $[\sum_j P(j) = 1]$; $P(X^c|i) = \prod_{d=1}^{D} \frac{\exp[\frac{1}{2\pi\sigma_{id}^2}(X_d^c - \mu_{id})^2}{\sqrt{2\pi}\sigma_{id}}$



$h_{t=1} = \sigma(W_{hh} h_{(t=0)(t=1)} + X W_{xh})$ localNOlin = Clustering, distlin = PCA $S_i \in \{0, 1\}^*$

Hopfield Net: $E = -\sum_i s_i b_i - \sum_{i<j} s_i s_j w_{ij}$ [Note Symmetric weights], Change One S_i, $E(S_i = 0) - E(S_i = 1) = b_i - \sum_j s_j w_{ij}$ $^*\Delta w_{ij} = 4(s_i - \frac{1}{2})(s_j - \frac{1}{2})$ or $w_{ij} = s_i s_j$, $s_i \in \{-1, 1\}$. Let Net Settle then Unlearn = ↓ Spurious minima. Simulated Annealing: Start w/ lots of noise to ↓ E landscape, and ↑ cross E barriers, then ↓ Noise = Sharpen E landscape, $P(S_i = 1) = 1/(1 + e^{-\Delta E_i / T})$

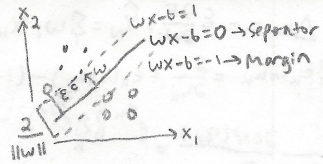
[Using Stochastic Binary threshold units.] Thermal (TE) Equilibrium – prob dist of Configs settled down

At low T, prob of good states >> bad states. $P_A / P_B = \exp(-(E_A - E_B)/T)$.

high bandwidth Stuff → image; low bandwidth Stuff → label

unsupervised (invert high bandwidth route) | supervised: image → label

Choose Hidden → get visible



$P(\vec{s}^\alpha) = \sum_{m \in models} \pi_m \prod_j \left(s_j^\alpha p_j^m + (1-s_j^\alpha)(1-p_j^m)\right)$ (if $s_j^\alpha=1$, if $s_j^\alpha=0$) [p_j = Prob of s_j^α being in training set]

Energy based = Boltz vs. Mixture Model = local = # Neurons exponential in N. = 1 Model for each Combo ⇒ want dist representation.

$P(v) = \sum_h P(h)P(v|h)$ ↳ Causal, generative model, (MOG) Pick π_i then generate image. Boltzmann Mach - E of joint config $S = \langle \underbrace{s_1 \cdots s_k}_{v} \underbrace{s_{k+1} \cdots s_n}_{h} \rangle$

prob joint config $P(v,h) = e^{-E_{vh}} / \sum_{v,h} e^{-E_{vh}}$, $P(v) = \sum_h e^{-E_{vh}} / \left(\sum_{vh} e^{-E_{vh}}\right)$ Cannot Compute so use MCMC sampling.

for training vecs $\{v_1 \cdots v_n\}$ maximize $\prod_i P(v_i) = \sum \log(p_i) = \log \sum_h e^{-E_{vh}} - \log \sum_{h,v} e^{-E_{hv}}$ and $\frac{\partial \log p(v)}{\partial w_{ij}} = \langle s_i s_j \rangle_v - \langle s_i s_j \rangle_{free}$

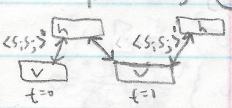
$P(v) = \frac{\sum_h e^{-E(h,v)}}{\sum_{h,v} e^{-E(h,v)}} \Rightarrow \frac{\partial \log P(v)}{\partial w_{ij}} = \frac{①\ \partial \log \sum_h e^{-E(h,v)}}{\partial w_{ij}} - \frac{②\ \partial \log \sum_{h,v} e^{-E(h,v)}}{\partial w_{ij}}$

② $= \frac{1}{\sum_{h,v} e^{-E(h,v)}} \cdot \sum_{h,v} \frac{\partial \exp(-E(h,v))}{\partial w_{ij}} = \frac{1}{\sum_{h,v} e^{-E(h,v)}} \sum_{h,v} e^{-E(h,v)} \frac{\partial(-E(h,v))}{\partial w_{ij}} = P(v,h) \cdot s_i s_j = \langle s_i s_j \rangle_{free}$

① $\frac{1}{\sum_h e^{-E(h,v)}} \sum_h \frac{\partial e^{-E(h,v)}}{\partial w_{ij}} = \frac{1}{\sum_h e^{-E(h,v)}} \sum_h e^{-E(h,v)} \cdot \frac{\partial(-E(h,v))}{\partial w_{ij}} = p(h|v) \cdot s_i s_j = \langle s_i s_j \rangle_v$ Note $E(v,h) = -\sum_i s_i b_i - \sum_{i<j} s_i s_j w_{ij}$

weight update $w_{ij} \leftarrow w_{ij} + \varepsilon \left[\frac{\partial \log p(v)}{\partial w_{ij}}\right] \rightarrow \langle s_i s_j \rangle_v - \langle s_i s_j \rangle_{free}$ → unlearning, ↓ prob of Nondata vecs.
↳ E(prob of states at TE when vector clamped to vis units)

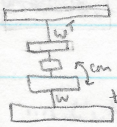
Learning = +ve phase: clamp V to Vis, let hid reach TE, Sample $s_i s_j$. Repeat $\forall V_i$'s ∈ Training set. -ve phase: Do Not clamp, get to TE, Sample $s_i s_j$. Gibbs Sampling → Start w/ training vec → Update hidden in parallel → then visable (reconstruction) → update hidden. Contrastive divergence



$\Delta w_{ij} = \varepsilon(\langle s_i s_j \rangle^0 - \langle s_i s_j \rangle^1) = \frac{1}{N} \sum_{n=1}^{N} [v_i^n \hat{h}_j^n - \hat{v}_i^n \hat{h}_j^n]$ Wake-Sleep algorithm

wake: use input data to get $h_j = 1/(1+\exp(-\sum_i v_i w_{ij} - b_j))$ then $h_j \leftarrow 1$ if $h_j >$ rand(0,1), 0 o.w.

Sleep: use h_j, get -ve data $\hat{v}_i = 1/(1+\exp(-\sum_j w_{ij} h_j - b_i))$; use $\hat{v}_i$ get -ve hid activities $\hat{h}_j = 1/(1+\exp(-\sum_i \hat{v}_i w_{ij} - b_j))$. $\Delta b_i = \frac{1}{N} \sum_{n=1}^{N} [v_i^{(n)} - \hat{v}_i^{(n)}]$, $\Delta b_j = \frac{1}{N} \sum [h_i^{(n)} - \hat{h}_i^{(n)}]$. $P(v) = \sum_h P(v|h) P(h)$ ← Train RBM to learn meaningful Patterns of activation. Training case info = log(# possible labels) ⇒ Need lots of labelled data. ↑ layers ⇒ ↑ or ↓ $\frac{\partial E}{\partial w_{ij}}$ Soln ⇒ greedy unsupervised learning then Fine-tune backprop. EQUIVARIANCE → Change in views = Change Neural Activity. 1) Place Coded = Discrete* change in property of Visual Entity = Discrete* change in which Neuron used for Encoding 2) Rate Coded = Real valued* change in Output, but not in which Neurons used. → Capsule: Info = prob ∃ Object, Pose of object ∈ $\mathbb{R}^n$ Steps



← Deep autoencoder. can treat as RBMs trained separately

1) Compute Capsule Outputs for Image 2) Apply transformation to output of capsule 3) Predict transformed image from transformed Output from Capsule. INVariance → Removes Spacial relationships. SVM - Choose Model, Structural risk Minimization

Terror = trainE + f(Size train set N, VC h, prob bound fails p); VC = How many data' Can learn ∀ label permutations. hyperplane ∈ $\mathbb{R}^K$ VC = K+1

$E_{test} < E_{train} + \left(h + h\log(2N/h) - \log(p/4)\right)/N)^{1/2}$ TRAIN $\underline{w} \cdot \underline{x}^c + b \geq 1 - \varepsilon^c$ (+ve Cases), $\leq -1 + \varepsilon^c$ (-ve cases) $\|w\|^2/2 + \lambda \sum_c \varepsilon^c$ small, $\varepsilon^c \geq 0$ Slack Variable. [Finding Max Margin Seperator using SVs] If No Seperating plane = ↑ Set of features, slack vars. - Can put plane closer to data' than the margin = Min dist to any data', ↑ ⇒ ↓ VC. Input Vecs → high D space → Kernal trick to avoid dot prod.

$b + \sum_{s \in SV} w_s K(x^{test}, x^s) > 0$ → Select: SV to max margin, Compute W for each SV, *Choose good Kernal function. *You Only do this $K(x_1, x_2) = \phi(x_1) \cdot \phi(x_2)$. SVs = Samples on margin.

MOG Generative Model → pick Gaussian w/ prob π_i, generate random point from Gaussian.

Stacked RBMs → 1) learn layer of Hidden features 2) Treat feature activation as data, learn 2nd layer.

weights in RBM define $P(v_i=1|h) = 1/(1+e^{-\sum_j h_j w_{ij}})$ ← Sample based on this dist.

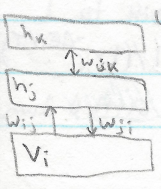
Generate data from RBM → 1) Let top RBM → TE 2) Single top-down pass to get States of lower layers.

CD → 1) Start w/ Train vec on Vis units. 2) Update hidden in parallel 3) Update visable (reconstruction) 4) Update hidden again

Wake Sleep → output [wake] ↑ ↓ [sleep] input. [wake] → Modified to ↑ prob they reconstruct activity below. [sleep] → modified to ↑ prob they reconstruct activity above.

input data → hid acts → +ve data → -ve hid acts.

(v_i) (h_j) $(\hat{v}_i)$ $(\hat{h}_j)$



Wake = generate $h_j = \sigma(\sum_i w_{ij} v_i + b_j) > rand(0,1)$ $\Delta w_{ji} = \varepsilon h_j (v_i - \sigma(\sum_j w_{ji} h_j + b_j))$, $h_k = \sigma(\sum_j w_{jk} h_j + b_k) > rand(0,1)$ $\Delta w_{kj} = \varepsilon \hat{h}_j (h_k - \hat{h}_k)$

Sleep = $\hat{h}_k = \sigma(\sum_j w_{jk} \hat{h}_j + b_k)$ $\hat{h}_j = \sigma(\sum_k w_{jk} \hat{h}_k + b_j)$ $\hat{v}_i = \sigma(\sum_j w_{ji} \hat{h}_j + b_j)$ $\Delta w_{ij} = \varepsilon \hat{v}_i (\hat{h}_j - \sigma(\sum_i w_{ij} \hat{v}_i + b_j))$

MoG = a Soft K-Means.