Linear Regression: MSE (mean-sq error) objective function: $J_D(\vec{w}, b) = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - (\vec{w}^T\vec{x}^{(i)} + b)\right)^2$

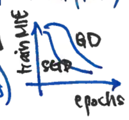
- gradient descent: $\vec{g} \leftarrow \nabla_\theta J(\theta) = \frac{1}{N}\sum_{i=1}^{N}\underbrace{(\vec{\theta}^T\vec{x}^{(i)} - y^{(i)})\vec{x}^{(i)}}_{\to J^{(i)}(\vec{\theta})}$, $\vec{\theta} \leftarrow \vec{\theta} - \gamma\vec{g}$ $O(MN)$ per iteration.

↳ local optimisation algo ⟺ converge to local min if exists ⟺ GDLR globally convergent.

minimising MSE: $\nabla J(\theta) = 0 \Leftrightarrow X^TX\theta = X^Ty \leftrightarrow$ lstsq(X,y) ⇐ find $\text{argmin}_\theta \|\theta\|$ s.t. normal eqn satisfied (soln exists, may not be unique)

↑ soln unique ⟺ N (# examples) ≥ # of LI features (e.g. 3D ⇒ 1 or 2 LI features if on line/plane)

Stochastic Gradient Descent: $\vec{\theta} \leftarrow \vec{\theta} - \gamma\nabla_\theta J^{(i)}(\vec{\theta})$, $i \sim U[1,N]$ or shuffle $[1,N]$



- epoch = single pass through training data (GD ⟷ 1 update/epoch, SGD ⟷ N updates)

	# steps to convergence	compute/step
GD	$O(\log 1/\varepsilon)^*$	$O(NM)$
SGD	$O(1/\varepsilon)^*$	$O(M)$

* WHP under assumptions
↓ learning rate ⇒ SGD behaves like GD
both: initial training MSE large due to uninformed init.

MLE: $\theta^{MLE} = \text{argmax}_\theta \prod_{i=1}^{N} p(y^{(i)}|\vec{x}^{(i)}, \vec{\theta}) = \text{argmax}_\theta\, p(D|\theta)$, $\theta^{MAP} = \text{argmax}_\theta\, p(D|\theta)p(\theta)$

[recall] (online) perceptron: $\hat{y} = \text{sign}(\vec{\theta}^T\vec{x}^{(t)})$, if misclassified, $\vec{\theta} \leftarrow \vec{\theta} + y^{(t)}\vec{x}^{(t)}$

Logistic Regression: $p(y|\vec{x}, \vec{\theta}) = y=1 \leftrightarrow \sigma(\vec{\theta}^T\vec{x})$, $y=0 \leftrightarrow 1-\sigma(\vec{\theta}^T\vec{x})$ ⟷ $\sum_{i=1}^{N}\left(y^i \ln\sigma^i + (1-y^i)\ln(1-\sigma^i)\right)$

- $\ell(\vec{\theta}) = \sum_{i=1}^{N}\log p(y^{(i)}|\vec{x}^{(i)}, \vec{\theta})$, $J(\vec{\theta}) = -\frac{\ell(\vec{\theta})}{N} \Rightarrow \frac{\partial J^{(i)}}{\partial\vec{\theta}} = -\left(y^{(i)} - \sigma(\vec{\theta}^T\vec{x})\right)\vec{x}^{(i)}$
- prediction: $\hat{y} = 1 \Leftrightarrow \sigma(\vec{\theta}^T\vec{x}) \geq 0.5$ ↖ $\vec{\theta} \mathrel{+}= \gamma\left(y^{(i)} - \sigma(\vec{\theta}^T\vec{x})\right)\vec{x}^{(i)}$ always updated.

Regularization: prevent overfitting (idea: Occam's razor): $\hat{\theta} = \text{argmin}_\theta\, J(\theta) + \lambda r(\theta)$

L1/Lasso: $\|\vec{\theta}\|_1 = \sum_{m=1}^{M}|\theta_m|$, L2/Ridge: $\|\vec{\theta}\|_2 = \sum_{m=1}^{M}\theta_m^2$

true function classifier → fit model params. training error used to choose $h \in \mathcal{H}$
* validation error to choose $\mathcal{H}$ (hyperparameters)

Learning Theory: true error rate $R(h) = \mathbb{E}_{x\sim p^*}[\mathbb{I}(c^*(x) \neq h(x))]$ is unknown

empirical risk/training error $\hat{R}(h) = \mathbb{E}_{x\sim D}[\ldots] = \frac{1}{M}\sum_{i=1}^{M}\mathbb{I}[y^{(i)} \neq h(x^{(i)})]$

↑ c^* may be unachievable, best achievable (true) risk minimizer $h^* = \text{argmin}_{h\in\mathcal{H}} R(h)$ unknown
only know empirical risk minimizer $\hat{h} = \text{argmin}_{h\in\mathcal{H}} \hat{R}(h)$. overfitting = $\hat{R}(h) - R(h)$

PAC (Probably Approximately Correct) criterion: $P(|R(h) - \hat{R}(h)| \leq \varepsilon) \geq 1-\delta\ \forall h \in \mathcal{H}$

↑ ε is diff between true & empirical risk, δ is probability of "failure"

ERM (empirical risk minimization) on $\mathcal{H}$ with M training ex. → realizable if $c^* \in \mathcal{H}$, agnostic if c^* might/might not be in $\mathcal{H}$

→ sample complexity = M needed in order to satisfy PAC for given ε, δ → finite if $|\mathcal{H}| < \infty$, infinite if $|\mathcal{H}| = \infty$.

→ Bayesian view: θ is RV, described by prior & posterior, MAP
find estimator that → θ quickly, even for worst-case distrib of data
Frequentist: θ as a constant, (regularized) MLE, consistency/convergence rates/robustness
↳ regularized MLE ≡ MAP if regularizer = log(prior), but justification different.

→ MAP estimate = mode of posterior; β-distribution $\propto \theta^{\alpha-1}(1-\theta)^{\beta-1}$ ↗ sharper if ↑α,β ↘ α moves mass to $\theta=1$, $\beta \to \theta=0$

$\vec{x} \leftrightarrow 6\times1$, $\vec{x}^{(1)}$ w bias (1 added)
↓ ↙ weights α is $3\times(6+1)$
$\vec{a} = \vec{\alpha}\vec{x}^{(1)} \leftrightarrow a_j = \alpha_{j,b} + \sum_{i=1}^{6}\alpha^*_{j,i}x_i\ \forall 1\leq j\leq 3$
↓ ↙ # of neurons in hidden layer is 3
$\vec{z} = \text{ReLU}(\vec{a}) = \max(0, \vec{a})$, $\vec{z}$ is 3×1 ⟷ alternative sigmoid$(u) = \frac{1}{1+e^{-u}}$ ↳ $\sigma'(u) = \sigma(u)(1-\sigma(u))$
↓ ↗ $4\times(3+1)$ → prepend 1 to 1st entry of z ↳ $\frac{\partial z}{\partial a} = \mathbb{I}[a>0]$
$\vec{b} = \vec{\beta}\vec{z} \leftrightarrow 4\times1$ * if β 1-row of non-zero values, $\vec{b}$ reducible to layer $\vec{w}$ single neuron (ie. all d−1 neurons have output 0)
↓
$\hat{y} = \text{softmax}(\vec{b}) = \frac{\exp(b_k)}{\sum_{l=1}^{4}\exp(b_l)}$ ↳ $\frac{\partial\hat{y}_l}{\partial b_k} = \hat{y}_l(\mathbb{I}[k=l] - \hat{y}_k)$
↓ ↙ 1-hot vector
$\ell(\hat{y}, \vec{y}) = -\sum_{i=1}^{4}y_i\ln(\hat{y}_i) \to \frac{\partial\ell}{\partial\hat{y}_i} = -\frac{y_i}{\hat{y}_i}$ & $\sum_l y_l = 1$

$\frac{\partial\ell}{\partial b_k} = \sum_l \frac{\partial\ell}{\partial\hat{y}_l}\frac{\partial\hat{y}_l}{\partial b_k} = \hat{y}_k - y_k$

→ LINEAR: $\vec{x} - \boxed{\vec{w}} \to \vec{z}$ $\left(\frac{\partial\ell}{\partial\beta_{kj}} = \frac{\partial\ell}{\partial b_k}z_j\right)$ used to calculate $\frac{\partial\ell}{\partial\alpha}$ ↳ (no bias bc. $\beta_{k,0}$ are not affected by values of α)

fwd: return $\vec{w}$ @ $\vec{x}^{(1)}$ ← cache $\vec{x}^{(1)}$

bck: $\frac{\partial\ell}{\partial\vec{w}} = \frac{\partial\ell}{\partial\vec{z}}\vec{x}^T$ (np.outer(dz, $\vec{x}^{(1)}$)), return $\frac{\partial\ell}{\partial\vec{x}} = (\vec{w}^*)^T\frac{\partial\ell}{\partial\vec{z}}$ → step: $\vec{w}$ -= lr * $\frac{\partial\ell}{\partial\vec{w}}$

→ ReLU: fwd cache $\max(0, \vec{x})$, bck ∂[in] = ∂[out] * (cache > 0)
Sigmoid: fwd cache $1/(1+\exp(-x))$, bck ∂[in] = ∂[out] cache (1 − cache)

→ Softmax Cross Entropy: fwd return ($\hat{y}$, $-\ln\hat{y}[y]$) (cross entropy loss), bck $y_k = [0, \cdots, 1, \cdots, 0]$ (k-index) for $\frac{\partial\ell}{\partial b_k}$

↳ if separate: $g_b = g_{\hat{y}}^T(\text{diag}(\hat{y}) - \hat{y}\hat{y}^T)$ + $\ell = -\vec{y}^T\ln\hat{y}$, $g_{\hat{y}} = -\frac{\vec{y}}{\hat{y}}\cdot g_\ell$ ← 1

① for finite $\mathcal{H}$ s.t. $c^* \in \mathcal{H}$ (realizable), and arbitrary distribution p^*, if # labelled training data pts $M \geq \frac{1}{\varepsilon}\left(\ln(|\mathcal{H}|) + \ln(\frac{1}{\delta})\right)$ then with prob $\geq 1-\delta$, $h \in \mathcal{H}$ with $\hat{R}(h) = 0$ have $R(h) \leq \varepsilon$

pf: E = event $\exists h \in \mathcal{H}$ with $\hat{R}(h) = 0$, $R(h) > \varepsilon$, then ⇒ $P(E) < \delta$ ← if k h with $R(h) > \varepsilon$ * union bound + $\ln(1+x) \leq x$

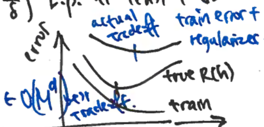
$P(E) < k(1-\varepsilon)^M \leq |\mathcal{H}|(1-\varepsilon)^M \Rightarrow \ln P(E) < \ln|\mathcal{H}| + M\ln(1-\varepsilon)$ ← $\leq -\varepsilon$

Ex: $\mathcal{H}$ = conjunctions over d boolean variables ⇒ $|\mathcal{H}| = 3^d$ ↗ 1, 0, absent.

↳ Cor: given training data set S s.t. $|S| = M$, all $h \in \mathcal{H}$ with $\hat{R}(h) = 0$ have $R(h) \leq \frac{1}{M}\left(\ln|\mathcal{H}| + \ln(\frac{1}{\delta})\right)$ ↗ w.p. at least $1-\delta$

② for finite $\mathcal{H}$ and arbitrary dist. p^*, if # labelled training dpts satisfies $R(h) < \varepsilon$ $M \geq \frac{1}{2\varepsilon^2}\left(\ln|\mathcal{H}| + \ln(\frac{2}{\delta})\right)$, then wp at least $1-\delta$, all $h \in \mathcal{H}$ satisfy $|R(h) - \hat{R}(h)| \leq \varepsilon$ $\hat{R}(h) + \frac{1}{M}\ln|\mathcal{H}| + \ln(\frac{1}{\delta})$

↳ Cor: ... s.t. $|S| = M$, all $h \in \mathcal{H}$ have $R(h) \leq \hat{R}(h) + \sqrt{\frac{1}{2M}\left(\ln|\mathcal{H}| + \ln\frac{2}{\delta}\right)}$ w.p. at least $1-\delta$.



Def: $\mathcal{H}$ shatters set of pts if it can classify them all possible ways

Sauer's lemma: sps $S_\mathcal{H}(M) = 2^d$ for $M \leq d$, but $S_\mathcal{H}(d+1) < 2^{d+1}$, then $S_\mathcal{H}(M) \in O(M^d)$

↳ VC($\mathcal{H}$) = size of largest set $\mathcal{H}$ can shatter (∃d pts... ∄ d+1 pts...)

→ halfspaces in d dimensions: VC = d+1 ↘ so $|\mathcal{H}| \to S_\mathcal{H}(M)$ (thms)

↖ if $S_\mathcal{H}(M) = 2^M\ \forall M$, not learnable/can memo at any |D|, if bounded, can memo at most d.

Algorithm 1 SGD

```
Initialize θ^(0)


s = 0
for t = 1, 2, ..., T do
    for i ∈ shuffle(1, ..., N) do
        Select the next training point (x_i, y_i)
        Compute the gradient g^(s) = ∇J_i(θ^(s−1))
        Update parameters θ^(s) = θ^(s−1) − η g^(s)
        Increment time step s = s + 1
    Evaluate average training loss J(θ) = (1/n) Σ_{i=1}^n J_i(θ)
return θ^(s)
```

$-l = -\log p(w^{(i)})$

Algorithm 1 Mini-Batch SGD

```
Initialize θ^(0)
Divide examples {1, ..., N} randomly into batches {I_1, ..., I_B}
where ⋃_{b=1}^B I_b = {1, ..., N} and ⋂_{b=1}^B I_b = ∅
s = 0
for t = 1, 2, ..., T do
    for b = 1, 2, ..., B do
        Select the next batch I_b, where m = |I_b|
        Compute the gradient g^(s) = (1/m) Σ_{i∈I_b} ∇J_i(θ^(s))
        Update parameters θ^(s) = θ^(s−1) − η g^(s)
        Increment time step s = s + 1
    Evaluate average training loss J(θ) = (1/n) Σ_{i=1}^n J_i(θ)
return θ^(s)
```

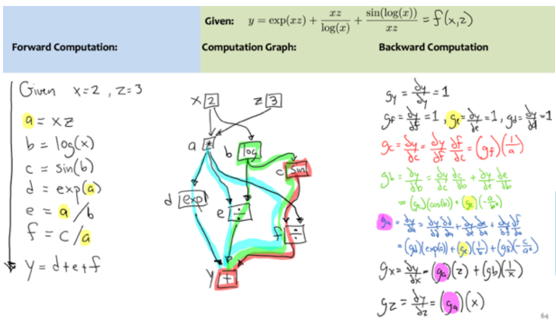
Convexity: $f: \mathbb{R}^D \to \mathbb{R}$ is convex if $\forall \mathbf{x^{(1)}}, \mathbf{x^{(2)}} \in \mathbb{R}^D$ and $0 \le c < 1$, $f\left(c\mathbf{x^{(1)}} + (1-c)\mathbf{x^{(2)}}\right) \le cf\left(\mathbf{x^{(1)}}\right) + (1-c)f\left(\mathbf{x^{(2)}}\right)$, linear functions are convex, ax^2 concave for $a < 0$

- MSE, MAE are convex, strictly convex iff $\mathbf{x}$ has full column rank (otherwise infinite minimizers in nullspace); adding strictly convex regulariser makes both strictly convex
- Convex → converge to global minima which might not exist (e.g. e^x) (exist for MSE/MAE)

Don't use test error in making model selection decisions!

Matrix multiplication for $[M \times N][N \times P] \in O(MNP)$, matrix inverse in $O(N^3)$

Conditional likelihood: iid samples $D = \{x^{(i)}, y^{(i)}\}$ from a pair of RVs (unlike likelihood function with single X RV with pmf $p(x \mid \theta)$), Y discrete with pmf $p(y \mid x, \theta)$



False positive rate FPR = FP / N
False negative rate FNR = FN / P
Positive predictive value/precision = TP / PP
Negative predictive value = TN / PN
True positive rate/recall = TP / P
Accuracy = (TP + TN)/(TP+FN+FP+TN)

		Predicted Label: +1	Predicted Label: −1	
True label	+1	True positive (TP)	False negative (FN)	Total positives (P) = TP + FN
True label	−1	False positive (FP)	True negative (TN)	Total negatives (N) = FP + TN
		Predicted positives (PP) = TP + FP	Predicted negatives (PN) = FN + TN	

Achieving fairness: (1) pre-processing data, (2) additional constraints during training, (3) post-processing predictions – premise for 1+2: if def of fairness satisfied in training data, then most models will preserve that relationship. A protected label, X applicant data, Y pred

1. **Independence** (selection rate parity): $h(X, A) \perp A$, prop of accepted applicants same for all genders (adjust penalty for predicting +ve in class till we get parity/use diff threshold), permits laziness (alw pred +1)/susceptible to adversaries (admit some randomly)
   - Prediction rate is the same across values of A, $P(h = 1 \mid A = a_1) = P(h = 1 \mid A = a_2)$
2. **Separation** (FPR = FNR): $h(X, A) \perp A \mid Y$, all good/bad applicants accepted with same prob rgdless of A, perpetuate existing bias (only access to target var thru historical data)
   - Among individuals of the same true label, $P(h = 1 \mid Y, A)$, classifier independent of A
3. **Sufficiency** (PPV = NPV): $Y \perp A \mid h(X, A)$, among people who receive the same prediction, the actual probability of being positive is the same across groups, $P(Y = 1 \mid h, A)$ independent of A (i.e., the info contained in $h(X, A)$ is sufficient, A becomes irrelevant)

If baseline rates of label across both values of A equal, then $Y \perp A$, both S's can be achieved.