CS 480 Cheat Sheet

Perceptron

We assume $(x_1, y_1), ..., (x_n, y_n)$ belongs to some distribution. Choose predictive function h_n such that $\max Pr(h(x_i) = y_i)$

Dot Product: $\langle w, x\rangle = \sum w_i x_i$

Padding: $\langle w, x\rangle + b = \langle (w, b), (x, 1)\rangle$

Note: $z = (w, b)$, $a_i = y_i(x_i, 1)$

Linear Seperable: if $s > 0$ and $Az > s1$

If data is not linearly seperable perceptron stalls.

Margin is determined by closest point to the hyperplane.

Perceptron finds a solution, may not be best solution.

l2 Norm: $||x||_2 = \sqrt{\sum_i x_i^2}$

Error Bound $\leq \frac{R^2||z||_2^2}{s^2}, R = \max ||a_i||_2$

Margin: $\gamma = \max_{||z||_2=1} \min_i \langle a_i, z\rangle$

One-versus-all: $\hat{y} = \text{argmax}_k \ w_k^T x + b_k$

One-versus-one: $\#\{x^T w_{k,k'} + b_{k,k'} > 0, x^T w_{k',k} + b_{k',k} < 0\}$

Linear Regression

Gradient: if $f(x)\mathbb{R}^d \to \mathbb{R}$, $\Delta f(v) = \left(\frac{\delta f}{\delta v_1}, ..., \frac{\delta f}{\delta v_d}\right) \mathbb{R}^d \to \mathbb{R}^d$

Hessian: $\Delta^2 f(v) = \begin{bmatrix} \frac{\delta^2 f}{\delta^2 v_1^2} & ... & \delta v_d^2 \delta v_1^2 \\ \vdots & & \vdots \\ \frac{\delta^2 f}{\delta v_1^2 \delta v_d^2} & ... & \delta^2 v_d^2 \end{bmatrix} : \mathbb{R}^d \to \mathbb{R}^{d \times d}$

Emprical Risk Minimization: $\text{argmin}_w \frac{1}{n} \sum_d l_w(x, y)$

Convexity #1: $f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2)$

Convexity #2: if $\Delta^2 f(x)$ is positive semi definite.

Positive Semidefinite: if $M \in \mathbb{R}^{d \times d}$, PSD iff $v^T M v \geq 0$

Loss function needs to be convex to optimizes.

Setting loss function to 0 optimizes our solution.

MLE principle: pick paramaters that maximize likelihood.

Ridge Regularization: $\arg\min_w ||Aw - z||_2^2 + \lambda||w||_2^2$

Lasso Regularization: $\arg\min_w ||Aw - z||_2^2 + \lambda||w||_2$

k-Nearest Neighbour Classification

Bays Optimal Classifier: $f^*(x) = \arg\max_c \Pr(y = c|x)$

NN Assumption: $Pr[y = c|x] \approx Pr[y' = c|x']$, $x \approx x'$

No classifier can do as good as bayes.

Cant be descriped as a parameter vector.

Can express non linear relationships.

Takes 0 training time, and $O(nd)$ to $O(d \log n)$ testing time.

Small values of k lead to overfitting.

Large values of k lead to high error.

1NN Limit: as $n \to \infty$ then $L_{1NN} \leq 2L_{Bayes}(1 - L_{Bayes})$

Logistic Regression

Classifications are take into account confidence: $\hat{y} \in (-1, 1)$

Bernoulli Model: $Pr[y = 1|x, w] = p(x, w) \in (-1, 1)$

Logit Transform: $\log\left(\frac{p(x,w)}{1-p(x,w)}\right) = \langle x, w\rangle = \frac{1}{1+exp(-\langle x,w\rangle)}$

Optimizing Loss: $\Delta_w l_w(x_i, y_i) = (p_i(x_i, w) - y_i)x_i$

Iterative Update: $w_t = w_{t-1} - \eta d_i$

Gradient Descent: $d_t = \frac{1}{n}\sum_i^n \Delta_w l_{wt-1}(x_i, y_i)$

Stochastic GD: $B \in [n], d - t = \frac{1}{|B|}\sum_{i \in B} \Delta_w l_{wt-1}(x_i, y_i)$

Newton's Method d_t is given by the equation below:

$$d_t = (\tfrac{1}{n}\textstyle\sum_i^n \Delta_w^2 l_{wt-1}(x_i, y_i))^{-1}(\tfrac{1}{n}\sum_i^n \Delta_w l_{wt-1}(x_i, y_i))$$

Multiclass Logisitc Regression where k = class: $Pr[y = k|x, w] = \frac{exp(\langle w_k, x\rangle)}{\sum_l exp(\langle w_l, x\rangle)}$

Hard-Margin SVM

Assume that dataset is linearlly seperable. Hard Margin SVM's will try to find the "best" solution. The best solution is the one that maximizes margin.

Optimize: $min_{w,b} \frac{1}{2}||w||_2^2 \ s.t \ y\hat{y} \geq 1$

Primal: $min_{w,b} \frac{1}{2}||w||_2^2 \ s.t \ y_i(\langle w, x_i\rangle + b) \geq 1$

Duel: $min_a \frac{1}{2}\sum_i \sum_j a_i a_j y_i y_j \langle x_i, x_j\rangle \ s.t \ \sum a_i y_i = 0$

Complimentary Slackness: $a_i(y_i(\langle w, x_i\rangle + b) - 1) = 0, \forall i$

Support Vector: if $a_i > 0$ then w=$\sum a_i y_i x_i$

Soft-Margin SVM

Data does not need to be linearlly seperable.

Soft-Margin: $\min_{w,b} \frac{1}{2}||w||_2^2 + C\sum_i \max(0, 1 - y_i\hat{y}_i)$

if $1 - y_i\hat{y}_i \leq 0 \implies$ Correct side of margin.

if $0 < 1 - y_i\hat{y}_i \leq 1 \implies$ Correctly classified, inside of margin.

if $y_i\hat{y}_i \leq 0 \implies$ incorrectly classified.

If C=0 ignore data, if C=∞, hard-margin.

Slack Variable: define γ_i such that $max(0, 1 - y_i\hat{y}_i) \leq \gamma_i$

Split in Two: $0 \leq \gamma_i$ and $1 - y_i\hat{y}_i \leq \gamma_i$

Duel Solution: Note $0 \leq \gamma_i$ and $1 - y_i\hat{y}_i \leq \gamma_i$ implies:

$=\max_{\alpha,\beta} min_{w,b,\gamma} \frac{1}{2}||w||_2^2 + \sum(C\gamma_i + \alpha(11 - y_i\hat{y}_i - \gamma_i) - \beta_i\gamma_i)$

$=\min_\alpha \frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j y_i y_j \langle x_i, x_j\rangle - \sum a_i$ s.t $\sum a_i y_i = 0$

if $a_i = 0$ then $y_i = 0$, point is classified correctly.

if $a_i > 0$ and $y_i = 0$, point is on margin.

if $a_i > 0$ and $y_i > 0$, point is on within margin.

Loss Function: L = $\frac{C}{n}\sum_i l_{w,b}(x_i, y_i) + \frac{1}{2}||w||_2^2$

Gradient Descent: $\frac{\delta L}{\delta w} = w + C/N \sum \ \delta_i$

if $1 - y_i\hat{y}_i \geq 0$, then $\delta = -y_i x_i$ else $\delta = 0$

Kernels

Map data to new space where it is linearlly seperable.

Padding Trick: $\o(x) = [w, 1]$ and $w = \langle x, p\rangle$

New Classifier: $\langle \o(x), w\rangle = \langle x, p\rangle + b > 0$

Quadratic Feature: $x^t Q x + \sqrt{2}x^T p + b$, wich gives us:

$\o(x) = [xx^t, \sqrt{2}x, 1]$ and $w = [Q, p, b]$

With feature map $\o : \mathbb{R}^d \to \mathbb{R}^{d \times d + d + 1}$, time O(d) to O($d^2$)

This can take infinite time in high dimensions. For the duel we only need to calculate dot product.

Kernal: k:$\mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ if k(x,x') = $l\angle \o(x), \o(x')\rangle$

A kernel if valid if its symmetric and positive semi-definite.

New Kernel: $K_{ij} = \langle \o(x^i), \o(x^j)\rangle = k(x^i, x^j)$

Classifiy New: sign($\sum a^i y^i k(x^i, x)$)

Polynomial Kernel t: $k(\langle x, x'\rangle + 1)^t$

Gaussian Basis: $exp(||x - x'||_2^2)$

SVM (Linear Kenrel): O(nd) train time, O(d) test time.

General Kernel: $O(n^2 d)$ train time O(nd) test time.

Decision Trees

Can do classification or regression, handle non-linear functions.

May fail on linear functions.

Start at one node, and split each. Select the pure node.

Loss:t*=$\text{argmin}_t\, l(\{(x_i, y_i) : x_i \leq t\}) + l(\{(x_i, y_i) : x_i > t\})$

Let p_c = frac of S with label c. $\hat{y} = arg\max_c p_c$

Misclassification Loss: l(s) = $1 - p_y$

Entropy Loss: l(s) = $-\sum_{classesc} p_c \log p_c$

Gini Index Loss: l(s) = $\sum_{classesc} p_c(1 - p_c)$

Regression: l(s) = $min_p \sum_{i \in S}(y_i - p)^2 = \sum_{i \in S}(y_i - y_s)^2$

We can stop based on run time, depth or splits.

Once a tree is fully grown we can prune it.

Bagging

Training on empirical mean gives a variance of: $E[\hat{\mu}] = \mu$

$Var[\hat{\mu}] = Var[\frac{1}{n}\sum X_i] = \frac{1}{n^2}Var[\sum X_i] = \sigma^2/n$

We can reduce variance by taking a sample of B points:

$Var[\hat{\mu}] = Var[\frac{1}{B}\sum X_i] = \frac{1}{B^2}Var[\sum X_i] = \sigma^2/Bn$

We can sample these points with replacement, and in practice this will work.

We aggregate by doing regression $f(x) = \frac{1}{B}\sum f^j(x)$.

Classification done by majority vote.

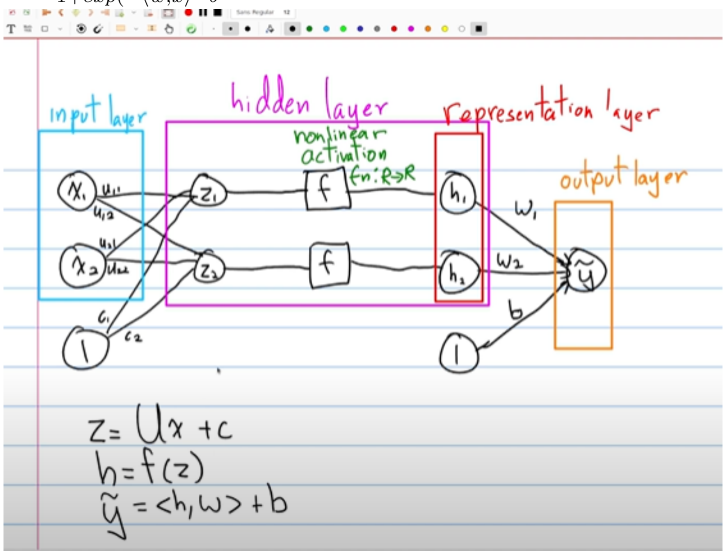
Random Forests Bootstrap but select $\sqrt{d}$ features.

Multilayer Perceptron

Neural Networks learn mapping from the data.

Sigmoid: $\sigma(t) = \frac{1}{1+e^{-t}}$

$\hat{y} = \frac{1}{1+exp(-\langle w,x\rangle - b})$



ReLU(t) $= \max(0, t)$

Loss $l_\theta(x, y) = -\sum^m y_i log y_i$

Tanh(t)$= \frac{e^t - e^{-1}}{e^t + e^{-t}}$

Gradient Descent $\theta^t = \theta^{t-1} - \eta\Delta L_{\theta t-1}$

Chain rule: $\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$

Any contionus function can be approximated well by a 2 layer nn.

Deep Networks

Most neural networks have many paramaters. We can minimize overfiting with:

Regularazation loss, gradient descent, and equivalently:

$\theta_t \leftarrow (1-\eta\lambda)\theta_{t-1} - \eta\Delta L\theta t - 1)(x, y)$

We can drop nodes and use normlizaiton of features:

mean$= \frac{1}{n}\sum X_i$, $X_i \leftarrow X_i - \mu$, $\sigma_j^2 = \frac{1}{n}\sum X_{i,j}$, $X_{i,j} = X_{i,j}/\sigma_j$

We do normilizaiton on each batch, such that:

$z^i = W^i h^{i-1} + b^i$ or $h^i = f(z^i)$

We can do normlization on each neuron (batchnorm) or each layer.

Batch GD: $\theta \leftarrow \theta - \eta * \frac{1}{n}\sum \delta l_\theta(x_i, j_i)$, optomize gradient.

Momentum: $v_t = \gamma v_{t-1} + (1-\gamma)\mu$, $\theta_t \leftarrow \theta_{t-1} - v_t$ **RMSProp** let $g \in R^p$, $G_{t,i} = \sum^t g_{j,i}^2$ and $\theta_t \leftarrow \theta_{t-1} - \frac{\mu}{\sqrt{G_{t,i}+\epsilon}} g_{t,i}$

- $\beta_1, \beta_2, \varepsilon$ hyperparameters
  - $\beta_1 = 0.9, \beta_2 = 0.999, \varepsilon = 10^{-8}$
- $m_{t,i} = \beta_1 m_{t-1,i} + (1-\beta_1)g_{t,i}$ (momentum)
- $v_{t,i} = \beta_2 v_{t-1,i} + (1-\beta_2)g_{t,i}^2$ (RMSProp)
- $\hat{m}_{t,i} = \frac{m_{t,i}}{1-\beta_1^t}$, $\hat{v}_{t,i} = \frac{v_{t,i}}{1-\beta_2^t}$
- $\theta_{t,i} \leftarrow \theta_{t-1,i} - \frac{\eta}{\sqrt{\hat{v}_{t,i}+\varepsilon}}\hat{m}_{t,i}$

[Perceptron] Algorthim

Algorithm: The Perceptron (Rosenblatt 1958)

Input: Dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i) \in \mathbb{R}^d \times \{\pm 1\} : i = 1, \ldots, n\}$, initialization $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$, threshold $\delta \geq 0$

Output: approximate solution $\mathbf{w}$ and b

```
1 for t = 1, 2, ... do
2 |  receive training example index I_t ∈ {1,...,n}      // the index I_t can be random
3 |  if y_{I_t}(w^T x_{I_t} + b) ≤ δ then
4 |  |  w ← w + y_{I_t} x_{I_t}                          // update only after making a "mistake"
5 |  |  b ← b + y_{I_t}
```

[Perceptron] Error Bound

- Theorem (informal): If $\exists z, s$ such that $Az \geq s\bar{1}$, perceptron makes at most $R^2\|z\|_2^2/s^2$ mistakes, where $R = \max\|a_i\|_2$.
  - Pick the "best" one to minimize $\|z\|_2^2/s^2$ and thus the number of mistakes

$$\min_{(z,s):Az\geq s\bar{1}} \frac{\|z\|_2^2}{s^2} = \min_{(z,s):\|z\|_2=1, Az\geq s\bar{1}} \frac{1}{s^2}$$

$$= \frac{1}{\left(\max_{(z,s):\|z\|_2=1, Az\geq s\bar{1}} s\right)^2} = \frac{1}{\left(\max_{\|z\|_2=1} \min_i \langle a_i, z\rangle\right)^2} = \frac{1}{\gamma^2}$$

$\gamma = \max_{\|z\|_2=1} \min_i \langle a_i, z\rangle$ is the *margin* of the solution wrt the dataset.

[Linear Regression] Convexity of Loss Function

- Loss fn $\|Aw - z\|_2^2 = (Aw - z)^T(Aw - z) = (w^TA^T - z^T)(Aw - z)$
  $= w^TA^TAw - z^TAw - w^TA^Tz + z^Tz$
  $= w^TA^TAw - 2w^TA^Tz + z^Tz$
- Claim: if $f(x) = x^TAx + x^Tb + c$, then $\nabla f(x) = (A + A^T)x + b$
- Thus $\nabla_w\|Aw - z\|_2^2 = 2A^TAw - 2A^Tz$
- Checking the Hessian, $\nabla_w^2\|Aw - z\|_2^2 = 2A^TA \succcurlyeq 0$
  - Why? Since $2v^TA^TAv = 2\|Av\|_2^2 \geq 0$ for any vector v

[Linear Regression] Deriving MLE

$y = \langle w, x\rangle + z$, where $z \sim N(0, \sigma^2)$

$$\hat{w} = \arg\max_w \Pr[(x_1, y_1), \ldots, (x_n, y_n)|w]$$
$$= \arg\max_w \prod_i \Pr[(x_i, y_i)|w]$$
$$= \arg\max_w \prod_i \Pr[y_i|x_i, w]\Pr[x_i|w]$$
$$= \arg\max_w \prod_i \Pr[y_i|x_i, w]$$
$$= \arg\max_w \prod_i \Pr[y_i|x_i, w]$$
$$= \arg\max_w \log\left(\prod_i \Pr[y_i|x_i, w]\right)$$
$$= \arg\max_w \sum_i \log(\Pr[y_i|x_i, w])$$

(Note: $y_i|x_i, w \sim N(\langle w, x\rangle, \sigma^2)$)

$$= \arg\max_w \sum \log\left(\frac{1}{\sqrt{2\pi\sigma}}\exp\left(-\frac{(y - \langle w, x\rangle)^2}{2\sigma^2}\right)\right)$$
$$= \arg\max_w \sum_i \log\left(\frac{1}{\sqrt{2\pi\sigma}}\right) + \log\left(\exp\left(-\frac{(y - \langle w, x\rangle)^2}{2\sigma^2}\right)\right)$$
$$= \arg\max_w \sum_i -\frac{(y - \langle w, x\rangle)^2}{2\sigma^2}$$
$$= \arg\min_w \sum_i (y - \langle w, x\rangle)^2$$

Loss function is the squared error!

[Linear Regression] Cross Validation

- Split training data into k sets (draw on board), e.g. $k = 10$ is common

For each λ:
  For $i = 1$ to k:
    $w_{\lambda,i}$ = train on all data but split i with hyperparameter λ
    perf$_{\lambda,i}$ = performance of $w_{\lambda,i}$ on the split i
  perf$_\lambda = \sum_i$ perf$_{\lambda,i}$
Return λ which has the biggest perf$_\lambda$

[KNN] Algorithm

- $\Pr_{y\sim D_{Y|X=x}}[y = c|x] \approx \Pr_{y'\sim D_{Y|X=x'}}[y' = c|x']$ when x and x' are close

Algorithm: kNN

Input: Dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i) \in \mathsf{X} \times \mathsf{Y} : i = 1, \ldots, n\}$, new instance $\mathbf{x} \in \mathsf{X}$, hyperparameter k

Output: $\mathbf{y} = \mathbf{y}(\mathbf{x})$

```
1 for i = 1, 2, ..., n do
2 |  d_i ← dist(x, x_i)                       // avoid for-loop if possible
3 find indices i_1, ..., i_k of the k smallest entries in d
4 y ← aggregate(y_{i_1}, ..., y_{i_k})
```

[Logistic Regression] MLE of $\hat{w}$

$$\hat{w} = \arg\min_w \frac{1}{n}\sum_{i=1}^n \log\big(\exp(-y_i\langle x_i, w\rangle) + \exp((1 - y_i)\langle x, w\rangle)\big)$$

Let $\tilde{y}_i = +1$ if $y_i = 1$, and $\tilde{y}_i = -1$ if $y_i = 0$.

$$\hat{w} = \arg\min_w \frac{1}{n}\sum_{i=1}^n \log(1 + \exp(-\tilde{y}_i\langle x_i, w\rangle))$$

[Logistic Regression] Logit Transform

$$\log\left(\frac{p(x,w)}{1-p(x,w)}\right) = \langle x, w\rangle$$

$$\frac{p(x,w)}{1-p(x,w)} = \exp(\langle x,w\rangle) \text{ (LHS: "odds ratio")}$$

$$p(x,w) = \exp(\langle x,w\rangle)\,(1-p(x,w))$$

$$p(x,w) = \exp(\langle x,w\rangle) - \exp(\langle x,w\rangle)\,p(x,w)$$

$$p(x,w)(1+\exp(\langle x,w\rangle)) = \exp(\langle x,w\rangle)$$

$$p(x,w) = \frac{\exp(\langle x,w\rangle)}{1+\exp(\langle x,w\rangle)}$$

$$p(x,w) = \frac{1}{1+\exp(-\langle x,w\rangle)} \triangleq \text{sigmoid}(\langle x,w\rangle)$$

[Logistic Regression] Deriving MLE #1

$$\hat{w} = \arg\max_w \prod_{i=1}^n \Pr[(x_i,y_i)|w]$$

$$= \arg\max_w \prod_{i=1}^n p(x_i,w)^{y_i}\left(1-p(x_i,w)\right)^{1-y_i}$$

(Let $p_i = p_i(x_i,w)$)

$$= \arg\max_w \log \prod_{i=1}^n p_i^{y_i}(1-p_i)^{1-y_i}$$

$$= \arg\max_w \sum_{i=1}^n \log\left(p_i^{y_i}(1-p_i)^{1-y_i}\right)$$

$$= \arg\max_w \sum_{i=1}^n y_i \log p_i + (1-y_i)\log(1-p_i)$$

("cross entropy loss")

Recall: $p(x,w) = \frac{1}{1+\exp(-\langle x,w\rangle)}$.

Suppose some $y_i = 1$. Then

$$y_i \log p_i + (1-y_i)\log(1-p_i) = \log p_i$$
$$= \log(1+\exp(-\langle x_i,w\rangle))^{-1}$$
$$= -\log(1+\exp(-\langle x_i,w\rangle))$$

Similarly, if $y_i = 0$, then

$$y_i \log p_i + (1-y_i)\log(1-p_i) = \log(1-p_i)$$
$$= \log\left(\frac{\exp(-\langle x_i,w\rangle)}{1+\exp(-\langle x_i,w\rangle)}\right)$$
$$= -\log(1+\exp(\langle x_i,w\rangle))$$

[Logistic Regression] Deriving MLE #2

$$\hat{w} = \arg\max_w \sum_{i=1}^n y_i \log p_i + (1-y_i)\log(1-p_i)$$

If $y_i = 1$, argument is $-\log(1+\exp(-\langle x_i,w\rangle))$

If $y_i = 0$, argument is $-\log(1+\exp(\langle x_i,w\rangle))$

$$\hat{w} = \arg\max_w \sum_{i=1}^n -\log\left(\exp(-y_i\langle x_i,w\rangle) + \exp\left((1-y_i)\langle x_i,w\rangle\right)\right)$$

$$\hat{w} = \arg\min_w \frac{1}{n}\sum_{i=1}^n \log\left(\exp(-y_i\langle x_i,w\rangle) + \exp\left((1-y_i)\langle x_i,w\rangle\right)\right)$$

[Logistic Regression] d_t Selection

- Gradient Descent
  - $d_t = \frac{1}{n}\sum_{i=1}^n \nabla_w \ell_{w_{t-1}}(x_i,y_i)$ (note, $= 0$ at optimum)
  - Running time?
- Stochastic gradient descent
  - Draw random set $B \subseteq [n]$, then let $d_t = \frac{1}{|B|}\sum_{i\in B}\nabla_w \ell_{w_{t-1}}(x_i,y_i)$
- Newton's Method
  - $d_t = \left(\frac{1}{n}\sum_{i=1}^n \nabla_w^2 \ell_{w_{t-1}}(x_i,y_i)\right)^{-1}\left(\frac{1}{n}\sum_{i=1}^n \nabla_w \ell_{w_{t-1}}(x_i,y_i)\right)$
  - Often needs fewer steps to converge, but more time/memory per step

[Hard Margin SVM] Goal

- SVM: Like perceptron, but try to maximize margin

$$\max_{w',b'} \gamma, \text{ s.t. } \|w'\|_2 = 1, y_i(\langle w', x_i\rangle + b') \geq \gamma \text{ for all } i$$

- Substitute $w' = \gamma w$, $b' = \gamma b$

$$\max_{\gamma w,\gamma b} \gamma, \text{ s.t. } \|w\|_2 = 1/\gamma, y_i(\langle \gamma w, x_i\rangle + \gamma b) \geq \gamma \text{ for all } i$$

$$\max_{\gamma w,\gamma b} \gamma, \text{ s.t. } \|w\|_2 = 1/\gamma, y_i(\langle w, x_i\rangle + b) \geq 1 \text{ for all } i$$

$$\max_{\gamma w,\gamma b} \frac{1}{\|w\|_2}, \text{ s.t. } y_i(\langle w, x_i\rangle + b) \geq 1 \text{ for all } i$$

$$\min_{w,b} \frac{1}{2}\|w\|_2^2 \text{ s.t. } y_i(\langle w, x_i\rangle + b) \geq 1 \text{ for all } i$$

[Hard Margin SVM] Duel Formation

$$\max_{\alpha\in\mathbf{R}^n,\alpha\geq 0} \min_{w,b} \frac{1}{2}\|w\|_2^2 - \sum_{i=1}^n \alpha_i\left(y_i(\langle w, x_i\rangle + b) - 1\right)$$

Fix some α for now, solve inner minimization. How? Set gradient = 0!

$$\frac{\partial}{\partial b} = -\sum_i \alpha_i y_i = 0, \qquad \frac{\partial}{\partial w} = w - \sum_i \alpha_i y_i x_i = 0$$

Substitute into above and rearrange...

$$\min_{\alpha\in\mathbf{R}^n,\alpha\geq 0} \frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j y_i y_j \langle x_i, x_j\rangle - \sum_i \alpha_i \text{ s.t. } \sum_i \alpha_i y_i = 0$$

[Soft Margin SVM] Goal

$$\min_{w,b} \frac{1}{2}\|w\|_2^2 + C\sum_i \max(0, 1 - y_i\hat{y}_i)$$

- (draw cases of $y_i\hat{y}_i$, versus hard-margin case)
  - If $1 - y_i\hat{y}_i \leq 0$, then on the correct side of margin
  - If $0 \leq y_i\hat{y}_i \leq 1$, correctly classified but within margin
  - If $y_i\hat{y}_i \leq 0$, incorrectly classified
- (draw hinge loss versus 0-1 loss, perceptron)
- If $C = 0$, ignore data, if $C = \infty$, hard-margin SVM

[Soft Margin SVM] Duel Formation

$$\min_{w,b,\gamma} \frac{1}{2}\|w\|_2^2 + C\sum_i \gamma_i \text{ s.t. } 0 \leq \gamma_i \text{ and } 1 - y_i\hat{y}_i \leq \gamma_i \text{ for all } i$$

- Introduce dual variables and take Lagrangian

$$\max_{\alpha,\beta\in\mathbf{R}^n,\alpha,\beta\geq 0} \min_{w,b,\gamma} \frac{1}{2}\|w\|_2^2 + \sum_i \left(C\gamma_i + \alpha_i(1 - y_i\hat{y}_i - \gamma_i) - \beta_i\gamma_i\right)$$

- Take derivative of inner problem, set to 0, substitute, simplify... (exercise)

$$\min_{\alpha\in\mathbf{R}^n, C\geq\alpha\geq 0} \frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j y_i y_j \langle x_i, x_j\rangle - \sum_i \alpha_i \text{ s.t. } \sum_i \alpha_i y_i = 0$$

[Soft Margin SVM] Optimization

- $\ell_{w,b}(x,y) = \max\left(0, 1 - y(\langle w, x\rangle + b)\right)$
- Optimize loss function

$$L = \frac{C}{n}\sum_i \ell_{w,b}(x_i,y_i) + \frac{1}{2}\|w\|_2^2$$

  - Normalize by n this time, same problem just rescaled
  - Note similarity to ridge regression (C on former vs λ on latter)
- Gradient descent: $\frac{\partial L}{\partial w} = w + \frac{C}{n}\sum \delta_i$
  - $\delta_i = -y_i x_i$ if $1 - y_i\hat{y}_i \geq 0$, $\delta_i = 0$ if $1 - y_i\hat{y}_i \leq 0$ (draw, note non-diff pt)

[Decision Tree] Example

$$S_L = \{(x_i,y_i) : x_{ij} \leq t\}, S_R = \{(x_i,y_i) : x_{ij} > t\}$$
$$(j^*, t^*) = \arg\min_{j,t} |S_L|\ell(S_L) + |S_R|\ell(S_R)$$

Gini index: $\sum_{classes\ c} \hat{p}_c(1-\hat{p}_c)$

Split on smokes?

No: $\hat{p}_0 = \frac{1}{4}, \hat{p}_1 = \frac{3}{4}$. Yes: $\hat{p}_0 = \frac{2}{3}, \hat{p}_1 = \frac{1}{3}$.

Cost: $4\cdot\left(\left(\frac{3}{4}\right)\left(\frac{1}{4}\right) + \left(\frac{1}{4}\right)\left(\frac{3}{4}\right)\right) + 6\cdot\left(\left(\frac{2}{3}\right)\left(\frac{1}{3}\right) + \left(\frac{1}{3}\right)\left(\frac{2}{3}\right)\right) = 4.16$

Split on age? (Cheat to save time: use 35 as split)

≤ 35: $\hat{p}_0 = 1, \hat{p}_1 = 0$. > 35: $\hat{p}_0 = \frac{1}{6}, \hat{p}_1 = \frac{5}{6}$.

Cost: $4\cdot((0)(1) + (1)(0)) + 6\cdot\left(\left(\frac{1}{6}\right)\left(\frac{5}{6}\right) + \left(\frac{5}{6}\right)\left(\frac{1}{6}\right)\right) = 1.66$

Age	Smokes	Cancer?
10	No	0
18	Yes	0
25	No	0
35	Yes	0
50	No	1
55	Yes	1
70	Yes	1
80	No	0
85	Yes	1
90	Yes	1

[Boosting] Hedge Algorithm

Hedge(β), where $\beta \in [0,1]$

1. Initialize $w^{(1)} = [1/n, \dots, 1/n] \in \mathbf{R}^n$
2. For $t = 1, \dots, T$
   1. Set $p^{(t)} = \frac{w^{(t)}}{\sum_i w_i^{(t)}}$ (normalize w into a distribution)
   2. Receive loss $\langle p^{(t)}, \ell^{(t)} \rangle$
   3. Update $w_i^{(t+1)} = w_i^{(t)} \beta^{\ell_i^{(t)}}$ (downweight experts based on loss)

Guarantee: $\sum_{t=1}^{T} \langle p^{(t)}, \ell^{(t)} \rangle \le \frac{1}{1-\beta}\left(\log n + \log(1/\beta) \min_i \sum_{t=1}^{T} \ell_i^{(t)}\right)$

[Boosting] AdaBoost Algorithm

1. Initialize $w^{(1)} = [1/n, \dots, 1/n] \in \mathbf{R}^n$
2. For $t = 1, \dots, T$
   1. Set $p^{(t)} = \frac{w^{(t)}}{\sum_i w_i^{(t)}}$ (normalize w into a distribution)
   2. Run WeakLearn on training set (with weights $p^{(t)}$)
      - Obtain classifier $h^{(t)}$ which maps (x, y) datapoints to $[0,1]$ (confidence in classification)
   3. Calculate error $\varepsilon_t = \sum_i p_i^{(t)} |h^{(t)}(x_i) - y_i|$ (should be < 0.5 by WeakLearn guarantees)
   4. Define $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$, if $\varepsilon_t \le 1/2$ set $w_i^{(t+1)} = w_i^{(t)} \beta_t^{1-|h^{(t)}(x_i) - y_i|}$
      - Note: If ε_t big, then β_t is big. Many errors, so don't downweight points!
3. $h(x) = 1$ if $\sum_{t=1}^{T} \log\left(\frac{1}{\beta_t}\right) h^{(t)}(x) \ge \frac{1}{2} \sum_{t=1}^{T} \log\left(\frac{1}{\beta_t}\right)$, 0 else

[Multilevel Perceptron] 2 Layer Perceptron

- $z = Ux + c,\ h = f(z),\ \tilde{y} = \langle h, w \rangle + b$
- Consider: $U = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, c = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, w = \begin{bmatrix} 2 \\ -4 \end{bmatrix}, b = -1$
  - Parameters to be learned, but suppose they're just given for now
- Choose $f(t) = \max(0, t) = \mathrm{ReLU}(t)$ (draw)
  - *Activation function* – this is a hyperparameter choice
- $x_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, y_1 = -1.\ z = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, h = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \tilde{y} = -1$
- $x_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, y_1 = 1.\ z = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, h = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \tilde{y} = 2 - 1 = 1$, etc.

[Multilevel Perceptron] 3 Layer Perceptron

- (Draw wider three-layer MLP, input $x \in \mathbf{R}^d$, output $\tilde{y} \in \mathbf{R}^m$)
  - (Illustrate depth and width, representation layer)
- $z^{(1)} = W^{(1)}x, h^{(1)} = f\left(z^{(1)}\right), z^{(2)} = W^{(2)}h^{(1)}, \dots$
- What to do with output $\tilde{y} \in \mathbf{R}^m$?
  - Put through *softmax* to get distribution over m classes (confidences of each)
  - $\hat{y}_i = \frac{\exp(\tilde{y}_i)}{\sum_{j=1}^{m} \exp(\tilde{y}_j)}$
- What loss function? Use the *cross-entropy* loss
  - $\ell_\theta(x, y) = -\sum_{i=1}^{m} y_i \log \hat{y}_i$
    - Use "one-hot encoding" of y: if $y = c$, then $y_c = 1$, and $y_i = 0$ for other entries
    - $\hat{y} = g_\theta(x)$, where g_θ is a (somewhat complicated) function

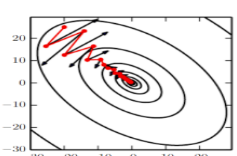
[Multilevel Perceptron] Back Propagation

- Simple case: $x \in \mathbf{R}, f, g: \mathbf{R} \to \mathbf{R}$
- Say $y = g(x), z = f(y) = f(g(x)$
- Chain rule: $\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}$
- More complex: $z = ux, h = f(z), y = wh, L = g(y)$ (draw)
- Can compute several derivatives easily: $\frac{dL}{dy}, \frac{dy}{dw}, \frac{dy}{dh}, \frac{dh}{dz}, \frac{dz}{du}$
- But we care about derivatives of L wrt parameters u, w
- $\frac{dL}{dw} = \frac{dL}{dy} \cdot \frac{dy}{dw}$ and $\frac{dL}{du} = \frac{dL}{dy} \cdot \frac{dy}{dh} \cdot \frac{dh}{dz} \cdot \frac{dz}{du}$

[Deep Networks] Momentum

Momentum

- Keep memory of previous gradient step
- Let $\gamma < 1$ (say $= 0.9$)
- $v_t = \gamma v_{t-1} + (1-\gamma)\eta \cdot \frac{1}{|B|} \sum_{i \in B} \nabla_{\theta_{t-1}} \ell_{\theta_{t-1}}(x_i, y_i)$
  - New step: weighted sum of old step and current gradient
- $\theta_t \leftarrow \theta_{t-1} - v_t$
- $v_t = 0.1 g_t + 0.1 \cdot 0.9 g_{t-1} + 0.1 \cdot 0.9^2 g_{t-2} + \cdots$
  - Total coefficient $1 - \gamma^t$
- Variant: Nesterov momentum



[A #1] Ridge Regression

$$\min_{\mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}} \frac{1}{2n} \left\| \begin{bmatrix} X & \mathbf{1}_n \\ \sqrt{2\lambda n} I_d & 0_d \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix} - \begin{bmatrix} \mathbf{y} \\ 0_d \end{bmatrix} \right\|_2^2,$$

$$= \min_{\mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}} \frac{1}{2n} \left\| \begin{bmatrix} X\mathbf{w} + b\mathbf{1}_n \\ \sqrt{2\lambda n}\mathbf{w} I_d \end{bmatrix} - \begin{bmatrix} \mathbf{y} \\ 0_d \end{bmatrix} \right\|_2^2,$$

$$= \min_{\mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}} \frac{1}{2n} \left\| \begin{bmatrix} X\mathbf{w} + b\mathbf{1}_n - \mathbf{y} \\ \sqrt{2\lambda n}\mathbf{w} I_d \end{bmatrix} \right\|_2^2,$$

$$= \min_{\mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}} \frac{1}{2n} \left[(X\mathbf{w} + b\mathbf{1}_n - \mathbf{y})^T \quad (\sqrt{2\lambda n}\mathbf{w} I_d)^T \right] \begin{bmatrix} X\mathbf{w} + b\mathbf{1}_n - \mathbf{y} \\ \sqrt{2\lambda n}\mathbf{w} I_d \end{bmatrix}$$

$$= \min_{\mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}} \frac{1}{2n} \left[\|X\mathbf{w} + b\mathbf{1}_n - \mathbf{y}\|_2^2 + 2\lambda n \|\mathbf{w}\|_2^2 \right],$$

$$= \min_{\mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}} \frac{1}{2n} \|X\mathbf{w} + b\mathbf{1}_n - \mathbf{y}\|_2^2 + \lambda \|\mathbf{w}\|_2^2,$$

[A #1] Ridge Regression Derivatives

$$= \tfrac{1}{2n} \left\{ \left[(X^T X) + (X^T X)^T \right] \mathbf{w} + 2X^T (b\mathbf{1}_n - \mathbf{y}) \right\} + 2\lambda \mathbf{w}$$

$$= \tfrac{1}{2n} \left\{ \left[(X^T X) + (X^T X) \right] \mathbf{w} + 2X^T (b\mathbf{1}_n - \mathbf{y}) \right\} + 2\lambda \mathbf{w}$$

$$= \tfrac{1}{2n} \left\{ \left[2(X^T X) \right] \mathbf{w} + 2X^T (b\mathbf{1}_n - \mathbf{y}) \right\} + 2\lambda \mathbf{w}$$

$$= \tfrac{1}{n} \left\{ X^T X \mathbf{w} + X^T (b\mathbf{1}_n - \mathbf{y}) \right\} + 2\lambda \mathbf{w}$$

$$\frac{\partial}{\partial \mathbf{w}} = \tfrac{1}{n} X^\top (X\mathbf{w} + b\mathbf{1}_n - \mathbf{y}) + 2\lambda \mathbf{w}$$

$$\frac{\partial}{\partial b} \left[\tfrac{1}{2n} \|X\mathbf{w} + b\mathbf{1}_n - \mathbf{y}\|_2^2 + \lambda \|\mathbf{w}\|_2^2 \right]$$

$$= \frac{\partial}{\partial b} \left\{ \tfrac{1}{2n} \left[\mathbf{w}^T X^T X \mathbf{w} + 2(b\mathbf{1}_n - \mathbf{y})^T X \mathbf{w} + (b^2 \mathbf{1}_n^T \mathbf{1}_n - 2b\mathbf{1}_n^T \mathbf{y} + \mathbf{y}^T \mathbf{y}) \right] + \lambda \mathbf{w}^T \mathbf{w} \right\}$$

$$= \tfrac{1}{n} \left[\mathbf{1}_n^T X \mathbf{w} + b\mathbf{1}_n^T \mathbf{1}_n - \mathbf{1}_n^T \mathbf{y} \right]$$

$$= \tfrac{1}{n} \mathbf{1}_n^T \left[X\mathbf{w} + b\mathbf{1}_n - \mathbf{y} \right]$$

$$\frac{\partial}{\partial b} = \tfrac{1}{n} \mathbf{1}_n^\top (X\mathbf{w} + b\mathbf{1}_n - \mathbf{y})$$

[A #2] Kernels

$$k(x, y) = \exp\left(-\alpha (x - y)^2\right)$$

$$= \exp(-\alpha(\mathbf{x}^2 + \mathbf{y}^2 - 2\mathbf{x}\mathbf{y}))$$

$$= \exp(-\alpha(\mathbf{x}^2 + \mathbf{y}^2)) \exp(2\alpha \mathbf{x}\mathbf{y})$$

$$= \exp(-\alpha(\mathbf{x}^2 + \mathbf{y}^2))(1 + \frac{2\alpha \mathbf{x}\mathbf{y}}{1!} + \frac{(2\alpha \mathbf{x}\mathbf{y})^2}{2!} + \frac{(2\alpha \mathbf{x}\mathbf{y})^3}{3!} \dots)$$

$$= exp(-\alpha(\mathbf{x}^2 + \mathbf{y}^2))(1 \cdot 1 + \sqrt{\frac{2\alpha}{1!}}\mathbf{x} \cdot \sqrt{\frac{2\alpha}{1!}}\mathbf{y} + \sqrt{\frac{(2\alpha)^2}{2!}}\mathbf{x}^2 \cdot \sqrt{\frac{(2\alpha)^2}{2!}}\mathbf{y}^2 + \sqrt{\frac{(2\alpha)^3}{3!}}\mathbf{x}^3 \cdot \sqrt{\frac{(2\alpha)^3}{3!}}\mathbf{y}^3 + \dots)$$

$$= [exp(-\alpha \mathbf{x}^2) \cdot (\sum_{n=0}^{\infty} \sqrt{\frac{(2\alpha)^n}{n!}}\mathbf{x}^n)] \cdot [exp(-\alpha \mathbf{y}^2) \cdot (\sum_{n=0}^{\infty} \sqrt{\frac{(2\alpha)^n}{n!}}\mathbf{y}^n)]$$

So, the corresponding feature map is $\phi(t) = exp(-\alpha t^2)[1, \sqrt{\frac{2\alpha}{1!}}t, \sqrt{\frac{(2\alpha)^2}{2!}}t^2, \dots]^T$.

Dual representation would be preferred.The primal representation of soft-margin support vector machine would be

$$\min_{\mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}} \frac{1}{2}\|\mathbf{w}\|_2^2 + C \sum^{n} \max(0, 1 - \mathbf{y}_i(\langle \mathbf{w}\phi(\mathbf{x}_i) \rangle + b))$$

[A #2] Duels

intermediate steps so that you can get partial credit.

Ans:

$$\min_{\mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}} \frac{1}{2}\|\mathbf{w}\|_2^2 + C \sum_{i=1}^{n} \max\{|y_i - (\mathbf{w}^\top \mathbf{x}_i + b)| - \varepsilon, 0\}$$

$$= \min_{\mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}} \frac{1}{2}\|\mathbf{w}\|_2^2 + C \sum_{i=1}^{n} \max\{y_i - (\mathbf{w}^\top \mathbf{x}_i + b), (\mathbf{w}^\top \mathbf{x}_i + b) - y_i, 0\}$$

Let $\mathbf{w}^\top \mathbf{x}_i + b = \hat{\mathbf{y}}_i$

$$= \min_{\mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}} \frac{1}{2}\|\mathbf{w}\|_2^2 + C \sum_{i=1}^{n} \gamma_i \quad s.t.\ \forall i \quad \max(y_i - \hat{\mathbf{y}}_i, \hat{\mathbf{y}}_i - y_i, 0) \le \gamma_i$$

$$= \min_{\mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}} \frac{1}{2}\|\mathbf{w}\|_2^2 + C \sum_{i=1}^{n} \gamma_i \quad s.t.\ \forall i \quad 0 \le \gamma_i,\ y_i - \hat{\mathbf{y}}_i \le \gamma_i,\ \hat{\mathbf{y}}_i - y_i \le \gamma_i$$

$$= \min_{\mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}} \max_{\alpha, \beta, \theta \ge 0} \frac{1}{2}\|\mathbf{w}\|_2^2 + \sum_{i=1}^{n} [C\gamma_i - \alpha_i \gamma_i + \beta_i(y_i - \hat{\mathbf{y}}_i - \varepsilon - \gamma_i) + \theta_i(\hat{\mathbf{y}}_i - y_i - \varepsilon - \gamma_i)]$$

$$= \max_{\alpha, \beta, \theta \ge 0} \min_{\mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}} \frac{1}{2}\|\mathbf{w}\|_2^2 + \sum_{i=1}^{n} [C\gamma_i - \alpha_i \gamma_i + \beta_i(y_i - \hat{\mathbf{y}}_i - \varepsilon - \gamma_i) + \theta_i(\hat{\mathbf{y}}_i - y_i - \varepsilon - \gamma_i)]$$

$$\frac{\partial}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n} (\beta_i \mathbf{x}_i - \theta_i \mathbf{x}_i) = 0,\ \mathbf{w} = \sum_{i=1}^{n} (\beta_i \mathbf{x}_i - \theta_i \mathbf{x}_i)$$

$$\frac{\partial}{\partial \mathbf{b}} = \sum_{i=1}^{n} -\beta_i + \theta_i = 0,\ \sum_{i=1}^{n} \beta_i = \sum_{i=1}^{n} \theta_i$$

$$\frac{\partial}{\partial \gamma} = \sum_{i=1}^{n} C - \beta_i - \theta_i - \alpha_i = 0,\ C = \beta_i - \theta_i - \alpha_i$$

$$= \max_{\alpha, \beta, \theta \ge 0} \frac{1}{2}\|\mathbf{w}\|_2^2 + \sum_{i=1}^{n} [C\gamma_i + \beta_i \mathbf{y}_i - \beta_i \hat{\mathbf{y}}_i - \beta_i \varepsilon - \beta_i \gamma_i + \theta_i \hat{\mathbf{y}}_i - \theta_i \mathbf{y}_i - \theta_i \varepsilon - \theta_i \gamma_i - \alpha_i \gamma_i]$$