1 Essentials

1.1 Adaptive Stepsize

Line search: Optimize step size at every step along gradient
Bold driver: Objective decrease → increase step size, objective increase → decrease step size

1.2 Loss functions

L1 loss: $l_1(\mathbf{w};x_i,y_i) = |y_i - \mathbf{w}^T x_i|$
Lp loss: $l_p(\mathbf{w};x_i,y_i) = |y_i - \mathbf{w}^T x_i|^p$
0/1 loss: $l_{0/1}(\mathbf{w};x_i,y_i) = \left\{ \begin{array}{l} 0, \text{ if } sign(\mathbf{w}^T x_i) = y_i \\ 1, \text{ else} \end{array} \right\}$
Perceptron loss: $l_{\text{perc}}(\mathbf{w};x_i,y_i) = \left\{ \begin{array}{l} 0, \text{ if } sign(\mathbf{w}^T x_i) = y_i \\ -y_i \mathbf{w}^T x_i, \text{ else} \end{array} \right\}$ $= max(0, -y_i \mathbf{w}^T x_i)$
Cost sensitive perceptron: $l_{\text{cs}}(\mathbf{w};x_i,y_i) = c_y * l_p(\mathbf{w};x_i,y_i)$
Hinge loss: $l_H(\mathbf{w}^T;x_i,y_i) = max(0, 1 - y_i \mathbf{w}^T x_i)$
Logistic loss: $l_{\text{logistic}}(\mathbf{w}^T, x_i, y_i) = log(1 + exp(-y_i \mathbf{w}^T x_i))$

1.3 Loss Function Derivatives

Perceptron loss: $\nabla_{\mathbf{w}} l_p = \left\{ \begin{array}{l} 0, \text{ if } -y_i \mathbf{w}^T x_i < 0 \\ -y_i x_i, \text{ else} \end{array} \right\}$

1.4 Distributions

1D-Gaussian: $P(X=x) = 1/\sqrt{2\pi\sigma^2} exp(-(x-\mu)^2/2\sigma^2)$
Bernoulli: $Ber(y;x) = \left\{ \begin{array}{l} x, \text{ if } y = +1 \\ 1-x, \text{ if } y = -1 \end{array} \right\}$

1.5 Matrix Calculus

Derivatives $\frac{\partial}{\partial x}\mathbf{A}\mathbf{x} = \mathbf{A}^T$
$\frac{\partial}{\partial x}\mathbf{x}^T\mathbf{A} = \mathbf{A}$
$\frac{\partial}{\partial x}\mathbf{x}^T\mathbf{x} = 2\mathbf{x}$
$\frac{\partial}{\partial x}\mathbf{x}^T\mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{x} + \mathbf{A}^T\mathbf{x}$
Ranks $rank(AB) \leq \min(rank(A), rank(B))$
Diverse X psd $\Rightarrow u^T X u \geq 0$
X pd $\Rightarrow u^T X u > 0$
X psd and Y pd $\Rightarrow X + Y$ pd
X pd $\Rightarrow$ invertible

1.6 Probabilistics

Multiplication: $P(A|B) = \frac{P(A,B)}{P(B)}$
Bayes: $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$

1.7 Gradient Descent

Normal: $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla_{\mathbf{w}} \hat{R}(\mathbf{w}_t)$
Stochastic: $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla_w l(\mathbf{w}_t, x', y')$ for random $(x', y') \in D$
SGD L2: $\mathbf{w}_{t+1} = \mathbf{w}_t(1 - 2\lambda\eta_t) - \eta_t \nabla_w l(\mathbf{w}_t, x', y')$

1.8 Fundamental assumptions

Optimal solution lies in span of data
Alternative Representation: $\mathbf{w}^* = \sum_{i=1}^n (\alpha_i y_i) x_i$ for some $\alpha_{1:n}$

2 Regression

2.1 Linear least sqares

Objective Function: $\hat{R}(\mathbf{w}) = \sum_{i=1}^n l_2(\mathbf{w};x_i,y_i)$
Closed Form: $\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T y$ with $\mathbf{X} = (x_1, x_2, \dots, x_n)^T$
Gradient: $\nabla_{\mathbf{w}} \hat{R}(\mathbf{w}) = -2\sum_{i=1}^n (y_i - \mathbf{w}^T x_i) x_i$
Runtime: Closed form $\Theta(n * d^2 + d^3)$, Gradient descent $\Theta(iter * n * d)$

2.2 Polynomial features

Aim: Fit non-linear functions via linear regression.
Solution: Use non-linear transformation of data
Ojbective Function: $\hat{R}(\mathbf{w}) = \sum_{i=1}^n (y_i - f(x))^2$
Transformation: $f(x) = \sum_{j=1}^d w_i \phi_i(x)$
Polynomial features: $x \to \phi(x)$

2.3 Ridge Regression

Objective Function: $\hat{R}(\mathbf{w}) = \sum_{i=1}^n l_2(\mathbf{w}, x_i, y_i) + \lambda \|\mathbf{w}\|_2^2$
Closed Form: $\mathbf{w}^* = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T y$
Gradient: $\nabla_{\mathbf{w}} \hat{R}(\mathbf{w}) = -2\sum_{i=1}^n (y_i - \mathbf{w}^T x_i) x_i + 2\lambda\mathbf{w}$
Notes: Scale of features matter. All features should be zero mean and unit variance
Use L1 regularizer to get LASSO for better feature selection

3 Classification

3.1 Nearest Neighbor

Idea: Use k closest neighbors to vote on new point x's class
Prediction: $\hat{y} = sign(\sum_{i:x_i \in KNN(x)} y_i)$ (binary case)

3.2 Perceptron algorithm

Stochastic gradient descent on perceptron loss

3.3 SVM

Objective Function: $\hat{R}(\mathbf{w}) = \sum_{i=1}^n l_H(\mathbf{w}, x_i, y_i) + \lambda \|\mathbf{w}\|_2^2$
Gradient: ???
Notes: Use L1 regularizer for better feature selection (L1 SVM)

3.4 Multiclass classification

1vAll: Train 1 classifier for each class. Choose classifier with biggest confidence.
1vAll prediction: $\hat{y} = \arg\max_{i=1:c} f_i(x)$ where $f_i(x)$ is the classifier for class i
1vAll Notes: Normalize weights $\hat{\mathbf{w}}_{\mathbf{i}} = \mathbf{w}_{\mathbf{i}}^* / \|\mathbf{w}_{\mathbf{i}}^*\|$ for determining confidence.
1v1: Train 1 classifier for each class pair. $(c(c-1)/2)$
1v1 prediction: Use voting to determine class

4 Kernels

4.1 Notation

Kernel Function: $k(x_i, x_j) = \phi(x_i)^T \phi(x_j)$
Gram Matrix: $\mathbf{K} = \begin{pmatrix} k(x_1,x_1), \dots, k(x_1,x_n) \\ \dots, \dots, \dots \\ k(x_n,x_1), \dots, k(x_n,x_n) \end{pmatrix}$
Kernelitem: $k_i = [y_1 k(x_i,x_1), y_2 k(x_i,x_2), \dots, y_n k(x_i,x_n)]$

4.2 General

Properties: inner product, symmetric, positive semidefinite
K. engineering: $k_1 + k_2$, $k_1 * k_2$, $c * k_1$ for $c > 0$, $f(k_1)$ for $f(x)$ polynomial or exp
Monomials of deg. m $k(x, x') = (x^T x')^m$
Monomials up to deg. m $k(x, x') = (1 + x^T x)^m$

4.3 Kernelized Linear Ridge Regression

Objective Function: $\hat{R}(\alpha) = \|\alpha^T \mathbf{K} - y\|_2^2 + \lambda \alpha^T \mathbf{K} \alpha$
Closed Form: $\alpha^* = (\mathbf{K} + \lambda\mathbf{I})^{-1} y$
Prediction: $\hat{y} = \sum_{i=1}^d \alpha_i^* k(x_i, x)$

4.4 Kernelized Perceptron

Objective Function: $\hat{R}(\alpha) = \sum_{i=1}^n \max(0, -y_i \alpha^T k_i)$
$\hat{R}(\alpha) = \sum_{i=1}^n \max(0, -y_i \sum_j \alpha_j y_j k(x_j, x_i))$

Prediction: $\hat{y} = sign(\sum_{i=1}^d \alpha_i^* y_i k(x_i, x))$
Optimize: if $\hat{y} \neq y_i$ set $\alpha_i = \alpha_i + \eta_t$

4.5 Kernelized SVM

Objective Function: $\hat{R}(\alpha) = \sum_{i=1}^n \max(0, 1 - y_i \alpha^T k_i) + \lambda \alpha^T \mathbf{K} \alpha$
Prediction: $\hat{y} = sign(\sum_{i=1}^d \alpha_i^* y_i k(x_i, x))$
Optimize: if $\hat{y} \neq y_i$ set $\alpha_i = \alpha_i + \eta_t$

5 Artificial Neural Networks

Transfer Function: $\sum_j w_j \varphi(\theta_j^T x)$, $w_j \mathrel{\hat{=}}$ hidden-to-output weight, $\theta_j^T \mathrel{\hat{=}}$ input-to-hidden weight
ReLU Act. Func:: $\varphi(z) = \max(z, 0)$

5.1 Propagation Algorithms

??

6 Practical Issues

6.1 Feature Selection

Greedy Forward: Start with no features. Always choose best feature to add next, until no improvement.
Greedy Backward: Start with all features. Always choose best feature to remove next, until no improvement.

6.2 Imbalanced Data

Subsampling Remove samples from majority class until balanced

Upsampling Repeat samples from minority class until balanced

Cost sensitive loss functions: See cost sensitive perceptron loss

6.3 Performance Metrics

Accuracy: $\frac{TP+TN}{TP+TN+FP+FN}$

Precision: $\frac{TP}{TP+FP}$

Recall: $\frac{TP}{TP+FN}$

F1-Score: $\frac{2TP}{2TP+FP+FN}$

What we want: Good F1-Score

7 Clustering

7.1 k-means

1: Initialize cluster centers at random

2: Assign each point to closest center

$z_i \leftarrow \arg\min_{j\in 1:k} \|x_i - \mu_j^{t-1}\|_2^2$

3: Update centers as mean of assigned points

$\mu_j^t \leftarrow 1/n_j \sum_{i:z_i=j} x_i$

Runtime $\Theta(iter * n * k * d)$

7.2 k-means++ (Initialization)

1: Start with random datapoint as center

2: Pick $\mu_j = x_i$ randomly s.t.

$P(\mu_j = x_i) = 1/Z \min_{l\in 1:j-1} \|x_i - \mu_l\|_2^2$

8 Probabilistic Modelling

8.1 Bayes Optimal Predictor

Assumption: $(x_i, y_i) \sim P(X,Y)$ i.i.d

Minimize: $\int P(x,y) l(y;h(x)) dx dy = \mathbb{E}_{x,y}[l(y;h(x))]$ by finding best $h(x)$

LS Solution: $h^*(x) = \mathbb{E}[Y|X=x] = \int P(Y|X=x) y dy$

Application: Estimate $P(Y|X=x)$ to predict label

8.2 Maximum Likelihood

Idea: Estimate parameters of model such that the likelihood of the labels is maximized

1: $\theta^* = \arg\max_\theta \hat{P}(y_1,\dots,y_n|x_1,\dots,x_n,\theta)$

$\Rightarrow \theta^* = \arg\min_\theta - \sum_i \log \hat{P}(y_i|x_i,\theta)$

2: Set derivative to zero, get θ^*

8.3 Maximum a Posteriori

Idea: Introduce assumption on distribution of parameters

1: $\arg\max_w P(w|x_{1:n}, y_{1:n}) = \arg\max_w \frac{P(w|x_{1:n})P(y_{1:n}|x_{1:n},w)}{P(y_{1:n}|x_{1:n})}$

$\Rightarrow \arg\min_w - \log P(w|x_{1:n}) - \log P(Y_{1:n}|x_{1:n},w) + \log P(y_{1:n}|x_{1:n})$

$\Rightarrow \arg\min_w - \log P(w) - \log P(Y_{1:n}|x_{1:n},w) + \log P(y_{1:n}|x_{1:n})$ (indep.)

$\Rightarrow \arg\min_w - \log P(w) - MLE + \log P(y_{1:n}|x_{1:n})$

$\Rightarrow \arg\min_w - \log P(w) - MLE$ (irrelevant, indep. of w)

$\Rightarrow \arg\min_w - \log \prod P(w_j) - MLE$ (iid)

2: Set derivative to zero

8.4 Logistic Regression

Link Function: $\sigma(\mathbf{w}'Tx) = \frac{1}{1+exp(-\mathbf{w}^T x)}$

Noise: Assume Bernoulli noise

Distribution: $P(y|x,\mathbf{w}) = Ber(y;\sigma(\mathbf{w}^T x)$

$\Rightarrow P(y|x,\mathbf{w}) = \frac{1}{1+exp(-y\mathbf{w}^T x)}$

Idea. Estimate above distribution using MLE

Gradient: $y\mathbf{x}\hat{P}(Y=-y|\mathbf{w},x)$

8.5 Bayesian Decision Theory

Idea: Assign cost to actions and minimize cost

Given: $P(y|x)$, Actions A, Cost function $y \times A \rightarrow \mathbb{R}$

Minimize: $a^* = \arg\min_{a\in A} \mathbb{E}_y[C(y,a)|x]$

$\Rightarrow a^* = \arg\min_{a\in A} \sum_y P(y|x)C(y,a)$ (discrete)

$\Rightarrow a^* = \arg\min_{a\in A} \int_y P(y|x)C(y,a)dy$ (cont.)

8.6 Uncertainity Sampling

Idea: Classify most uncertain points first

1: Estimate $\hat{P}(y_i|x_i)$ given D

2: Pick most uncertain data point

3: Classify point and set $D \leftarrow D \cup (x_i, y_i)$

4: Restart at **1**

9 Generative Modeling

1: Estimate prior $P(y)$

2: Estimate conditional $P(x|y)$

3: Then: $P(y|x) = \frac{1}{P(x)} P(y)P(x|y)$ and

$P(x,y) = P(x|y)P(x)$ with

$p(x) = \sum_y P(y)P(x|y)$

Prediction: $\hat{y} = \arg\max_y P(y|x)$

9.1 Naive Bayes Classifier

Class label: $P(Y=y) = p_y$ (categorical)

$\Rightarrow p_y = \frac{Count(Y=y)}{n}$

Features: $P(X_1,\dots,X_n|Y) = \prod_{i=1}^d P(X_i|Y)$ (independent)

$\Rightarrow$ Use MLE to estimate

Gauss NBC: $P(X_i|y) = \mathcal{N}(X_i|\mu_{y,i}, \sigma_{y,i}^2)$

$\mu_{y,i} = \frac{1}{Count(Y=y)} \sum_{j:y_j=y} x_{j,i}$

$\sigma_{y,i}^2 = \frac{1}{Count(Y=y)} \sum_{j:y_j=y} (x_{j,i} - \mu_{y,i})^2$

9.2 Gaussian Bayes Classifier

Class label: $P(Y=y) = p_y$ (categorical)

$\Rightarrow p_y = \frac{Count(Y=y)}{n}$

Features: $P(x|y) = \mathcal{N}(x, \mu_y, \Sigma_y)$ (multivariate)

Estimates: $\mu_y = \frac{1}{Count(Y=y)} \sum_{i:y_i=y} x_i$

$\Sigma_y = \frac{1}{Count(Y=y)} \sum_{i:y_i=y} ((x_i - \mu_y)(x_i - \mu_y)^T$

9.3 Outlier Detection

If $P(x) < \mathcal{T} \mathrel{\hat{=}}$ Threshold, throw away point.

10 Mixture Models for Clustering

Idea: Model each cluster j as $P(x|\theta_j)$

Assumption iid: $P(D|\theta) = \prod_{i=1}^n \sum_{j=1}^k w_j P(x_i|\theta_j)$

Minimization difficult! $\Rightarrow$ Soft/Hard EM

10.1 Hard EM

for t=1, ...

1: $z_i^t = \arg\max_z P(z|x_i, \theta^{t-1})$

$\Rightarrow z_i^t = \arg\max_z P(z|\theta^{t-1})P(x_i|z,\theta^{t-1})$

2: $\theta^t = \arg\max_\theta P(D^t|\theta)$

10.2 Soft EM

for t=1, ...

E-Step: $y_j^t(x) = P(Z=j|x,\Sigma,\mu,w)$

$\Rightarrow y_j^t(x) = \frac{w_j P(x|\Sigma_j,\mu_j)}{\sum_l w_l P(x|\Sigma_l,\mu_l)}$

M-Step: $\theta^t = \arg\max_\theta Q(\theta;\theta^{t-1})$

$Q(\theta;\theta^{t-1}) = \mathbb{E}_{z_{1:n}}[\log P(x_{1:n}, z_{1:n}|\theta)|x_{1:n}, \theta^{t-1}]$

$\Rightarrow \theta^t = \arg\max_\theta \sum_{i=1}^n \sum_{z_i=1}^k y_{z_i}(x_i) \log P(x_i, z_i|\theta)$

M-Step Gaussian: $w_j^t \leftarrow \frac{1}{n} \sum_{i=1}^n y_j^t(x_i)$

$\mu_j^t \leftarrow \frac{\sum_{i=1}^n y_j^t(x_i) * x_i}{\sum_{i=1}^n y_j^t(x_i)}$

$\Sigma_j^t \leftarrow \frac{\sum_{i=1}^n y_j^t(x_i)(x_i - \mu_j^t)(x_i - \mu_j^t)^T}{\sum_{i=1}^n y_j^t(x_i)}$

11 Markov Chains / Markov Model

Markov Assumption: $\forall t : P(Y_t|Y_1,\dots,Y_{t-1}) = P(Y_t|Y_{t-1})$

Stationary Assumption: $\forall t,y,y' : P(Y_{t+1} = y|Y_t = y') = P(Y_t = y|Y_{t-1} = y')$

Markov Chain: $p^t = [p_1^t, p_2^t, \dots, p_c^t]$

$T_{y',y} = P(Y_{t+1} = y|Y_t = y') = \theta_{y|y'}$

$\Rightarrow p^{t+1} = p^t * T$

MLE Estimation: $\hat{p}_y = \frac{Count(Y_1=y)}{m}$

$\hat{\theta}_{y|y'} = \frac{Count(Y_t=y, Y_{t-1}=y')}{Count(Y_{t-1}=y')}$