Course Notes

EECS 445

Introduction to Machine Learning

Honglak Lee - Fall 2015

Contributors: Max Smith

Latest revision: September 28, 2015

Contents

Abstract

Theory and implementation of state-of-the-art machine learning algorithms for large-scale real-world applications. Topics include supervised learning (regression, classification, kernel methods, neural networks, and regularization) and unsupervised learning (clustering, density estimation, and dimensionality reduction).

1 Introduction

- Formal definition: A computer program A is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T, as measured by P, improves with experience E
- Informal definition: Algorithms that improve their prediction performance at some task with experience (or data)
- e.g., spam filtering, handwritten digit recognition
- **Training**: given some example data you update parameters of your machine learning algorithm
- **Testing**: evaluating how well your algorithm performs on new data
- Machine learning tasks:
  - Supervised learning:
    - Classification
    - Regression
  - Unsupervised learning:
    - Clustering
    - Density estimation
    - Dimensionality reduction
  - Reinforcement Learning
    - Learning to act

1.1 Supervised Learning

- Goal:
  - Given data X in feature space and the labels Y
  - Learn to predict Y from X
- Labels could be discrete or continuous
  - Discrete labels: classification
  - Continuous labels: regression
- Classification:
  - Given a feature space (e.g., words in a document)
  - Predict a label space (e.g., topic of document)
- Regression:
  - Given a continuous feature space (e.g., market infromation up to time t)
  - Predict a label space (e.g., shapre price "$24.50")

1.2 Unsupervised Learning

- Goal:
  - Given data X without any labels
  - Learn the structures of the data
- "Learning without teacher"
- Clustering:
  - "Grouping into similar examples"
  - TODO: Image
- Lecture cut early, possibly add skipped here.

1.3 Feature Extraction

- Represent data in terms of vectors
  - Featurs are statistics or attributes that describe the data
  - Practitioners tend to tern this data into a feature space
  - e.g., For housing data useful features may be: number of rooms, square footage, etc.
- You can also consider domain knowledge, namely, use knowledge of how the task work to inject features into the domain
  - e.g., for OCR, aspect ratio of tight bounding boxes, existence of of vertical/horizontal strokes

2 Linear Regression

2.1 Notation

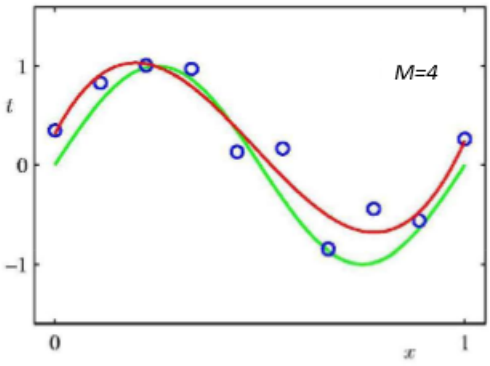
- $x \in \mathbb{R}^D$: data
- $\phi(x) \in \mathbb{R}^M$: features for $\vec{x}$
- $t \in \mathbb{R}$: continuous-valued labels
- $\vec{x}^{(n)} \equiv \vec{x}_n$: n-th training example
- $\vec{t}^{(n)} \equiv \vec{t}_n$: n-th targe value

2.2 1D Inputs

- In the 1D case ($x \in \mathbb{R}^1$)
- Given a set of observations $x^{(1)}, \ldots, x^{(N)}$ and corresponding targe values $t^{(1)}, \ldots, t^{(N)}$
- We want to learn a function $y(\vec{x}, W) \approx t$ to predict future values

$$y(\vec{x}, \vec{w}) = \sum_{j=0}^{M-1} \vec{w}_j \phi_j(\vec{x}) = \vec{w}^T \phi(x)$$

– e.g., (green = solution, red = 3-rd polynomial approximation)



– For simplicity, we add a *bias function*: $\phi_0(\vec{x}) = 1$

$$\vec{\phi} = 1, x, x^2, x^3, \ldots$$

2.3 Basis Function

– Function to construct features from raw data.

– e.g.,

– Polynomial: $\phi_j(x) = x^j$

– Gaussian: $\phi_j(x) = exp(-\frac{(x-\mu_j)^2}{2s^2})$

– Sigmoid: $\phi_j(x) = \sigma(\frac{x-\mu_j}{s})$

2.4 Objective Function

– We will use of sum-of-square errors:

$$E(w) = \frac{1}{2}\sum_{n=1}^{N}(y(x^{(n)}, w) - t^{(n)})^2$$

2.5 Batch Gradient Descent

– Given data (x, y) initial w, repeat until convergence:

$$\vec{w} = \vec{w} - \eta\nabla_{\vec{w}}E(\vec{w})$$

$$\nabla_{\vec{w}}E(w) = \sum_{n=1}^{N}\left(\sum_{k=0}^{M-1} w_k\phi_k(\vec{x}^{(n)}) - t^{(n)}\right)\phi(\vec{x}^{(n)}) = \sum_{n=1}^{N}\left(\vec{w}^T\phi(\vec{x}^{(n)}) - \vec{t}^{(n)}\right)\phi(x^{(n)})$$

2.6 Overfitting

- An implicit way to tell is when the coeffecients become unreasonably large
- Solutions:
  - Reduce order
  - Add more data point
  - Reselect features, some may be harming you
- If you have a small number of data points, then you should use low order polynomial (small number of features)
- As you obtain more data points, you can gradually increase the order of the polynomial (more features)
- Controlling model complexity: **regularization**

3 Linear Regression Pt. 2

3.1 Probability

- **Experiment**: procedure that yields an outcome
- **Sample space**: set of all possible outcomes in the experiment, denoted as Ω (or S)
- **Event**: subset of the sample space Ω (i.e., an event is a set consisting of individual outcomes)
- **Probability measure**: function from events to probability levels
- **Probability space**: $(\Omega, \mathcal{F}, P)$

Axioms of Probability

- $P(A) \geq 0, \forall A \in \mathcal{F}$
- $P(\Omega) = 1$
- If $A_1, A_2, \ldots$ are disjoint events, then:

$$P(\Cup_i A_i) = \sum_i P(A_i)$$

Set probabilities

- $A \subset B \rightarrow P(A) \leq P(B)$
- $P(A \cap B) \leq min(P(A), P(B))$
- $P(A \cup B) \leq P(A) + P(B)$
- $P(\Omega$
  $A) = 1 - P(A)$

– If $A_1, \ldots, A_k$ are a set of disjoint events such that $\biguplus_{i=1}^{k} A_i = \Omega$, then:

$$\sum_{i=1}^{k} P(A_k) = 1$$

Conditional Probability

– For events $A, B \in \mathcal{F}$ with $P(B) > 0$, we may write the conditional probability of A given B:

$$TODO$$

3.2 Bayes' Rule

– Using chain rule we may see **Bayes' rule**:

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}$$

– Often written:

$$P(B_i|A) = \frac{P(A|B_i)P(B_i)}{\sum_i P(A|B_i)P(B_i)}$$

– Where B_i are a partition of Ω (note the bottom is just the law of total probability)

–

3.3 Likelihood Functions

– Bayes' allows us to compute the posterior of w given data D:

$$p(w|D) = \frac{p(D|w)p(w)}{p(D)}$$

$$p(D) = \sum_w p(D|w)p(w)$$

– The likelihood unction $p(D|w)$ is evaluated for observbed data D as a function of w.

– Namely,

$$posterior \propto likelihood \times prior$$

$$p(\vec{w}|D) \propto p(D|\vec{w})p(\vec{w})$$

– We do this because we typically have a model ($p(\vec{w})$), and then we can improve our likelihood of prediction by observing data ($p(D|\vec{w})$

3.4 Maximum Likelihood

- Maximum likelihood:
  - choose parameter setting w that maximizes likelihood function $p(D|w)$
  - Choose the value of w that maximizes the probability of observed data
  - The negative log of the likelihood is called the negative log-likelihood (e.g., a loss function to minimize)
  - Maximizing likelihood is equivalent to minimizing the loss
- MAP (maximum a posteriori) estimation
  - Equivalent to mazimizing $p(w|D) \propto p(D|w)p(w)$

3.5 The Gaussian Distribution

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^(1/2)} e^{-\frac{1}{2\sigma^2}(x-\mu)^2}$$

TODO INSERT FUNCTION AND GRAPH

- Expected value: $E(x) = \mu = \int p(x)xdx$
- Variance: $Var(x) = E(x^2) - E(x)^2 = \sigma^2$

3.6 Maximum Likelihood w

- Assume a stochastic model:
$$t = y(x, w) + \epsilon \text{ where } \epsilon \ \mathcal{N}(0, \beta^{-1})$$
- $\beta^{-1} = \sigma^2$
- This gives a likelihood function:
$$p(t|x, w, \beta) = \mathcal{N}(t|y(x, w), \beta^{-1})$$
- With inputs $X = \{x^{(1)}, \ldots, x^{(N}\}$ and targe values $t = \{t^{(1)}, \ldots, t^{(N)}\}$, the data likelihood is:
$$p(t|X, w, \beta) = \Pi_{n=1}^{N} \mathcal{N}(t^{(n)}|w^T\phi(x^{(n)}), \beta^{-1})$$
- Log likelihood is:
$$TODO - 16$$
- where:
$$TODO - 16$$

3.7 Log Likelihood

– The log likelihood is:

$$\ln p(\vec{t}|\vec{w},\beta) = \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi) - \beta E_D(\vec{w})$$

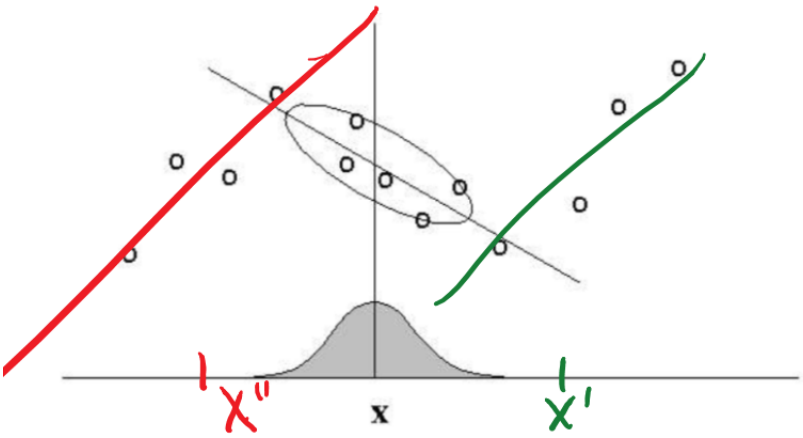
– Derivation:

$$\begin{aligned}
\log(P(t^{(1)},t^{(2)},\ldots,t^{(n)}|x^{(1)},x^{(2)},\ldots,x^{(n)},w)) &= \log P(\vec{t}|\vec{x},\vec{w}) \\
&= \log\left[\Pi_{i=1}^{N} P(t^{(i)}|x^{(i)},w)\right] \\
&= \log\left[\Pi_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{|t^{(i)}-w^T\phi(x^{(i)})|^2}{2\sigma^2}\right)\right] \\
&= \log\left[\Pi_{i=1}^{N} \frac{\sqrt{\beta}}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}\beta|t^{(i)}-w^T\phi(x^{(i)})|^2\right)\right] \\
&= \sum_{i=1}^{N}\log\left(\frac{\sqrt{\beta}}{\sqrt{2\pi}}\right) + \log\left(\exp\left(-\frac{1}{2}\beta|t^{(i)}-w^T\phi(x^{(i)})|^2\right)\right) \\
&= \sum_{i=1}^{N}\log\left(\frac{\sqrt{\beta}}{\sqrt{2\pi}}\right) + -\frac{1}{2}\beta|t^{(i)}-w^T\phi(x^{(i)})|^2
\end{aligned}$$

– Maximizing the likelihood is summarized as:

$$0 = (\phi^T t)^T - w^T(\phi^T\phi)$$

3.8 Locally weighted linear regression



– Main idea: when predicting $f(x)$, give high weights for "neighbors" of x

– In locally weighted regression, points are weighted by prioximity to the current x in question using a kernel. A regression is computed using the weighted points

– Use simple features, like when you're not sure what a good feature function would be

– Weighted based on proximity of neighbors

– Do linear regression at each position based on neighbors

- Approximation of local neighborhood
- Final curve is made by dragging x across and rerunning regression each time

Definition 3.1 (Locally-weighted linear regression)**.** 1. Fit $\vec{w}$ to minimize $\sum_i r^{(i)} \left(t^{(i)} - \vec{w}^T \phi(x^{(i)})\right)^2$

2. Predict: $\vec{w}^T \phi(x^{(i)})$

- Remarks:
  - Standard choice: $r^{(i)} = \exp\left(-\frac{||x^{(i)}-x||^2}{2\tau^2}\right)$, where $\tau =$"kernel width"
  - $r^{(i)}$ depends on x (query point), and you solve linear regression for each query point x
  - The problem can be formualted as a modified version of least squares
  - CHoice of τ can cause overfitting/underfitting

4 Classification

4.1 Classification problem

- The task of classification:
  - Given an input vector $\vec{x}$, assign it to one of K distinct classes C_k where $k = 1, \ldots, K$
- Representing the assignment:
  - For $K = 2$
    - $t = 1$ means that $\vec{x}$ is in C_1
    - $t = 0$ means that $\vec{x}$ is in C_2
  - For $K > 2$
    - Use $1 - of - K$ (one-hot) encoding
    - e.g., $\vec{t} = (0, 1, 0, 0, 0)^T$ means that $\vec{x}$ is in C_2
- Training: train a classifier $h(x)$ from training data:

$$\{(x^{(i)}), t^{(i)}\}_{i\in[1,N]}$$

- Testing evualation:
  - Data:

$$\{(x_{test}^{(i)}, t_{test}^{(j)})\}_{i\in[1,N],j\in[1,m]}$$

  - The learning algorithm produces predictions:

$$h(x_{test}^{(1)}), \ldots, h(x_{test}^{(m)})$$

  - 0-1 loss:

$$\text{Classification error:} A = \frac{1}{m}\sum_{j=1}^{m} h(x_{test}^{(j)}) \neq t_{test}^{(j)}$$

4.2 Strategies to classification

- Nearest neighbor classification
  - Given query data $\vec{x}$, find the closest training points and do majority vote
- Discriminant functions
  - Learn a function $y(\vec{x})$ that maps $\vec{x}$ onto some C_j
- Learn the distributions of $p(C_k|\vec{x})$
  - **Discriminative models**: directly model $p(C_k|\vec{x})$ and learn parameters from the training set
  - **Generative models**: Learn class densities $p(x|C_k)$ and priors $p(C_k)$

4.3 Probabilistic discriminative models

- Model decision boundary as a function of input $\vec{x}$
- Learn $P(C_k|\vec{x})$ over data (e.g., maximum likelihood)
- Directly predict class labels from inputs

4.4 Logistic Regression

- Models that class posterior using a sigmoid applied to a linear function of the feature vector:

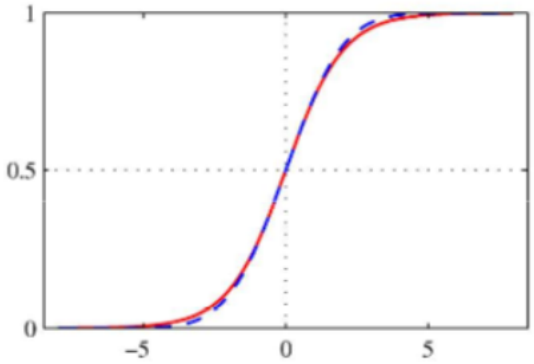
$$p(C_1|\phi) = y(\phi) = \sigma(\vec{w}^T\phi(\vec{x}))$$

- We can solve the parameter $\vec{w}$ by maximizing the likelihood of the training data

Sigmoid and Logit functions

- The **logistic sigmoid** function is:

$$\sigma(a) = \frac{1}{1+\exp(-a)}$$



- Its inverse is the **logit** function (aka log odd ratio):

$$a = \ln\left(\frac{\sigma}{1-\sigma}\right)$$

4.5 Likelihood function

– Depending on the label y, the likelihood of x is defined as:

$$P(t=1|\vec{x},\vec{w}) = \sigma(\vec{w}^T\phi(x))$$

$$P(t=0|\vec{x},\vec{w}) = 1-\sigma(\vec{w}^T\phi(x))$$

– Therefore:

$$P(t|x,w) = \sigma(w^T\phi(x))^t(1-\sigma(w^T-\sigma(w^T\phi(x)))^{1-t}$$

– The complete likelihood function for the data set:

$$p(t|w,x) = \Pi_{n=1}^{N}(y^{(n)})^{(t)}(1-y^{(n)})^{1-t^{(n)}}$$

$$\text{where } y^{(n)} = p(C_1|\phi(\vec{x}^{(n)}) = \sigma(w^T\phi(x^{(n)})$$

4.6 Derivation - Gradient

$$\log P(t|w) = \sum_{n=1}^{N} t^{(n)}\log y^{(n)} + (1-t^{(n)})\log(1-y^{(n)})$$

$$\sigma^{(n)} = \sigma(w^T\phi(x^{(n)})) = y^{(n)}$$

$$\nabla_w \log P(t|w)$$

$$\sum_{n=1}^{N}\nabla_w\left(t^{(n)}\log\sigma(w^T\phi(x^{(n)})) + (1-t^{(n)})\log(1-\sigma(w^T\phi(x^{(n)})))\right)$$

$$\sum_{n=1}^{N}\left(t^{(n)}\frac{\sigma^{(n)}(1-\sigma^{(n)})}{\sigma^{(n)}} - (1-t^{(n)}\frac{\sigma^{(n)}(1-\sigma^{(n)})}{1-\sigma^{(n)}}\right)\nabla_w(w^T\phi(x^{(n)}))$$

$$\frac{\partial}{\partial s}\sigma(s) = \frac{\partial}{\partial s}\left(\frac{1}{1+\exp(-s)}\right) = \sigma(s)(1-\sigma(s))$$

$$\sum_{n=1}^{N}\left(t^{(n)}(1-\sigma^{(n)})\nabla_w(w^T\phi(x^{(n)})) - (1-t^{(n)})\sigma^{(n)}\nabla_w(w^T\phi(x^{(n)}))\right)$$

$$\sum_{n=1}^{n}(t^{(n)}-\sigma^{(n)})\phi(x^{(n)})$$

4.7 Least-squares closed-form

$$w_{ML} = (\phi^T\phi)^{-1}\phi^T t$$

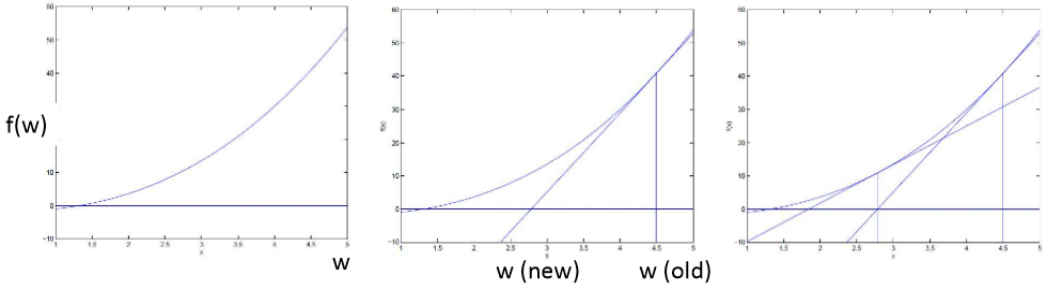
– WIth an $N \times N$ diagonal weight matrix R:

$$w_{WLS} = (\phi^T R\phi)^{-1}\phi^T Rt$$

4.8 Newton's Method

- Goal: Minimizing a general function $l(w)$ (one dimensional case)
  - Approach: solve for $f(w) = \frac{\partial l(w)}{\partial w} = 0$
  - How to solve this problem?
- Newton's method (aka Newton-Raphson method):
  - Repeat until convergence:

$$w := w - \frac{f(w)}{f'(w)}$$



- We iteratively solve by looking at the where the tangent line intercepts the x-axis and move on until the tangent line has a slope of zero.
- f' represents the curvative, so if the curvature is large, it forces a small step size so we don't overshoot.

4.9 Multivariate case

$$w := w - H^{-1}\nabla_w l$$

- Where H is the Hessian matrix:

$$H_{ij}(w) = \frac{\partial^2 l(w)}{\partial w_i \partial w_j}$$

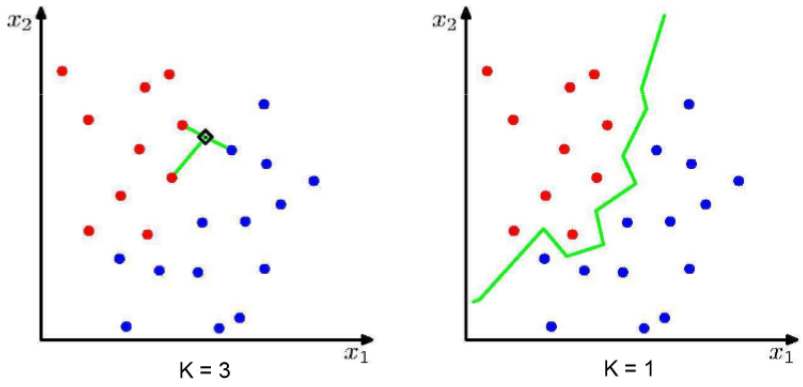
4.10 K-Nearest Neighbor Classification

- Training Method:
  - Save the training examples (no sophisticated learning)
- At prediction (testing) time:
  - Given a test (query) example x, find the k training examples that are closest to the x

$$kNN(x) = \{(x^{(1)}, t^{(1)}), \ldots, (x'^{(k)}, t'^{(k)})\}$$

  - Predict the most frequent class amount t_i's from ("majority vote"):

$$y = \max_t \sum_{(x',t') \in kNN(x)} l(t' = t)$$



- K acts as a smother (bias-variance trade-off)
- Classification performance gnerally improves as N (training set size) increases
- For $N \to \infty$ the error rate of hte 1-nearest-neighbor classifier is enver mroe than twice the optimal error

4.11 Factors (hyperparameters affectin kNN)

- Distance metric $D(x, x')$
  - How to define distance between two examples between x and x'?
- The value of K
  - K determiens how much we "smooth out" the prediction

4.12 kNN: Classification vs Regression

- For classification: we take "majority vote" from the targe labels
- For regression: we take "average" from the target labels

5 Classification 2: Softmax, Probabilistic generative models

5.1 Softmax Regression

- For multiclass case, we can use softmax regression
- Softmax regression can be viewed as a generalization of logistic regression
- Recall that, logistic regression (binary classification) models class conditional probability as:

$$p(t = 1|x; w) = \frac{\exp(w^T \phi(x))}{1 + \exp(w^T \phi(x))}$$

$$p(t = 0|x; w) = \frac{1}{1 + \exp(w^T \phi(x))}$$

  - Note that tehse probabilities sum to 1

– For multiclass classification (with K classes), we use the following model:

$$p(t = k : x; w) = \frac{\exp(w_k^T \phi(x))}{1 + \sum_{j=1}^{K-1} \exp(w_j^T \phi(x))}$$

$$p(t = K : x; w) = \frac{1}{1 + \sum_{j=1}^{K-1} \exp(w_j^T \phi(x))}$$

  – Note that these probabilities sum to 1
  – This is euqivalent when seting $w_k = 0$

Log-likelihood and learning

– Defining $w_k = 0$, we can write as:

$$p(t = k : x; w) = \frac{\exp(w_j^T \phi(x))}{\sum_{j=1}^{K-1} \exp(w_j^T \phi(x))}$$

$$p(t|x; w) = \prod_{k=1}^{K} \left[\frac{\exp(w_j^T \phi(x))}{\sum_{j=1}^{K-1} \exp(w_j^T \phi(x))} \right]^{I(t=k)}$$

– Log-likelihood:

$$\log p(D|w) = \sum_i \log p(t^{(i)}|x^{(i)}, w)$$

$$= \sum_i \log \prod_{k=1}^{M} [\,]^{I(t^{(i)}=k)}$$

5.2 Probabilistic generative models

– Goal: Learn the distributions $p(C_k|\vec{x})$

  – Discriminative models: Directly model $p(C_k|\vec{x})$ and learn parameters from the training set
    * Logistic regression
    * Softmax regression
  – Generative models: Learn class densities $p(\vec{x}|C_k)$ and priors $p(C_k)$
    * Gaussian discriminant analysis
    * Naive bayes

Bayes' Theorem

– Bayes' theorem reduces the classificaqtion problem $p(C_k|x)$ to estimating the distribution of the data

– Densisty estimation problems are easy to learn from labeled training data: $p(C_k), p(x|C_k)$

– Maximum likelihood parameter estimation

- For two classes, Bayes' theorem says:

$$p(C_1|x) = fracp(x|C_1)p(C_1)p(x|C_1)p(C_1) + p(x|C_2)p(C_2)$$

- Use **log odds**:

$$a = \ln \frac{p(C_1|x)}{p(C_2|x)} = \ln \frac{p(x|C_1)p(C_1)}{p(x|C_2)p(C_2)}$$

- Then we can define the posterior via the sigmoid:

$$p(C_1|x) = \sigma(a)$$

5.3 Comparing teh approaches: Discriminative vs. Generative

- The **generative** approach is typically model-based, and makes it possible to generate synthetic data from $p(x|C_k)$
  - By comparing the synthetic data and read data, we get a sense of how good the generative model is
- The **discriminative** approach will typically have fewer parameters to estimate and have less assumptions about data distribution.
  - Linear (e.g., logistic regression) versus quadratic (e.g., Gaussian discriminant analysis) if the dimension of the input
  - Less generative assumptions about the data (however, construction the features may need prior knowledge)

5.4 Gaussian Discriminant Analysis

- Prior distribution: $p(C_k)$ (constant)
- Likelihood: $p(x|C_k)$ (gaussian distribution)

$$p(x|C_k) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k)\right\}$$

  - Here Σ represents the covariance matrix.
  - $D = 1 \rightarrow \Sigma = \sigma^2$
  - Spherical case ($\Sigma \propto \mathcal{I}$)
  - Diagonal covariance
  - Full covariance (non-diagonals can be non-zero expressing relationships)
- Classification: use Bayes' rule
- Basic GDA assums same covariance for all classes
  - The below shows class-sepcific density and decision boundary
  - Note lienar decision boundary TODO: PICTURE

Class-Conditional Densities

– Suppose we model $p(x|C_k)$ as Gaussians with the same covariance matrix.

$$p(x|C_k) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k)\right\}$$

– This gives us:

$$w = \Sigma^{-1}(\mu_1 - \mu_2)$$
$$w_0 = -\frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1 + \frac{1}{2}\mu_2^T\Sigma^{-1}\mu_2 + \ln\frac{p(C_1)}{p(C_2)}$$

– Derivation:

$$\begin{aligned}
P(x, C_1) &= P(x|C_1)P(C_1) \\
&= \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\right) P(C_1) \\
P(x, C_2) &= P(x|C_2)P(C_2) \\
&= \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_2)^T\Sigma^{-1}(x-\mu_2)\right) P(C_2) \\
\text{log-odds: } a &= \ln(\frac{P(x, C_1)}{P(x, C_2)}) = \ln(\frac{P(C_1)p(x|C_1)}{P(C_2)P(X|C_2)}) \\
a &= \ln(\frac{P(x, C_1)}{P(x, C_2)}) \\
&= \frac{\frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\right) P(C_1)P(C_1)}{\frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_2)^T\Sigma^{-1}(x-\mu_2)\right) P(C_2)P(C_2)} \\
&= (-\frac{1}{2}(x-\mu_1)\Sigma^{-1}(x-\mu_1) + \ln(P(C_1))) - (-\frac{1}{2}(x-\mu_2)\Sigma^{-1}(x-\mu_2) + \ln(P(C_2))) \\
&= (\mu_1-\mu_2)^T\Sigma^{-1}x - \frac{1}{2}\mu_1\Sigma^{-1}\mu_1 + \frac{1}{2}\mu_2\Sigma^{-1}\mu_2 + \log\frac{P(C_1)}{P(C_2)} \\
&= (\Sigma^{-1}(\mu_1-\mu_2))^T x + w_0
\end{aligned}$$

Linear Decision Boundaries

– At decision boundary, we have $p(C_1|x) = p(C_2|x)$

– With the same covariance matrices, the boundary is linear

– It gives you a linear relgation due to cancellation of 2nd-order terms

Learning parameters via maximum likelihood

– Given the training data $\{(x^{(1)}, t^{(1)}), \ldots, (x^{(N)}, t^{(N)})\}$ and a generative model ("shared covariance")

$$p(t) = \phi^t(1-\phi)^{1-t}$$
$$p(x|t=0) = \frac{1}{\sqrt{2\pi}|\Sigma|^{1/2}} \exp(-\frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0))$$

$$p(x|t=1) = \frac{1}{\sqrt{2\pi}|\Sigma|^{1/2}} \exp(-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1))$$

– MLE:

$$\phi = \frac{1}{N}\sum_{i=1}^{N} 1\{t^{(1)}=1\}$$

$$\mu_0 = \frac{\sum_{i=1}^{N} 1\{t^{(i)}=0\}x^{(i)}}{\sum_{i=1}^{N} 1\{t^{(i)}=0\}}$$

$$\mu_1 = \frac{\sum_{i=1}^{N} 1\{t^{(i)}=1\}x^{(i)}}{\sum_{i=1}^{N} 1\{t^{(i)}=1\}}$$

A Linear Algebra Review

Definition A.1 (Gradient (vector))**.**

$$f(x) : \mathbb{R}^{n\times n} \to \mathbb{R}$$

$$\nabla_x f(x) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix}$$

Definition A.2 (Gradient (matrix))**.**

$$f(x) : \mathbb{R}^{n\times n} \to \mathbb{R}$$

$$\nabla_A f(A) = \begin{bmatrix} \frac{\partial f(A)}{\partial A_{11}} & \cdots & \frac{\partial f(A)}{\partial A_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f(A)}{\partial A_{m1}} & \cdots & \frac{\partial f(A)}{\partial A_{mn}} \end{bmatrix}$$

Definition A.3 (Hessian)**.**

$$f : \mathbb{R}^n \to \mathbb{R}$$

$$\nabla_x^2 f(x) = \left[\frac{\partial^2 f(x)}{\partial x_r \partial x_c}\right]$$

A.1 Matrix Calculus Examples

– e.g.,

$$\vec{a}^T C\vec{b}; a \in \mathbb{R}^m, C \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^n$$

$$\sum_{i=1}^{m} a_i (Cb)^i$$

$$\sum_{i=1}^{m} a_i \sum_{j=1}^{n} C_{ij} b_j$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} C_{ij} a_i b_j$$

– e.g.,

$$\frac{\partial f(x)^T A g(x)}{\partial x}; f : \mathbb{R}^k \to \mathbb{R}^m, g : \mathbb{R}^k \to \mathbb{R}^n, A : m \times n$$

$$\begin{aligned}
\frac{\partial f(x)^T A g(x)}{\partial x_k} &= \frac{\partial}{\partial x_k} \left(\sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij} f(x)_i g(x)_j \right) \\
&= \sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij} \frac{\partial}{\partial x_k} f(x)_i g(x)_j \\
&= \sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij} \left(g(x)_j \frac{\partial f(x)_i}{\partial x_k} + f(x)_i \frac{\partial g(x)_j}{\partial x_k} \right) \\
&= \sum_{i=1}^{m} \sum_{j=1}^{n} \left(A_{ij} g(x)_j \frac{\partial f(x)_i}{\partial x_k} \right) + \sum_{i=1}^{m} \sum_{j=1}^{n} \left(A_{ij} f(x)_i \frac{\partial g(x)_j}{\partial x_k} \right)
\end{aligned}$$