Math Review

X and **Y** are **independent** iff $\mathbb{P}(\mathbf{X},\mathbf{Y}) = \mathbb{P}(\mathbf{X})\mathbb{P}(\mathbf{Y})$
X and **Y** are **uncorrelated** iff $\mathbb{E}(\mathbf{X},\mathbf{Y}) = \mathbb{E}(\mathbf{X})\mathbb{E}(\mathbf{Y})$
Expected value of $g(X)$: $E[g(X)] = \int_{-\infty}^{\infty} g(x) f(x) dx$
Variance $\sigma^2 = E[(X-\mu)^2] = E[X^2] - \mu^2$
Determinant of matrix is product of its eigenvalues.

$f(\vec{x}) = \mathbf{A}\vec{x} + \vec{x}^\top \mathbf{A} + \vec{x}^\top \vec{x} + \vec{x}^\top \mathbf{A}\vec{x} \Rightarrow \frac{df(\vec{x})}{d\vec{x}} = \mathbf{A}^\top + \mathbf{A} + 2\vec{x} + \mathbf{A}\vec{x} + \mathbf{A}^\top \vec{x}$

$$\nabla_{\vec{x}}(\vec{y}\cdot\vec{z}) = (\nabla_{\vec{x}})\vec{z} + (\nabla_{\vec{x}})\vec{y} \qquad \nabla_{\vec{x}} f(\vec{y}) = (\nabla_{\vec{x}}\vec{y})(\nabla_{\vec{y}} f(\vec{y}))$$

$$\nabla_w w^T A w = (A + A^T) w \qquad \mathbf{H}_{i,j} = \frac{\partial^2 f}{\partial x_i \partial x_j}$$

Perceptron (Lecture 2)

$f(\vec{x}) = \vec{w}\cdot\vec{x} + \alpha = \sum_{i=1}^{d} w_i x_i + \alpha$,
Goal: find w s.t all constraints $y_i X_i \cdot w \geq 0$. Define a risk function and optimize it, where the loss is defined as $L(z, y_i) = -y_i z$ if $y_i z < 0$, else 0. Therefore risk $R(w) = \sum_{i\in V} -y_i X_i \cdot w$

Decision boundary, a **hyperplane** in $\mathbb{R}^d$:
$H = \{\vec{x} \in \mathbb{R}^d : f(\vec{x}) = 0\} = \{\vec{x} \in \mathbb{R}^d : \vec{w}\cdot\vec{x} + \alpha = 0\}$

$\vec{w}$ is the **normal** of the hyperplane,
α is the **offset** of the hyperplane from origin,
$\frac{f(\vec{x})}{\|\vec{w}\|}$ is the **signed distance** from the $\vec{x}$ to hyperplane $\mathcal{H}$.

Perceptron algorithm,
Input: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n) \in \mathbb{R}^d \times \{\pm 1\}$
while some $y_i \neq \text{sign}(\vec{w}\cdot\mathbf{x}_i)$
    pick some misclassified $(\mathbf{x}_i, y_i)$
    $\vec{w} \leftarrow \vec{w} + y_i \mathbf{x}_i$

Given a **linearly separable data**, perceptron algorithm will take no more than $\frac{R^2}{\gamma^2}$ updates to **converge**, where $R = \max_i \|\mathbf{x}_i\|$ is the radius of the data,
$\gamma = \min_i \frac{y_i(\vec{w}\cdot\mathbf{x}_i)}{\|\vec{w}\|}$ is the margin.
Also, $\frac{\vec{w}\cdot\vec{x}}{\|\vec{w}\|}$ is the signed distance from H to $\vec{x}$ in the direction $\vec{w}$.

Gradient descent view of perceptron, minimize margin cost function
$J(\vec{w}) = \sum_i (-y_i(\vec{w}\cdot\mathbf{x}_i))_+$ with $\vec{w} \leftarrow \vec{w} - \eta\nabla J(\vec{w})$

Support Vector Machine (Lecture 3, 4)

Hard margin SVM,
This method makes the margin as wide as possible. The signed distance from the hyperplane to X_i is $\frac{f(\mathbf{x}_i)}{\|w\|}$ Hence the margin is
$\min_i \frac{1}{\|w\|}|w\cdot X_i + \alpha| \geq \frac{1}{\|w\|} \implies \min_{\vec{w}} \|\vec{w}\|^2$, such that
$y_i \vec{w}\cdot\mathbf{x}_i \geq 1 (i = 1, \ldots, n)$
Soft margin SVM,
$\min_{\vec{w}} \|\vec{w}\|^2 + C\sum_{i=1}^{n} \xi_i$

Regularization and SVMs: Simulated data with many features $\phi(\vec{x})$; C controls trade-off between margin $1/\|\vec{w}\|$ and fit to data; Large C: focus on fit to data (small margin is ok). More overfitting. Small C: focus on large margin, less tendency to overfit. Overfitting increases with: less data, more features.

Decision Theory (Lecture 6)

Bayes Theorem: $\underbrace{\mathbb{P}(Y = C|X)}_{\text{Poster. Prob}} = \frac{\mathbb{P}(X|Y=C)\overbrace{\mathbb{P}(Y=C)}^{\text{Prior Prob.}}}{\mathbb{P}(X)}$ Assume $(\mathbf{X},\mathbf{Y})$ are chosen i.i.d according to some probability distribution on $\mathcal{X}\times\mathcal{Y}$. **Risk** is misclassification probability: $R(r) = \mathbb{E}(L(r(\mathbf{X}),\mathbf{Y})) = \mathbb{P}(r(\mathbf{X}) \neq \mathbf{Y}) =$
$\sum_{\vec{x}} \left[L(r(\vec{x}), 1)\mathbb{P}(Y = 1|x) + L(r(x), -1)\mathbb{P}(Y = -1|X = \vec{x})\right] \times \mathbb{P}(\vec{x})$

$= \mathbb{P}(Y = 1)\sum_x L(r(\vec{x}), 1)\mathbb{P}(\vec{x}|Y = 1) + \mathbb{P}(Y = -1)\sum_x L(r(\vec{x}), -1)\mathbb{P}(\vec{x}|Y = -1)$

Bayes Decision Rule is
$$r^*(x) = \begin{cases} 1, & \text{if } L(-1,1)\mathbb{P}(\mathbf{Y} = 1|x) > L(1,-1)\mathbb{P}(\mathbf{Y} = -1|x) \\ -1, & \text{otherwise.} \end{cases},$$
and the optimal risk (Bayes risk) $R^* = \inf_r R(r) = R(r^*)$

Generative and Discriminative Models (Lecture 6)

Discriminative models: $\mathbb{P}(\mathbf{X},\mathbf{Y}) = \mathbb{P}(\mathbf{X})\mathbb{P}(\mathbf{Y}|\mathbf{X})$.
Estimate $\mathbb{P}(\mathbf{Y}|\mathbf{X})$, then pretend our estimate $\hat{\mathbb{P}}(\mathbf{Y}|\mathbf{X})$ is the actual $\mathbb{P}(\mathbf{Y}|\mathbf{X})$ and plug in bayes rule expression.

Generative model: $\mathbb{P}(\mathbf{X},\mathbf{Y}) = \mathbb{P}(\mathbf{Y})\mathbb{P}(\mathbf{X}|\mathbf{Y})$.
Estimate $\mathbb{P}(\mathbf{Y})$ and $\mathbb{P}(\mathbf{X}|\mathbf{Y})$, then use bayes theorem to calculate $\mathbb{P}(\mathbf{Y}|\mathbf{X})$ and use discriminative model.

Gaussian class conditional densities $\mathbb{P}(\mathbf{X}|Y = +1)$,$\mathbb{P}(\mathbf{X}|Y = -1)$ (with the same variance), the posterior probability is **logistic**:
$\mathbb{P}(Y = +1|\vec{x}) = \frac{1}{1+\exp(-\vec{x}\cdot\vec{w}-\beta_0)}$,
$\vec{w} = \Sigma^{-1}(\mu_1 - \mu_0)$, $\beta_0 = \frac{\mu_0'\Sigma^{-1}\mu 0 - \mu_1\Sigma^{-1}\mu 1}{2} + \log\frac{\mathbb{P}(Y=1)}{\mathbb{P}(Y=0)}$

Multivariate Normal Distribution (Lecture 7)

$\vec{x} \in \mathbb{R}^d : p(x) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} e^{(-\frac{1}{2}(\vec{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\vec{x}-\boldsymbol{\mu}))}$

QDA: Class-conditional densities $X_C \sim \mathcal{N}(\vec{\mu}_C, \Sigma_C)$. Optimal decision rule $r^*(x)$ for 0–1 loss: Choose class **C** that maxes $\mathbb{P}(Y = C|X) \propto f_C(x)\pi_C$. Parameters estimated via MLE:
LDA: Assumes equal covariance matrices across classes ($\Sigma_C = \Sigma$), simplifying to linear decision surfaces.

$\boldsymbol{\Sigma} = \mathbb{E}(\mathbf{X}-\boldsymbol{\mu})(\mathbf{X}-\boldsymbol{\mu})'$
Symmetric: $\boldsymbol{\Sigma}_{i,j} = \boldsymbol{\Sigma}_{ji}$
Non-negative diagonal entries: $\boldsymbol{\Sigma} i, i \geq 0$
Positive semidefinite: $\forall \mathbf{v} \in \mathbb{R}^d, \mathbf{v}'\boldsymbol{\Sigma}\mathbf{v} \geq 0$

Given a d-dimensaional Gaussian $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$,
matrix $\mathbf{A} \in \mathbb{R}^{m\times d}$ and vector $\mathbf{b} \in \mathbb{R}^m$, define $\mathbf{Y} = \mathbf{AX} + \mathbf{b}$.
Then $\mathbf{Y} \sim \mathcal{N}(\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^\top)$

Given a d-dimensaional Gaussian $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$,
with $\boldsymbol{\Sigma}$ positive definite,
$\mathbf{Y} = \boldsymbol{\Sigma}^{-\frac{1}{2}}(\mathbf{X}-\boldsymbol{\mu}) \sim \mathcal{N}(\vec{0}, \mathbf{I},)$

MLE's

Maximum a posterior probability: the mode of the posterior. If uniform prior, MAP is MLE; if not uniform prior, MAP is Penalized MLE.

Prior: $\hat{\pi}_C = \mathbb{P}(Y = C) = \frac{N_C}{n}$
Mean: $\hat{\mu}_C = \mathbb{E}[\mathbf{X}|Y = C] = \frac{1}{N_C}\sum_{i:Y_i=C} X_i$
Covariance: $\hat{\Sigma}_C = \frac{1}{N_C}\sum_{i:Y_i=C}(X_i - \hat{\mu}_C)(X_i - \hat{\mu}_C)^\top$
Pooled Cov: $\hat{\Sigma} = \frac{1}{n}\sum_{C_k}\sum_{i:Y_i=C_k}(X_i - \hat{\mu}_{C_k})(X_i - \hat{\mu}_{C_k})^\top$

Discriminant Analysis (Lecture 7)

Discriminant Functions for LDA and QDA for class C, denoted $Q_C(\vec{x})$ is:
$$Q_C(\vec{x}) = \ln\left(\frac{f_{\vec{X}|Y=C}(\vec{x})\pi_C}{(2\pi)^{\frac{d}{2}}}\right) = -\frac{1}{2}(\vec{x}-\boldsymbol{\mu}_C)^T\boldsymbol{\Sigma}_C^{-1}(\vec{x}-\boldsymbol{\mu}_C) - \frac{1}{2}\ln|\boldsymbol{\Sigma}_C| + \ln\frac{\pi_C}{(2\pi)^{\frac{d}{2}}}$$
Here we have, the class-conditional density $f_{\vec{X}|Y=C}(\vec{x})$, prior probability π_C, the mean vector of class C, and covariance matrix $\boldsymbol{\Sigma}_C$ of class C.

Linear Decision Function between Classes C and D: Compares classes by:
$$Q_C(\vec{x}) - Q_D(\vec{x}) = (\boldsymbol{\mu}_C - \boldsymbol{\mu}_D)^T\boldsymbol{\Sigma}^{-1}\vec{x} - \frac{1}{2}(\boldsymbol{\mu}_C^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_C - \boldsymbol{\mu}_D^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_D) + \ln\frac{\pi_C}{\pi_D}.$$
This expression includes a linear term $(\boldsymbol{\mu}_C - \boldsymbol{\mu}_D)^T\boldsymbol{\Sigma}^{-1}\vec{x}$, which describes how the decision boundary is formed linearly in the feature space, and a constant term that adjusts the boundary based on the means of the classes and their prior probabilities.

Multi-class LDA: The optimal class for a given feature vector $\vec{x}$ is chosen by

Choose class C such that $Q_C(\vec{x})$ is maximized.

Effectively determining the most likely class based on the feature measurements and statistical properties of each class.

Linear Regression (Lecture 10)

Objective Function: The goal in linear regression is to minimize the sum of squared residuals (RSS), expressed as:
$$\text{RSS}(\vec{w}) = \sum_{i=1}^{n}(\vec{x}_i^\top\vec{w} + \alpha - y_i)^2 = \|\mathbf{X}\vec{w} - \vec{y}\|^2.$$
Where, $\mathbf{X} \in \mathbb{R}^{n\times(d+1)}$ is the **design matrix**, and $\vec{y}$ is the **response vector**.
Minimize via Calculus: $\nabla\text{RSS}(\vec{w}) = 2\mathbf{X}^\top\mathbf{X}\vec{w} - 2\mathbf{X}^\top\vec{y}$.
Setting this gradient to zero leads to the **normal equations**:
$$\mathbf{X}^\top\mathbf{X}\vec{w} = \mathbf{X}^\top\vec{y},$$
Solving for the optimal weights $\vec{w}^*$:
$$\vec{w}^* = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\vec{y},$$
$(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top$ is known as the *Moore-Penrose pseudo-inverse* $X^\dagger$ of **X**.
Projection Theorem also leads to the normal equations, by asserting that the solution $\vec{w}^*$ projects the residuals orthogonally onto the column space of **X**:
$$\mathbf{X}^\top(\vec{y} - \mathbf{X}\vec{w}) = 0 \implies \mathbf{X}^\top\mathbf{X}\vec{w} = \mathbf{X}^\top\vec{y},$$

Logistic Regression (Lecture 10, 11)

Fits "probabilities" to data and usually used for classification.
$\mathbb{P}(Y = 1|\vec{x}_i) = \frac{1}{1+\exp(-\vec{w}^\top\vec{x}_i)} = \sigma(\vec{w}^\top\vec{x}_i)$
Logistic Loss: $L(z, y) = -y\ln z - (1-y)\ln(1-z)$

Cost Function: $J(\vec{w}) = -\sum_{i=1}^{n} L\left(\sigma(\vec{w}^\top \vec{x}_i), y_i\right)$
Find $\vec{w}^* = \arg\min J(\vec{w})$, Cost function is convex, and solved by gradient descent. Converges to max margin classifier.

Let $\sigma_i = \sigma(\vec{w}^\top \vec{x}_i)$ and $\vec{\sigma} = \begin{bmatrix}\sigma_1 & \cdots & \sigma_n\end{bmatrix}^\top$

$\nabla_{\vec{w}} \sigma_i = \sigma_i(1-\sigma_i)\vec{x}_i$
$\nabla_{\vec{w}} J(\vec{w}) = \mathbf{X}^\top(\vec{\sigma} - \vec{y})$
$\nabla^2_{\vec{w}} J(\vec{w}) = \mathbf{X}^\top \operatorname{diag}(\vec{\sigma}(1-\vec{\sigma}))\mathbf{X}$

Gradient descent:
$\vec{w}^{(t+1)} = \vec{w}^{(t)} - \eta \nabla_{\vec{w}} J(\vec{w}^{(t)}) : O(nd)$ per step
Stochastic gradient descent:
$\vec{w}^{(t+1)} = \vec{w}^{(t)} - \eta \nabla L_i(\vec{w}^{(t)}) : O(d)$ per step
Newton's method:
$\vec{w}^{(t+1)} = \vec{w}^{(t)} - \left(\nabla^2 J(\vec{w}^{(t)})\right)^{-1} \nabla J(\vec{w}^{(t)})$

LDA vs. Logistic Regression

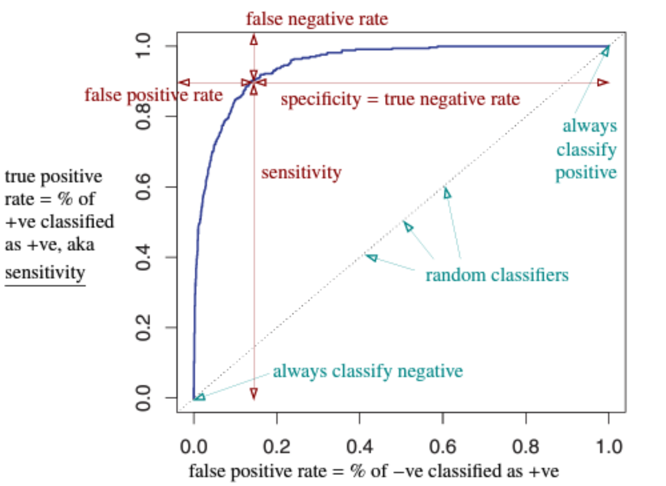
Advantages of LDA:

- For well-separated classes, LDA is stable; log. reg. is suprisingly unstable.
- > 2 classes? Easy & elegant; log. reg. needs modifying. (softmax regr.)
- slightly more accurate when classes nearly normal, especially if n is small.

Advantages of Log. Regression:

More emphasis on desc. boundary; always separates linearly separable pts.
More robust on some non-Gaussian distributions (e.g., dists w/ large skews.)
Naturally fits labels between 0 and 1.

ROC CURVES



Polynomial Regression (Lecture 11)

Replace each x_i with feature vector $\Phi(x_i)$ with all terms of degree p polynomial. e.g.,$\Phi(x_i) = \begin{bmatrix} x_{i1}^2 & x_{i1}x_{i2} & x_{i2}^2 & x_{i1} & x_{i2} & 1 \end{bmatrix}^\top$

Log. Reg. + quadratic features = same form of posteriors as QDA.

Statistical Justification For Regression (Lecture 12)

Typical model of reality: $\forall X_i,\ y_i = g(X_i) + \epsilon_i : \epsilon_i \sim D'$ where D' has mean 0.
Ideal approach: choose $h(X_i) = \mathbb{E}_y[Y|X = X_i] = g(X_i) + \mathbb{E}[\epsilon_i] = g(X_i)$

Least Squares Cost Function From MLE

Suppose $\epsilon_i \sim \mathcal{N}(0, \sigma^2) \implies y_i \sim \mathcal{N}\left(g(\vec{x}_i), \sigma^2\right)$. We're going to try to estimate $g(\vec{x})$, which is defined by some weights $\vec{w}$. The probability of y_i, given it's parameters for its distribution, is a pdf $f(y_i)$ & the log likelihood is:

$$\ell(g; \mathbf{X}, \vec{y}) = \ln\left(\prod_{i=1}^{n} f(y_i)\right) = \sum_{i=1}^{n} \ln\left(f(y_i)\right) = -\frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(y_i - g(\vec{x}_i)\right)^2 - C$$

Maximumizing this is equivalent to minimizing $\sum_{i=1}^{n}\left(y_i - g(\vec{x}_i)\right)^2 = \text{RSS}$

Logistic Loss from MLE

Consider $P[\vec{x}_i \in \text{class C}] = y_i$ as the actual probability, with $h(\vec{x}_i)$ as the predicted probability. For β hypothetical repetitions, where $y_i\beta$ samples belong to class C and $(1-y_i)\beta$ do not. The likelihood of h given the data is:

$$\mathscr{L}(h; \mathbf{X}, \vec{y}) = \prod_{i=1}^{n} h(\vec{x}_i)^{y_i\beta}(1-h(\vec{x}_i))^{(1-y_i)\beta}$$

$$\ell(h; \mathbf{X}, \vec{y}) = \beta\sum_{i=1}^{n}\left[y_i \ln\left(h(\vec{x}_i)\right) + (1-y_i)\ln\left(1-h(\vec{x}_i)\right)\right]$$

Maximizing the log likelihood $\ell(h)$ is equivalent to minimizing logistic losses.

The Bias-Variance Decomposition

There are 2 sources of error in a hypothesis h:

bias: error due to inability of hypothesis h to fit g perfectly.
variance: error due to fitting random noise in data, e.g., we fit linear g with a linear h, yet $h \neq g$.

Model: $X_i \sim D, \epsilon_i \sim D'$ and $y_i = g(X_i) + \epsilon_i$. Fit hypothesis h to $\mathbf{X}, \vec{y}$. Now h is a r.v., because of the random noise ϵ. Let z be an arbitrary pt and $\gamma = g(z) + \epsilon$
Now the risk fn when loss is the squared errror is:
$R(h) = \mathbb{E}[L(h(z), \gamma)] \longleftarrow$ exp over possible training set X, y and values of γ
$R(h) = \mathbb{E}[(h(z) - \gamma)^2]$
$R(h) = \mathbb{E}[h(z)^2] + \mathbb{E}[\gamma^2] - 2\mathbb{E}[\gamma h(z)]$
$R(h) = \text{Var}(h(z)) + \mathbb{E}[h(z)]^2 + \text{Var}(\gamma) + \mathbb{E}[\gamma]^2 - 2\mathbb{E}[\gamma]\mathbb{E}[h(z)]$
$R(h) = \left(\mathbb{E}[h(z)] - \mathbb{E}[\gamma]\right)^2 + \text{Var}(h(z)) + \text{Var}(\gamma)$
$R(h) = \underbrace{\left(\mathbb{E}[h(z)] - \mathbb{E}[\gamma]\right)^2}_{\text{bias}^2 \text{ of method}} + \underbrace{\text{Var}(h(z))}_{\text{variance of method}} + \underbrace{\text{Var}(\epsilon)}_{\text{irreducible error}}$

- **Underfitting**: Too much bias
- **Overfitting**: Too much variance
- Training Error reflects bias but not variance
- Test error reflects both.

Linear Regression Regularization (Lecture 13)

Trading off bias/variance: some increase in bias can give big decrease in var.

Ridge regression is like $L2$ regularization:
$\hat{\vec{w}} = \arg\min_{\vec{w}} \|\vec{y} - \mathbf{X}\vec{w}\|^2 + \lambda\|\vec{w}\|^2$, for some cost function J on classifier h
Minimized w/ Calculus
$\hat{\vec{w}}^{\text{ridge}} = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\vec{x}^\top\vec{y}$

Lasso regression is like $L1$ regularization:
$\hat{\vec{w}} = \arg\min_{\vec{w}} \|\vec{y} - \mathbf{X}\vec{w}\|^2 + \lambda\|w\|_1$
Minimized w/ Calculus (it's also a quadratic program)

While ridge regression leads to reduced, but rare non-zero values of the weights, Lasso regression forces some weights to be zero.

Bayesian analysis: Ridge regression is equivalent to a MAP estimate with a gaussian prior. Lasso regression is equivalent to a MAP estimate with a Laplace prior.

Misc

Centering X: This involves subtracting $\boldsymbol{\mu}^T$ from each row of $\mathbf{X}$. Symbolically, $\mathbf{X}$ transforms into $\bar{\mathbf{X}}$.
Decorrelating X: This process applies a rotation $\mathbf{Z} = \bar{\mathbf{X}}\mathbf{V}$, where $\text{Var}(\mathbf{R}) = \mathbf{V}\Lambda\mathbf{V}^T$. This step rotates the sample points to the eigenvector coordinate system.
Sphering: $\bar{\mathbf{X}}$: applying transform $\mathbf{W} = \bar{\mathbf{X}}\text{Var}(\mathbf{R})^{-\frac{1}{2}}$
whitening X: centering + sphering, $\mathbf{X} \to \mathbf{W}$

Decision Trees (Lecture 14)

Cuts x-space into rectangular cells, and works well with both categorical and quantitative features.

Two types of nodes:

1. **internal**: test feature values & branch accordingly
2. **leaf**: they specify the class $h(x)$

For classification the learning algorithm is a greedy, top-down learning heuristic. Let S be subset of sample pts indices
Learning algorithm

```
function GrowTree(S)
    if all y_i = C for all i ∈ S and some class C then
        return new leaf(C)
    else
        Choose best splitting feature j and splitting value β
        S_L = {i ∈ S : X_ij < β}
        S_R = {i ∈ S : X_ij ≥ β}
        return new node(j, β, GrowTree(S_L), GrowTree(S_R))
    end if
end function
```