KNN Find k examples $\left\{\mathbf{x}^{(i)}, t^{(i)}\right\}$ closest to the test instance $\mathbf{x}$ and then output majority $\arg\max_{t^z} \sum_{r=1}^{k} \delta(t^{(z)}, t^{(r)})$. Define $\delta(a,b) = 1$ if $a = b$, 0 otw. **Choice of k:** Rule is $k < \sqrt{n}$, small k may overfit, while large may under-fit. **Curse of Dim:** In high dimensions, "most" points are approximately the same distance. **Computation Cost:** 0 (minimal) at training/ no learning involved. Query time find N distances in D dimension $\mathcal{O}(ND)$ and $\mathcal{O}(N \log N)$ sorting time.

Entropy $H(X) = -\mathbb{E}_{X\sim p}\left[\log_2 p(X)\right] = -\sum_{x\in X} p(x)\log_2 p(x)$ **Multi-class:** $H(X,Y) = -\sum_{x\in X}\sum_{y\in Y} p(x,y)\log_2 p(x,y)$ **Properties:** H is non-negative, $H(Y|X) \leq H(Y)$, $X \perp Y \implies H(Y|X) = H(Y)$, $H(Y|Y) = 0$, and $H(X,Y) = H(X|Y) + H(Y) = H(Y|X) + H(X)$

Expected Conditional Entropy $H(Y|X) = \mathbb{E}_{X\sim p(x)}[H(Y|X)] = \sum_{x\in X} p(x) H(Y|X=x) = -\sum_{x\in X}\sum_{y\in Y} p(x,y)\log_2 p(y|x) = -\mathbb{E}_{(X,Y)\sim p(x,y)}\left[\log_2 p(Y|X)\right]$ **Information Gain** $IG(Y|X) = H(Y) - H(Y|X)$

Bias Variance Decomposition Using the square error loss $L(y,t) = \frac{1}{2}(y-t)^2$, **Bias ($\uparrow \implies$ under-fitting):** How close is our classifier to true target. **Variance ($\uparrow \implies$ overfitting):** How widely dispersed are out predictions as we generate new datasets

$$\begin{aligned}
\mathbb{E}_{\mathbf{x},\mathcal{D}}\left[\left(h_{\mathcal{D}}(\mathbf{x}) - t\right)^2\right] =& \mathbb{E}_{\mathbf{x},\mathcal{D}}\left[\left(h_{\mathcal{D}}(\mathbf{x}) - \mathbb{E}_{\mathcal{D}}\left[h_{\mathcal{D}}(\mathbf{x})\right] + \mathbb{E}_{\mathcal{D}}\left[h_{\mathcal{D}}(\mathbf{x})\right] - t\right)^2\right] \\
=& \mathbb{E}_{\mathbf{x},\mathcal{D}}\left[\left(h_{\mathcal{D}}(\mathbf{x}) - \mathbb{E}_{\mathcal{D}}\left[h_{\mathcal{D}}(\mathbf{x})\right]\right)^2 + \left(\mathbb{E}_{\mathcal{D}}\left[h_{\mathcal{D}}(\mathbf{x})\right] - t\right)^2 + 2\left(h_{\mathcal{D}}(\mathbf{x}) - \mathbb{E}_{\mathcal{D}}\left[h_{\mathcal{D}}(\mathbf{x})\right]\right)\left(\mathbb{E}_{\mathcal{D}}\left[h_{\mathcal{D}}(\mathbf{x})\right] - t\right)\right] \\
=& \underbrace{\mathbb{E}_{\mathbf{x},\mathcal{D}}\left[\left(h_{\mathcal{D}}(\mathbf{x}) - \mathbb{E}_{\mathcal{D}}\left[h_{\mathcal{D}}(\mathbf{x})\right]\right)^2\right]}_{\text{variance}} + \underbrace{\mathbb{E}_{\mathbf{x}}\left[\left(\mathbb{E}_{\mathcal{D}}\left[h_{\mathcal{D}}(\mathbf{x})\right] - t\right)^2\right]}_{\text{bias}}
\end{aligned}$$

Bagging with Generating Distribution Suppose we could sample m independent training sets $\{\mathcal{D}_i\}_{i=1}^m$ from $p_{dataset}$. Learn $h_i := h_{\mathcal{D}_i}$ and out final predictor is $h = 1/m\sum_{i=1}^m h_i$. **Bias Unchanged:** $\mathbb{E}_{\mathcal{D}_1,\ldots,\mathcal{D}_m \overset{iid}{\sim} p_{\text{dataset}}}[h(\mathbf{x})] = \frac{1}{m}\sum_{i=1}^m \mathbb{E}_{\mathcal{D}_i\sim p_{\text{dataset}}}[h_i(\mathbf{x})] = \mathbb{E}_{\mathcal{D}\sim p_{\text{dataset}}}[h_{\mathcal{D}}(\mathbf{x})]$ **Variance Reduced:** $\text{Var}_{\mathcal{D}_1,\ldots,\mathcal{D}_m}[h(\mathbf{x})] = \frac{1}{m^2}\sum_{i=1}^m \text{Var}[h_i(\mathbf{x})] = \frac{1}{m}\text{Var}[h_{\mathcal{D}}(\mathbf{x})]$

Bootstrap Aggregation Take a single dataset $\mathcal{D}$ with n sample and generate m new datasets, each by sampling n training examples from $\mathcal{D}$, with replacement. We then the average the predictions. We have the reduction in variance to be $\text{Var}\left(\frac{1}{m}\sum_{i=1}^m h_i(\mathbf{x})\right) = \frac{1}{m}(1-\rho)\sigma^2 + \rho\sigma^2$

Random Forest Upon bootstrap aggregation, for each bag we choose a random set of features to make the trees grow on (decorrelates predictions, lower ρ).

Bayes Optimality $\mathbb{E}_{\mathbf{x},\mathcal{D},t|\mathbf{x}}\left[\left(h_{\mathcal{D}}(\mathbf{x}) - t\right)^2\right] = \underbrace{\mathbb{E}_{\mathbf{x}}\left[\left(\mathbb{E}_{\mathcal{D}}\left[h_{\mathcal{D}}(\mathbf{x})\right] - y_*(\mathbf{x})\right)^2\right]}_{\text{bias}} + \underbrace{\mathbb{E}_{\mathbf{x},\mathcal{D}}\left[\left(h_{\mathcal{D}}(\mathbf{x}) - \mathbb{E}_{\mathcal{D}}\left[h_{\mathcal{D}}(\mathbf{x})\right]\right)^2\right]}_{\text{variance}} + \underbrace{\mathbb{E}_{\mathbf{x}}[\text{Var}[t|\mathbf{x}]]}_{\text{Bayes}}$

Feature Mapping Some time we want fit a polynomial curve, we can do this using a feature map $y = \mathbf{w}^\top\boldsymbol{\psi}(x)$ where $\boldsymbol{\psi}(x) = \left[1, x, x^2, \ldots\right]^\top$. In general the feature map could be anything.

Ridge Regression $\mathbf{w}_\lambda^{Ridge} = \underset{\mathbf{w}}{\text{argmin}}\mathcal{J}_{\text{reg}}(\mathbf{w}) = \underset{\mathbf{w}}{\text{argmin}}\frac{1}{2}\|\mathbf{X}\mathbf{w} - \mathbf{t}\|_2^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2 = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{t}$ When $\lambda = 0$ this is just OLS.

Gradient Descent Consider the some cost function $\mathcal{J}$ and we want to optimize it.

- **GD:** $\mathbf{w} \leftarrow \mathbf{w} - \alpha\frac{\partial\mathcal{J}}{\partial\mathbf{w}}$; **GD w/ Reg** $\mathbf{w} \leftarrow \mathbf{w} - \alpha\left(\frac{\partial\mathcal{J}}{\partial\mathbf{w}} + \lambda\frac{\partial\mathcal{R}}{\partial\mathbf{w}}\right) = (1-\alpha\lambda)\mathbf{w} - \alpha\frac{\partial\mathcal{J}}{\partial\mathbf{w}}$
- **mSGD:** Choose mini batch $\mathcal{M} \subset \{1, ..., N\}$ and update $\mathbf{w} \leftarrow \mathbf{w} - \frac{\alpha}{|\mathcal{M}|}\sum_{i=1}^{|\mathcal{M}|}\frac{\partial\mathcal{L}^{(i)}}{\partial\mathbf{w}}$ **Reasonable size** would be $|\mathcal{M}| \approx 100$
- **SGD:** Choose i at uniform; $\mathbf{w} \leftarrow \mathbf{w} - \alpha\frac{\partial\mathcal{L}^{(i)}}{\partial\mathbf{w}}$; **Pro//Cons:** Progress w/o seeing all data//High Variance & Not efficiently vectorized

Cross Entropy Loss $\mathcal{L}_{CE} = -t\log y - (1-t)\log(1-y)$ **Logistic CE** $\mathcal{L}_{LCE}(z,t) = \mathcal{L}_{CE}(\sigma(z),t) = t\log(1+e^{-z}) + (1-t)\log(1+e^z)$

Multi-class Classification

- **Softmax Function** Natural generalization of logistic func: $y_k = \text{softmax}\left(z_1, \ldots, z_K\right)_k = \frac{e^{z_k}}{\sum_{k'} e^{z_{k'}}}$; iuputs z_k are called logits.
- **CE Loss, Vectorized** $\mathcal{L}_{\text{CE}}(\mathbf{y},\mathbf{t}) = -\sum_{k=1}^K t_k\log y_k = -\mathbf{t}^\top(\log\mathbf{y})$ where the log is applied elementwise.
- **Softmax Regression** $\mathbf{z} = \mathbf{W}\mathbf{x}+\mathbf{b}$, $\mathbf{y} = \text{softmax}(\mathbf{z})$, and $\mathcal{L}_{CE} = -\mathbf{t}^\top(\log\mathbf{y})$; GD Updates is $\mathbf{w}_k \leftarrow \mathbf{w}_k - \alpha\frac{1}{N}\sum_{i=1}^N\left(y_k^{(i)} - t_k^{(i)}\right)\mathbf{x}^{(i)}$ where $\mathbf{w}_k$ means the k-th row of $\mathbf{W}$

Activation Functions **Identity** $y = z$ **ReLU** $y = \max(0,z)$ **Soft ReLU** $y = \log(1+e^z)$ **Thresholding** $y = 1$ if $z > 0$ else 0. **Logistic** $y = \frac{1}{1+e^{-z}}$ **tanh** $y = \frac{e^z - e^{-z}}{e^z + e^{-z}}$

Multilayer Perceptron

- **Modularity of Layers** $\mathbf{h}^{(1)} = f^{(1)}(\mathbf{x}) = \phi\left(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}\right)$, $\mathbf{h}^{(2)} = f^{(2)}\left(\mathbf{h}^{(1)}\right) = \phi\left(\mathbf{W}^{(2)}\mathbf{h}^{(1)} + \mathbf{b}^{(2)}\right)$, $\ldots$, $\mathbf{y} = f^{(L)}\left(\mathbf{h}^{(L-1)}\right) = f^{(L)}\circ\cdots\circ f^{(1)}(\mathbf{x})$
- **Choice of Last Layer Activation Func** Regression: $\mathbf{y} = f^{(L)}\left(\mathbf{h}^{(L-1)}\right) = \left(\mathbf{w}^{(L)}\right)^T\mathbf{h}^{(L-1)} + b^{(L)}$; Binary Classification: $\mathbf{y} = f^{(L)}\left(\mathbf{h}^{(L-1)}\right) = \sigma\left(\left(\mathbf{w}^{(L)}\right)^T\mathbf{h}^{(L-1)} + b^{(L)}\right)$
- **Back Propagation** Suppose $\mathcal{L}$ what I want to optimize, then for some variable $\mathbf{w}$ that we want to optimize w.r.t., $\frac{\partial\mathcal{L}}{\partial\mathbf{w}} =: \overline{\mathbf{w}}$
- **Back Prop Cost** Forward: one add-multiplicity operation per weight; Backward: two add-multiplicity operations per weight $\implies$ the Backward pass is about as expensive as two Forward passes. (cost is linear in # of layers, quadratic in # of units per layer)

Statistic on Samples

- **Sample Mean** $\hat{\boldsymbol{\mu}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}^{(i)}$. $\hat{\boldsymbol{\mu}}$ roughly quantifies where your data is located in space
- **Sample Cov** $\hat{\boldsymbol{\Sigma}} = \frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}}\right)\left(\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}}\right)^{\top}$ quantifies the shape of spread of the data

Euclidean projection Let $\mathcal{S}$ denote the subspace with dim $= k$ that is spanned by the basis $\{\mathbf{u}_1, ..., \mathbf{u}_K\} \subseteq \mathbb{R}^D$. Then,

- Any vector $\mathbf{y} \in \mathcal{S}$ could be represented as $\mathbf{y} = \sum_{i=1}^{K} z_i\mathbf{u}_i$, for some $z_1, ..., z_k \in \mathbb{R}$
- The projection of $\mathbf{x}$ onto $\mathcal{S}$ is given as $\text{Proj}_{\mathcal{S}}(\mathbf{x}) = \sum_{i=1}^{K}(\mathbf{x}^{\top}\mathbf{u}_i)\mathbf{u}_i = \sum_{i=1}^{K} z_i\mathbf{u}_i \quad$ where $\quad z_i = \mathbf{x}^{\top}\mathbf{u}_i$

Principle Component Analysis - Projection onto Subspace

- Let $\{\mathbf{u}_k\}_{k=1}^{K}$ be an orthonormal basis of the subspace $\mathcal{S}$.
- Define $\mathbf{U}$ to be a matrix with columns $\{\mathbf{u}_k\}_{k=1}^{K}$ then $\mathbf{z} = \mathbf{U}^T(\mathbf{x} - \hat{\boldsymbol{\mu}})$. Here the $\mathbf{z}$ is called the code vector
- Also, $\tilde{\mathbf{x}} = \hat{\boldsymbol{\mu}} + \mathbf{U}\mathbf{z} = \hat{\boldsymbol{\mu}} + \mathbf{U}\mathbf{U}^T(\mathbf{x} - \hat{\boldsymbol{\mu}})$ is called the reconstruction of $\mathbf{x}$
- Note: $\mathbf{U}\mathbf{U}^T$ is the projector matrix and $\mathbf{U}^T\mathbf{U} = I$ is the identity matrix.
- $\mathbf{x}$ and $\tilde{\mathbf{x}}$ are both in $\mathbb{R}^D$ while $\tilde{\mathbf{x}}$ lives in a low dimensional subspace in $\mathbb{R}^D$. The code vector $\mathbf{x}$ is in $\mathbb{R}^K$, and is the low dim representation of the vector $\mathbf{x}$

PCA - Learning Subspace

- **Criteria I:** Minimize the reconstruction error: Find vectors in a subspace that are closest to data points, $\min_{\mathbf{U}} \frac{1}{N}\sum_{i=1}^{N}\left\|\mathbf{x}^{(i)} - \tilde{\mathbf{x}}^{(i)}\right\|^2$
- **Criteria II:** Maximize the variance of reconstructions: Find subspaces where data has the most variability, $\max_{\mathbf{U}} \frac{1}{N}\sum_{i}\left\|\tilde{\mathbf{x}}^{(i)} - \hat{\boldsymbol{\mu}}\right\|^2$
- **Proof: Criteria I $\equiv$ Criteria II**; It suffices to show that $\frac{1}{N}\sum_{i=1}^{N}\left\|\mathbf{x}^{(i)} - \tilde{\mathbf{x}}^{(i)}\right\|^2 = \text{ const } - \frac{1}{N}\sum_{i}\left\|\tilde{\mathbf{x}}^{(i)} - \hat{\boldsymbol{\mu}}\right\|^2$

Work Flow of EM and GGM

- Step 0; Assume (1). Pick a cluster (with probability π_k) and (2). sample from it using the parameters μ_k, Σ_k. (Some gaussian distribution)
- Step 1; The log-likelihood is $\log P(X|\pi, \mu_k, \Sigma_k) = \sum_{i=1}^{N} \log \sum_{k=1}^{K} \pi_k N(X_n|\mu_k, \Sigma_k)$
- Step 2; Optimize via EM
- Step 2.1 Compute responsibilities (E)
- Step 2.2 Maximize (M)

$$\mu_k = \frac{1}{N_k} \sum_k$$

- Notice that Generative models are called "generative" because they try to learn the underlying distribution, while the discriminative analysis procedures (eg NN) are trying to simply map data.