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 underfit. **Curse of Dim:** In high dimensions, "most" points are approximately the same distance. **Computation Cost:** 0 (minimal) at trianing/ 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, $H(Y|X)\le H(Y)$, $X\perp Y \implies H(Y|X)=H(Y)$, $H(Y|Y)=0$, 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 ($\uparrow \implies$ underfitting):** 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 trianing 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:** $\mathbb{E}_{\mathcal{D}_1,\ldots,\mathcal{D}_m \overset{iid}{\sim} p_{\text{dataset}}}\left[h(\mathbf{x})\right] = \frac{1}{m}\sum_{i=1}^m \mathbb{E}_{\mathcal{D}_i\sim p_{\text{dataset}}}\left[h_i(\mathbf{x})\right] = \mathbb{E}_{\mathcal{D}\sim p_{\text{dataset}}}\left[h_{\mathcal{D}}(\mathbf{x})\right]$ **Variance Reduced:** $\operatorname{Var}_{\mathcal{D}_1,\ldots,\mathcal{D}_m}[h(\mathbf{x})] = \frac{1}{m^2}\sum_{i=1}^m \operatorname{Var}\left[h_i(\mathbf{x})\right] = \frac{1}{m}\operatorname{Var}\left[h_{\mathcal{D}}(\mathbf{x})\right]$

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 $\operatorname{Var}\left(\frac{1}{m}\sum_{i=1}^m h_i(\mathbf{x})\right) = \frac{1}{m}(1-\rho)\sigma^2+\rho\sigma^2$

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

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 $\mathbf{w}_\lambda^{Ridge} = \underset{\mathbf{w}}{\operatorname{argmin}}\mathcal{J}_{\text{reg}}(\mathbf{w}) = \underset{\mathbf{w}}{\operatorname{argmin}}\frac{1}{2}\|\mathbf{X}\mathbf{w}-\mathbf{t}\|_2^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2 = \left(\mathbf{X}^T\mathbf{X}+\lambda\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{t}$ 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** $\mathcal{L}_{LCE}(z,t) = \mathcal{L}_{CE}(\sigma(z),t) = t\log(1+e^{-z})+(1-t)\log(1+e^{z})$

Multiclass Classification

- **Softmax Function** Natural generalization of logistic func: $y_k = \operatorname{softmax}\left(z_1,\ldots,z_K\right)_k = \frac{e^{z_k}}{\sum_{k'}e^{z_{k'}}}$; 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** $\mathbf{z}=\mathbf{W}\mathbf{x}+\mathbf{b}$, $\mathbf{y}=\operatorname{softmax}(\mathbf{z})$, 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)