601.475 - Machine Learning

Kaushik Srinivasan

March 13, 2019

Topic 1 — Introduction to ML

Machine Learning - Using *Experience* to gain *expertise*. Design of algorithms of

- improve their performance
- at some task
- with experience

Supervised Learning

- **Classification** - Discrete Labels
- **Regression** - Continuous Labels

Task: Given $X \in \mathcal{X}$, predict $Y \in \mathcal{Y}$
Construct **prediction rule** $f : \mathcal{X} \to \mathcal{Y}$
Performance: Risk $R(f) \equiv \mathbb{E}_{XY}[\text{loss}(Y, f(X))]$, $(X,Y) \sim P_{XY}$
Experience: Training data $\{(X_i, Y_i)\}_{i=1}^n \sim P_{XY}$ (unknown)

$$\{(X_i, Y_i)\}_{i=1}^n \longrightarrow \textbf{Learning Algorithm} \longrightarrow \hat{f}_n$$

Unsupervised Learning

Task: Given $X \in \mathcal{X}$, learn $f(X)$

- Density Estimation
- Clustering
- Embedding

Performance Measure in Supervised L.

0/1 Loss: $loss(X, f(X)) = 1_{\{f(X) \neq Y\}}$
Square Loss : $loss(Y, f(X)) = (f(X) - Y)^2$

$loss(Y, f(X))$ - Measure of closeness between true label Y and prediction $f(X)$

$$(X, Y) \sim P_{XY}$$

Risk $R(f) \equiv \mathbb{E}_{XY}[\text{loss}(Y, f(X))]$

$loss(Y, f(X))$	Risk $R(f)$
$1_{f(X) \neq Y}$ **0/1 Loss**	$P(f(X) \neq Y)$ **Probability of Error**
$(f(X) - Y)^2$ **Square Loss**	$\mathbb{E}[(f(X) - Y)^2]$ **Mean Square Error**

Bayes Optimal Rule

Ideal Goal: Construct **prediction rule** $f^* : \mathcal{X} \to \mathcal{Y}$

$$f^* = \arg\min_f \mathbb{E}_{XY}[\text{loss}(Y, f(X))]$$

Best possible performance:

Bayes Risk $R(f^*) \leq R(f)$ for all f

Issues in ML

A good Machine Learning Algorithm:

- Does not *overfit* training data
- *Generalizes* well to test data.

Performance Revisited

Expected Risk (Generalization Error)

$$\mathbb{E}_{D_n}\left[R(\hat{f}_n)\right] \equiv \mathbb{E}_{D_n}\left[\mathbb{E}_{XY}[\text{loss}(Y, \hat{f}_n(X))]\right]$$

Ideal Goal: Construct **prediction rule** $f^* : \mathcal{X} \to \mathcal{Y}$

$$f^* = \arg\min_f \mathbb{E}_{XY}[\text{loss}(Y, f(X))]$$

Practical Goal: Given $\{X_i, Y_i\}_{i=1}^n$, $\hat{f}_n : \mathcal{X} \to \mathcal{Y}$, **learn** prediction rule

$$\hat{f}_n = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n [\text{loss}(Y_i, f(X_i))]$$

By Law of Large Numbers: $(n \longrightarrow \infty)$

$$\frac{1}{n} \sum_{i=1}^n [\text{loss}(Y_i, f(X_i))] \longrightarrow \mathbb{E}_{XY}[\text{loss}(Y, f(X))]$$

Consistency and Rates of Convergence

Excess Risk:

$$\mathbb{E}_{D_n}\left[R(\hat{f}_n)\right] - R(f^*)$$

is consistent if Excess Risk $\to 0$ as n $\to \infty$

How to Approach a ML Algorithm

1. Consider your Goal $\to$ definition of task T.
2. Consider nature of experience E.
3. Choose type of output O to Learn
4. Choose performance measure P.
5. Choose representation for input X.
6. Choose set of possible solutions H.
7. Choose or design learning algorithm.

Topic 2 — Linear Regression

Formal setup

- Input data space $\mathcal{X}$
- Output (label, target) space $\mathcal{Y}$
- Unknown probability distribution $p(\cdot,\cdot)$ over $\mathcal{X} \times \mathcal{Y}$
- We are given labelled examples $(\mathbf{x}_i, y_i)$, $i = 1, \dots, N$ sampled i.i.d. from $p; \mathbf{x}_i \in \mathcal{X},\ y_i \in \mathcal{Y}$
- Goal: for any future $\mathbf{x}$, accurately preduct y (drawn according tto p) in other words: learn a mapping $f : \mathcal{X} \to \mathcal{Y}$

Types of Supervised Problems

Goal: learn $f : \mathcal{X} \to \mathcal{Y}$

- **Regression:** $\mathcal{Y} = \mathbb{R}$, learn (continuous) function f
- **Classification:** $\mathcal{Y} = \{1, \dots, C\}$, learn a separator between classes.

Linear Functions

General Form: $f(\mathbf{x}; \mathbf{w}) = w_0 + w_1 x_1 + \ldots + w_d x_d$
Where $x_0 \equiv 1$.

- 1D case ($\mathcal{X} = \mathbb{R}$): a line
- $\mathcal{X} = \mathbb{R}^2$: a plane
- Hyperplane in general: d-D case

Loss Function

A **loss function**: $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ maps prediction to cost, given true value. Standard choice of regression is squared loss: it's symmetric, non-negative, gives 0 loss for correct prediction.

$$\ell(\hat{y}, y) = (\hat{y} - y)^2$$

Empirical loss: LSQ minimizes empirical loss when ℓ is squared loss.

$$L(\mathbf{w}, \mathbf{X}, \mathbf{y}) = \frac{1}{N} \sum_{i=1}^{N} \ell(f(\mathbf{x}_i; \mathbf{w}), y_i)$$

Goal: Minimize the expected loss (**Risk**)

$$R(\mathbf{w}) = \mathbb{E}_{(\mathbf{x}_0, y_0) \sim p(\mathbf{x}, y)} \left[\ell(f(\mathbf{x}_0; \mathbf{w}), y_0)\right]$$

Emperical Risk Minimization (ERM) approach: to the extent that the training set is a representation of the underlying distribution $p(\mathbf{x}, y)$ the empirical loss serves as a proxy for the risk (expected loss). To minimize square loss -

$$f(\mathbf{x}; \mathbf{w}) = \mathbf{w} \cdot \mathbf{x}$$

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{i=1}^{N} (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2$$

How to we find $\mathbf{w}^* = [w_0^*, w_1^*, \dots, w_d^*]$

Least Squares

We need to minimize L w.r.t. $\mathbf{w}$

$$L(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \mathbf{w} \cdot \mathbf{x})^2$$

$$\frac{\partial}{\partial w_j} L(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial}{\partial w_j} (y_i - \mathbf{w} \cdot \mathbf{x})^2 = 0$$

$$= -\frac{2}{N} \sum_{i=1}^{N} (y_i - \mathbf{w} \cdot \mathbf{x}) x_{ij} = 0$$

$$\boxed{\sum_{i=1}^{N} (y_i - \mathbf{w} \cdot \mathbf{x}_i) = 0}$$

Necessary Conditions:

1. Errors have zero mean
2. Errors are uncorrelated with the data..

Least Squares in Matrix Form

$$\mathbf{X} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1d} \\ \vdots & & \vdots & \\ 1 & x_{N1} & \cdots & x_{Nd} \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} w_0 \\ \vdots \\ w_d \end{bmatrix}$$

Prediction: $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$, errors: $\mathbf{y} - \mathbf{X}\mathbf{w}$

$$L(\mathbf{w}, \mathbf{X}, \mathbf{y}) = \frac{1}{N} (\mathbf{y} - \mathbf{X}\mathbf{w}) \cdot (\mathbf{y} - \mathbf{X}\mathbf{w}) = \frac{1}{N} (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w})$$

$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} = \frac{1}{N} \frac{\partial}{\partial \mathbf{w}} [\mathbf{y}^T\mathbf{y} - \mathbf{w}^T\mathbf{X}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\mathbf{w} + \mathbf{x}^T\mathbf{X}^T\mathbf{X}\mathbf{w}] = 0$$

$$= \frac{1}{N} [\mathbf{0} - (\mathbf{y}^T\mathbf{X})^T + 2\mathbf{X}^T\mathbf{X}\mathbf{w}] = 0$$

$$= -\frac{2}{N} \left(\mathbf{X}^T\mathbf{y} - \mathbf{X}^T\mathbf{X}\mathbf{w}\right) = 0$$

$$\Rightarrow \boxed{\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}}$$

ML Paradox: The more training data we have, the "worse" the fit, but our prediction ability improves.

Best Unrestricted Predictor

$$f^* = \arg\min_{f : \mathcal{X} \to \mathbb{R}} \mathbb{E}_{(\mathbf{x}_0, y_0) \sim p(\mathbf{x}, y)} \left[(f(\mathbf{x}_0) - y)^2\right]$$

Chain rule of Probability: $p(\mathbf{x}, y) = p(y|\mathbf{x})p(\mathbf{x})$

By Def: $\mathbb{E}_{p(y, \mathbf{x})}[g(y, \mathbf{x})] = \int_{\mathbf{x}} \int_{y} g(y, \mathbf{x}) p(y|\mathbf{x}) p(\mathbf{x}) \, dy d\mathbf{x}$

$$\mathbb{E}_{(\mathbf{x}_0, y_0) \sim p(\mathbf{x}, y)} \left[(f(\mathbf{x}_0) - y_0)^2\right]$$

$$= \mathbb{E}_{\mathbf{x}_0 \sim p(\mathbf{x})} \left[\mathbb{E}_{y_0 \sim p(y|\mathbf{x})} \left[(f(\mathbf{x}_0) - y)^2 | \mathbf{x}_0\right]\right]$$

$$= \int_{\mathbf{x}_0} \left\{\mathbb{E}_{y_0 \sim p(y|\mathbf{x})} \left[(f(\mathbf{x}_0) - y_0 | \mathbf{x}_0)^2\right]\right\} p(\mathbf{x}_0) d\mathbf{x}_0$$

minimizing the inner conditional expectation for each $\mathbf{x}_0$

$$\frac{\partial}{\partial f(\mathbf{x})}\mathbb{E}_{p(y|\mathbf{x})}\left[(f(\mathbf{x}_0)-y_0|\mathbf{x}_0)^2\right] = 2\mathbb{E}_{p(y|\mathbf{x})}\left[f(\mathbf{x}_0)-y_0|\mathbf{x}_0\right]$$
$$0 = 2\big(f(\mathbf{x}_0)-\mathbb{E}_{p(y|\mathbf{x})}[y_0|\mathbf{x}_0]\big)$$

$$\boxed{\hat{y}(\mathbf{x}_0) = f^*(\mathbf{x}_0) = \mathbb{E}_{p(y|\mathbf{x})}[y_0|\mathbf{x}_0]}$$

Generative vs. Discriminative Approach

Generative	Discriminative
• Estimate Joint Probability Density $p(\mathbf{x}, y)$ • Normalize to find conditional $p(y\|\mathbf{x})$	• Estimate the density $p(y\|\mathbf{x})$ from data • no need $p(\mathbf{x}, y)$

Decomposition of Error

We can understand the expected loss in this manner. $\hat{\mathbf{w}}$ are LSQ estimates from training data. $\mathbf{w}^*$ are *optimal* linear regression parameters.

$$y-\hat{\mathbf{w}}\cdot\mathbf{x} = (y-\mathbf{w}^*\cdot\mathbf{x}) + (\mathbf{w}^*\cdot\mathbf{x}-\hat{\mathbf{w}}\cdot\mathbf{x})$$

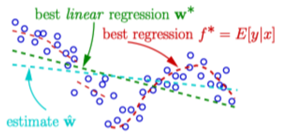
$$\begin{aligned}\mathbb{E}_{p(\mathbf{x},y)}\left[(y-\hat{\mathbf{w}}\cdot\mathbf{x})^2\right] &= \mathbb{E}_{p(\mathbf{x},y)}\left[(y-\mathbf{w}^*\cdot\mathbf{x})^2\right] \\ &\quad + \mathbb{E}_{p(\mathbf{x},y)}[(y-\mathbf{w}^*\cdot\mathbf{x}) \\ &\qquad (\mathbf{w}^*\cdot\mathbf{x}-\hat{\mathbf{w}}\cdot\mathbf{x})] \\ &\quad + \mathbb{E}_{p(\mathbf{x},y)}\left[(\mathbf{w}^*\cdot\mathbf{x}-\hat{\mathbf{w}}\cdot\mathbf{x})^2\right] \\ &= \mathbb{E}_{p(\mathbf{x},y)}\left[(y-\mathbf{w}^*\cdot\mathbf{x})^2\right] \\ &\quad + \mathbb{E}_{p(\mathbf{x},y)}\left[(\mathbf{w}^*\cdot\mathbf{x}-\hat{\mathbf{w}}\cdot\mathbf{x})^2\right]\end{aligned}$$

Approximation Error

$$\mathbb{E}[(y-\mathbf{w}^*\cdot\mathbf{x})^2]$$

Estimation Error

$$\mathbb{E}_{p(\mathbf{x},y)}\left[(\mathbf{w}^*\cdot\mathbf{x}-\hat{\mathbf{w}}\cdot\mathbf{x})^2\right]$$



Staistical view of Regression

$$y = f(\mathbf{x};\mathbf{w}) + \nu, \qquad \nu \sim \mathcal{N}(\nu; 0, \sigma^2)$$

Where *noise* ν accounts everything not captured by f. Given input data $\mathbf{x}$, the label y is a random variable.

$$\begin{aligned}p(y|\mathbf{x};\mathbf{w},\sigma) &= \mathcal{N}(y; f(\mathbf{x};\mathbf{w}), \sigma^2) \\ &= \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(y-f(\mathbf{x};\mathbf{w}))^2}{2\sigma^2}\right)\end{aligned}$$

To sample y for a given $\mathbf{x}$

Maximum Likelihood Estimation

$$\begin{aligned}\hat{\mathbf{w}}_{ML} &= \arg\max_{\mathbf{w}} p(Y|X;\mathbf{w},\sigma) \\ &= \arg\max_{\mathbf{w}} \prod_{i=1}^{N}\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(y_i-f(\mathbf{x}_i;\mathbf{w}))^2}{2\sigma^2}\right)\end{aligned}$$

We take the log likelihood because it is also monotonically increasing, and easier to use.

$$\begin{aligned}\log p(Y|X;\mathbf{w},\sigma) &= \sum_{i=1}^{N}\log p(y_i|\mathbf{x}_i;\mathbf{w},\sigma) \\ &= \sum_{i=1}^{N}\left[-\frac{(y_i-f(\mathbf{x}_i;\mathbf{w}))^2}{2\sigma^2} - \log\sigma\sqrt{2\pi}\right] \\ &= -\frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i-f(\mathbf{x}_i;\mathbf{w}))^2 - N\log\sigma\sqrt{2\pi}\end{aligned}$$

Red terms are independent of $\mathbf{w}$. Now we define *log-loss* as the negative conditional density of training data.

$$L(f(\mathbf{x};\mathbf{w}), y) = -\log p(y_i|\mathbf{x};\mathbf{w},\sigma)$$

Maximizing log likelihood is always equivalent to *minimizing* log loss.

$$\begin{aligned}\arg\max_{\mathbf{w}}\sum_{i=1}^{N}\log p(y_i|\mathbf{x}_i;\mathbf{w},\sigma) &= \arg\max_{\mathbf{w}} -\sum_{i=1}^{N}(y_i-f(\mathbf{x}_i;\mathbf{w}))^2 \\ &= \arg\min_{\mathbf{w}}\sum_{i=1}^{N}(y_i-f(\mathbf{x}_i;\mathbf{w}))^2\end{aligned}$$

Maximum a Posteriori Estimation

Now given a prior $p(\theta)$ about our parametters, the maximum posterior is

$$\hat{\theta}_{MAP} = \arg\max_{\Theta} p(\theta|X) = \arg\max_{\Theta}\frac{p(X|\theta)p(\theta)}{p(X)}$$

The choice of prior matters! Bayesian approach, Utilitarian approach & regularization. If uniform $p(\theta)$, then $MAP = MLE$

Polynomial Regression

Consider the 1-D case where $f : x \to y$

$$f(x;\mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_m x^m$$

No longer linear in x but linear in $\mathbf{w}$! We define $\phi(x) = [1, x, x^2, \ldots, x^m]^T$, then $f(x;\mathbf{w}) = \mathbf{w}\cdot\boldsymbol{\phi}(x)$. So least squares solution:

$$\hat{\mathbf{w}} = (\mathbf{X^TX})^{-1}\mathbf{X^Ty} \text{ where } \mathbf{X} = \begin{bmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^m \\ 1 & x_2 & x_2^2 & \ldots & x_2^m \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & x_N & x_N^2 & \ldots & x_N^m \end{bmatrix}$$

$$f(\mathbf{x};\mathbf{w}) = w_0\phi_0(\mathbf{x}) + w_1\phi_1(\mathbf{x}) + \ldots + w_m\phi_m(\mathbf{x}) = \mathbf{w}\cdot\boldsymbol{\phi}(\mathbf{x})$$

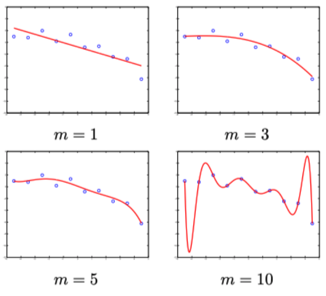
where $\phi_j(\mathbf{x}) : \mathcal{X} \to \mathbb{R}$, $j = 1\ldots, m$ are basis functions.

$$f(\mathbf{x};\mathbf{w}) = \mathbf{w}\cdot\boldsymbol{\phi}(\mathbf{x})$$

Is still linear in $\mathbf{w}$ even when $\boldsymbol{\phi}$ is non-linear in inputs $\mathbf{x}$, $\phi_0(\mathbf{x}) \equiv 0$

$$\hat{\mathbf{w}} = (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-\mathbf{1}}\mathbf{X}^{\mathbf{T}}\mathbf{y}$$

$$\mathbf{X} = \begin{bmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \phi_2(\mathbf{x}_1) & \dots & \phi_m(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \phi_2(\mathbf{x}_2) & \dots & \phi_m(\mathbf{x}_2) \\ \dots & \dots & \dots & \dots & \dots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \phi_2(\mathbf{x}_N) & \dots & \phi_m(\mathbf{x}_N) \end{bmatrix}$$



How to avoid overfitting

- if a model overfits (it is *too* sensitive to data) it will be unstable - removing part of the data will change the fit significantly.
- We can *hold out* part of the data - this is *validation* set
- Fit the model to the rest and test on heldout data.
- Problem - if heldout set too small, we are susceptible to chance
- Problem - if heldout set too large, we get pessimistic (training on too little data compared to what we do).

Topic 3 — Regularization

Controlling for overfitting

1. More complex model (10th degree) overfits more than simple model (linear)
2. Pure ERM would always prefer complex model
3. Validation is a way to control for this in *model selection*

Intuitively, complexity of model measured by the number of "degrees of freedom" $\rightarrow$ more complex model more likely to overfit. Caused by finite training data.

Controlling for overfitting

1. **Idea 1**: Restrict model complexity based on amount of data $\rightarrow\approx$ 10 examples per parameter
2. **Idea 2**: Directly peanalize by number of parameters. Akaike information criterion (AIC).

$$\max \left[\log p(X \mid \hat{\mathbf{w}}) - \#\text{params}\right]$$

But: definition of model complexity as number of parameters is too simplistic.
3. **Idea 3**: consider the behavior of values of $\mathbf{w}^*$

Intuition. Should penalize not the parameters, but the number of bits required to encode the parameters $\rightarrow$ with finite parameter values, these are the same.

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \left\{ \frac{1}{2} \sum_{i=1}^{N} \log p(\text{data}_i; \mathbf{w}) - \text{penalty}(\mathbf{w}) \right\}$$

Ridge Regression (L_2)

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \left\{ \sum_{i=1}^{N} (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2 + \lambda \sum_{j=1}^{m} w_j^2 \right\}$$

where $\mathbf{w} = [w_0, w_1, w_2, \dots, w_m]$. **Careful!** Solution is *not invariant* to scaling, we should normalize input before solving.

$$\mathbf{w}^*_{\text{ridge}} = (\lambda \mathbf{I} + \mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

We can write the following as an optimization problem

$$\min_{\mathbf{w}} \sum_{i=1}^{N} (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2$$

$$\text{subject to } \sum_{j=1}^{m} w_j^2 \leq t$$

Lasso Regression (L_1)

$$\mathbf{w}^*_{\text{lasso}} = \arg\max_{\mathbf{w}} \left\{ -\sum_{i=1}^{N} (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2 - \lambda \sum_{j=1}^{m} |w_j| \right\}$$

- Still concave(has a unique maximum), but not smooth (differentiable)
- Can solve using convex programming methods
- "lasso" $\rightarrow$ least absolute shrinkage and selection operator.

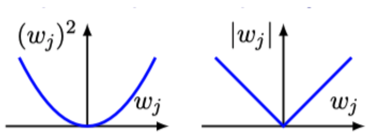
We can write the following as an optimization problem

$$\min_{\mathbf{w}} \sum_{i=1}^{N} (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2$$

$$\text{subject to } \sum_{j=1}^{m} |w_j| \leq t$$

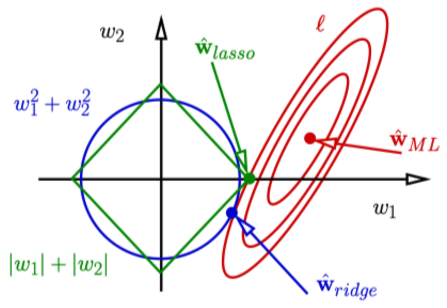
Geometry of surfaces

We can compare the shape of penalty as a function of w_j



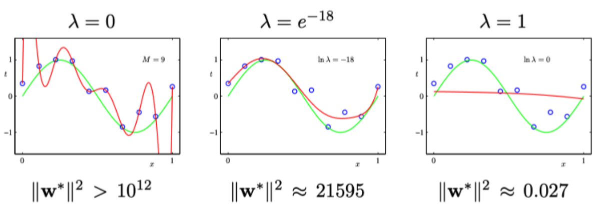
An equivalently formulation of L_p regularization is constrained maximization.

$$\hat{\mathbf{w}} = \underset{\mathbf{w}:\sum_{j=1}^{m}|w_j|^p\le\beta}{\arg\max} -\sum_{i=1}^{N}(y_i - \mathbf{w}\cdot\mathbf{x}_i)^2$$



- sufficiently large λ (small β) lasso leads to *sparsity*
- must explicitly solve optimization problem using Lagrange multipliers

Choice of λ



Most often λ is chosen by cross-validation.

Topic 4 — Gradient Descent

Often times, we cannot always solve the closed form solution $\mathbf{w}^* = (\mathbf{X}^\mathbf{T}\mathbf{X})^{-1}\mathbf{X}^\mathbf{T}\mathbf{y}$ e.g. matrix is too large/difficult to calculate the pseudoinverse. **Gradient ascent/descent**

- Gradient ascent - "hill climbing" on function surface.
- start at a random spot and make steps in direction of maximal altitude.
- Equivalent: *gradient descent* on *convex* loss $-\log p(y \mid \mathbf{x}; \mathbf{w})$

Gradient descent algorithm

- Iteration counter $t = 0$
- Initialize $\mathbf{w}^{(t)}$ (to zero or a small random vector)
- for $t = 1, \ldots$: compute gradient

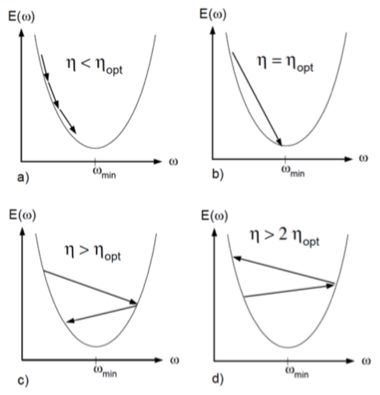
$$\mathbf{g}^{(t)} = \nabla f\big(\mathbf{X}, \mathbf{y}; \mathbf{w}^{(t-1)}\big)$$

  update model

$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} - \eta\mathbf{g}^{(t)}$$

- *learning rate* η controls step size

The size of η is important for the rate of convergence



Gradient descent convergence

- Minimum number of iterations (time budget) $\to$ may not have converged
- Minimum required change in objective value (loss)
- Minimum required change in model parameters ($\mathbf{w}$)

Estimation Theory

- *Estimator* $\hat{\theta}$ of parameter θ estimates it given input data.
- The *bias* of an estimator $\hat{\theta}$ is

$$\text{bias}(\hat{\theta}) \triangleq \mathbb{E}_X\left[\hat{\theta} - \theta\right]$$

- An *unbiased* estimator $\mathbb{E}_X\left[\hat{\theta}\right] = \theta$

Consistency of Estimator

- With enough data, bias *may* nott be a problem.
- Estimator $\hat{\theta}$ is *consistent* if

$$\lim_{N\to\infty} \hat{\theta}_N \xrightarrow{p} \theta$$

Estimation and Regression

- True model $y = F(\mathbf{x}) + \nu$, 0 mean additive noise ν
- Approximate F by $f(\mathbf{x}; \hat{\mathbf{w}}) \in \mathcal{F}$ with $\hat{\mathbf{w}}$ estimated from data X.
- $\hat{f}(\mathbf{x}) = f(\mathbf{x}; \hat{w})$ estimate based on particular X
- $\bar{f}(\mathbf{x}) = \mathbb{E}_X[f(\mathbf{x}; \hat{\mathbf{w}})]$ average estimate over training sets X..

- $f^*(\mathbf{x}) = f(\mathbf{x}; \arg\min_w \mathbb{E}_{p(\mathbf{x},y)}[(y - f(\mathbf{x};\mathbf{w}))^2])$ the best estimate by function in $\mathcal{F}$

Bias-Variance decomposition

Denote $\bar{\theta} = \mathbb{E}[\hat{\theta}]$

$$\begin{aligned} E\left[(\hat{\theta}-\theta)^2\right] &= \mathbb{E}\left[(\hat{\theta}-\bar{\theta}+\bar{\theta}-\theta)^2\right] \\ &= \mathbb{E}\left[(\hat{\theta}-\bar{\theta})^2\right] + 2(\bar{\theta}-\theta)\underbrace{\mathbb{E}\left[\hat{\theta}-\bar{\theta}\right]}_{=0} + \mathbb{E}\left[(\bar{\theta}-\theta)^2\right] \\ &= \mathbb{E}\left[(\hat{\theta}-\bar{\theta})^2\right] - (\bar{\theta}+\theta)^2 \\ &= \mathrm{var}(\hat{\theta}) + \mathrm{bias}^2(\hat{\theta}) \end{aligned}$$

- bias^2 term $\Leftrightarrow$ approximation error
- variance $\Leftrightarrow$ estimation error due to finite data.

Bias-Variance in Regression

For a single $\mathbf{x}_0$

$$\mathbb{E}_X\left[(y_0-\hat{f}(\mathbf{x}_0))^2\right] = (y_0-\bar{f}(\mathbf{x}_0))^2 + \underbrace{\mathbb{E}_X\left[(\hat{f}(\mathbf{x}_0)-\bar{f}(\mathbf{x}_0))^2\right]}_{\text{variance}}$$

The first term can be further decomposed

$$(y_0-\bar{f}(\mathbf{x}_0))^2 = \underbrace{(y_0-F(\mathbf{x}_0))^2}_{\text{noise}} + \underbrace{(F(\mathbf{x}_0)-\hat{f}(\mathbf{x}_0))^2}_{\text{bias}^2}$$

- noise term is *irreducible*
- bias^2 is due to difference between f and F

$$\mathbb{E}[\text{squared loss}] = \mathrm{bias}^2 + \mathrm{var} + \mathrm{noise}$$

Bias-Variance Tradeoff: Theory

Cramer-Rao inequality: for an unbiased estimator $\hat{\theta}_N$

$$\mathrm{var}(\hat{\theta}_N) \geq \frac{1}{\mathbb{E}\left[(\frac{\partial}{\partial\theta}\log p(\mathbf{X};\theta))^2\right]} = \frac{1}{\mathcal{I}(\theta)}$$

where $\mathcal{I}(\theta)$ is the *Fischer Information*. Intuitively, it measures the amount of data X provides about parameter θ

Topic 5 — Logistic Regression

Classification as Regression

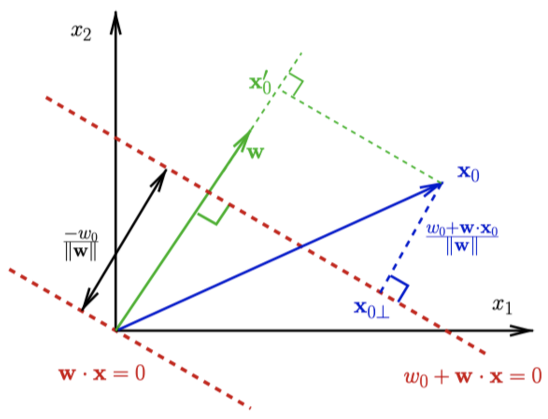
$$f(\mathbf{x};\hat{\mathbf{w}}) = w_0 + \hat{\mathbf{w}}\cdot\mathbf{x}$$

Can't just take $\hat{y} = f(\mathbf{x};\hat{\mathbf{w}})$ as it won't be valid label. Hence we use a decision rule for $y \in \{-1, 1\}$

$$y = \begin{cases} 1 & f(\mathbf{x};\hat{\mathbf{w}}) \geq 0 \\ -1 & f(\mathbf{x};\hat{\mathbf{w}}) < 0 \end{cases}$$

$\hat{y} = \mathrm{sign}(w_0 + \hat{\mathbf{w}}\cdot\mathbf{x})$ is also a valid *linear classifier* which transforms $\mathbb{R} \to \{1, -1\}$. Linear equation $w_0 + \hat{\mathbf{w}}\cdot\mathbf{x} = 0$ separates the space into two "half-spaces"

Classification as Regression


x_2
$\mathbf{x}_0'$
$\mathbf{w}$
$\mathbf{x}_0$
$\frac{-w_0}{\|\mathbf{w}\|}$
$\frac{w_0+\mathbf{w}\cdot\mathbf{x}_0}{\|\mathbf{w}\|}$
$\mathbf{x}_{0\perp}$
x_1
$\mathbf{w}\cdot\mathbf{x}=0$
$w_0+\mathbf{w}\cdot\mathbf{x}=0$

- $\mathbf{w}\cdot\mathbf{x} = 0$: a line passing through the origin and *orthogonal* to $\mathbf{w}$
- $\mathbf{w}\cdot\mathbf{x} + w_0 = 0$ shifts the line along $\mathbf{w}$.
- $\mathbf{x}'$ is the projection of $\mathbf{x}$ on $\mathbf{w}$
- Set up new coordinate system $\mathbf{x} \to (w_0 + \mathbf{w}\cdot\mathbf{x})/||\mathbf{w}||$

Linear Classifiers

$$\hat{y} = h(\mathbf{x}) = \mathrm{sign}(w_0 + \mathbf{w}\cdot\mathbf{x})$$

- Classifying using linear decision boundary effectively reduces data dimension to 1.
- Want to minimize the expected 0/1 loss for classifier $h : \mathcal{X} \to \mathcal{Y}$, which for $(\mathbf{x}, y)$ is

$$L(h(\mathbf{x}), y)\begin{cases} 0 & \text{if } h(\mathbf{x}) = y \\ 1 & \text{if } h(\mathbf{x}) \neq y \end{cases}$$

Risk of a classifier

- The risk (expected loss) of a C-way classifier $h(\mathbf{x})$

$$\begin{aligned} R(h) &= \mathbb{E}_{\mathbf{x},y}[L(h(\mathbf{x}), y)] \\ &= \int_{\mathbf{x}} \sum_{c=1}^{C} L(h(\mathbf{x}), y) p(\mathbf{x}, y = c)\ d\mathbf{x} \\ &= \int_{\mathbf{x}} \left[\sum_{c=1}^{C} L(h(\mathbf{x}), y) p(y = c|\mathbf{x})\right] p(\mathbf{x}) d\mathbf{x} \end{aligned}$$

- It is enough to minimize the *conditional risk* for any

$\mathbf{x}$

$$\begin{aligned} R(h|\mathbf{x}) &= \sum_{c=1}^{C} L(h(\mathbf{x}), c) p(y = c|\mathbf{x}) \\ &= 0 \cdot p(y = h(\mathbf{x})|\mathbf{x}) + 1 \cdot \sum_{c \neq h(\mathbf{x})} p(y = c|\mathbf{x}) \\ &= \sum_{c \neq h(\mathbf{x})} p(y = c|\mathbf{x}) = 1 - p(y = h(\mathbf{x})|\mathbf{x}) \end{aligned}$$

- To minimize conditional risk given $\mathbf{x}$, the classifier must decide

$$h(\mathbf{x}) = \arg\max_c p(y = c|\mathbf{x})$$

- This is the *best possible classifier* in terms of generalization i.e. expected misclassification rate on new examples.

Logistic Model

Define the decision boundary directly

$$\log \frac{p(y = 1|\mathbf{x})}{p(y = 0|\mathbf{x})} = w_0 + \mathbf{w} \cdot \mathbf{x} = 0$$

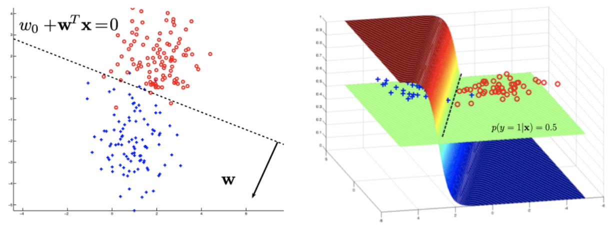
We define the logistic function

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- $\sigma(0) = 1/2$. Shift the crossing to z: $\sigma(x - z)$
- change the "slope": $\sigma(ax)$

Logistic Function in $\mathbb{R}^d$

- direction of $\mathbf{w}$ determines orientation
- w_0 determines location
- $||\mathbf{w}||$ determines slope



Logistic Function in $\mathbb{R}^d$

- *Regression:* observe values, measure residuals under the model
- *Logistic Regression:* observe values, measure their probability under the model

$$\begin{aligned} p(y_i|\mathbf{x}_i; \mathbf{w}) &= \begin{cases} \sigma(w_0 + \mathbf{w} \cdot \mathbf{x}_i) & \text{if} y_i = 1 \\ 1 - \sigma(w_0 + \mathbf{w} \cdot \mathbf{x}_i) & \text{if} y_i = 0 \end{cases} \\ &= \sigma(w_0 + \mathbf{w} \cdot \mathbf{x}_i)^{y_i} (1 - \sigma(w_0 + \mathbf{w} \cdot \mathbf{x}_i))^{1 - y_i} \end{aligned}$$

- the log likelihood of $\mathbf{w}$

$$\log p(Y|X; \mathbf{w}) = \sum_{i=1}^{N} \log p(y_i|\mathbf{x}_i; \mathbf{w})$$

- set derivatives to 0

$$\frac{\partial}{\partial w_0} \log p(Y|X; \mathbf{w}) = \sum_{i=1}^{N} (y_i - \sigma(w_0 + \mathbf{w} \cdot \mathbf{x}_i)) = 0$$

$$\frac{\partial}{\partial w_j} \log p(Y|X; \mathbf{w}) = \sum_{i=1}^{N} (y_i - \sigma(w_0 + \mathbf{w} \cdot \mathbf{x}_i)) x_{ij} = 0$$

- Treat $y_i - p(y_i|\mathbf{x}_i) = y_i - \sigma(w_0 + \mathbf{w} \cdot \mathbf{x}_i)$ as *prediction error* of the model on $\mathbf{x}_i, y_i$