# Problem Solving with AI Techniques
# (Generalized) Linear Models

Paul Weng

UM-SJTU Joint Institute

VE593, Fall 2018

# Framework of Linear Regression

- Labeled i.i.d. data: $\mathcal{D} = \{(\boldsymbol{x}^1, y^1), (\boldsymbol{x}^2, y^2), \ldots, (\boldsymbol{x}^N, y^N)\}$ with $\boldsymbol{x}^i \in \mathbb{R}^D, y^i \in \mathbb{R}$
- Usual Trick: Redefine $\boldsymbol{x}^i \leftarrow (x_0^i = 1, x_1^i, \ldots, x_D^i)^\intercal$
- Notations: $\boldsymbol{X} = \left(\boldsymbol{x}^1, \ldots, \boldsymbol{x}^N\right)^\intercal \in \mathbb{R}^{N\times(D+1)}$, $\boldsymbol{y} = (y^1, \ldots, y^N)^\intercal \in \mathbb{R}^N$

- Regression model: $y = f(\boldsymbol{x}) + \varepsilon$ where $f \in \mathcal{H}$ and $\varepsilon$ noise
- Linear regression model: $f_{\boldsymbol{w}}(\boldsymbol{x}) = w_0 + \sum_{j=1}^{D} w_j \boldsymbol{x}_j = \boldsymbol{w}^\intercal \boldsymbol{x}$
- Problem: learn weights $\boldsymbol{w} = (w_0, w_1, \ldots, w_D) \in \mathbb{R}^{D+1}$ that fits $\mathcal{D}$

# Method of Least Squares

Based on squared-error loss, minimize

$$R_{\mathcal{D}}(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{N}\big(f_{\boldsymbol{w}}(\boldsymbol{x}^i) - y^i\big)^2 = \frac{1}{2}\sum_{i=1}^{N}\big(\sum_{j=0}^{D} w_j \boldsymbol{x}_j^i - y^i\big)^2$$

$$= \frac{1}{2}\sum_{i=1}^{N}\big(\boldsymbol{w}^\intercal \boldsymbol{x}^i - y^i\big)^2 = \frac{1}{2}(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^\intercal(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$$

# Method of Least Squares

Based on squared-error loss, minimize

$$
\begin{aligned}
R_{\mathcal{D}}(\boldsymbol{w}) &= \frac{1}{2}\sum_{i=1}^{N}\left(f_{\boldsymbol{w}}(\boldsymbol{x}^i) - y^i\right)^2 = \frac{1}{2}\sum_{i=1}^{N}\Big(\sum_{j=0}^{D} w_j \boldsymbol{x}_j^i - y^i\Big)^2 \\
&= \frac{1}{2}\sum_{i=1}^{N}\left(\boldsymbol{w}^\intercal \boldsymbol{x}^i - y^i\right)^2 = \frac{1}{2}(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^\intercal(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})
\end{aligned}
$$

It is a convex optimization problem: compute the gradient and cancel it.

$$
\begin{aligned}
\frac{\partial R_{\mathcal{D}}(\boldsymbol{w})}{\partial w_k} &= \sum_{i=1}^{N}\Big(\sum_{j=0}^{D} w_j \boldsymbol{x}_j^i - y^i\Big)\boldsymbol{x}_k^i \\
\nabla_{\boldsymbol{w}} R_{\mathcal{D}}(\boldsymbol{w}) &= \sum_{i=1}^{N}\left(\boldsymbol{w}^\intercal \boldsymbol{x}^i - y^i\right)\boldsymbol{x}^i = \boldsymbol{X}^\intercal(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})
\end{aligned}
$$

# Closed Form Solution

$$
\begin{aligned}
&\nabla_{\boldsymbol{w}} R_{\mathcal{D}}(\boldsymbol{w}) = 0 \\
&\boldsymbol{X}^\intercal(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}) = 0 \\
&\boldsymbol{X}^\intercal\boldsymbol{X}\boldsymbol{w} - \boldsymbol{X}^\intercal\boldsymbol{y} = 0 \\
&\boldsymbol{X}^\intercal\boldsymbol{X}\boldsymbol{w} = \boldsymbol{X}^\intercal\boldsymbol{y} \quad (*) \\
&\boldsymbol{w} = (\boldsymbol{X}^\intercal\boldsymbol{X})^{-1}\boldsymbol{X}^\intercal\boldsymbol{y} \quad \text{if } \boldsymbol{X}^\intercal\boldsymbol{X} \text{ is invertible}
\end{aligned}
$$

- In practice, we don't invert $\boldsymbol{X}^\intercal\boldsymbol{X}$, but solve (*).
- Computational complexity: $O(D^2N) = O(D^2N + DN + D^3)$

# (Batch) Gradient Descent

- Gradient descent: given dataset $\boldsymbol{X}, \boldsymbol{y}$ and initial guess $\boldsymbol{w}_0$
  Repeat until convergence:

$$\boldsymbol{w}_t \leftarrow \boldsymbol{w}_{t-1} - \alpha \nabla_{\boldsymbol{w}} R_{\mathcal{D}}(\boldsymbol{w}_{t-1})$$

where $\nabla_{\boldsymbol{w}} R_{\mathcal{D}}(\boldsymbol{w}) = \sum_{i=1}^{N} \left(\boldsymbol{w}^\intercal \boldsymbol{x}^i - y^i\right) \boldsymbol{x}^i = \boldsymbol{X}^\intercal(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$

---

BatchGradientDescent$(T, \mathcal{D})$
initialize $\boldsymbol{w}$
**for** $t = 1$ *to* $T$ **do**
  $\boldsymbol{w}' \leftarrow 0$
  **for** $i = 1$ *to* $N$ **do** $\boldsymbol{w}' \leftarrow \boldsymbol{w}' - \alpha\left(\boldsymbol{w}^\intercal \boldsymbol{x}^i - y^i\right)\boldsymbol{x}^i$ ;
  $\boldsymbol{w} \leftarrow \boldsymbol{w}'$
return $\boldsymbol{w}$

---

# (Stochastic) Gradient Descent

- Issue: $N$ may be large
- Idea: No need to loop over all instances in $\mathcal{D}$
- Stochastic gradient descent: update one random instance per iteration

$$\boldsymbol{w}_t \leftarrow \boldsymbol{w}_{t-1} - \alpha \nabla_{\boldsymbol{w}} R_{\mathcal{D}}(\boldsymbol{w}_{t-1} \,|\, \boldsymbol{x}^i)$$

where $\nabla_{\boldsymbol{w}} R_{\mathcal{D}}(\boldsymbol{w} \,|\, \boldsymbol{x}^i) = \left(\boldsymbol{w}^\intercal \boldsymbol{x}^i - y^i\right)\boldsymbol{x}^i$ with $\boldsymbol{x}^i, y^i$ in $\boldsymbol{X}, \boldsymbol{y}$

StochasticGradientDescent$\left(T, \mathcal{D}\right)$
initialize $\boldsymbol{w}$
**for** $t = 1$ *to* $T$ **do**
  select $\boldsymbol{x}^i, y^i$ in $\mathcal{D}$
  $\boldsymbol{w}' \leftarrow \boldsymbol{w}' - \alpha\left(\boldsymbol{w}^\intercal \boldsymbol{x}^i - y^i\right)\boldsymbol{x}^i$
return $\boldsymbol{w}$

# Linear Basis Function Regression

- Labeled i.i.d. data: $(\boldsymbol{x}^1, y^1), (\boldsymbol{x}^2, y^2), \dots, (\boldsymbol{x}^N, y^N)$ with $\boldsymbol{X} \in \mathbb{R}^{D\times N}, \boldsymbol{y} \in \mathbb{R}^N$ Notation: $\boldsymbol{X} = (\boldsymbol{x}^1, \dots, \boldsymbol{x}^N)$, $\boldsymbol{y} = (y^1, \dots, y^N)$
- Basis functions and features: $\phi(\boldsymbol{x}) = (\phi_0(\boldsymbol{x}), \phi_1(\boldsymbol{x}), \dots, \phi_M(\boldsymbol{x}))^\intercal \in \mathbb{R}^{M+1}$ with $\phi_0(\boldsymbol{x}) = 1$
- Regression model: $y = f(\boldsymbol{x}) + \varepsilon$ where $f \in \mathcal{C}$ and $\varepsilon$ noise
- Linear basis function regression model:
$$f_{\boldsymbol{w}}(\boldsymbol{x}) = w_0 + \sum_{j=1}^{M} w_j \phi_j(\boldsymbol{x}) = \boldsymbol{w}^\intercal \phi(\boldsymbol{x})$$
- The previous two methods apply with $\boldsymbol{X}$ replaced by $\boldsymbol{\Phi} = \big(\phi(\boldsymbol{x}^1), \dots, \phi(\boldsymbol{x}^N)\big)^\intercal \in \mathbb{R}^{N\times(M+1)}$

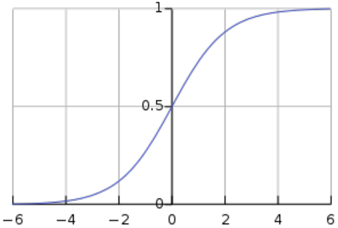
# Framework of Logistic Regression

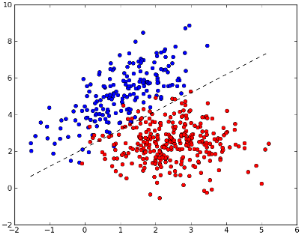
- i.i.d. data (notation with trick): $\boldsymbol{X} = \left(\boldsymbol{x}^1, \dots, \boldsymbol{x}^N\right)^\intercal \in \mathbb{R}^{N \times (D+1)}$,
  $\boldsymbol{y} = (y^1, \dots, y^N)^\intercal \in \{0, 1\}^N$
- Probabilistic binary classifier: $f(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \eta(\boldsymbol{x}) \geq \frac{1}{2} \\ 0 & \text{if } \eta(\boldsymbol{x}) < \frac{1}{2} \end{cases}$
  where $\eta(\boldsymbol{x}) = \mathbb{P}(Y = 1 \,|\, X = \boldsymbol{x}) = 1 - \mathbb{P}(Y = 0 \,|\, X = \boldsymbol{x})$
  Note: for each $\boldsymbol{x}$, $\eta(\boldsymbol{x})$ defines a Bernoulli distribution.
- Logistic regression assumes: $\eta$ is based on a sigmoid (or logistic) function:

$$\eta_{\boldsymbol{w}}(\boldsymbol{x}) = g(\boldsymbol{w}^\intercal \boldsymbol{x}) = \frac{1}{1 + \exp(-\boldsymbol{w}^\intercal \boldsymbol{x})} \text{ where } g(x) = \frac{1}{1 + \exp(-x)}$$

- Probabilistic model $\Rightarrow$ $\boldsymbol{w}$ can be estimated by Maximum Likelihood (or MAP)

# Logistic Regression is a Generalized Linear Model (GLM)



- GLM: response is a function of a linear function
- Logistic regression = linear classifier: decision boundary is linear

$$
\begin{aligned}
\eta_{\boldsymbol{w}}(\boldsymbol{x}) = \frac{1}{1+\exp(-\boldsymbol{w}^\intercal \boldsymbol{x})} &= \frac{1}{2} \\
1+\exp(-\boldsymbol{w}^\intercal \boldsymbol{x}) &= 2 \\
\exp(-\boldsymbol{w}^\intercal \boldsymbol{x}) &= 1 \\
\boldsymbol{w}^\intercal \boldsymbol{x} &= 0
\end{aligned}
$$

# Step 1: Compute Likelihood

As $\mathcal{D} = (\boldsymbol{X}, \boldsymbol{y})$ has been generated i.i.d.,

$$\begin{aligned}
\mathbb{P}(\mathcal{D} \mid \boldsymbol{w}) &= \prod_{i=1}^{N} \mathbb{P}(\boldsymbol{x}_i, y_i \mid \boldsymbol{w}) \\
&= \prod_{i=1}^{N} \mathbb{P}(y_i \mid \boldsymbol{x}_i, \boldsymbol{w}) \mathbb{P}(\boldsymbol{x}_i \mid \boldsymbol{w}) \\
&= \prod_{i=1}^{N} \mathbb{P}(y_i \mid \boldsymbol{x}_i, \boldsymbol{w}) \mathbb{P}(\boldsymbol{x}_i) \\
&= \prod_{i=1}^{N} \eta_{\boldsymbol{w}}(\boldsymbol{x}_i)^{y_i} (1 - \eta_{\boldsymbol{w}}(\boldsymbol{x}_i))^{1-y_i} \mathbb{P}(\boldsymbol{x}_i) \\
\log \mathbb{P}(\mathcal{D} \mid \boldsymbol{w}) &= \sum_{i=1}^{N} y_i \log \eta_{\boldsymbol{w}}(\boldsymbol{x}_i) + (1 - y_i) \log(1 - \eta_{\boldsymbol{w}}(\boldsymbol{x}_i)) + \log \mathbb{P}(\boldsymbol{x}_i)
\end{aligned}$$

# Step 2: Maximize Likelihood

- Maximizing the log likelihood (w.r.t. $\boldsymbol{w}$) is equivalent to maximizing

$$\begin{aligned} L(\boldsymbol{w}, \mathcal{D}) &= \sum_{i=1}^{N} y_i \log \eta_{\boldsymbol{w}}(\boldsymbol{x}_i) + (1 - y_i) \log(1 - \eta_{\boldsymbol{w}}(\boldsymbol{x}_i)) \\ &= \sum_{i=1}^{N} y_i \log g(\boldsymbol{w}^\intercal \boldsymbol{x}_i) + (1 - y_i) \log(1 - g(\boldsymbol{w}^\intercal \boldsymbol{x}_i)) \end{aligned}$$

Note: $g'(t) = \frac{\exp(-t)}{(1+\exp(-t))^2} = g(t)(1 - g(t))$

- $L$ is concave in $\boldsymbol{w}$: compute gradient and cancel it!

# Iterative Method

$$\nabla_{\boldsymbol{w}} L(\boldsymbol{w}, \mathcal{D}) = \sum_{i=1}^{N} y_i \boldsymbol{x}_i (1 - g(\boldsymbol{w}^\intercal \boldsymbol{x}_i)) - (1 - y_i) \boldsymbol{x}_i g(\boldsymbol{w}^\intercal \boldsymbol{x}_i)$$
$$= \sum_{i=1}^{N} \boldsymbol{x}_i (y_i - g(\boldsymbol{w}^\intercal \boldsymbol{x}_i))$$

- $\nabla_{\boldsymbol{w}} L(\boldsymbol{w}, \mathcal{D}) = 0$ defines a system of non-linear equations
- Issue: no closed-form solution
- Solution: gradient ascent

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha \nabla_{\boldsymbol{w}} L(\boldsymbol{w}, \mathcal{D})$$

- Other solution: Newton (also called Newton-Raphson) method

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \mathbb{H}^{-1} \nabla_{\boldsymbol{w}} L(\boldsymbol{w}, \mathcal{D})$$

where $\mathbb{H} = \left(\frac{\partial^2 L(\boldsymbol{w}, \mathcal{D})}{\partial w_i \partial w_j}\right)$ is called the Hessian of $L$.

# Multi-class Logistic Regression

- i.i.d. data (notation with trick): $\boldsymbol{X} = \left(\boldsymbol{x}^1, \ldots, \boldsymbol{x}^N\right)^\intercal \in \mathbb{R}^{N\times(D+1)}$,
  $\boldsymbol{y} = (y^1, \ldots, y^N)^\intercal \in \{1, 2, \ldots, K\}^N$
- Probabilistic multiclass classifier: $f(\boldsymbol{x}) = \arg\max_k \mathbb{P}(Y = k \,|\, X = \boldsymbol{x})$
  Note: for each $\boldsymbol{x}$, $\mathbb{P}(Y \,|\, X = \boldsymbol{x})$ is a categorical distribution.
- Logistic regression assumes: $\eta$ is based on a softmax function:

$$\mathbb{P}_{\boldsymbol{W}}(Y = k \,|\, X = \boldsymbol{x}) = \frac{\exp(\boldsymbol{w}_k^\intercal \boldsymbol{x})}{\sum_{k=1}^K \exp(\boldsymbol{w}_k^\intercal \boldsymbol{x})}$$

  where $\boldsymbol{W} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K) \in \mathbb{R}^{(D+1)\times K}$
- $\boldsymbol{W}$ can be estimated by Maximum Likelihood (or MAP)

# Logistic Regression with Linear Basis Functions

- Labeled i.i.d. data: $(\boldsymbol{x}^1, y^1), (\boldsymbol{x}^2, y^2), \ldots, (\boldsymbol{x}^N, y^N)$ with $\boldsymbol{X} \in \mathbb{R}^{D\times N}, \boldsymbol{y} \in \{1, \ldots K\}^N$ Notation: $\boldsymbol{X} = (\boldsymbol{x}^1, \ldots, \boldsymbol{x}^N)$, $\boldsymbol{y} = (y^1, \ldots, y^N)$
- Basis functions and features: $\phi(\boldsymbol{x}) = (\phi_0(\boldsymbol{x}), \phi_1(\boldsymbol{x}), \ldots, \phi_M(\boldsymbol{x}))^\intercal \in \mathbb{R}^{M+1}$ with $\phi_0(\boldsymbol{x}) = 1$
- Model:
$$\mathbb{P}_{\boldsymbol{W}}(Y = k \mid X = \boldsymbol{x}) = \frac{\exp(\boldsymbol{w}_k^\intercal \phi(\boldsymbol{x}))}{\sum_{k=1}^K \exp(\boldsymbol{w}_k^\intercal(\boldsymbol{x}))}$$
where $\boldsymbol{W} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K) \in \mathbb{R}^{(M+1)\times K}$
- $\boldsymbol{W}$ can be estimated by Maximum Likelihood (or MAP)

# How Logistic Regression Fits the General Framework?

- What is the Loss Function Optimized for Logistic Regression?

# How Logistic Regression Fits the General Framework?

- What is the Loss Function Optimized for Logistic Regression?
- Recall: $\min_{h \in \mathcal{C}} R_\mu(h) = \mathbb{E}_{(X,Y)\sim\mu}[\ell(h(X), Y)]$ with $\mu \in \mathcal{P}$

# How Logistic Regression Fits the General Framework?

- What is the Loss Function Optimized for Logistic Regression?
- Recall: $\min_{h \in \mathcal{C}} R_\mu(h) = \mathbb{E}_{(X,Y)\sim\mu}[\ell(h(X), Y)]$ with $\mu \in \mathcal{P}$
- Approximated by $\min_{H \in \mathcal{H}} R_{\mathcal{D}}(H) = \min_{H \in \mathcal{H}} \sum_{i=1}^{N} \ell(H(\boldsymbol{x}^i), y^i)$

# How Logistic Regression Fits the General Framework?

- What is the Loss Function Optimized for Logistic Regression?
- Recall: $\min_{h \in \mathcal{C}} R_\mu(h) = \mathbb{E}_{(X,Y)\sim\mu}[\ell(h(X), Y)]$ with $\mu \in \mathcal{P}$
- Approximated by $\min_{H \in \mathcal{H}} R_{\mathcal{D}}(H) = \min_{H \in \mathcal{H}} \sum_{i=1}^{N} \ell(H(\boldsymbol{x}^i), y^i)$
- Here, $\max L(\boldsymbol{w}, \mathcal{D}) = \max \sum_{i=1}^{N} y^i \log \eta_{\boldsymbol{w}}(\boldsymbol{x}^i) + (1 - y^i) \log(1 - \eta_{\boldsymbol{w}}(\boldsymbol{x}^i))$

# How Logistic Regression Fits the General Framework?

- What is the Loss Function Optimized for Logistic Regression?
- Recall: $\min_{h \in \mathcal{C}} R_\mu(h) = \mathbb{E}_{(X,Y)\sim\mu}[\ell(h(X), Y)]$ with $\mu \in \mathcal{P}$
- Approximated by $\min_{H \in \mathcal{H}} R_{\mathcal{D}}(H) = \min_{H \in \mathcal{H}} \sum_{i=1}^{N} \ell(H(\boldsymbol{x}^i), y^i)$
- Here, $\max L(\boldsymbol{w}, \mathcal{D}) = \max \sum_{i=1}^{N} y^i \log \eta_{\boldsymbol{w}}(\boldsymbol{x}^i) + (1 - y^i) \log(1 - \eta_{\boldsymbol{w}}(\boldsymbol{x}^i))$
- Therefore, this loss function, also called log loss, is a cross-entropy or KL-divergence:

$$\ell(\eta_{\boldsymbol{w}}(\boldsymbol{x}), Bern(y)) = -y \log \eta_{\boldsymbol{w}}(\boldsymbol{x}) - (1 - y) \log(1 - \eta_{\boldsymbol{w}}(\boldsymbol{x}))$$
$$\ell(\eta_{\boldsymbol{w}}(\boldsymbol{x}), Bern(y)) = D(Bern(y) || \eta_{\boldsymbol{w}}(\boldsymbol{x}))$$

# Framework of Perceptron

- i.i.d. data (notation with trick): $\boldsymbol{X} = \left(\boldsymbol{x}^1, \dots, \boldsymbol{x}^N\right)^\intercal \in \mathbb{R}^{N\times(D+1)}$,
  $\boldsymbol{y} = (y^1, \dots, y^N)^\intercal \in \{-1, 1\}^N$

- Binary classifier: $f(\boldsymbol{x}) = \begin{cases} 1 & \text{if } h(\boldsymbol{x}) \geq 0 \\ -1 & \text{if } h(\boldsymbol{x}) < 0 \end{cases}$
  where $h(\boldsymbol{x})$ is a function that defines the decision boundary.

- Perceptron assumes that $h$ is a linear function: $h_{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{w}^\intercal \boldsymbol{x}$
- How can we learn $\boldsymbol{w}$?

# Perceptron Algorithm

- With $\ell(\hat{y}, y) = \max(0, -y\hat{y})$, $\min\limits_{H} R_{\mathcal{D}}(H) = \sum_{i=1}^{N} \max(0, -y\boldsymbol{w}^\intercal \boldsymbol{x}^i)$
- Convex optimization problem: use (stochastic) (sub)gradient descent

---

Perceptron$\big(T, \mathcal{D}\big)$
initialize $\boldsymbol{w}$
**for** $i = 1$ *to* $T$ **do**
    select $\boldsymbol{x}^i, y^i$ in $\mathcal{D}$
    **if** $sign(\boldsymbol{w}^\intercal \boldsymbol{x}^i) \neq y^i$ **then**
        $\boldsymbol{w} \leftarrow \boldsymbol{w} + y^i \boldsymbol{x}^i$
return $\boldsymbol{w}$

---

- Simple interpretation: adjust $\boldsymbol{w}$ if there's an error
- Guaranteed to converge only if problem linearly separable
- What could we do if it's not linearly separable?

# Adaline Algorithm

- Issue: previous update does not take into account the size of the error
- Idea: Use instead $\ell(y, y') = (y - y')^2$, $\min_H R_{\mathcal{D}}(H) = \sum_{i=1}^{N} (y - \boldsymbol{w}^\intercal \boldsymbol{x}^i)^2$
- Convex optimization problem: use (stochastic) gradient descent

---

Adaline$(T, \mathcal{D})$
initialize $\boldsymbol{w}$
**for** $i = 1$ *to* $T$ **do**
    select $\boldsymbol{x}^i, y^i$ in $\mathcal{D}$
    $\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha(y^i - \boldsymbol{w}^\intercal \boldsymbol{x}^i)\boldsymbol{x}^i$
return $\boldsymbol{w}$

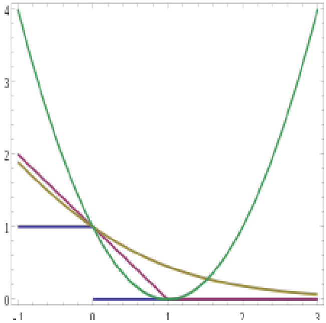
---

- Does this look familiar?

1. Linear Regression
2. Logistic Regression
3. Perceptron
4. Discussions
   - Summary
   - Bias-Variance Trade-off

# Summary

- Given dataset $\mathcal{D}$
- General approach for ERM:
  - Choose $\mathcal{H} = \{h_{\boldsymbol{\theta}} : \mathcal{X} \rightarrow \mathcal{Y}\}$
  - Choose loss function $\ell$
  - Apply stochastic gradient descent to get $\boldsymbol{\theta}^*$
- How well will $h_{\boldsymbol{\theta}^*}$ do on new data?




Examples of loss functions for classification

# Bias-Variance Decomposition for Regression

- Where does the error made by our trained model come from?
- Assumption: $y = f(\boldsymbol{x}) + \varepsilon$ with uncorrelated noise $\varepsilon$ ($\mathbb{E}[\varepsilon] = 0$ and $\mathbb{V}[\varepsilon] = \sigma^2$)
- The error for a given $\boldsymbol{x}$ can be decomposed as follows:

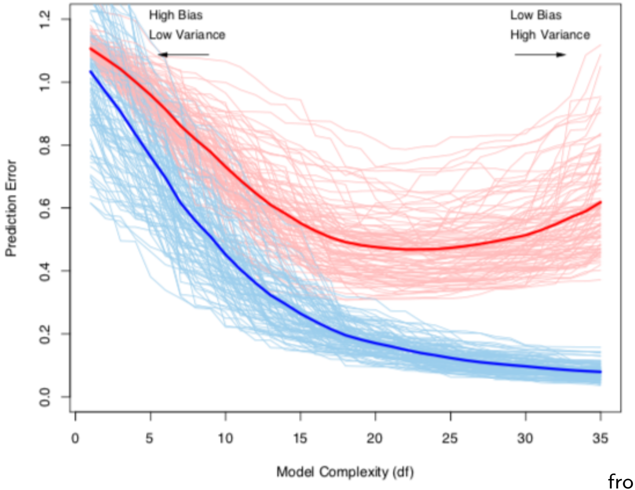
$$\mathbb{E}[(H(\boldsymbol{x}) - Y)^2] = \text{irreducible error } + \text{bias}^2 + \text{ variance}$$

where the expectation over the distributions of $\mathcal{D}$ and $\varepsilon$.

- Proof:

$$\begin{aligned}
\mathbb{E}[(H(\boldsymbol{x}) - Y)^2] &= \mathbb{E}[H(\boldsymbol{x})^2] - 2\mathbb{E}[H(\boldsymbol{x})Y] + \mathbb{E}[Y^2] \\
&= \mathbb{V}[H(\boldsymbol{x})] + \mathbb{E}[H(\boldsymbol{x})]^2 - 2\mathbb{E}[H(\boldsymbol{x})f(\boldsymbol{x})] + \mathbb{V}[Y] + \mathbb{E}[Y]^2 \\
&= \mathbb{V}[H(\boldsymbol{x})] + \mathbb{E}[H(\boldsymbol{x})]^2 - 2\mathbb{E}[H(\boldsymbol{x})f(\boldsymbol{x})] + f(\boldsymbol{x})^2 + \mathbb{V}[\epsilon] \\
&= \sigma^2 + \mathbb{E}[H(\boldsymbol{x}) - f(\boldsymbol{x})]^2 + \mathbb{V}[H(\boldsymbol{x})]
\end{aligned}$$

# Bias-Variance Tradeoff




from Hastie et al.

- Underfitting vs overfitting

# Regularization

- Idea: penalize complex hypothesis $H$
- Regularized ERM: $\min_H R_{\mathcal{D}}(H) - \lambda\rho(H)$
  where $\lambda$ is a hyperparameter, $\rho(H)$ is a complexity measure of $H$
- Examples for linear models:
  - L1 regularization: $\min_{\boldsymbol{w}} R_{\mathcal{D}}(H_{\boldsymbol{w}}) - \lambda||\boldsymbol{w}||_1$
  - L2 regularization: $\min_{\boldsymbol{w}} R_{\mathcal{D}}(H_{\boldsymbol{w}}) - \lambda||\boldsymbol{w}||_2$
- Learning procedure: Apply batch or stochastic gradient descent

# Sample Complexity

- Intuitive definition: # training samples needed to learn target function

- Regret $\hat{\mathcal{R}}_{\mathcal{D}}(H) = R_\mu(H) - \inf_{C \in \mathcal{C}} R_\mu(C)$
- Expected Regret $\mathcal{R}_{N,\mu}(H) = \mathbb{E}_{\mathcal{D}}[\hat{\mathcal{R}}_{\mathcal{D}}(H)]$
- PAC model (Probably, Approximately Correct) with $\delta \in (0, 1)$, $\epsilon > 0$

$$\mathbb{P}(\hat{R}_{\mathcal{D}}(H) \leq \epsilon) \geq 1 - \delta$$

- Sample complexity for expected regret with $\epsilon > 0$: $N$ s.t. $R_{N,\mu}(H) \leq \epsilon$
- Sample complexity in PAC setting: $N$ s.t. previous inequality holds