Intro into Machine Learning

Linear Models. Numerical optimization. Logistic Regression. Figures of Merits. Overfitting. Model Selection. Regularization.

Fifth Machine Learning in High Energy Physics Summer School,
MLHEP 2019, August 1—10

Alexey Artemov[1,2]

[1] Skoltech [2] Higher School of Economics

Lecture overview

- Linear models for regression
- Numerical and stochastic optimization at a glance
- Linear models for classification
- Figures of merits
- Overfitting: how to fool the linear regression
- Regularization
- A Bayesian perspective on regularization

“Physics-based” vs. machine-learned models

- Some criteria for machine learning to be applied in a dependency recovery setting:

“Physics-based” vs. machine-learned models

- Some criteria for machine learning to be applied in a dependency recovery setting:
  - Little prior knowledge of the dependency exists

“Physics-based” vs. machine-learned models

- Some criteria for machine learning to be applied in a dependency recovery setting:
  - Little prior knowledge of the dependency exists
  - The dependency has a complex form too hard for manual examination

“Physics-based” vs. machine-learned models

- Some criteria for machine learning to be applied in a dependency recovery setting:
  - Little prior knowledge of the dependency exists
  - The dependency has a complex form too hard for manual examination
  - A sample from the dependency of sufficiently large size is available

“Physics-based” vs. machine-learned models

› Would one want to apply machine learning for . . .

“Physics-based” vs. machine-learned models

- Would one want to apply machine learning for . . .
  - . . . Newton’s mechanics?

“Physics-based” vs. machine-learned models

- Would one want to apply machine learning for . . .
  - . . . Newton’s mechanics?
  - . . . suggesting music to radio listeners?

“Physics-based” vs. machine-learned models

- Would one want to apply machine learning for . . .
  - . . . Newton’s mechanics?
  - . . . suggesting music to radio listeners?
  - . . . sorting integers?

“Physics-based” vs. machine-learned models

- Would one want to apply machine learning for . . .
  - . . . Newton’s mechanics?
  - . . . suggesting music to radio listeners?
  - . . . sorting integers?
  - . . . controlling steel production?

“Physics-based” vs. machine-learned models

- Would one want to apply machine learning for . . .
    - . . . Newton’s mechanics?
    - . . . suggesting music to radio listeners?
    - . . . sorting integers?
    - . . . controlling steel production?
    - . . . sorting strawberries?

Machine learning as function approximation

- An unknown distribution D generates instances $(\mathbf{x}_1, \mathbf{x}_2, \ldots)$ independently

Machine learning as function approximation

- An unknown distribution D generates instances $(\mathbf{x}_1, \mathbf{x}_2, \ldots)$ independently
- An unknown function $f : \mathbb{X} \to \mathbb{Y}$ generates responses $(y_1, y_2, \ldots)$ for them such that $y_i = f(\mathbf{x}_i), i = 1, 2, \ldots$

Machine learning as function approximation

- An unknown distribution D generates instances $(\mathbf{x}_1, \mathbf{x}_2, \ldots)$ independently
- An unknown function $f : \mathbb{X} \to \mathbb{Y}$ generates responses $(y_1, y_2, \ldots)$ for them such that $y_i = f(\mathbf{x}_i), i = 1, 2, \ldots$
- The machine learning problem: choose a plausible hypothesis $h : \mathbb{X} \to \mathbb{Y}$ from the hypothesis space $\mathbb{H}$

Machine learning as function approximation

- An unknown distribution D generates instances $(\mathbf{x}_1, \mathbf{x}_2, \ldots)$ independently
- An unknown function $f : \mathbb{X} \to \mathbb{Y}$ generates responses $(y_1, y_2, \ldots)$ for them such that $y_i = f(\mathbf{x}_i), i = 1, 2, \ldots$
- The machine learning problem: choose a plausible hypothesis $h : \mathbb{X} \to \mathbb{Y}$ from the hypothesis space $\mathbb{H}$
- The error of a hypothesis h is the deviation from the true f measured by the loss function (an example for regression):

$$Q(h, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} (f(\mathbf{x}_i) - h(\mathbf{x}_i))^2$$

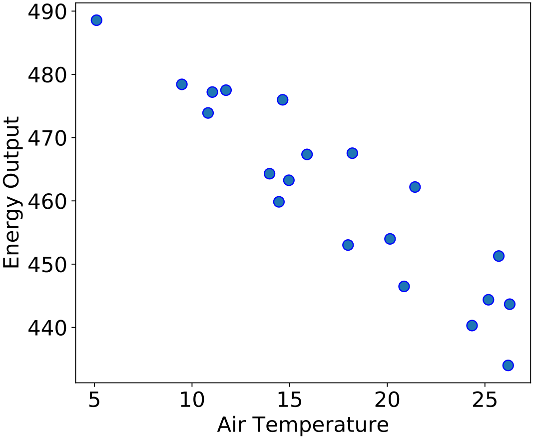
Machine learning as function approximation

- An unknown distribution D generates instances $(\mathbf{x}_1, \mathbf{x}_2, \ldots)$ independently
- An unknown function $f : \mathbb{X} \to \mathbb{Y}$ generates responses $(y_1, y_2, \ldots)$ for them such that $y_i = f(\mathbf{x}_i), i = 1, 2, \ldots$
- The machine learning problem: choose a plausible hypothesis $h : \mathbb{X} \to \mathbb{Y}$ from the hypothesis space $\mathbb{H}$
- The error of a hypothesis h is the deviation from the true f measured by the loss function (an example for regression):

$$Q(h, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} (f(\mathbf{x}_i) - h(\mathbf{x}_i))^2$$

- Learning: the search for the optimal hypothesis $h \in \mathbb{H}$ w. r. t. the fixed loss function

Function approximation: energy usage

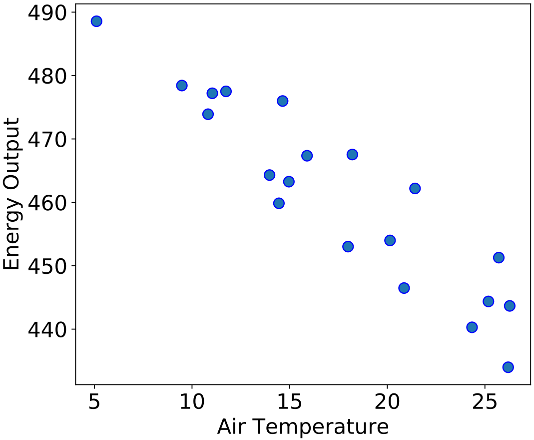


- **The goal:** obtain some fit $f(x)$ plausible for every x_i and y_i

Function approximation: energy usage

instances?

responces?



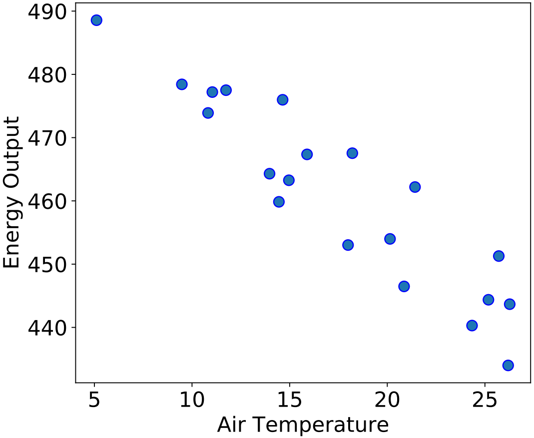
- **The goal:** obtain some fit $f(x)$ plausible for every x_i and y_i

Function approximation: energy usage

instances?

responces?

hypothesis?



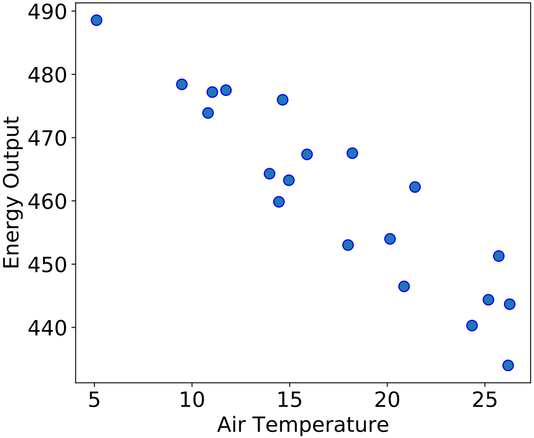
› **The goal:** obtain some fit $f(x)$ plausible for every x_i and y_i

Function approximation: energy usage

instances?

responces?

hypothesis?



loss?

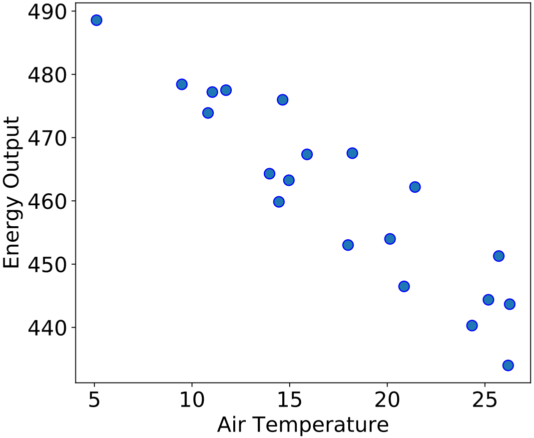
- **The goal:** obtain some fit $f(x)$ plausible for every x_i and y_i

Function approximation: energy usage

instances?

responces?

hypothesis?



loss?

learning?

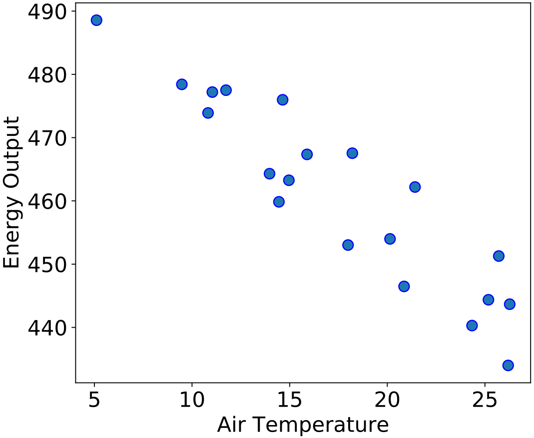
- **The goal:** obtain some fit $f(x)$ plausible for every x_i and y_i

Function approximation: energy usage

instances?

responces?

hypothesis?



loss?

learning?

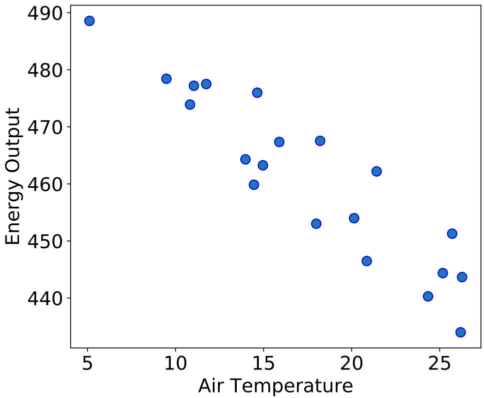
approximation?

- **The goal:** obtain some fit $f(x)$ plausible for every x_i and y_i

Linear models for regression

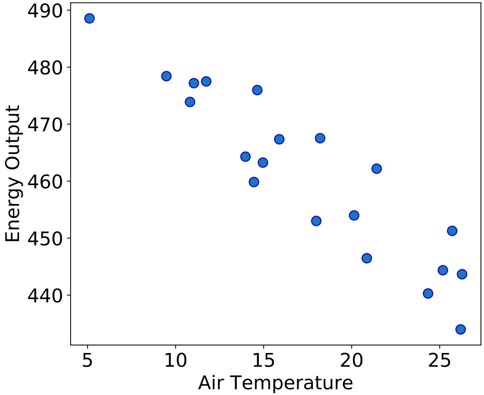
Univariate linear regression

- A single feature (regressor) x:
  Air Temperature



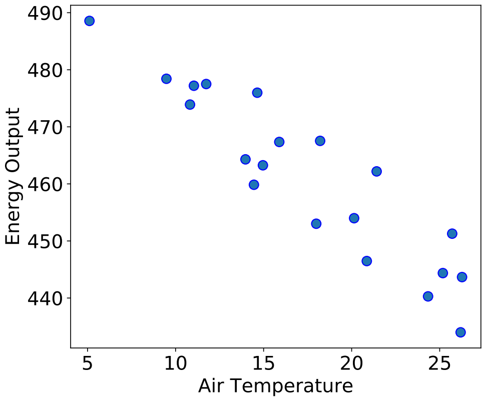
Univariate linear regression

- A single feature (regressor) x:
  Air Temperature
- A single dependent variable y:
  Energy Output



Univariate linear regression

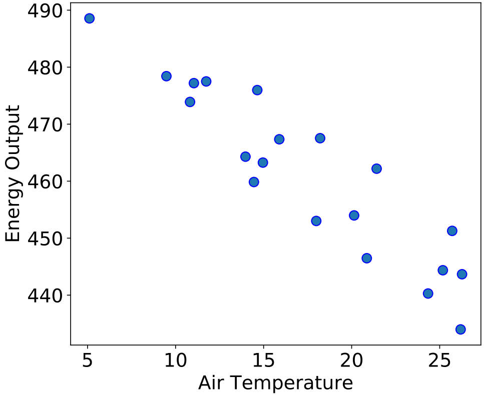
- A single feature (regressor) x:
  Air Temperature
- A single dependent variable y:
  Energy Output
- Training set $X^\ell = \left\{(x_i, y_i)\right\}_{i=1}^{20}$



Univariate linear regression

- A single feature (regressor) x: Air Temperature
- A single dependent variable y: Energy Output
- Training set $X^\ell = \left\{(x_i, y_i)\right\}_{i=1}^{20}$
- The regression model:

$$y_i = h(x_i; \mathbf{w}) + \varepsilon_i$$

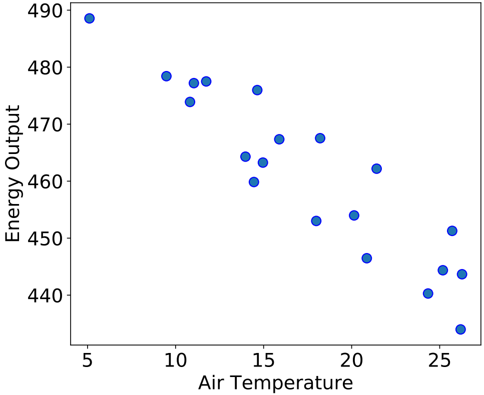


Univariate linear regression

- A single feature (regressor) x: Air Temperature
- A single dependent variable y: Energy Output
- Training set $X^\ell = \left\{(x_i, y_i)\right\}_{i=1}^{20}$
- The regression model:

$$y_i = h(x_i; \mathbf{w}) + \varepsilon_i$$

- Linear model: $y_i = w_1 x_i + w_0 + \varepsilon_i$

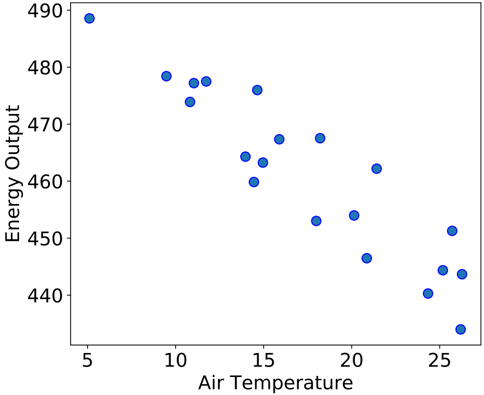


Univariate linear regression

- A single feature (regressor) x:
  Air Temperature
- A single dependent variable y:
  Energy Output
- Training set $X^\ell = \left\{(x_i, y_i)\right\}_{i=1}^{20}$
- The regression model:

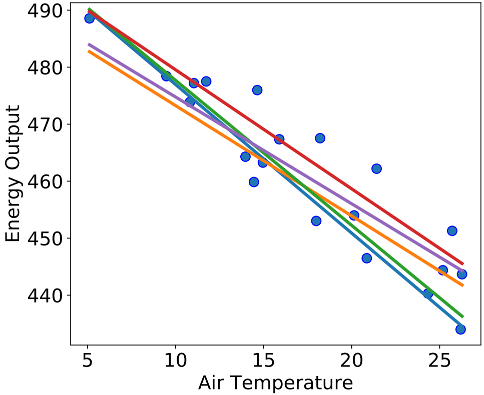
$$y_i = h(x_i; \mathbf{w}) + \varepsilon_i$$

- Linear model: $y_i = w_1 x_i + w_0 + \varepsilon_i$
- The goal: given X^ℓ, find $\mathbf{w} = (w_1, w_0)$



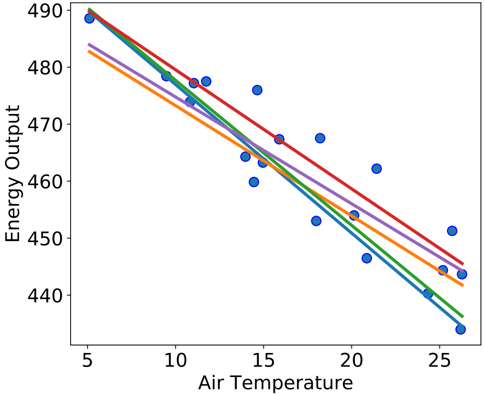
Univariate linear regression

- Which fit to choose?



Univariate linear regression

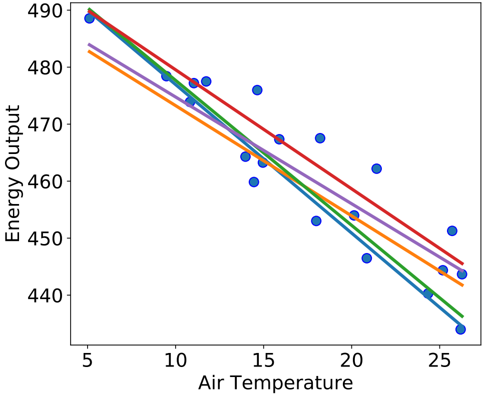
- Which fit to choose?
- With the linear model being fixed, depends on the data and the loss function!



Univariate linear regression

- Which fit to choose?
- With the linear model being fixed, depends on the data and the loss function!
- Mean square (L2) loss (MSE):

$$Q(h, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} (y_i - h(x_i))^2$$



Some other evaluation metrics for regression

- Mean square (L2) loss (MSE): $\mathrm{MSE}(h, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} (y_i - h(x_i))^2$

Some other evaluation metrics for regression

- Mean square (L2) loss (MSE): $\mathrm{MSE}(h, X^\ell) = \frac{1}{\ell}\sum_{i=1}^{\ell}(y_i - h(x_i))^2$
- Root MSE: $\mathrm{RMSE}(h, X^\ell) = \sqrt{\frac{1}{\ell}\sum_{i=1}^{\ell}(y_i - h(x_i))^2}$

Some other evaluation metrics for regression

- Mean square (L2) loss (MSE): $\mathrm{MSE}(h, X^\ell) = \frac{1}{\ell}\sum\limits_{i=1}^{\ell}(y_i - h(x_i))^2$
- Root MSE: $\mathrm{RMSE}(h, X^\ell) = \sqrt{\frac{1}{\ell}\sum\limits_{i=1}^{\ell}(y_i - h(x_i))^2}$
- Coefficient of determination: $R^2(h, X^\ell) = 1 - \frac{\sum_{i=1}^{\ell}(y_i - h(x_i))^2}{\sum_{i=1}^{\ell}(y_i - \mu_y)^2}$

  with $\mu_y = \frac{1}{\ell}\sum_{i=1}^{\ell} y_i$

Some other evaluation metrics for regression

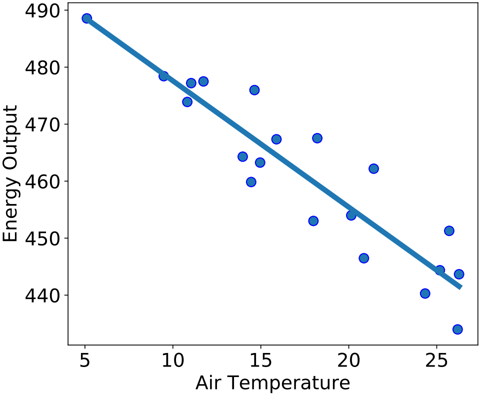
- Mean square (L2) loss (MSE): $\mathrm{MSE}(h, X^\ell) = \frac{1}{\ell}\sum\limits_{i=1}^{\ell}(y_i - h(x_i))^2$
- Root MSE: $\mathrm{RMSE}(h, X^\ell) = \sqrt{\frac{1}{\ell}\sum\limits_{i=1}^{\ell}(y_i - h(x_i))^2}$
- Coefficient of determination: $R^2(h, X^\ell) = 1 - \frac{\sum_{i=1}^{\ell}(y_i - h(x_i))^2}{\sum_{i=1}^{\ell}(y_i - \mu_y)^2}$

  with $\mu_y = \frac{1}{\ell}\sum_{i=1}^{\ell} y_i$
- Mean absolute error: $\mathrm{MAE}(h, X^\ell) = \frac{1}{\ell}\sum\limits_{i=1}^{\ell}|y_i - h(x_i)|$

Univariate linear regression

- With the loss fixed, the linear problem reduces to optimization:

$$\frac{1}{\ell}\sum_{i=1}^{\ell}(y_i - w_1 x_i - w_0)^2 \to \min_{(w_0,w_1)\in\mathbb{R}^2},$$



Univariate linear regression

- With the loss fixed, the linear problem reduces to optimization:

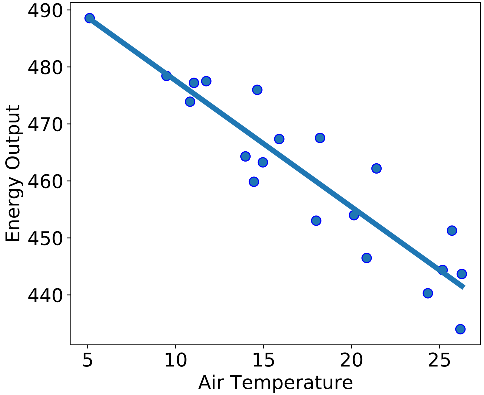
$$\frac{1}{\ell}\sum_{i=1}^{\ell}(y_i - w_1 x_i - w_0)^2 \to \min_{(w_0,w_1)\in\mathbb{R}^2},$$

to which an analytical solution is available

$$\widehat{w}_1 = \frac{\sum_{i=1}^{\ell}(x_i - \mu_x)(y_i - \mu_y)}{\sum_{i=1}^{\ell}(x_i - \mu_x)^2},$$

$$\widehat{w}_0 = \mu_y - \widehat{w}_1 \mu_x$$

with $\mu_x = \frac{1}{\ell}\sum_{i=1}^{\ell} x_i, \quad \mu_y = \frac{1}{\ell}\sum_{i=1}^{\ell} y_i$



Multivariate linear regression

- Multiple features (regressors) $\mathbf{x}_i = (x_{1i}, \dots x_{di})$ available for each y_i
- The model:

$$
\begin{aligned}
y_1 &= w_1 x_{11} + \dots w_d x_{d1} + \varepsilon_1, \\
y_2 &= w_1 x_{12} + \dots w_d x_{d2} + \varepsilon_2, \\
&\dots \\
y_\ell &= w_1 x_{1\ell} + \dots w_d x_{d\ell} + \varepsilon_\ell,
\end{aligned}
$$

is often written in matrix-vector form as

$$
\begin{bmatrix} y_1 \\ \vdots \\ y_\ell \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{d1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1\ell} & x_{2\ell} & \dots & x_{d\ell} \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_d \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_\ell \end{bmatrix} \quad \longleftrightarrow \quad \boldsymbol{y} = \boldsymbol{X}\mathbf{w} + \boldsymbol{\varepsilon}
$$

Multivariate linear regression: the solution

- The problem: minimize MSE

$$Q(h, X^l) = \sum_{i=1}^{\ell} \Big(y_i - \sum_{k=1}^{d} w_k x_{ki} \Big)^2 \equiv \|\boldsymbol{y} - \boldsymbol{X}\mathbf{w}\|^2 \to \min_{\mathbf{w} \in \mathbb{R}^d}$$

Multivariate linear regression: the solution

- The problem: minimize MSE

$$Q(h, X^l) = \sum_{i=1}^{\ell} \Big(y_i - \sum_{k=1}^{d} w_k x_{ki}\Big)^2 \equiv \|\boldsymbol{y} - \boldsymbol{X}\mathbf{w}\|^2 \to \min_{\mathbf{w} \in \mathbb{R}^d}$$

- Solve analytically via computing the gradient

$$\nabla_{\mathbf{w}} \|\boldsymbol{y} - \boldsymbol{X}\mathbf{w}\|^2 = 2(\boldsymbol{y} - \boldsymbol{X}\mathbf{w})\boldsymbol{X} = 0$$

Multivariate linear regression: the solution

- The problem: minimize MSE

$$Q(h, X^l) = \sum_{i=1}^{\ell} \Big(y_i - \sum_{k=1}^{d} w_k x_{ki} \Big)^2 \equiv \|\boldsymbol{y} - \boldsymbol{X}\mathbf{w}\|^2 \to \min_{\mathbf{w} \in \mathbb{R}^d}$$

- Solve analytically via computing the gradient

$$\nabla_{\mathbf{w}} \|\boldsymbol{y} - \boldsymbol{X}\mathbf{w}\|^2 = 2(\boldsymbol{y} - \boldsymbol{X}\mathbf{w})\boldsymbol{X} = 0$$

- The solution

$$\mathbf{w}^* = (\boldsymbol{X}^\intercal \boldsymbol{X})^{-1} \boldsymbol{X}^\intercal \boldsymbol{y}$$

Numerical and stochastic optimization at a glance

A quick intro into the Numerical Optimization

- Consider the optimization problem in $\mathbb{R}^d$

$$f(\mathbf{x}) \to \min_{\mathbf{x} \in \mathbb{R}^d} \qquad \text{(such as } f(\mathbf{w}) \equiv \sum_{i=1}^{\ell} \left(y_i - \sum_{k=1}^{d} w_k x_{ki} \right)^2 \to \min_{\mathbf{w} \in \mathbb{R}^d} \text{)}$$

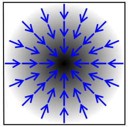
A quick intro into the Numerical Optimization

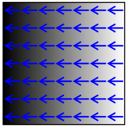
- Consider the optimization problem in $\mathbb{R}^d$

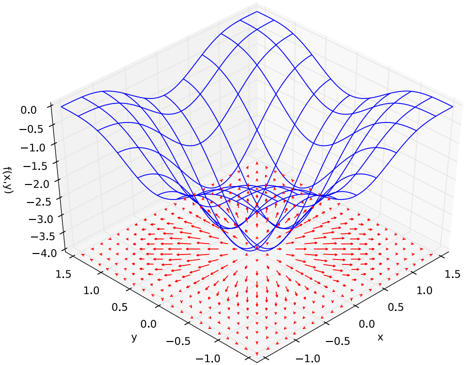
$$f(\mathbf{x}) \to \min_{\mathbf{x} \in \mathbb{R}^d} \qquad \text{(such as } f(\mathbf{w}) \equiv \sum_{i=1}^{\ell} \left(y_i - \sum_{k=1}^{d} w_k x_{ki} \right)^2 \to \min_{\mathbf{w} \in \mathbb{R}^d} \text{)}$$

- In general, solved using the numerical methods such as the gradient descent
- Gradients: directions in $\mathbb{R}^d$ pointing towards steepest function increase

$$\nabla_{\mathbf{x}} f(\mathbf{x}) \equiv \left(\frac{\partial f(\mathbf{x})}{\partial x_1}, \ldots, \frac{\partial f(\mathbf{x})}{\partial x_d} \right)$$







The gradient descent algorithm

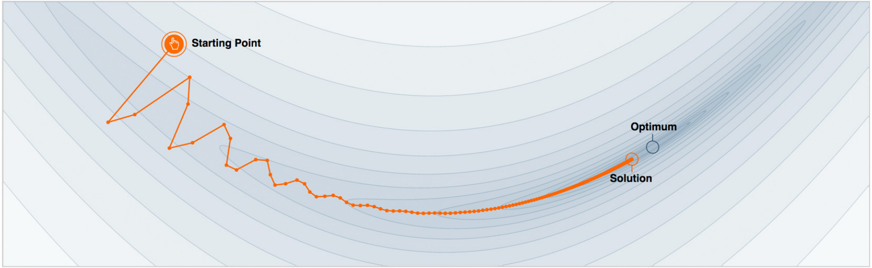
- The gradient descent procedure iterates from $\mathbf{x}^{(0)}$ as
$$\mathbf{x}^{(k)} \leftarrow \mathbf{x}^{(k-1)} - \alpha_k \nabla_{\mathbf{x}} f\big(\mathbf{x}^{(k-1)}\big)$$
with α_k controlling the kth step size

The gradient descent algorithm

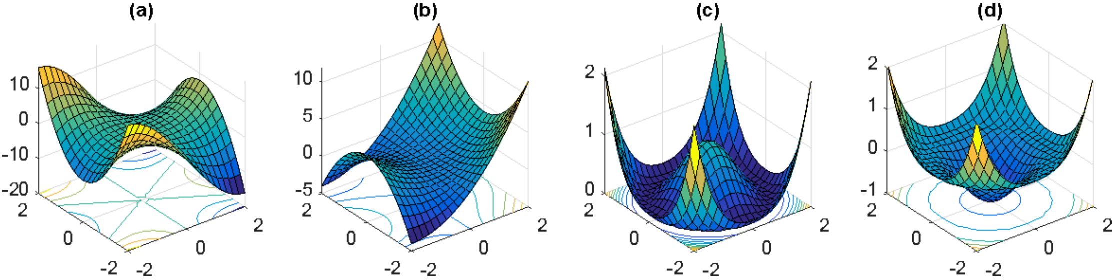
- The gradient descent procedure iterates from $\mathbf{x}^{(0)}$ as

$$\mathbf{x}^{(k)} \leftarrow \mathbf{x}^{(k-1)} - \alpha_k \nabla_{\mathbf{x}} f(\mathbf{x}^{(k-1)})$$

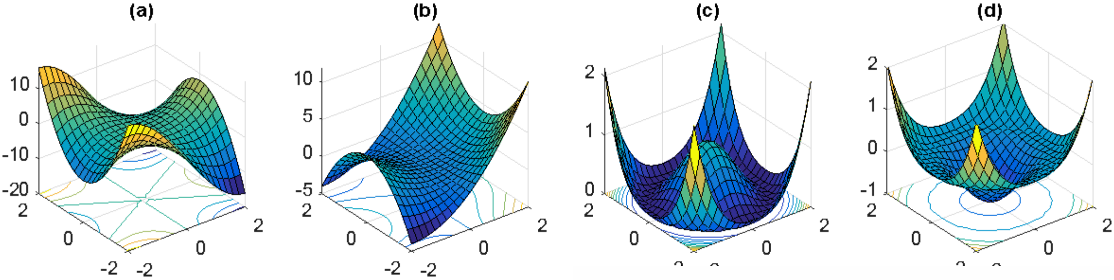
  with α_k controlling the kth step size
- For smooth convex functions with a single minimum $\mathbf{x}^*$, k steps of gradient descent achieve accuracy $f(\mathbf{x}^{(k)}) - f(\mathbf{x}^*) = \mathcal{O}(1/k)$



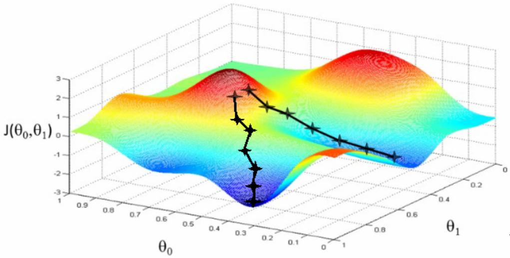
The gradient descent with non-convex targets



The gradient descent with non-convex targets



- Trajectories of gradient descent over non-convex functions may (and will) not always end up in a single optimum



The stochastic gradient descent algorithm

- Machine learning: many additive targets $f(\mathbf{w}) = \sum_{i=1}^{\ell} f_i(\mathbf{w})$, computationally inefficient for large ℓ

The stochastic gradient descent algorithm

- Machine learning: many additive targets $f(\mathbf{w}) = \sum\limits_{i=1}^{\ell} f_i(\mathbf{w})$, computationally inefficient for large ℓ
- Use subsamples for gradient estimation: the Stochastic Gradient Descent (SGD)
  1. Pick $i_k \in \{1, \ldots, \ell\}$ at random;
  2. Compute $\mathbf{w}^{(k)} \leftarrow \mathbf{w}^{(k-1)} - \alpha_k \nabla_{\mathbf{w}} f_{i_k}(\mathbf{w}^{(k-1)})$

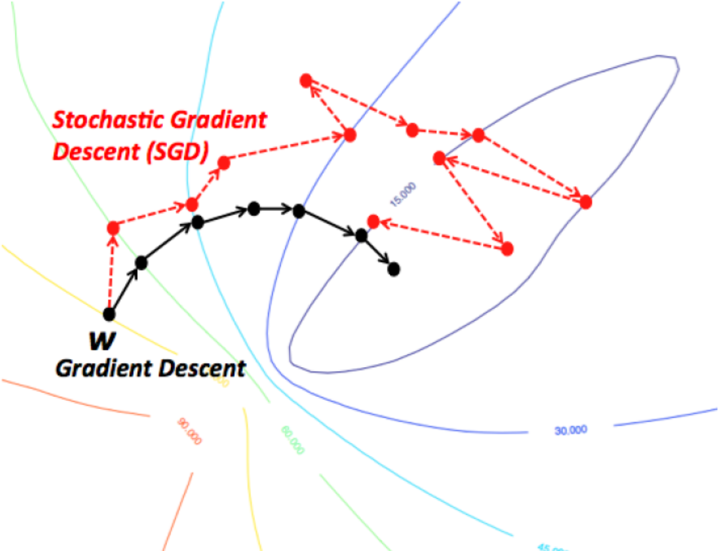
The stochastic gradient descent algorithm

- Machine learning: many additive targets $f(\mathbf{w}) = \sum\limits_{i=1}^{\ell} f_i(\mathbf{w})$, computationally inefficient for large ℓ
- Use subsamples for gradient estimation: the Stochastic Gradient Descent (SGD)
  1. Pick $i_k \in \{1, \dots, \ell\}$ at random;
  2. Compute $\mathbf{w}^{(k)} \leftarrow \mathbf{w}^{(k-1)} - \alpha_k \nabla_{\mathbf{w}} f_{i_k}(\mathbf{w}^{(k-1)})$
- For smooth convex functions with a single minimum $\mathbf{x}^*$, k steps of SGD achieve accuracy $f(\mathbf{w}^{(k)}) - f(\mathbf{w}^*) = \mathcal{O}(1/\sqrt{k})$

The stochastic gradient descent algorithm

- Machine learning: many additive targets $f(\mathbf{w}) = \sum\limits_{i=1}^{\ell} f_i(\mathbf{w})$, computationally inefficient for large ℓ
- Use subsamples for gradient estimation: the Stochastic Gradient Descent (SGD)
    1. Pick $i_k \in \{1, \ldots, \ell\}$ at random;
    2. Compute $\mathbf{w}^{(k)} \leftarrow \mathbf{w}^{(k-1)} - \alpha_k \nabla_{\mathbf{w}} f_{i_k}(\mathbf{w}^{(k-1)})$
- For smooth convex functions with a single minimum $\mathbf{x}^*$, k steps of SGD achieve accuracy $f(\mathbf{w}^{(k)}) - f(\mathbf{w}^*) = \mathcal{O}(1/\sqrt{k})$
- Batching, variance reduction, momentum hacks available to improve the convergence rate to $\mathcal{O}(1/k)$

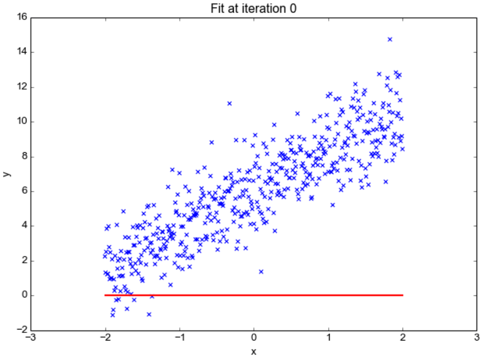
Gradient descent VS stochastic gradient descent



An example: SGD for multivariate linear regression

- Initialize with some $\mathbf{w}^{(0)}$
- Gradient in i_kth object is

  $\nabla_{\mathbf{x}} f_{i_k}(\mathbf{w}) = 2(y_{i_k} - \mathbf{x}_{i_k}^{\intercal}\mathbf{w})\mathbf{x}_{i_k} \qquad (\in \mathbb{R}^d)$
- Compute updates using SGD:
  $\mathbf{w}^{(k)} \leftarrow \mathbf{w}^{(k-1)} - \alpha_k \nabla_{\mathbf{w}} f_{i_k}(\mathbf{w}^{(k-1)})$



Linear models for classification

Binary classification

- An unknown distribution D generates instances $(\mathbf{x}_1, \mathbf{x}_2, \ldots)$

Binary classification

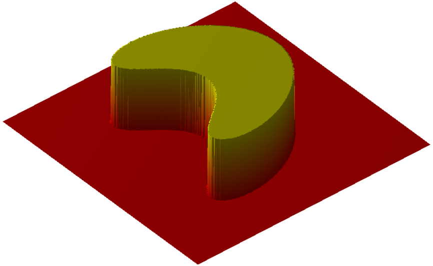
- An unknown distribution D generates instances $(\mathbf{x}_1, \mathbf{x}_2, \ldots)$
- An unknown function $f : \mathbb{X} \to \mathbb{Y}$ generates labels $(y_1, y_2, \ldots)$ for them such that $y_i = f(\mathbf{x}_i)$, and $y_i \in \{-1, +1\}$

Binary classification

- An unknown distribution D generates instances $(\mathbf{x}_1, \mathbf{x}_2, \dots)$
- An unknown function $f : \mathbb{X} \to \mathbb{Y}$ generates labels $(y_1, y_2, \dots)$ for them such that $y_i = f(\mathbf{x}_i)$, and $y_i \in \{-1, +1\}$
- Any examples of such functions? Implications?

Binary classification

- An unknown distribution D generates instances $(\mathbf{x}_1, \mathbf{x}_2, \ldots)$
- An unknown function $f : \mathbb{X} \to \mathbb{Y}$ generates labels $(y_1, y_2, \ldots)$ for them such that $y_i = f(\mathbf{x}_i)$, and $y_i \in \{-1, +1\}$
- **Any examples of such functions? Implications?**



There exist sets A^+, A^- such that $A^+ \equiv \{i : \mathbf{x}_i \in D : y_i = +1\}$ and $A^- \equiv \{i : \mathbf{x}_i \in D : y_i = -1\}$ with indicator functions $\chi_{A^+}(\cdot), \chi_{A^-}(\cdot)$ (displayed on the left)

Binary classification

- An unknown distribution D generates instances $(\mathbf{x}_1, \mathbf{x}_2, \ldots)$
- An unknown function $f : \mathbb{X} \to \mathbb{Y}$ generates labels $(y_1, y_2, \ldots)$ for them such that $y_i = f(\mathbf{x}_i)$, and $y_i \in \{-1, +1\}$
- The classification problem: choose a plausible hypothesis (classifier) $h : \mathbb{X} \to \mathbb{Y}$ from the hypothesis space $\mathbb{H}$

Binary classification

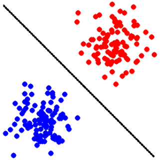
- An unknown distribution D generates instances $(\mathbf{x}_1, \mathbf{x}_2, \ldots)$
- An unknown function $f : \mathbb{X} \to \mathbb{Y}$ generates labels $(y_1, y_2, \ldots)$ for them such that $y_i = f(\mathbf{x}_i)$, and $y_i \in \{-1, +1\}$
- The classification problem: choose a plausible hypothesis (classifier) $h : \mathbb{X} \to \mathbb{Y}$ from the hypothesis space $\mathbb{H}$
- The error of the classifier h is the probability (over D) that it will fail

$$Q(h, D) = \mathrm{Pr}_{\mathbf{x} \sim D}[f(\mathbf{x}) \neq h(\mathbf{x})]$$

usually estimated by the accuracy metric

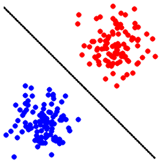
$$Q(h, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} [f(\mathbf{x}_i) \neq h(\mathbf{x}_i)]$$

Linear models for classification



- Linear model: $h(\mathbf{x}) = \text{sign}\Big(\sum_{i=1}^{d} w_i x_i + w_0\Big) = \text{sign}\big(\mathbf{w}^\intercal \mathbf{x} + w_0\big)$

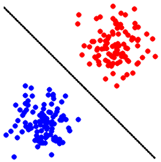
Linear models for classification



- Linear model: $h(\mathbf{x}) = \text{sign}\Big(\sum_{i=1}^{d} w_i x_i + w_0\Big) = \text{sign}\big(\mathbf{w}^\intercal \mathbf{x} + w_0\big)$
- The learning problem is discrete over $\mathbf{w} \in \mathbb{R}^d$:

$$Q(h, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} [\text{sign}(\mathbf{w}^\intercal \mathbf{x}_i) \neq h(\mathbf{x}_i)] \to \min_{\mathbf{w} \in \mathbb{R}^d}$$

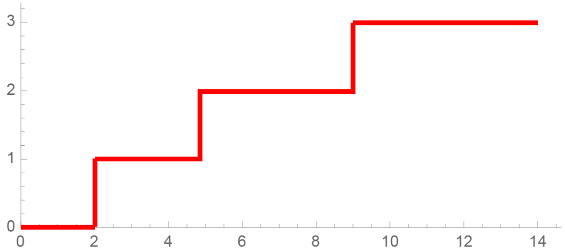
Linear models for classification



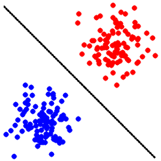
- Linear model: $h(\mathbf{x}) = \text{sign}\Big(\sum_{i=1}^{d} w_i x_i + w_0\Big) = \text{sign}\big(\mathbf{w}^\intercal \mathbf{x} + w_0\big)$
- The learning problem is discrete over $\mathbf{w} \in \mathbb{R}^d$:

$$Q(h, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} [\text{sign}(\mathbf{w}^\intercal \mathbf{x}_i) \neq h(\mathbf{x}_i)] \to \min_{\mathbf{w} \in \mathbb{R}^d}$$

(cannot optimize using gradient descent)



Linear models for classification

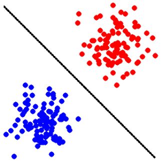


- Linear model: $h(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{d} w_i x_i + w_0\right) = \text{sign}\left(\mathbf{w}^\intercal \mathbf{x} + w_0\right)$
- The learning problem is discrete over $\mathbf{w} \in \mathbb{R}^d$:

$$Q(h, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} [\text{sign}(\mathbf{w}^\intercal \mathbf{x}_i) \neq h(\mathbf{x}_i)] \to \min_{\mathbf{w} \in \mathbb{R}^d}$$

- The solution: optimize a differentiable upper bound for $Q(h, X^\ell)$!

Linear models for classification



- Linear model: $h(\mathbf{x}) = \text{sign}\Big(\sum_{i=1}^{d} w_i x_i + w_0\Big) = \text{sign}\big(\mathbf{w}^\intercal \mathbf{x} + w_0\big)$
- The learning problem is discrete over $\mathbf{w} \in \mathbb{R}^d$:

$$Q(h, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} [\text{sign}(\mathbf{w}^\intercal \mathbf{x}_i) \neq h(\mathbf{x}_i)] \to \min_{\mathbf{w} \in \mathbb{R}^d}$$

- The solution: optimize a differentiable upper bound for $Q(h, X^\ell)$!
- $Q(h, X^\ell)$ can be written using $Q(h, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} L(M_i)$

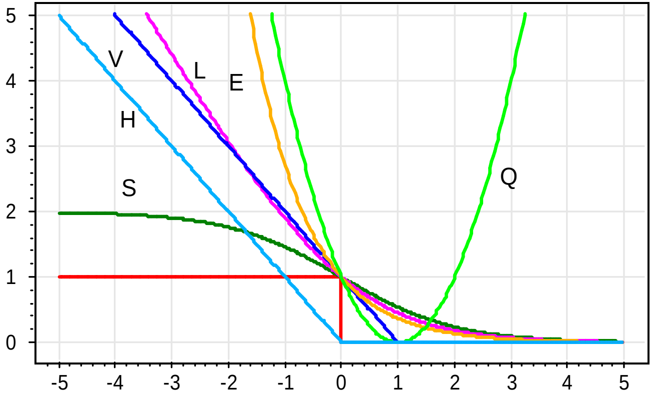
  where $L(M_i) = [M_i < 0] \equiv [y_i \mathbf{w}^\intercal \mathbf{x}_i < 0]$
- Upper-bounding $L(M)$ yields upper bounds for $Q(h, X^\ell)$

Linear models for classification: upper bounds

Multiple approximations to accuracy

- $L_{\mathsf{L}}(M) = \log(1 + e^{-M})$
- $L_{\mathsf{H}}(M) = \max(0, 1 - M)$
- $L_{\mathsf{P}}(M) = \max(0, -M)$
- $L_{\mathsf{E}}(M) = e^{-M}$
- $L_{\mathsf{S}}(M) = 2/(1 + e^{M})$

and their respective optimization procedures give rise to various learning algorithms



The logistic regression model

- Training set $X^\ell = \left\{(\mathbf{x}_i, y_i)\right\}_{i=1}^{\ell}$ where $y_i \in \{-1, +1\}$

The logistic regression model

- Training set $X^\ell = \big\{(\mathbf{x}_i, y_i)\big\}_{i=1}^{\ell}$ where $y_i \in \{-1, +1\}$
- We seek an algorithm h such that $h(\mathbf{x}) = \mathrm{P}(y = +1|\mathbf{x})$ (model probability!)

The logistic regression model

- Training set $X^\ell = \{(\mathbf{x}_i, y_i)\}_{i=1}^{\ell}$ where $y_i \in \{-1, +1\}$
- We seek an algorithm h such that $h(\mathbf{x}) = \mathrm{P}(y = +1|\mathbf{x})$ (model probability!)
- A probability that an instance $(\mathbf{x}_i, y_i)$ is encountered in X^ℓ

$$h(\mathbf{x}_i)^{[y_i=+1]} + (1 - h(\mathbf{x}_i))^{[y_i=-1]}$$

The logistic regression model

- Training set $X^\ell = \{(\mathbf{x}_i, y_i)\}_{i=1}^{\ell}$ where $y_i \in \{-1, +1\}$
- We seek an algorithm h such that $h(\mathbf{x}) = \mathrm{P}(y = +1|\mathbf{x})$ (model probability!)
- A probability that an instance $(\mathbf{x}_i, y_i)$ is encountered in X^ℓ

$$h(\mathbf{x}_i)^{[y_i=+1]} + (1 - h(\mathbf{x}_i))^{[y_i=-1]}$$

- Entire X^ℓ likelihood:

$$L(X^\ell) = \prod_{i=1}^{\ell} h(\mathbf{x}_i)^{[y_i=+1]} + (1 - h(\mathbf{x}_i))^{[y_i=-1]}$$

The logistic regression model

- Training set $X^\ell = \left\{(\mathbf{x}_i, y_i)\right\}_{i=1}^{\ell}$ where $y_i \in \{-1, +1\}$
- We seek an algorithm h such that $h(\mathbf{x}) = \mathrm{P}(y = +1|\mathbf{x})$ (model probability!)
- A probability that an instance $(\mathbf{x}_i, y_i)$ is encountered in X^ℓ

$$h(\mathbf{x}_i)^{[y_i=+1]} + (1 - h(\mathbf{x}_i))^{[y_i=-1]}$$

- Entire X^ℓ likelihood:

$$L(X^\ell) = \prod_{i=1}^{\ell} h(\mathbf{x}_i)^{[y_i=+1]} + (1 - h(\mathbf{x}_i))^{[y_i=-1]}$$

is often written via log-likelihood (of which the negative is log-loss)

$$\log L(X^\ell) = \sum_{i=1}^{\ell} [y_i = +1] \log h(\mathbf{x}_i) + [y_i = -1] \log(1 - h(\mathbf{x}_i))$$

The logistic regression model

- The choice of h: sigmoid function

$$h(\mathbf{x}) = \sigma(\mathbf{w}^\intercal \mathbf{x})$$

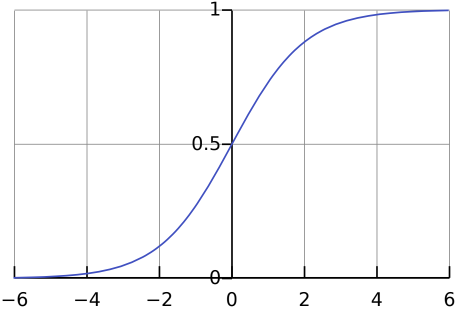
  where $\sigma(x) \in [0, 1]$

- Typical choice: the logistic function

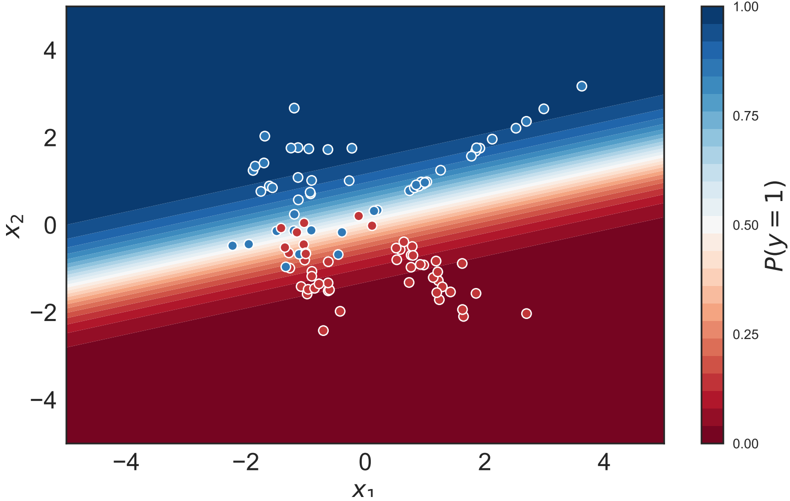
$$\sigma(\mathbf{w}^\intercal \mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^\intercal \mathbf{x})}$$

- Plugging the logistic function into the loss yields an approximation of accuracy

$$\sum_{i=1}^{\ell} (1 + \exp(\mathbf{w}^\intercal \mathbf{x})) \to \min_{\mathbf{w} \in \mathbb{R}^d}$$



The logistic regression model



Figures of merits

Classification quality evaluation: accuracy

- Given a labeled sample $X^\ell = \left\{(\mathbf{x}_i, y_i)\right\}_{i=1}^{\ell}$, $y_i \in \{-1, +1\}$, and some candidate h, how well does h perform on X^ℓ?

Classification quality evaluation: accuracy

- Given a labeled sample $X^\ell = \left\{(\mathbf{x}_i, y_i)\right\}_{i=1}^{\ell}$, $y_i \in \{-1, +1\}$, and some candidate h, **how well does h perform on X^ℓ?**
- Let the thresholded decision rule be $a(x) = [h(x) > t]$ (t: hyperparameter)

Classification quality evaluation: accuracy

- Given a labeled sample $X^\ell = \left\{(\mathbf{x}_i, y_i)\right\}_{i=1}^{\ell}$, $y_i \in \{-1, +1\}$, and some candidate h, how well does h perform on X^ℓ?
- Let the thresholded decision rule be $a(x) = [h(x) > t]$ (t: hyperparameter)
- Obvious choice: accuracy

$$\text{accuracy}(a, X^\ell) = \frac{1}{\ell}\sum_{i=1}^{\ell}[a(\mathbf{x}_i) = y_i]$$

- Example: Higgs challenge — selection of the interesting signal $H \to \tau\tau$ decay against the already known backround
- 164,333 background, 85,667 signal events (66% background)

Classification quality evaluation: confusion matrix

	Label $y = 1$	Label $y = -1$
Decision $a(x) = 1$	True Positive (TP)	False Positive (FP)
Decision $a(x) = -1$	False negative (FN)	True Negative (TN)

Classification quality evaluation: confusion matrix

	Label $y = 1$	Label $y = -1$
Decision $a(x) = 1$	True Positive (TP)	False Positive (FP)
Decision $a(x) = -1$	False negative (FN)	True Negative (TN)

- Rates are often more informative:

$$\text{False Positive Rate aka FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}},$$

$$\text{True Positive Rate aka TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}},$$

Classification quality evaluation: confusion matrix

	Label $y = 1$	Label $y = -1$
Decision $a(x) = 1$	True Positive (TP)	False Positive (FP)
Decision $a(x) = -1$	False negative (FN)	True Negative (TN)

- Rates are often more informative:

$$\text{False Positive Rate aka FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}},$$

$$\text{True Positive Rate aka TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}},$$

- While accuracy can be expressed, too

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

Classification quality: the receiver operating curve

- Often $h(\mathbf{x})$ is more valuable than its thresholded version $a(x) = [h(x) > t]$

Classification quality: the receiver operating curve

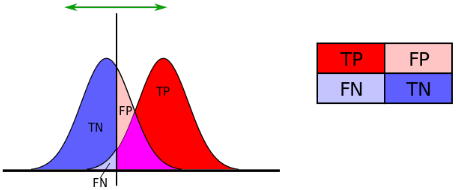
- Often $h(\mathbf{x})$ is more valuable than its thresholded version $a(x) = [h(x) > t]$
- Consider two-dimensional space with coordinates (TPR(t), FPR(t)), corresponding to various choices of the threshold t

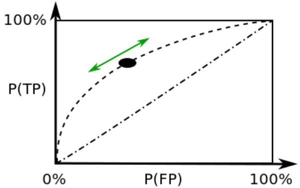
Classification quality: the receiver operating curve

- Often $h(\mathbf{x})$ is more valuable than its thresholded version $a(x) = [h(x) > t]$
- Consider two-dimensional space with coordinates (TPR(t), FPR(t)), corresponding to various choices of the threshold t
- The plot TPR(t) vs. FPR(t) is called the receiver operating characteristic (ROC) curve

Classification quality: the receiver operating curve

- Often $h(\mathbf{x})$ is more valuable than its thresholded version $a(x) = [h(x) > t]$
- Consider two-dimensional space with coordinates (TPR(t), FPR(t)), corresponding to various choices of the threshold t
- The plot TPR(t) vs. FPR(t) is called the receiver operating characteristic (ROC) curve
- Area under curve (ROC-AUC) reflects classification quality





Source: Wikipedia

Classification quality: imbalanced data

- Recall the Higgs: 164,333 background vs. 85,667 signal events (66% background)

Classification quality: imbalanced data

- Recall the Higgs: 164,333 background vs. 85,667 signal events (66% background)
- TPR(t) vs. FPR(t) / ROC is bad for imbalanced data: for $\ell = 1000$, $n_- = 950$ (high background noise), $n_+ = 50$ (low signal), a trivial rule $h(\mathbf{x}) = -1$ ("treat everything as background") would yield:

Classification quality: imbalanced data

- Recall the Higgs: 164,333 background vs. 85,667 signal events (66% background)
- TPR(t) vs. FPR(t) / ROC is bad for imbalanced data: for $\ell = 1000$, $n_- = 950$ (high background noise), $n_+ = 50$ (low signal), a trivial rule $h(\mathbf{x}) = -1$ ("treat everything as background") would yield:
  - accuracy$(a, X^\ell) = 0.95$ (bad)

Classification quality: imbalanced data

- Recall the Higgs: 164,333 background vs. 85,667 signal events (66% background)
- TPR(t) vs. FPR(t) / ROC is bad for imbalanced data: for $\ell = 1000$, $n_- = 950$ (high background noise), $n_+ = 50$ (low signal), a trivial rule $h(\mathbf{x}) = -1$ ("treat everything as background") would yield:
  - accuracy$(a, X^\ell) = 0.95$ (bad)
  - TPR$(a, X^\ell) = 0.$ (OK)

Classification quality: imbalanced data

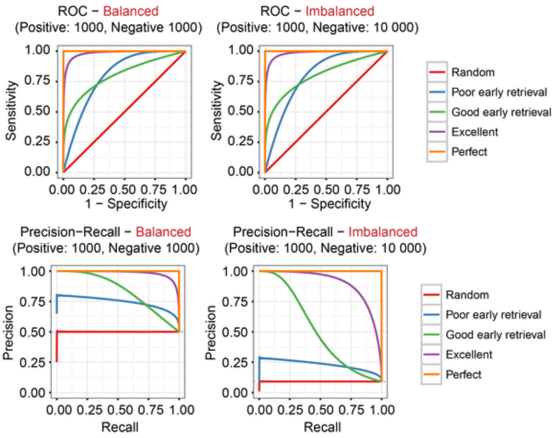
- Recall the Higgs: 164,333 background vs. 85,667 signal events (66% background)
- TPR(t) vs. FPR(t) / ROC is bad for imbalanced data: for $\ell = 1000$, $n_- = 950$ (high background noise), $n_+ = 50$ (low signal), a trivial rule $h(\mathbf{x}) = -1$ ("treat everything as background") would yield:
  - accuracy$(a, X^\ell) = 0.95$ (bad)
  - TPR$(a, X^\ell) = 0.$ (OK)
  - FPR$(a, X^\ell) = 0.$ (bad)

Classification quality: imbalanced data

- Recall the Higgs: 164,333 background vs. 85,667 signal events (66% background)
- TPR(t) vs. FPR(t) / ROC is bad for imbalanced data: for $\ell = 1000$, $n_- = 950$ (high background noise), $n_+ = 50$ (low signal), a trivial rule $h(\mathbf{x}) = -1$ ("treat everything as background") would yield:
  - accuracy$(a, X^\ell) = 0.95$ (bad)
  - TPR$(a, X^\ell) = 0.$ (OK)
  - FPR$(a, X^\ell) = 0.$ (bad)
- Criteria better suited for imbalanced problems:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \qquad \text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

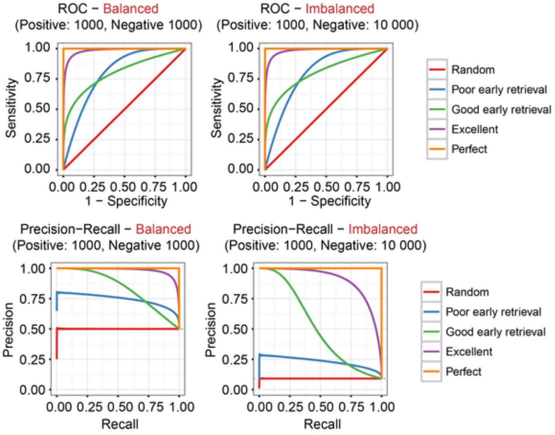
Classification quality: imbalanced data



Source: classeval.wordpress.com

› The plot recall vs. precision is called the precision-recall (PR) curve

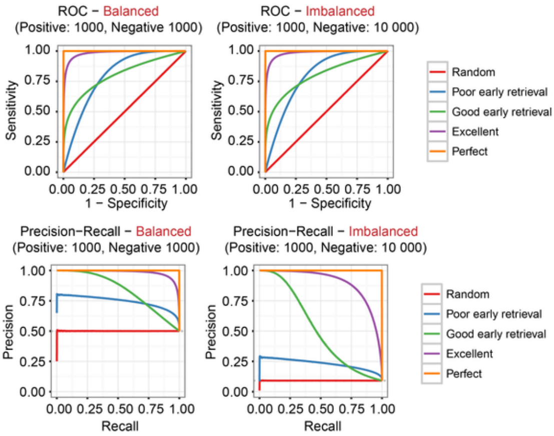
Classification quality: imbalanced data



Source: classeval.wordpress.com

- The plot recall vs. precision is called the precision-recall (PR) curve
- Recall(t) vs. Precision(t) is good for imbalanced data: for $\ell = 1000$, $n_- = 950$ (high background noise), $n_+ = 50$ (low signal), a trivial rule $h(\mathbf{x}) = -1$ would yield:

Classification quality: imbalanced data



Source: classeval.wordpress.com

- The plot recall vs. precision is called the precision-recall (PR) curve
- Recall(t) vs. Precision(t) is good for imbalanced data: for $\ell = 1000$, $n_- = 950$ (high background noise), $n_+ = 50$ (low signal), a trivial rule $h(\mathbf{x}) = -1$ would yield:
  - Recall$(a, X^\ell) = 0.$ (OK)
  - Precision$(a, X^\ell) = 0.$ (OK)

Overfitting: how to fool the linear regression

Generalization and overfitting

- Training set memorization: for seen $(\mathbf{x}, y) \in X^\ell$, $h(\mathbf{x}) = y$

Generalization and overfitting

- Training set memorization: for seen $(\mathbf{x}, y) \in X^\ell$, $h(\mathbf{x}) = y$
- Generalization: equally good performance on both new and seen instances

Generalization and overfitting

- Training set memorization: for seen $(\mathbf{x}, y) \in X^\ell$, $h(\mathbf{x}) = y$
- Generalization: equally good performance on both new and seen instances
- How to assess model's generalization ability?

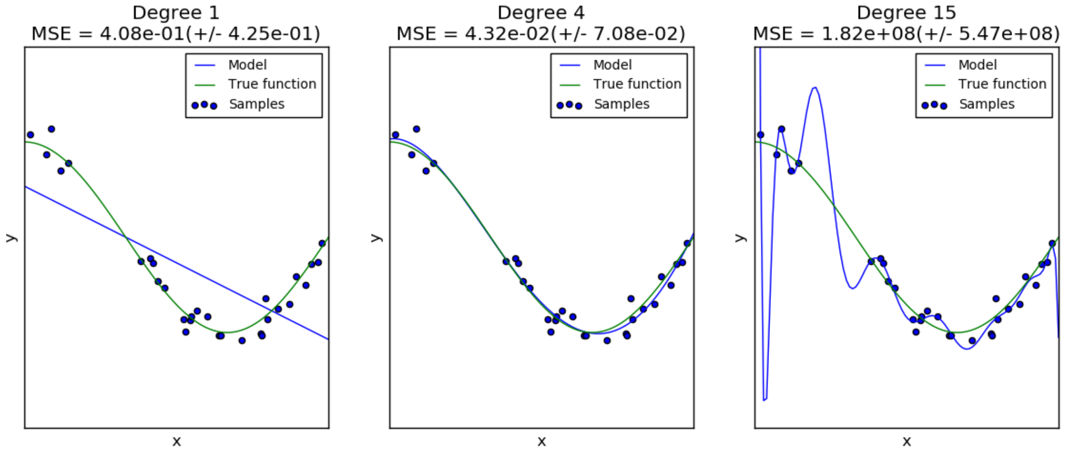
Generalization and overfitting

- Training set memorization: for seen $(\mathbf{x}, y) \in X^\ell$, $h(\mathbf{x}) = y$
- Generalization: equally good performance on both new and seen instances
- How to assess model's generalization ability?
- Consider an example:
  - $y = \cos(1.5\pi x) + \mathcal{N}(0, 0.01)$, $x \sim \mathrm{Uniform}[0, 1]$
  - Features: $\{x\}$, $\{x, x^2, x^3, x^4\}$, $\{x, \dots, x^{15}\}$
  - The model is linear w. r. t. features: $f(\mathbf{x}) = \mathbf{w}^\intercal \phi(\mathbf{x})$

Generalization and overfitting

- Training set memorization: for seen $(\mathbf{x}, y) \in X^\ell$, $h(\mathbf{x}) = y$
- Generalization: equally good performance on both new and seen instances
- How to assess model's generalization ability?
- Consider an example:
  - $y = \cos(1.5\pi x) + \mathcal{N}(0, 0.01)$, $x \sim \text{Uniform}[0, 1]$
  - Features: $\{x\}$, $\{x, x^2, x^3, x^4\}$, $\{x, \dots, x^{15}\}$
  - The model is linear w. r. t. features: $f(\mathbf{x}) = \mathbf{w}^\intercal \phi(\mathbf{x})$
- How well do the regression models perform?

Polynomial fits of different degrees



Model validation and selection

- We have free parameters in models:

Model validation and selection

- We have free parameters in models:
  - polynomial degree d, subset of features in multivariate regression, kernel width in kernel density estimates, ...

Model validation and selection

- We have free parameters in models:
  - polynomial degree d, subset of features in multivariate regression, kernel width in kernel density estimates, ...
- Model selection: how to select optimal hyperparameters for a given classification problem?

Model validation and selection

- We have free parameters in models:
  - polynomial degree d, subset of features in multivariate regression, kernel width in kernel density estimates, ...
- Model selection: how to select optimal hyperparameters for a given classification problem?
- Validation: how to estimate true model performance?

Model validation and selection

- We have free parameters in models:
  - polynomial degree d, subset of features in multivariate regression, kernel width in kernel density estimates, ...
- Model selection: how to select optimal hyperparameters for a given classification problem?
- Validation: how to estimate true model performance?
- Can we use entire dataset to fit the model?

Model validation and selection

- We have free parameters in models:
  - polynomial degree d, subset of features in multivariate regression, kernel width in kernel density estimates, ...
- Model selection: how to select optimal hyperparameters for a given classification problem?
- Validation: how to estimate true model performance?
- Can we use entire dataset to fit the model?
- Yes, but we will likely get overly optimistic performance estimate

Model validation and selection

- We have free parameters in models:
  - polynomial degree d, subset of features in multivariate regression, kernel width in kernel density estimates, ...
- Model selection: how to select optimal hyperparameters for a given classification problem?
- Validation: how to estimate true model performance?
- Can we use entire dataset to fit the model?
- Yes, but we will likely get overly optimistic performance estimate
- The solution: rely on held-out data to assess model performance

Assessing generalization ability: train / validation

- Split training set into two subsets:

$$X^\ell = X^\ell_{\mathsf{TRAIN}} \cup X^\ell_{\mathsf{VAL}}$$

Assessing generalization ability: train / validation

- Split training set into two subsets:

$$X^\ell = X^\ell_{\mathsf{TRAIN}} \cup X^\ell_{\mathsf{VAL}}$$

- Train a model h on X^ℓ_{TRAIN}

Assessing generalization ability: train / validation

- Split training set into two subsets:

$$X^\ell = X^\ell_{\text{TRAIN}} \cup X^\ell_{\text{VAL}}$$

- Train a model h on X^ℓ_{TRAIN}
- Evaluate model h on X^ℓ_{VAL}

Assessing generalization ability: train / validation

- Split training set into two subsets:

$$X^\ell = X^\ell_{\text{TRAIN}} \cup X^\ell_{\text{VAL}}$$

- Train a model h on X^ℓ_{TRAIN}
- Evaluate model h on X^ℓ_{VAL}
- Assess quality using $Q(h, X^\ell_{\text{VAL}})$

Assessing generalization ability: train / validation

- Split training set into two subsets:

$$X^\ell = X^\ell_{\text{TRAIN}} \cup X^\ell_{\text{VAL}}$$

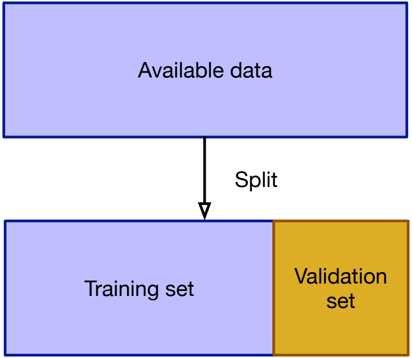
- Train a model h on X^ℓ_{TRAIN}
- Evaluate model h on X^ℓ_{VAL}
- Assess quality using $Q(h, X^\ell_{\text{VAL}})$
- **Data-hungry:** can we afford the ”luxury” of setting aside a portion of the data for testing?

Assessing generalization ability: train / validation

- Split training set into two subsets:

$$X^\ell = X^\ell_{\text{TRAIN}} \cup X^\ell_{\text{VAL}}$$

- Train a model h on X^ℓ_{TRAIN}
- Evaluate model h on X^ℓ_{VAL}
- Assess quality using $Q(h, X^\ell_{\text{VAL}})$
- **Data-hungry:** can we afford the "luxury" of setting aside a portion of the data for testing?
- **May be imprecise:** the holdout estimate of error rate will be misleading if we happen to get an "unfortunate" split



Assessing generalization ability: cross-validation

- Split training set into subsets of equal size
  $X^\ell = X_1^\ell \cup \ldots \cup X_K^\ell$

Assessing generalization ability: cross-validation

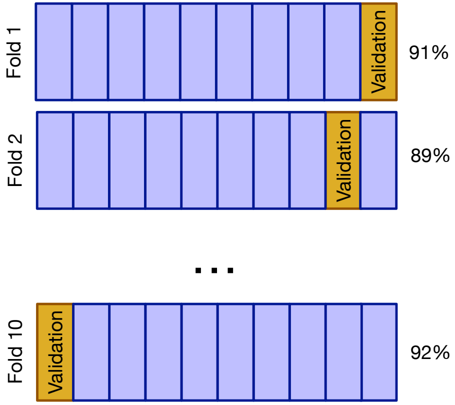
- Split training set into subsets of equal size $X^\ell = X_1^\ell \cup \ldots \cup X_K^\ell$
- Train K models $h_1, \ldots, h_K$ where each model h_k is trained on all subsets but X_k^ℓ

Assessing generalization ability: cross-validation

- Split training set into subsets of equal size $X^\ell = X_1^\ell \cup \ldots \cup X_K^\ell$
- Train K models $h_1, \ldots, h_K$ where each model h_k is trained on all subsets but X_k^ℓ
- Assess quality using $\mathrm{CV} = \frac{1}{K} \sum_{k=1}^{K} Q(h_k, X_k^\ell)$ (K-fold)

Assessing generalization ability: cross-validation

- Split training set into subsets of equal size $X^\ell = X_1^\ell \cup \ldots \cup X_K^\ell$
- Train K models $h_1, \ldots, h_K$ where each model h_k is trained on all subsets but X_k^ℓ
- Assess quality using $\mathrm{CV} = \frac{1}{K} \sum\limits_{k=1}^{K} Q(h_k, X_k^\ell)$ (K-fold)
- Leave-one-out cross-validation: $X_k^\ell = \{(\mathbf{x}_k, y_k)\}$ (yes, train ℓ models!)



Cross-validation method: drawbacks

$$\mathrm{CV} = \frac{1}{K} \sum_{k=1}^{K} Q(h_k, X_k^\ell)$$

Many folds:

- Small bias: the estimator will be very accurate
- Large variance: due to small split sizes
- Costly: many experiments, large computational time

Cross-validation method: drawbacks

$$\mathrm{CV} = \frac{1}{K}\sum_{k=1}^{K} Q(h_k, X_k^{\ell})$$

Many folds:

- Small bias: the estimator will be very accurate
- Large variance: due to small split sizes
- Costly: many experiments, large computational time

Few folds:

- Cheap, computationally effective: few experiments
- Small variance: average over many samples
- Large bias: estimated error rate conservative or smaller than the true error rate

Regularization

Ad-hoc regularization: motivation

- Consider the multivariate linear regression problem with $\boldsymbol{X} \in \mathbb{R}^{d \times d}$

$$\|\boldsymbol{y} - \boldsymbol{X}\mathbf{w}\|^2 \to \min_{\mathbf{w} \in \mathbb{R}^d}$$

Ad-hoc regularization: motivation

- Consider the multivariate linear regression problem with $\boldsymbol{X} \in \mathbb{R}^{d \times d}$

$$\|\boldsymbol{y} - \boldsymbol{X}\mathbf{w}\|^2 \to \min_{\mathbf{w} \in \mathbb{R}^d}$$

- Analytic solution involves computing the product $\boldsymbol{R} = (\boldsymbol{X}^\intercal \boldsymbol{X})^{-1} \boldsymbol{X}^\intercal$

Ad-hoc regularization: motivation

- Consider the multivariate linear regression problem with $\boldsymbol{X} \in \mathbb{R}^{d \times d}$
$$\|\boldsymbol{y} - \boldsymbol{X}\mathbf{w}\|^2 \to \min_{\mathbf{w} \in \mathbb{R}^d}$$
- Analytic solution involves computing the product $\boldsymbol{R} = (\boldsymbol{X}^\intercal \boldsymbol{X})^{-1} \boldsymbol{X}^\intercal$
- If $\boldsymbol{X} = \text{diag}(\lambda_1, \ldots, \lambda_d)$ with $\lambda_1 > \lambda_2 > \ldots > \lambda_d \to 0$ (meaning we're in eigenbasis of $\boldsymbol{X}$) then
$$\begin{aligned}\boldsymbol{R} &= (\boldsymbol{X}^\intercal \boldsymbol{X})^{-1} \boldsymbol{X}^\intercal = \\ &= \big(\text{diag}(\lambda_1, \ldots, \lambda_d)\text{diag}(\lambda_1, \ldots, \lambda_d)\big)^{-1} \text{diag}(\lambda_1, \ldots, \lambda_d) = \\ &= \text{diag}\left(\frac{1}{\lambda_1}, \ldots, \frac{1}{\lambda_d}\right), \quad \text{leading to huge diagonal values in } \boldsymbol{R}\end{aligned}$$

Ad-hoc regularization: L2

- **Regularization:** replace fit with fit + penalty as in

$$Q(\mathbf{w}) \to Q_\alpha(\mathbf{w}) = Q(\mathbf{w}) + \alpha R(\mathbf{w})$$

Ad-hoc regularization: L2

- **Regularization:** replace fit with fit + penalty as in

$$Q(\mathbf{w}) \to Q_\alpha(\mathbf{w}) = Q(\mathbf{w}) + \alpha R(\mathbf{w})$$

- $R(\mathbf{w})$ is called the regularizer, $\alpha > 0$ – the regularization constant

Ad-hoc regularization: L2

- **Regularization:** replace fit with fit + penalty as in

$$Q(\mathbf{w}) \to Q_\alpha(\mathbf{w}) = Q(\mathbf{w}) + \alpha R(\mathbf{w})$$

- $R(\mathbf{w})$ is called the regularizer, $\alpha > 0$ – the regularization constant
- Regularized multivariate linear regression problem

$$\|\boldsymbol{y} - \boldsymbol{X}\mathbf{w}\|^2 + \alpha\|\mathbf{w}\|_2^2 \to \min_{\mathbf{w}\in\mathbb{R}^d}$$

Ad-hoc regularization: L2

- **Regularization:** replace fit with fit + penalty as in

$$Q(\mathbf{w}) \to Q_\alpha(\mathbf{w}) = Q(\mathbf{w}) + \alpha R(\mathbf{w})$$

- $R(\mathbf{w})$ is called the regularizer, $\alpha > 0$ – the regularization constant
- Regularized multivariate linear regression problem

$$\|\boldsymbol{y} - \boldsymbol{X}\mathbf{w}\|^2 + \alpha\|\mathbf{w}\|_2^2 \to \min_{\mathbf{w}\in\mathbb{R}^d}$$

- Regularized analytic solution available

$$\mathbf{w}^* = (\boldsymbol{X}^\intercal \boldsymbol{X} + \alpha \boldsymbol{I})^{-1}\boldsymbol{X}^\intercal \boldsymbol{y}$$

Why L2 regularization works

- Analytic solution: compute the regularized operator

$$\boldsymbol{R} = (\boldsymbol{X}^\intercal \boldsymbol{X} + \alpha \boldsymbol{I})^{-1} \boldsymbol{X}^\intercal$$

Why L2 regularization works

- Analytic solution: compute the regularized operator

$$\boldsymbol{R} = (\boldsymbol{X}^\intercal \boldsymbol{X} + \alpha \boldsymbol{I})^{-1} \boldsymbol{X}^\intercal$$

- If $\boldsymbol{X} = \text{diag}(\lambda_1, \ldots, \lambda_d)$ with $\lambda_1 > \lambda_2 > \ldots > \lambda_d \to 0$ (meaning we're in eigenbasis of $\boldsymbol{X}$) then

$$\begin{aligned}\boldsymbol{R} &= (\boldsymbol{X}^\intercal \boldsymbol{X} + \alpha \boldsymbol{I})^{-1} \boldsymbol{X}^\intercal = \\ &= \Big(\text{diag}(\lambda_1, \ldots, \lambda_d)\text{diag}(\lambda_1, \ldots, \lambda_d) + \text{diag}(\alpha, \ldots, \alpha)\Big)^{-1} \text{diag}(\lambda_1, \ldots, \lambda_d) = \\ &= \text{diag}\left(\frac{\lambda_1}{\lambda_1^2 + \alpha}, \ldots, \frac{\lambda_d}{\lambda_d^2 + \alpha}\right),\end{aligned}$$

smoothing diagonal values in $\boldsymbol{R}$

More regularizers!

- L2 regularized multivariate linear regression problem

$$\|\boldsymbol{y} - \boldsymbol{X}\mathbf{w}\|^2 + \alpha\|\mathbf{w}\|^2 \to \min_{\mathbf{w}\in\mathbb{R}^d}$$

- L1 regularized regression (LASSO)

$$\|\boldsymbol{y} - \boldsymbol{X}\mathbf{w}\|^2 + \alpha\|\mathbf{w}\|_1 \to \min_{\mathbf{w}\in\mathbb{R}^d}$$

- L1/L2 regularized regression (Elastic Net)

$$\|\boldsymbol{y} - \boldsymbol{X}\mathbf{w}\|^2 + \alpha_1\|\mathbf{w}\|_1 + \alpha_2\|\mathbf{w}\|_2^2 \to \min_{\mathbf{w}\in\mathbb{R}^d}$$

More regularizers!

- L2 regularized multivariate linear regression problem

$$\|\boldsymbol{y} - \boldsymbol{X}\mathbf{w}\|^2 + \alpha\|\mathbf{w}\|^2 \to \min_{\mathbf{w}\in\mathbb{R}^d}$$

- L1 regularized regression (LASSO)

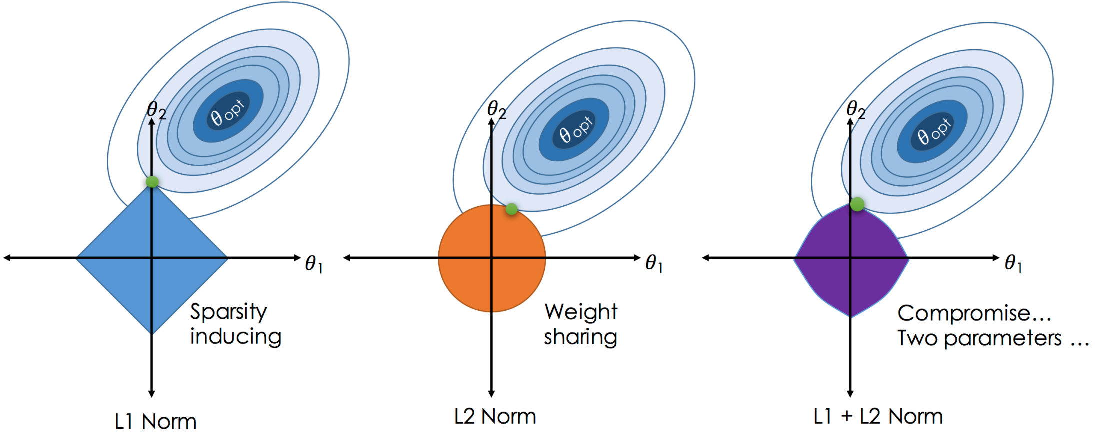
$$\|\boldsymbol{y} - \boldsymbol{X}\mathbf{w}\|^2 + \alpha\|\mathbf{w}\|_1 \to \min_{\mathbf{w}\in\mathbb{R}^d}$$

- L1/L2 regularized regression (Elastic Net)

$$\|\boldsymbol{y} - \boldsymbol{X}\mathbf{w}\|^2 + \alpha_1\|\mathbf{w}\|_1 + \alpha_2\|\mathbf{w}\|_2^2 \to \min_{\mathbf{w}\in\mathbb{R}^d}$$

- Convex $Q(\mathbf{w})$: unconstrained optimization $Q(\mathbf{w}) + \alpha\|\mathbf{w}\|_1$ is equivalent to constrained problem $Q(\mathbf{w})$ s.t. $\|\mathbf{w}\|_1 \leqslant C$

Geometric interpretation of regularizers



Picture credit: http://www.ds100.org/sp17/assets/notebooks/linear_regression/Regularization.html

Another interpretation of regularizers

Figure: Large parameter space

Figure: Regularized models

A Bayesian perspective on regularization

Principled definition of regularizers?

- **L2 regularized** linear regression

$$\|\boldsymbol{y} - \boldsymbol{X}\mathbf{w}\|^2 + \alpha\|\mathbf{w}\|^2 \to \min_{\mathbf{w}\in\mathbb{R}^d}$$

works better in some instances (many correlated features, etc.)

Principled definition of regularizers?

- **L2 regularized** linear regression

$$\|\boldsymbol{y} - \boldsymbol{X}\mathbf{w}\|^2 + \alpha\|\mathbf{w}\|^2 \to \min_{\mathbf{w}\in\mathbb{R}^d}$$

  works better in some instances (many correlated features, etc.)
- But why exactly we formulate penalty as $\alpha\|\mathbf{w}\|^2$?

Principled definition of regularizers?

- **L2 regularized** linear regression

$$\|\boldsymbol{y} - \boldsymbol{X}\mathbf{w}\|^2 + \alpha\|\mathbf{w}\|^2 \to \min_{\mathbf{w}\in\mathbb{R}^d}$$

  works better in some instances (many correlated features, etc.)
- But why exactly we formulate penalty as $\alpha\|\mathbf{w}\|^2$?
- Is there any **principled** way of defining regularizers?

Principled definition of regularizers?

- **L2 regularized** linear regression

$$\|\boldsymbol{y} - \boldsymbol{X}\mathbf{w}\|^2 + \alpha\|\mathbf{w}\|^2 \to \min_{\mathbf{w}\in\mathbb{R}^d}$$

  works better in some instances (many correlated features, etc.)
- But why exactly we formulate penalty as $\alpha\|\mathbf{w}\|^2$?
- Is there any **principled** way of defining regularizers?
- Recall: we wanted $\mathbf{w}$ to take **small values** $\to$ α controls how small they must be

Principled definition of regularizers?

- **L2 regularized** linear regression

$$\|\boldsymbol{y} - \boldsymbol{X}\mathbf{w}\|^2 + \alpha\|\mathbf{w}\|^2 \to \min_{\mathbf{w}\in\mathbb{R}^d}$$

  works better in some instances (many correlated features, etc.)
- But why exactly we formulate penalty as $\alpha\|\mathbf{w}\|^2$?
- Is there any **principled** way of defining regularizers?
- Recall: we wanted $\mathbf{w}$ to take **small values** $\to$ α controls how small they must be
- In (Bayesian) statistics, **priors** are used to characterize variables with known (or assumed) distributions of values $\to$ use priors on weights $\mathbf{w}$!

The prior and the posterior distributions

- x is a mathematical entity

The prior and the posterior distributions

- x is a mathematical entity

Questions:

- What kinds of mathematical entities can have priors?
- What is a prior for x?

The prior and the posterior distributions

- x is a mathematical entity

Questions:

- What kinds of mathematical entities can have priors?
- What is a prior for x?

Answers:

- Prior probability distribution **(prior)**: "the probability distribution that would express one's beliefs about this quantity before some evidence is taken into account." (wiki)
- Most mathematical objects can have priors

The prior and the posterior distributions

- x is a mathematical entity

Questions:

- What kinds of mathematical entities can have priors?
- What is a prior for x?

Answers:

- Prior probability distribution (**prior**): "the probability distribution that would express one's beliefs about this quantity before some evidence is taken into account." (wiki)
- Most mathematical objects can have priors
- Sources: past experiments, expert assessment, ...

A typical scenario in Bayesian data analysis would be:

$$\boxed{\text{design a prior} \rightarrow \text{collect evidence (data)} \rightarrow \text{compute a posterior}}$$

Expressing prior knowledge using math

- How to technically impose the “small values prior” on $\mathbf{w}$?

Expressing prior knowledge using math

- How to technically impose the “small values prior” on $\mathbf{w}$?
- Let $P(\mathbf{w} < \varepsilon)$ be high, and $P(\mathbf{w} \geqslant \varepsilon)$ be (exponentially) low $\rightarrow$ find a distribution that satisfies this condition

Expressing prior knowledge using math

- How to technically impose the “small values prior” on $\mathbf{w}$?
- Let $P(\mathbf{w} < \varepsilon)$ be high, and $P(\mathbf{w} \geqslant \varepsilon)$ be (exponentially) low $\rightarrow$ find a distribution that satisfies this condition
- Example: Gaussian $\mathbf{w} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})$

Prior knowledge for linear regression

- Linear model for regression:

$$\boldsymbol{y} = \boldsymbol{X}\mathbf{w} + \boldsymbol{\varepsilon} \implies y_i = f(x_i) + \varepsilon_i$$

Prior knowledge for linear regression

- Linear model for regression:

$$\boldsymbol{y} = \boldsymbol{X}\mathbf{w} + \boldsymbol{\varepsilon} \implies y_i = f(x_i) + \varepsilon_i$$

- Obtaining MLE for linear regression:
  - $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \sigma_\varepsilon^2 \boldsymbol{I})$;
  - $\sigma_\varepsilon^2 = \mathrm{const}$ (unknown);
  - $\boldsymbol{\varepsilon}$ is the only random variable

Prior knowledge for linear regression

- Linear model for regression:

$$\boldsymbol{y} = \boldsymbol{X}\mathbf{w} + \boldsymbol{\varepsilon} \implies y_i = f(x_i) + \varepsilon_i$$

- Obtaining MLE for linear regression:
  - $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \sigma_\varepsilon^2 \boldsymbol{I})$;
  - $\sigma_\varepsilon^2 = \mathrm{const}$ (unknown);
  - $\boldsymbol{\varepsilon}$ is **the only random variable**

- Likelihood for $\boldsymbol{\varepsilon}$:

$$L = \prod_i P_\varepsilon(\varepsilon_i) \implies \log L = -\sum_i \log P_\varepsilon(\varepsilon_i) = -\sum_i \log P_\varepsilon(y_i - f(x_i))$$

Prior knowledge for linear regression

› MLE for $\mathbf{w}$...

$$L = -\sum_i \log P_\varepsilon(y_i - f(x_i)) =$$

$$\sum_i \left[Z(\sigma_\varepsilon^2) - \frac{(y_i - f(x_i))^2}{2\sigma_\varepsilon^2} \right] \sim$$

$$\sum_i (y_i - f(x_i))^2 \to \min_{\mathbf{w}} \implies \text{MSE problem!}$$

Prior knowledge for linear regression

› MLE for $\mathbf{w}$...

$$L = -\sum_i \log P_\varepsilon(y_i - f(x_i)) =$$

$$\sum_i \left[Z(\sigma_\varepsilon^2) - \frac{(y_i - f(x_i))^2}{2\sigma_\varepsilon^2} \right] \sim$$

$$\sum_i (y_i - f(x_i))^2 \to \min_{\mathbf{w}} \implies \text{MSE problem!}$$

› ...is **the same as the solution for the least squares problem** (with Gaussian errors)

Prior knowledge for linear regression

- Data model: $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \sigma_\varepsilon^2 \boldsymbol{I})$.

Prior knowledge for linear regression

- Data model: $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \sigma_\varepsilon^2 \boldsymbol{I})$.
- Weights prior: $\mathbf{w} \sim \mathcal{N}(0, \Sigma_w)$;

Prior knowledge for linear regression

- Data model: $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \sigma_\varepsilon^2 \boldsymbol{I})$.
- Weights prior: $\mathbf{w} \sim \mathcal{N}(0, \Sigma_w)$;
- Computing posterior we get:

$$
\begin{aligned}
P(\mathbf{w} \mid \boldsymbol{y}, \boldsymbol{X}) &\propto P(\boldsymbol{y} \mid \mathbf{w}, \boldsymbol{X}) P(\mathbf{w}) \\
&\propto \exp\left[-\frac{1}{2\sigma_\varepsilon^2}(\boldsymbol{y} - \boldsymbol{X}\mathbf{w})^T(\boldsymbol{y} - \boldsymbol{X}\mathbf{w})\right] \cdot \exp\left[-\frac{1}{2}\mathbf{w}^T \Sigma_w^{-1} \mathbf{w}\right] = \\
&\propto \exp\left[-\frac{1}{2}(\mathbf{w} - \mathbf{w}^*)^T \boldsymbol{A}_w (\mathbf{w} - \mathbf{w}^*)\right]
\end{aligned}
$$

where:

- $\boldsymbol{A}_w = \frac{1}{\sigma_\varepsilon^2}\boldsymbol{X}\boldsymbol{X}^T + \Sigma_w^{-1}$;
- $\mathbf{w}^* = \frac{1}{\sigma_\varepsilon^2}\boldsymbol{A}_w^{-1}\boldsymbol{X}\boldsymbol{y}$.